Hedera™ Hashgraph

Hedera: A Public Hashgraph Network & Governing Council

The trust layer of the internet

Dr. Leemon Baird, Mance Harmon, and Paul Madsen

Vision

To build a trusted, secure, and empowered digital future for all.

Mission

We are dedicated to building a trusted and secure online world that empowers you.

Where you can work, play, buy, sell, create, and engage socially.

Where you have safety and privacy in your digital communities.

Where you are confident when interacting with others.

Where this digital future is available to all.

Hello future.

Executive Summary

Distributed ledger technologies (DLT) have the potential to disrupt and transform existing markets in multiple industries. However, in our opinion there are five fundamental obstacles to overcome before distributed ledgers can be widely accepted and adopted by enterprises. In this paper we will examine these obstacles and discuss why Hedera Hashgraph is well-suited to support a vast array of applications and become the world's first mass-adopted public distributed ledger.

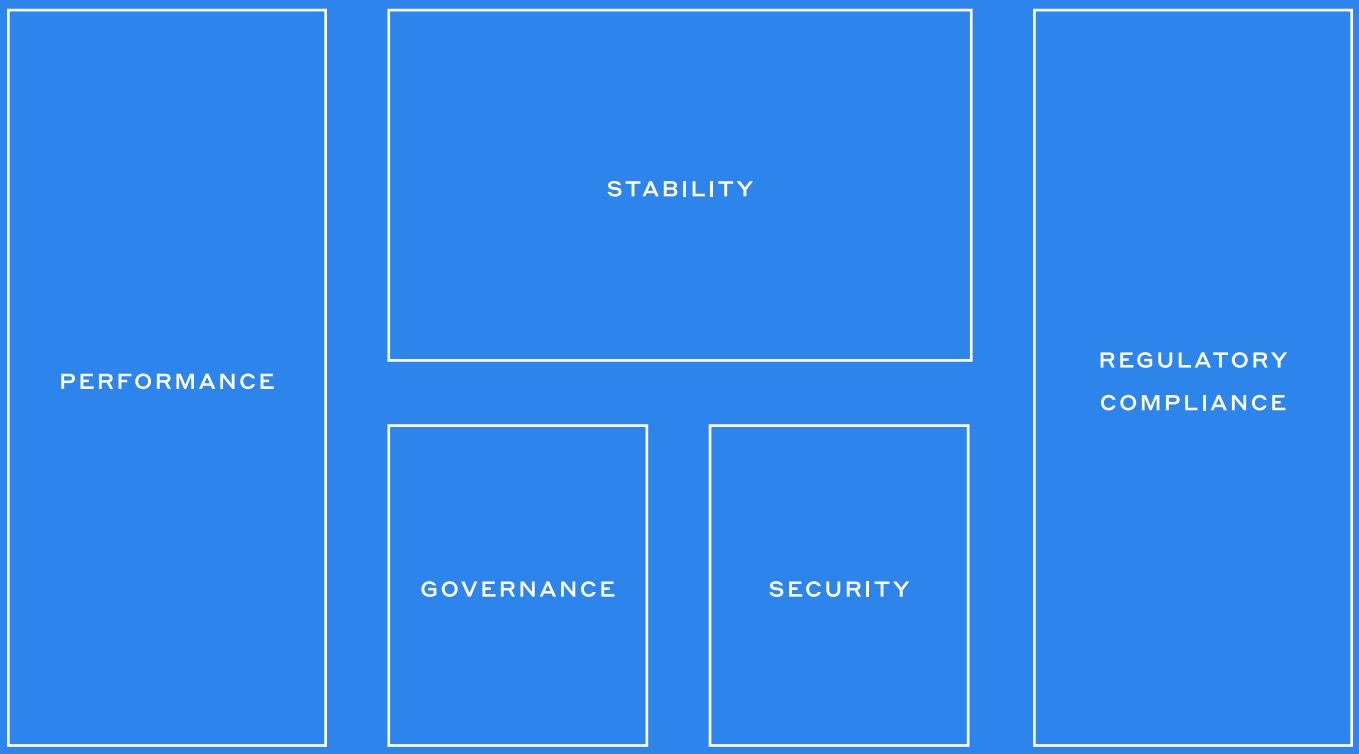
1. **PERFORMANCE** - The most compelling use cases for DLT require hundreds of thousands of transactions per second, and many require consensus latency measured in seconds. These performance metrics are orders of magnitude beyond what current public DLT platforms can achieve.

2. **SECURITY** - If public DLT platforms are to facilitate the transfer of trillions of dollars of value, they will be targeted by hackers, and so will need the strongest possible network security. Having the strongest possible security starts with the consensus algorithm itself, with its security properties formally proven mathematically. Security vulnerabilities and attack vectors shouldn't be mitigated; they should be eliminated entirely. To improve performance metrics, some public DLT platforms are making compromises with regard to decentralization, and in so doing are potentially compromising security.

3. **GOVERNANCE** - A general-purpose public ledger should be governed by representatives from a broad range of market and geographic sectors, each with world-class expertise. Those responsible for network governance need technical expertise so they can competently manage the platform's underlying software. They need business and economics expertise so they can manage business operations of the organization and its cryptocurrency. They need legal expertise to help navigate the evolving regulatory environment. In other words, the network should be governed by a decentralized group of globally recognized industry leaders, representative of every market in the world.

4. **STABILITY** - Without technical and legal mechanisms to enforce the decisions of the governing body, public DLT platforms are at risk of devolving into chaos. Strong security and mature governance will enable a stable platform — one that engenders the necessary trust and confidence among those that would build commercial or sensitive applications on it.

5. **REGULATORY COMPLIANCE** - We expect that governments will continue to extend policy objectives to users, enterprises, and developers utilizing public ledgers and associated cryptocurrencies and tokens. We consider that a public distributed ledger must be capable of providing necessary tools for all members of its ecosystem to comply with applicable laws and regulations, such as the European Union's General Data Privacy Regulations (GDPR), and enable appropriate identity management to conduct sanctions screening and facilitate Know Your Customer (KYC) and Anti Money Laundering (AML) checks.



What is required to move the DLT industry forward and enable it to realize its full potential?

A platform that provides a combination of high performance, strong security, effective governance, technical and legal controls to ensure the platform's stability, and tools to enable regulatory compliance. Only then do we think mainstream markets will trust a DLT platform enough to adopt it.

Introducing Hedera — a public hashgraph network and governing body designed to address the needs of mainstream markets.

The Hedera network will be governed by a council of leading global enterprises, across multiple industries and geographies. Its vision is a cyberspace that is trusted and secure, without the need for centralized parties with inordinate influence. Its licensing and governance model protects users by eliminating the risk of forking, protecting the integrity of the codebase, and providing open access to review the underlying software code. Platform governance will be decentralized through the Hedera Governing Council, which will have a term-limited, rotating set of governing members that each have equal voting rights over key decisions relating to the platform.[1]

The Hedera network is a distributed ledger platform that resolves the factors that constrain adoption of public DLT by the mainstream.

1. **PERFORMANCE** - The platform is built on the hashgraph distributed consensus algorithm, invented by Dr. Leemon Baird. The hashgraph consensus algorithm provides near-perfect efficiency in bandwidth usage and consequently can process hundreds of thousands of transactions per second in a single shard (a fully connected, peer-to-peer mesh of nodes in a network). Initially, we anticipate that the Hedera network will be able to process 10,000 cryptocurrency transactions per second. Consensus latency is measured in seconds, not minutes, hours, or days.

2. **SECURITY** - The hashgraph consensus algorithm achieves the gold standard for security in the field of distributed consensus: asynchronous Byzantine Fault Tolerance (aBFT). Other platforms that use coordinators, leaders, or communication timeouts to improve performance tend to be vulnerable to Distributed Denial of Service (DDoS) attacks. Hashgraph is resilient to these types of attacks against the consensus algorithm because there is no such leader. Achieving this level of security at scale is a fundamental advance in the field of distributed systems.

   Many applications require that the consensus order of transactions match the actual order in which the transactions are received by the network. It should not be possible for a single party to prevent the flow of transactions into the network, nor influence the order of transactions in the eventual network consensus. A fair consensus algorithm ensures that if a user can submit a transaction to the network at all, then the transaction will be received by the network and the order in which it was received will be a fair ordering. Hashgraph uniquely ensures that the actual order transactions are received by the network will be reflected in the consensus order. In other words, hashgraph ensures both Fair Access and Fair Ordering.

   Formal proofs of the aBFT and fairness properties for the hashgraph consensus algorithm have been available for public review since June 2016. Furthermore, the hashgraph algorithm was validated as aBFT by a math proof checked by computer using the Coq system in October 2018.[2]

3. **GOVERNANCE** - The Hedera network will be governed by a council of up to 39 leading global enterprises. Hedera Council members will bring needed experience in process and business expertise that has been absent in previous public ledger platforms. Council membership is designed (i) to reflect a range of industries and geographies, (ii) to have highly respected brands and trusted market positions, and (iii) to encompass competing perspectives.
The terms of governance ensure that no single Council member will have control, and no small group of members will have undue influence over the body as a whole.

4. **STABILITY** - Hedera relies on both technical and legal controls to ensure the stability of the platform.

Hedera technical controls enable two capabilities.

i) **First, the hashgraph technology ensures that software clients validate the pedigree of the Hedera hashgraph ledger prior to use through a shared state mechanism.** It isn't possible for a network node to fork the official version of the Hedera hashgraph platform, make changes, and then have those changes accepted as valid. If the original hashgraph platform and the copy are changed independently, software clients using the Hedera platform will know which is the valid version and which is not.

ii) **Second, the hashgraph technology makes it possible for the Hedera Council to specify the software changes to be made to network nodes, precisely when those changes are to be adopted**, and to confirm that they have been adopted. When the Hedera Council releases a software update, network nodes will have their software automatically updated at exactly the same moment. Any node with invalid software (i.e., one that didn't install the software update) will no longer be able to modify the ledger or have the world accept their version of the ledger as legitimate.

Hedera legal controls ensure the platform will not fork into a competing platform and cryptocurrency.

iii) The Hedera codebase will be governed by the Hedera Governing Council, and will be released for public review with Version 1.0, expected to be released in 2020. It will not be open source, but anyone will be able to read the source code, recompile it, and verify that it is correct. No license will be required to use the Hedera platform. No license will be required to write software that uses the services of the Hedera platform. No license will be required to build smart contracts on top of the Hedera platform. Applications built upon the Hedera platform can be open source or proprietary. They do not require any license or any approval from Hedera. Hedera will make the code publicly available for anyone to review it (at Version 1.0), while ensuring stability by using hashgraph software patents defensively to prevent forks. In this way, Hedera will provide a transparent codebase that will provide the stability that markets demand

for mainstream adoption of a public ledger.

The combination of technical and legal controls provide the governing body with the mechanisms needed to enable meaningful governance, and to bring the stability that we think is required for broad-based adoption.

5. **REGULATORY COMPLIANCE** – The Hedera technical framework includes controlled mutability of the network state and the potential to request or attach additional data to transactions, such as identity certificates. These features enable future functionality such as the erasure of personal data and other uploaded files and opt-in verified identity mechanisms – all optional and within the control of the end users. We intend to work with regulators and encourage development of such tools to allow enterprises to fulfill their consumer protection and regulatory compliance obligations.

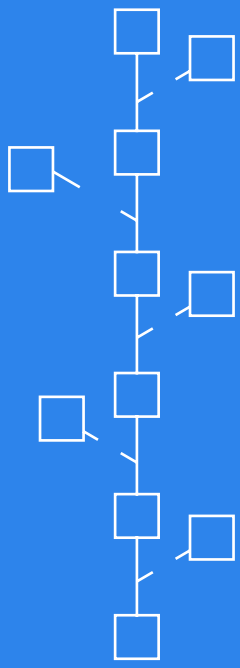
[1] Swirlds, Inc. is a Delaware corporation that holds the patent titles to the hashgraph consensus algorithm. Swirlds is a permanent member of the Hedera Governing Council.

[2] See Appendix 3 for a full definition of the hashgraph algorithm, including proofs of aBFT. Also see https://www.hedera.com/blog/coq-proof-completed-by-carnegie-mellon-professor-confirms-hashgraph-consensus-algorithm-is-asynchronous-byzantine-fault-tolerant for access to the formalized aBFT proof.

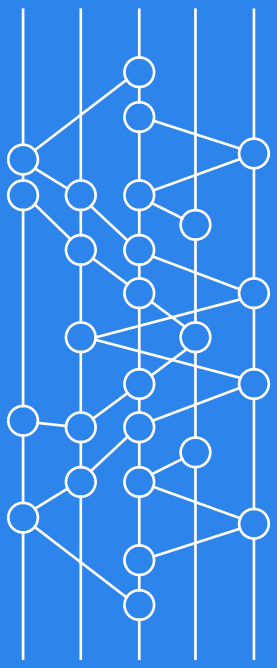
Part 1
An introduction to Hedera Hashgraph

The hashgraph data structure and consensus algorithm provides a new platform for distributed consensus. This introduction gives an overview of how hashgraph works, and some of its properties.

The goal of a distributed consensus algorithm is to allow a community of users to come to an agreement on the order in which some of the users generated **transactions**, when no single member is trusted by everyone. In this way, it is a system for generating trust, when individuals do not already trust each other. Hashgraph achieves this in a fundamentally new way.



BLOCKCHAIN



HASHGRAPH

A blockchain is like a tree that is continuously pruned as it grows – this pruning is necessary to keep the branches of blocks from growing out of control and to ensure the ledger consists of just one chain of blocks. In hashgraph, rather than pruning new growth, such growth is woven back into the body of the ledger.

In both blockchain and hashgraph ledgers, any user can create a transaction, which will eventually be put into a container (the "block") and will then spread throughout the distributed network.

In blockchain, those "blocks" of containers are intended to form a single, long chain. If two blocks are created at the same time, the network nodes will eventually choose one chain to continue and discard the other one, lest the blockchain "fork" into two different chains. It is like a growing tree that is constantly having all but one of its branches chopped off.

In hashgraph, every container of transactions is incorporated into the ledger — none are discarded — so it is more efficient than blockchains. All the branches continue to exist forever, and are woven together into a single whole.

Furthermore, blockchain fails if the new containers arrive too quickly, because new branches sprout faster than they can be pruned. That is why blockchain needs proof-of-work or some other mechanism to artificially slow down the growth. In hashgraph, though, nothing is thrown away. There is no harm in the hashgraph data structure growing quickly. Every member can create transactions and containers whenever they want.

Finally, because the hashgraph does not require pruning of would-be "forks" (as every container of transactions is incorporated into the ledger), hashgraph allows more powerful mathematical guarantees, such as Byzantine agreement and fairness. Distributed databases such as Paxos are Byzantine, but do not guarantee fairness in the ordering of transactions. Blockchain is neither Byzantine nor fair. The hashgraph algorithm accomplishes being ***fair, fast, Byzantine, ACID compliant, efficient, inexpensive, timestamped, and DoS resistant.***

Performance

COST

A hashgraph distributed ledger is **inexpensive** to operate compared to blockchain distributed ledgers, as it avoids energy-intensive proof-of-work. Individuals and organizations who want to run hashgraph nodes will not need to purchase expensive custom mining rigs. Instead, they will be able to run hashgraph nodes via readily available hardware that is less expensive than such specialized mining rigs.

EFFICIENCY

The hashgraph is **100% efficient**, as that term is used in the blockchain community. In blockchain, work is sometimes wasted mining a block that later is considered stale and is discarded by the network of nodes.

In hashgraph, the equivalent of a "block" of transactions never becomes stale. Hashgraph is also efficient in its use of bandwidth. Whatever the amount of bandwidth required to inform all the nodes of a given transaction (even without achieving consensus on a timestamp for that transaction), hashgraph adds only a very small overhead of additional bandwidth to achieve a consensus timestamp and put the transactions into order. Additionally, the hashgraph voting algorithm does not require any additional messages be sent in order for nodes to vote on validating transactions (or those votes to be counted) beyond those messages by which the network nodes learned of the transaction itself.

THROUGHPUT

The hashgraph is ***fast***. It is limited only by the bandwidth. If each network node has enough bandwidth to download and upload a given number of transactions per second, the network as a whole can handle close to that many transactions per second. Even a fast home internet connection could enable a hashgraph node to be fast enough to handle transaction volume equal to that of the entire global VISA card network.

THE FOLLOWING CHARTS GIVE PERFORMANCE RESULTS FOR HASHGRAPH TECHNOLOGY UNDER TEST CONDITIONS.

The charts on the previous pages show the results of tests performed using hashgraph nodes hosted by Amazon AWS m4.4xlarge instances. Figure 1 shows performance for a single region (Virginia), Figure 2 shows results for two regions 2,000 miles apart on opposite sides of the continental United States (Virginia and Oregon), and Figure 3 shows results for 8 regions (Virginia, Oregon, Canada, Sao Paulo, Australia, Seoul, Tokyo, and Frankfurt).

Each line on the chart is for a different number of instances (computers), which is shown to the right of the line. In every case, the instances were distributed evenly across the number of regions being used.

The horizontal axis is the number of 100-byte transactions per second for which the ledger achieved consensus. In these experiments, this throughput ranges from less than 50,000 tps up to almost 500,000 tps. On most of the lines, the second dot from the left is 10,000 tps.

The vertical axis is the average number of seconds from when a node first creates a transaction until it knows the exact consensus order and consensus timestamp for it. This isn't just a time to a first confirmation — it is the time until a 100% certain finality is reached.

In all of the experiments, this latency was under 11 seconds. Various experiments had latencies down to less than 0.04 seconds.

In the graphs, there are clear tradeoffs between throughput, latency, number of computers, and geographic distribution. For 32 computers running at 50,000 transactions per second, consensus finality is reached in 3 seconds when the network is spread across 8 regions spanning the globe. When the network stretches only 2,000 miles across the U.S., this drops to 1.5 seconds. In a single region, it drops to 0.75 seconds.

If it is desired to keep the latency under the 7 seconds required by credit cards, while still achieving 200,000 transactions per second, it is possible to use 32 computers in eight regions, or use 64 computers in two regions, or use 128 computers in one region.

It is important to note that these tests are purely for achieving consensus on transaction order and timestamps. They do not include the time to process transactions. For example, if every transaction is digitally signed, then these results suggest that a great deal of processing power might be needed to verify hundreds of thousands of digital signatures per second. It is possible that GPU implementations could be helpful.

In addition, if a transaction is a request to store a large file, then bandwidth limitations would slow down the network's ability to process that transaction. Network pricing for file storage transactions will be structured to make it relatively expensive to store large files on the ledger, which should disincentivize any efforts to slow down the network in this way.

Figure 1
Hashgraph Latency vs Throughput
1 region, m4.4xlarge

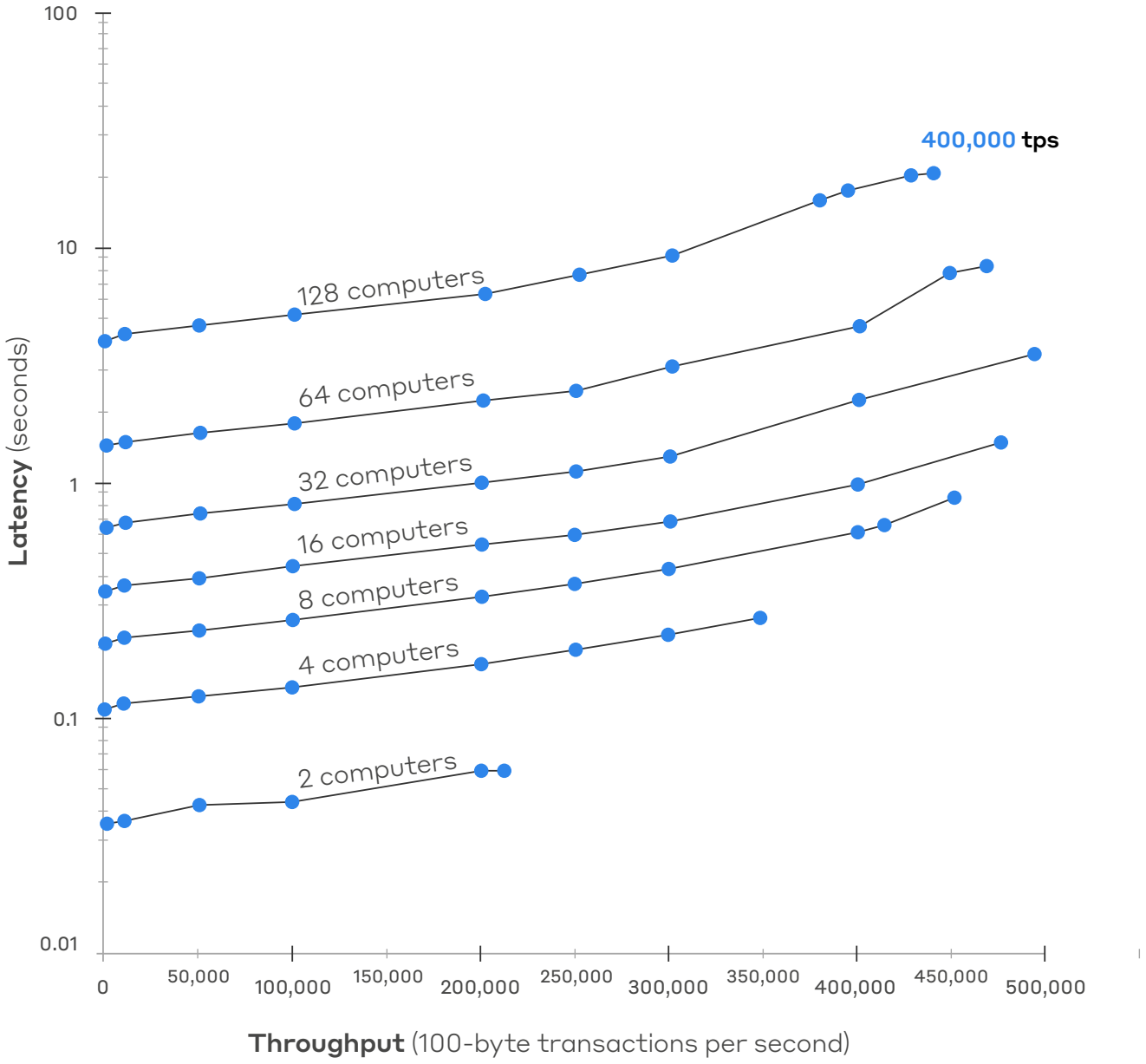


Figure 2
Hashgraph Latency vs Throughput
2 regions, m4.4xlarge

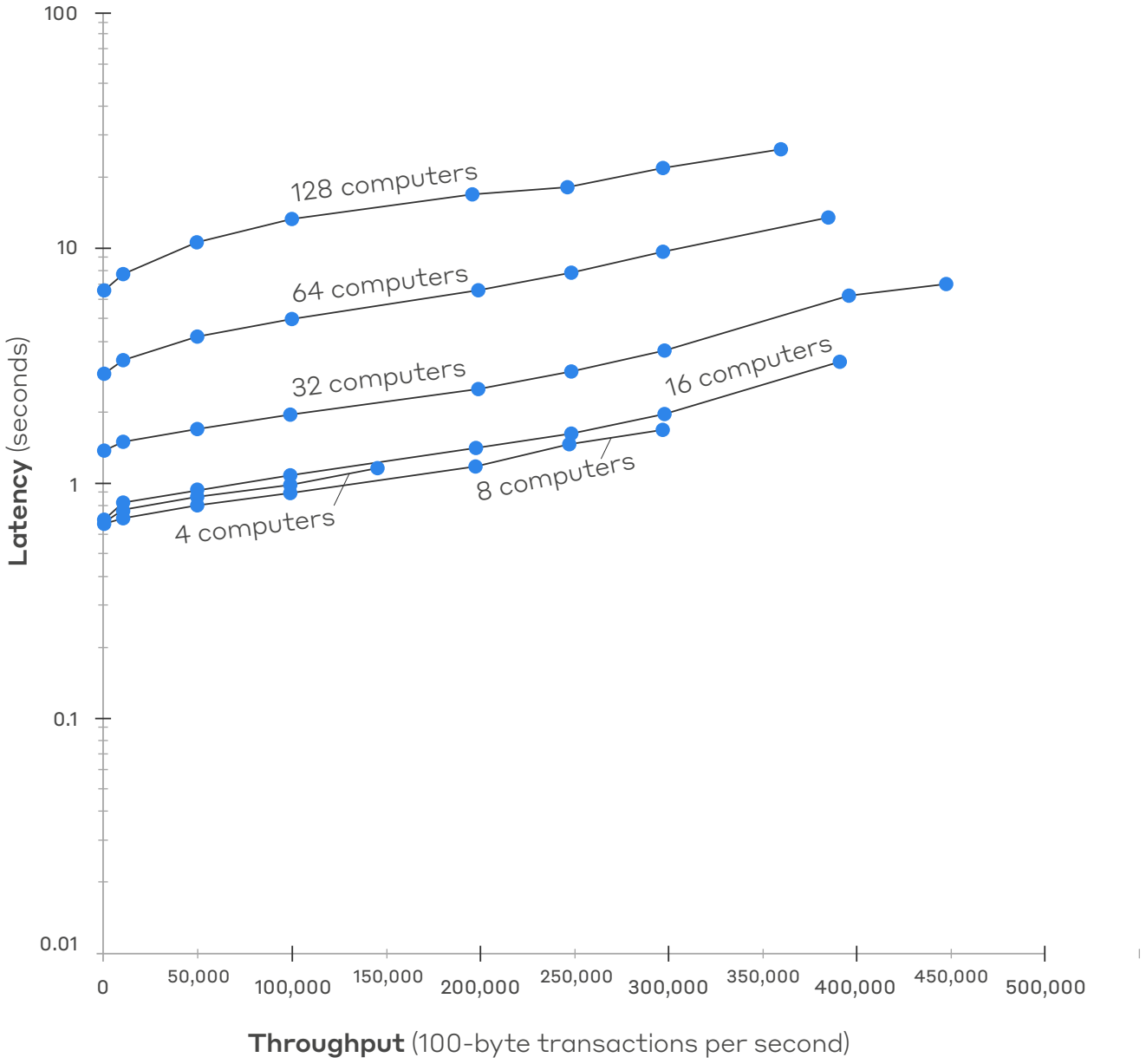
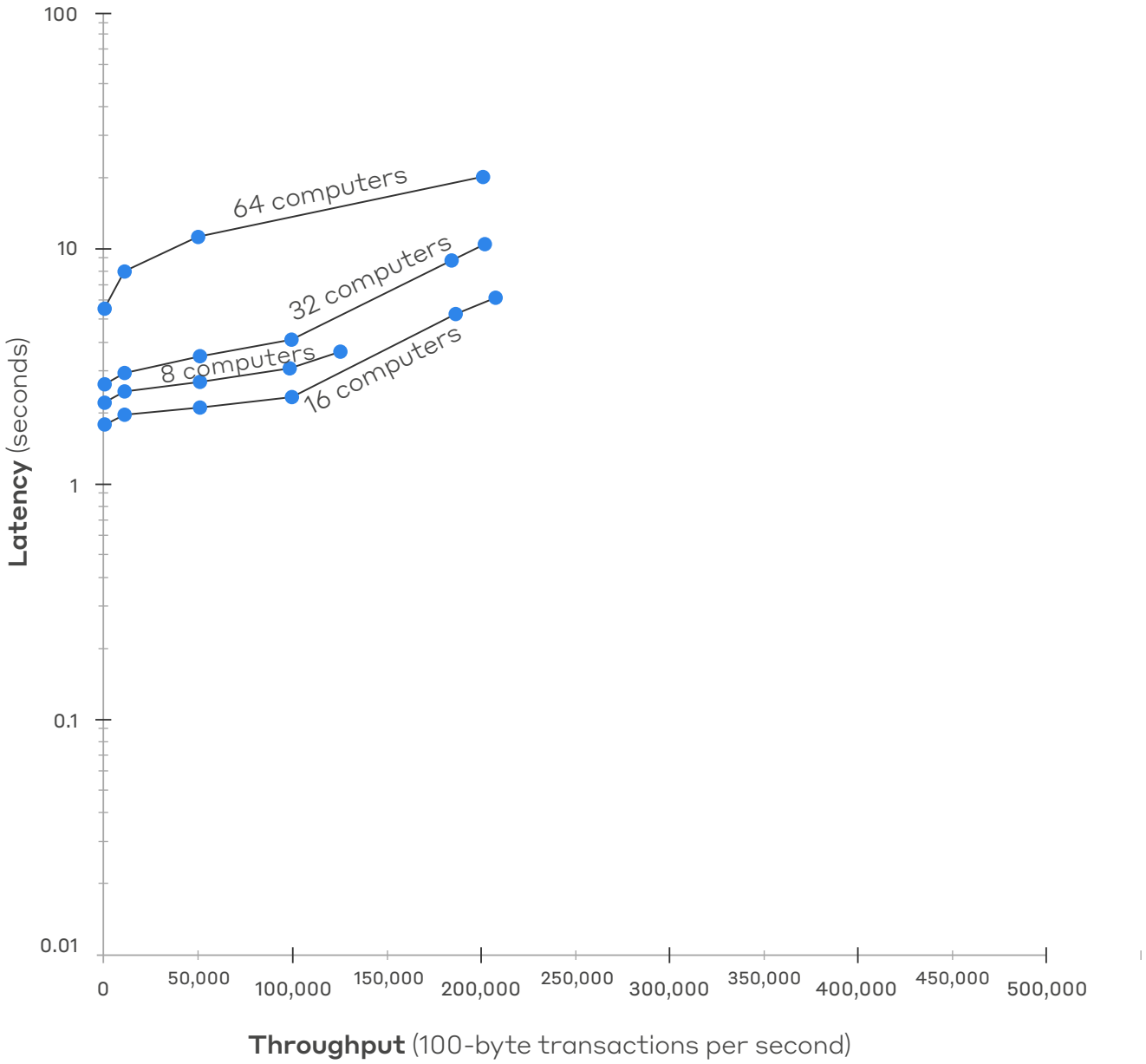


Figure 3
Hashgraph Latency vs Throughput
8 regions, m4.4xlarge



STATE EFFICIENCY

Once a network transaction occurs, within **seconds** all nodes in the network will know where that transaction should be placed in a history of transactions with **100% certainty**. More importantly, every node will know that every other node knows this. At that point, the network can just incorporate the effects of the transaction and, unless needed for future audit or compliance, discard the transaction data. So, in a minimal cryptocurrency system, each node would only need to store the current balance of each network account that is not empty. The nodes would not need to remember the full history of the transactions that resulted in those balances all the way back to “genesis.”

Security

CRYPTOGRAPHY

All Hedera network communications are encrypted with TLS 1.2, all transactions are digitally signed, and the hashgraph is constructed using cryptographic hashes. All the algorithms and key sizes were chosen to be compliant with the CNSA Suite security standard. This is the standard required for protecting U.S. government Top Secret information. It specifies using AES-256, RSA 3072, SHA-384, and ECDSA and ECDH with p-384, along with ephemeral keys for perfect forward secrecy.

ASYNCHRONOUS BYZANTINE FAULT TOLERANCE

The hashgraph algorithm is asynchronous Byzantine Fault Tolerant (aBFT). This is a technical term meaning that no single node (or small group of nodes) can prevent the network from reaching consensus, nor can they change the consensus once it has been reached. Each Hedera network node will eventually reach a point where it "knows" for sure that the network has reached consensus. Blockchain platforms do not have a guarantee of Byzantine agreement, because a node never reaches certainty that agreement has been achieved. Rather there is just a probability that increases over time. Blockchain is also non-Byzantine because it does not automatically deal with network partitions — i.e., if a group of miners is isolated from the rest of the internet, multiple chains ("forks") could grow that conflict with each other on the order of transactions.

It is worth noting that the term "Byzantine Fault Tolerant" (BFT) is sometimes used in a weaker sense of resistance to malicious behavior by other consensus algorithms. We use it in its original, stronger sense in that even if we assume (1) some attackers will collude to stop or skew consensus and (2) that some attackers could even control the internet itself (with some limits) and slow or prevent the delivery of messages, the network will eventually reach consensus and every node eventually knows consensus has been reached. Hashgraph is Byzantine, even by this stronger definition. So long as attackers have less than 1/3 of the total stake of hbars, they will be unable to stop consensus or even skew transaction timestamps or consensus order.

There are different degrees of BFT, depending on the assumptions made about the network and transmission of messages.

The strongest form of BFT is asynchronous BFT — meaning that the network can achieve consensus even if malicious actors are able to control the network and delete or slow down messages of their choosing. The only assumptions made are that more than 2/3 are following the protocol correctly, and that if messages are repeatedly sent from one node to another over the internet, eventually one will get through, and then eventually another will, and so on. Some systems are partially asynchronous, which are secure only if the attackers do not have too much power and do not manipulate the timing of messages too much. For instance, a partially asynchronous system could prove Byzantine under the assumption that messages get passed over the internet in ten seconds. However, this assumption ignores the reality of botnets, Distributed Denial of Service attacks, and malicious firewalls.

A full technical report describing the hashgraph data structure and algorithm, including mathematical proofs that Hashgraph is asynchronous BFT, is included in the Appendix.

ACID COMPLIANCE

The hashgraph is **ACID compliant**. ACID (Atomicity, Consistency, Isolation, Durability) is a database term, and applies to the hashgraph when it is used as a distributed database. A network of nodes use it to reach a consensus on the order in which transactions occurred. After reaching consensus, each node feeds those transactions to that node's local copy of the database, sending each one in the consensus order. If the local database has all the standard properties of a database (ACID), then the network as a whole can be said to have a single, distributed database with those same properties. In blockchain, there is never a moment when you know that consensus has been reached, so it would not be ACID compliant

DISTRIBUTED DENIAL OF SERVICE ATTACK RESILIENCE

One form of Denial of Service (DoS) attack occurs when an attacker is able to flood an honest node on a network with meaningless messages, preventing that node from performing other (valid) duties and roles. A Distributed Denial of Service (DDoS) uses public services or devices to unwittingly amplify that DoS attack — making them an even greater threat.

In a DLT network, a DDoS attack could target the nodes that contribute to the definition of consensus and, potentially, prevent that consensus from being established.

The hashgraph is DDoS resilient as it empowers no single node or small number of nodes with special rights or responsibilities in establishing consensus. Both Bitcoin and hashgraph are distributed in a way that resists DDoS attacks. An attacker might flood one node or miner with packets of data to temporarily disconnect it from the internet. But the network as a whole will continue to operate normally. An attack on the system as a whole would require flooding a large fraction of the nodes with packets, which is more difficult. There have been a number of proposed alternatives to blockchain based on "leaders" or "round robin" models. These have been proposed to avoid the proof-of-work costs of Bitcoin, but they have the drawback of being sensitive to DDoS attacks. If the attacker attacks the current leader node, and switches to attacking the new leader as soon as one is chosen, then the attacker can freeze the entire system while still attacking only one node at a time. Hashgraph avoids this problem, while still avoiding the energy requirements of proof-of-work.

.

Fairness

Hashgraph is fair because there is no leader node or miner given special permissions for determining the consensus timestamp assigned to a transaction. Instead, the consensus timestamps for transactions are calculated via an automated voting process in the algorithm through which the nodes collectively and democratically establish the consensus. We can distinguish between three aspects of fairness.

FAIR ACCESS

Hashgraph is fundamentally fair because no individual node can stop a transaction from entering the system, or even delay it very much. If one or a few malicious nodes attempt to prevent a given transaction from being delivered to the rest of the network and so be added into consensus, the random nature of the hashgraph gossip protocol through which nodes communicate messages to each other will ensure that the transaction flows around that blockage.

FAIR TIMESTAMPS

Hashgraph gives each transaction a consensus timestamp that is based on when the majority of the network nodes received that transaction. This consensus timestamp is fair, because it is not possible for a malicious node to corrupt it and make it differ by very much from that time. Every transaction is assigned a consensus time, which is the median of the times at which each node says it first received it. Received here refers to the time that a given node was first passed the transaction from another node through gossip. This is part of the consensus, and so has all the guarantees of being Byzantine. If more than two-thirds of participating nodes are honest and have reliable clocks on their computer, then the timestamp itself will be honest and reliable, because it is generated by an honest and reliable node or falls between two times that were generated by honest and reliable nodes. Because hashgraph takes the median of all these times, the consensus timestamp is robust. Even if a few of the clocks are a bit off, or even if a few of the nodes maliciously give times that are far off, the consensus timestamp is not significantly impacted.

This consensus timestamping is useful for things such as a legal obligation to perform some action by a particular time. There will be a consensus on whether an event happened by a deadline, and the timestamp is resistant to manipulation by an attacker. In blockchain, each block contains a timestamp, but it reflects only a single clock: the one on the computer of the miner who mined that block.

FAIR TRANSACTION ORDER

Transactions are put into order according to their timestamps. Because the timestamps assigned to individual transactions are fair, so is the resulting order. This is critically important for some use cases. For example, imagine a stock market, where Alice and Bob both try to buy the last available share of a stock at the same moment for the same price. In blockchain, a miner might put both of those transactions in a single block and have complete freedom to choose in what order they occur. Or the miner might choose to only include Alice's transaction, and delay Bob's to a future block. In hashgraph, there is no way for an individual node to unduly affect the consensus order of those transactions. The best Alice can do is to invest in a better internet connection so that her transaction reaches everyone before Bob's. That's the fair way to compete.

Governance

A governance model for a public ledger will define the rules and policies that control the evolution of the node software, issuance of coins, and the reward model that incentivizes network participants. There are multiple stakeholders whose interests and motivations must be balanced: network node operators, developers building applications on the platform, businesses relying on those applications, end-users of those applications, and relevant regulatory bodies.

The Hedera Governing Council is a limited liability company that will have up to 39 members, all to be well-known enterprises from diverse industries and geographies. Hedera's licensing and governance model protects network users by eliminating the risk of forks, guaranteeing the integrity of the codebase, and providing open access to review the underlying software code. Under the governance model, all Governing Members will have equal voting rights and each Governing Member (with the exception of Swirlds) will serve a limited term, ensuring that no single Governing Member or group of Governing Members has centralized control.

The Hedera network has a model of permissioned governance with a phased plan for permissionless or open consensus.

Governance is permissioned in that governance decisions are made by the members of the Hedera Governing Council. The Council establishes policy for Council membership, sets the network rules, manages the platform's treasury of coins, and approves changes to the platform codebase. Our governance model is based on the original model used by National BankAmericard Inc., founded in 1968, which was later renamed VISA. We are designing our governance model in a way that ensures the Governing Members can be trusted to do what is in the best interest of the Hedera platform, and not be unduly influenced by individual Council members or node operators. In addition to Governing Members, Hedera expects to have participating organizations contribute to the Hedera network ecosystem by providing advisory services as appropriate, but such organizations will not have voting privileges.

The open consensus model relates to the process by which the nodes join the network and reach a consensus on the order of transactions in the platform. The model is designed to prevent consolidation of power over consensus by encouraging the emergence of a decentralized network with, eventually, millions of nodes. It prevents collusion by a few to attack the system such as by counterfeiting the cryptocurrency, modifying the ledger inappropriately, or influencing the consensus order of transactions. We inhibit collusion by weighting the votes within the hashgraph algorithm of a particular node based on the node's stake of coins. Loosely stated, each node casts one vote for each coin of the Hedera native currency (hbars) it owns. Initially, all of the nodes in the Hedera network will be operated by Council members, so consensus will start on a permissioned basis. As network use grows, the Council will allow new node operators to join the network and be paid for their services in maintaining the hashgraph. The number of nodes is expected to grow large over time, ensuring consensus voting privileges are distributed to many nodes. A full discussion of the staking model is included in the section below.

This system of permissioned governance with open consensus will build more public trust than a purely closed system. This is essential to the success of a global DLT platform.

PERMISSIONED GOVERNANCE

We designed the Hedera governance model to ensure that the Council can be trusted to govern the network fairly. Council members will have equal governing rights and limited terms, ensuring that governance is decentralized. Deliberation and debate will be open to all Council members and controlled by none.

The Governing Members will also participate in committees that provide oversight of Hedera operations. Committees will include but are not limited to a Technical Steering & Product Committee, a Finance Committee, and a Legal & Regulatory Committee. The Governing Members are organizations that span a broad range of business sectors, and our objective is that, collectively, the membership will contribute industry-leading representation to the range of Hedera committees.

Stability

The hard forks that Bitcoin and Ethereum have experienced have arguably damaged the network effect of their corresponding currencies, creating confusion and uncertainty in the marketplace. Similarly, the explosion of altcoins (and the dubious legitimacy and value of many of them) does not engender the necessary confidence in businesses and consumers considering adopting cryptocurrencies.

Historically, open source software developers have recognized the value of maintaining a single baseline of code and ensuring that the best ideas from the community are included for the benefit of the whole. However, when combining an open source project with a cryptocurrency, the traditional incentive structure is turned upside down. The distributed ledger technologies that have been most widely adopted are also those that have split the most. This dynamic causes chaos in the industry, and directly impedes the adoption of public ledgers by mainstream markets.

Hedera technical and legal controls ensure the platform will not fork into a competing platform and cryptocurrency.

Technical controls

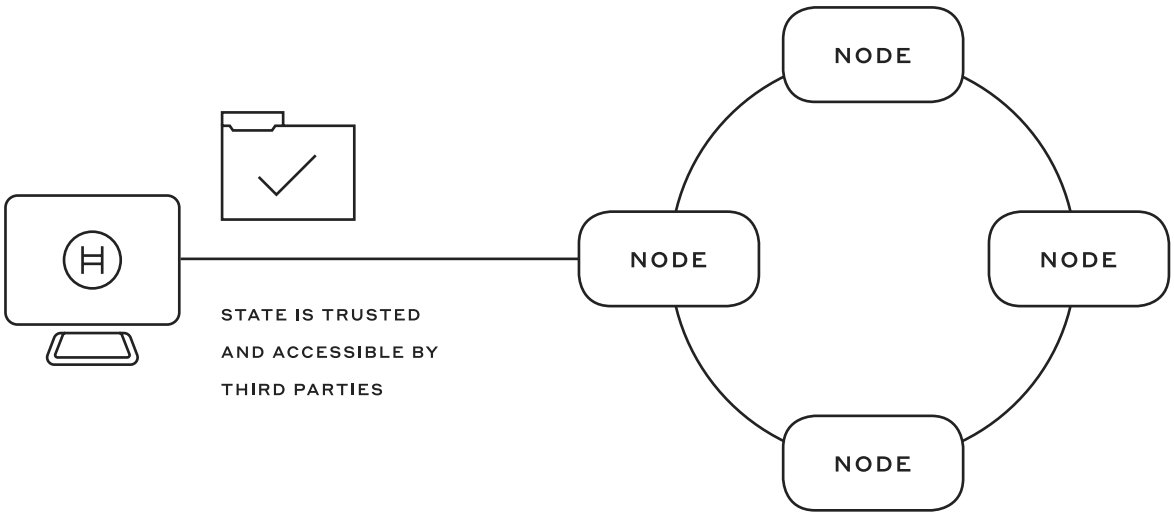
SIGNED STATE PROOFS

All nodes maintain a copy of the state. For instance, each node knows the balances of all network participants' crypto accounts. At the end of each round of the algorithm, each node calculates the new state by processing all transactions that were received in that round and before. Each node then digitally signs a hash of that shared state, puts it in a transaction, and gossips it out to the network. Then it collects those signatures from all other nodes.

When a client requests some aspect of the state, all nodes will be able to construct and return a small file carrying the collected signatures and other cryptographic material to prove to the client (or another party) that the returned data is indeed the true, consensus state.

The state is organized as a Merkle tree, so a third party can be given a proof that consists of a small part of the state, plus the path from there to the root of the Merkle tree (including siblings of those vertices in the tree), plus the signatures, and an address book history for the public keys.

The diagram below represents how a third party will be confident that the state it receives from one of the nodes does indeed represent the consensus state of the full network.



LEDGER ID

The proof will also include an "address book", which is a list of the public keys of all the nodes, along with each node's stake. A third party will need this address book in order to check the signatures on the state (or portion of the state).

The proof must also include an "address book history." This is a sequence of address books, where each address book is signed by nodes from the previous address book. Any given address book must be signed by a set of nodes that control more than 2/3 of the stake of the platform's coins, according to the network's list of nodes and their stake from the previous address book. This chain of address books extends back to the genesis address book, which was signed by the initial nodes that created the ledger.

The hash of the genesis address book is important. It serves as a unique identifier of the ledger. It is the "name" of the ledger.

HANDLING FORKS

If a small number of nodes wants to split off from the network and create a new ledger that is a fork of the current one, these nodes have the technical ability to do so, and can even create the initial state of their new ledger to be identical to the old ledger. So it is a fork. However, they will not be able to create an address book history reaching back to the genesis address book, with the nodes of each address book signing the next one, because the majority of nodes (who are not forking) will not sign the address book for the minority of nodes who are forking. This forces the new fork to have a new genesis address book, and therefore a new unique identifier, and therefore a new name. Consequently, those creating the fork will be unable to fool anybody into thinking the fork is the legitimate ledger.

When a client submits a transaction to a node to send to the ledger, the client will receive in response from the node the cryptographic proof that their transaction has affected the shared state correctly. When Alice transfers cryptocurrency to Bob, both of them will be able to receive a cryptographic proof that the transaction succeeded. This proof includes the signatures reaching back to the genesis address book. So they not only verify that the transfer occurred, they verify that it occurred on the correct ledger. If a ledger forks, no client will ever be confused about which ledger they are dealing with because only one ledger at a time can have that name.

Furthermore, if the network's stake of hbars were to split 50/50 between two groups of nodes, neither group of nodes would be able to prove a connection to the genesis address book. Rather than a fork, it would be the complete deconstruction of one ledger and the creation of two new ones. This would greatly reduce the value of the ledger to the nodes, because they would no longer be able to earn fees from the clients who want to access the original ledger. And all of the original cryptocurrency would, in a very real sense, cease to exist. This creates an enormous disincentive to forking.

In this way, it is impossible to effectively create a deceptive fork of the Hedera ledger that would confuse users. Even if dishonest nodes create a forked copy of the Hedera ledger in order to try to deceive users into thinking the copy is the legitimate ledger, users would know that ledger isn't valid because those nodes hosting the illegitimate fork would not be able to provide a valid state proof. And there is little incentive for nodes to openly create a forked copy, because there is unlikely to be much demand from users to shift to using an invalid version. So there are strong incentives to avoid forks, even aside from any legal incentives.

The cryptographic proofs and unique identifiers that prevent forks are also critically important for secure sharding. They allow shards to send each other messages, with assurance that the message from a given shard was truly the consensus of that shard.

Legal Controls and Transparency

The Hedera codebase will be governed by the Hedera Governing Council. Version 1.0 of the codebase, which is expected in 2020, will be "open review," meaning that anyone will be able to read the source code, recompile it, and verify that it is correct.

No license will be required to use the Hedera platform. No license will be required to write software that uses the services of the Hedera platform. No license will be required to build smart contracts on top of the Hedera platform. Applications built upon the Hedera platform can be open source or proprietary. They do not require any license or any approval from Hedera. Software developed using the platform APIs will not be encumbered in any way. Software developers will have complete ownership and discretion on the licensing they choose for their applications that use the Hedera platform.

Swirlds owns the intellectual property rights in the hashgraph consensus algorithm. The Hedera Governing Council has a license from Swirlds to use the hashgraph consensus algorithm and associated technology for the Hedera distributed public ledger platform. In exchange for that license, the Hedera Governing Council will pay Swirlds 10% of network revenue (with monthly minimums) and Swirlds will own 5% of Hedera coins. Swirlds will continue to require licenses for use of the hashgraph technology in private, permissioned networks, but no license will be required for distributed applications that run on Hedera's public platform. Hedera and Swirlds will use the patent rights associated with the hashgraph algorithm defensively to legally prohibit the forking of the codebase and the creation of a competing platform and currency. Developers are free to build distributed applications on top of the Hedera platform with associated native tokens.

In summary, Hedera will simultaneously embrace open review of the software code, while bringing stability to the platform and cryptocurrency by controlling the license. In this way, Hedera will provide a transparent codebase, assuring the stability that markets demand for mainstream adoption.

Regulatory Compliance

We expect that governments will continue to extend policy objectives to users, enterprises, and developers utilizing public ledgers and associated cryptocurrencies and tokens. We consider it a fundamental goal that the Hedera network can provide necessary tools for all members of its ecosystem to comply with applicable laws and regulations, including existing regimes such as the European Union's General Data Privacy Regulations (GDPR) and anti-money laundering (AML) obligations. We will continue to work with regulators to enable compliance with new and changing laws and regulations as well. The sections below outline some of the key elements of the Hedera network that allow for a path to regulatory compliance.

SELF-CUSTODY

Just like with most other distributed ledgers, all transactions that affect the state of a Hedera account must be signed by the account's private key. This enables end users who retain possession of their private key to have sole control over their accounts and never relinquish custody of any funds to Hedera, developers, or enterprises. Even cryptocurrency transactions are completely peer-to-peer without intermediaries taking possession of hbars. Developers can either design and build wallets and applications on the Hedera network that utilize self-custody of private keys and funds or choose to host the private keys of their users and comply with the regulatory obligations arising from such custodial relationship.

DATA SELF-SOVREIGNITY

Unlike most other distributed ledgers, the Hedera network implements controlled mutability. When data is published on the Hedera network the publisher can define an authorization policy for future deletion and/or modification by specifying which keys have those authorizations. This allows developers to build applications that can comply with the "right to erasure" requirement under the GDPR or allow its users to maintain full control of which data is public and for how long.

PRIVACY MANAGEMENT

The Hedera network also allows users to manage their own identities and transaction requirements. By default, accounts are pseudonymous, like most other distributed ledgers. However, the Hedera technology framework can support future implementations that allow users to attach their real identity (as asserted by an accredited party acting as a Certificate Authority) to be logically bound to their ledger account so that, when using that account to move funds, appropriate KYC checks can be performed A counterparty could choose to require full proof of identity (including full legal name) or specific identity characteristics (such as age) consistent with their compliance program in order to accept a transaction on the network. User remain in control of their personal information and whether to provide proof of identity or characteristics to a counterparty; the counterparty remains in control of meeting compliance obligations before engaging in a transaction.

The system is opt-in, in that a user must explicitly choose to avail themselves of the mechanism and, if they do not, their account transactions will remain pseudonymous. This choice, however, may prevent them from engaging in certain financial transactions.

The system is designed to provide the appropriate balance of:

1. Government visibility
2. Security
3. User privacy

Comparable to showing a driver's license when creating a new bank account, the model has a user attach a hash of a digital certificate created by a recognized identity provider to their account. This attachment will take the form of a transaction sent out to the network.

This transaction:

1. May need to be signed by both the user's private key and that of the identity provider
2. Can stipulate what parties are authorized to subsequently detach the hash (and so revoke the binding between the identity and the account).

As long as the attachment between account and certificate has not been revoked, by either the user or the identity provider, it can be used to establish that the account is bound to a known user whenever funds move in or out of that account. If and when appropriate, the identity provider can revoke the binding simply by sending a signed transaction to the network.

As an example of how it might work, consider a user trying to send money from their Hedera account to a US bank. The user would provide to the bank the certificate as well as their account address. The bank would look up the account and confirm that the account had the corresponding hash for the certificate, and that the certificate was issued by a trusted identity provider. Only if all these checks were confirmed would the bank authorize the transaction and accept the funds. The bank might be asked by the corresponding government to send the certificate and transaction details, either in real-time (perhaps based on the amount of the transfer) or on a schedule.

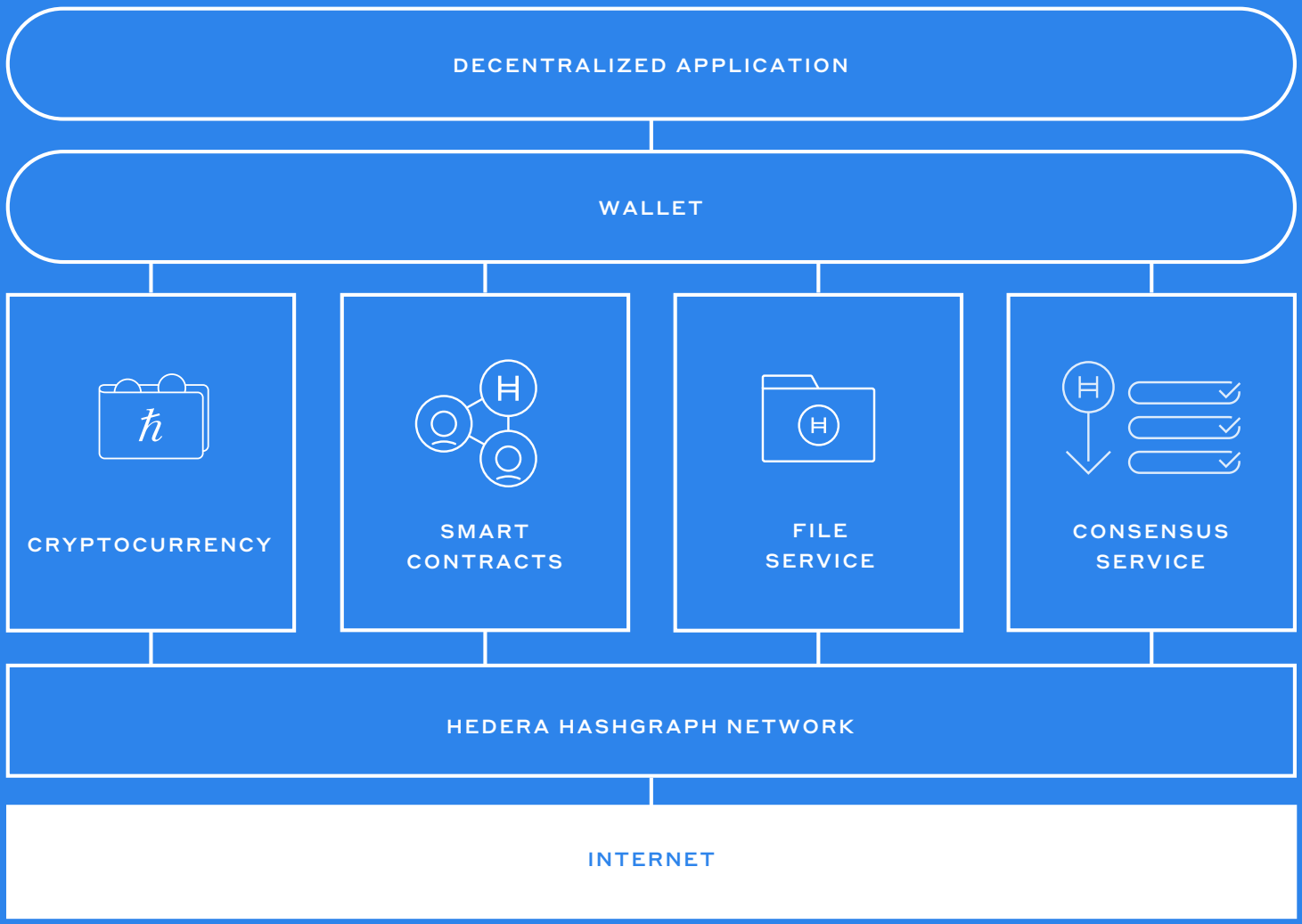
ANTI-MONEY LAUNDERING

Some developers and enterprises utilizing the Hedera network may have anti-money laundering reporting obligations. For such entities, on-network identity certificates and the use of network data from mirror nodes could form the basis of a compliance program. Mirror nodes, as discussed in more depth below, will eventually be able to run by anyone and serve as "read-only" observers of network activity. With such mirror nodes, all public data on the network can be viewed, stored, and analyzed to conduct investigations and flag suspicious behavior.

Hedera is a founding member of the Distributed Ledger Foundation and will work with the broader DLT community and governments to ensure that regulatory requirements can be satisfied, while maintaining privacy and security.

Hedera directly resolves the five fundamental obstacles to mainstream market adoption of public ledger technology: Performance, Security, Stability, Governance, and Regulatory Compliance. The hashgraph data structure and consensus algorithm provide a best-in-class, unmatched combination of performance and security. The Hedera platform and Hedera Governing Council will provide transparency, open innovation, platform stability, tools to enable opt-in KYC and AML, and global, cross-industry expertise to provide governance and decision making for a globally distributed network and cryptocurrency.

Part 2
Architecture


DECENTRALIZED APPLICATION
WALLET
CRYPTOCURRENCY
SMART CONTRACTS
FILE SERVICE
CONSENSUS SERVICE
HEDERA HASHGRAPH NETWORK
INTERNET

INTERNET LAYER

The Hedera network nodes are all computers on the internet, communicating by TCP/IP connections protected by TLS encryption with ephemeral keys for perfect forward secrecy. Nodes are addressed by IP address and port, rather than by symbolic names, so attacks on the DNS system will not affect the network.

HASHGRAPH CONSENSUS LAYER

The nodes take transactions from clients and share them throughout the network with a gossip protocol. Then all nodes run the hashgraph consensus algorithm to reach agreement on a consensus timestamp for each transaction and its consensus order in history. Each node then applies the effects of the transactions in consensus order to modify its copy of the shared state. In this way, all nodes maintain an identical consensus state (within any given shard).

SERVICES LAYER

CRYPTOCURRENCY

The cryptocurrency is designed to be fast, which leads to low network fees, making very small microtransactions practical. When the Hedera platform is running at scale, any user will be able to run a node in the network and earn cryptocurrency payments for doing so. Any user will be able to create an account by simply creating a key pair, without any name or address attached to it. Optionally, provisions are made to allow a user to attach hashes of identity certificates. These could come from any third-party certificate authority or identity authority that the user chooses. This is intended to allow regulatory compliance, for cryptocurrency accounts that will be used in a jurisdiction with Know Your Customer (KYC) or Anti-Money Laundering (AML) laws. More detail is given in the Regulatory Compliance section.

FILE STORAGE

The file system allows users to store information, with consensus on exactly what is stored and what is not stored. Every node in the shard stores the same files, so they will not be lost if one of the nodes crashes. Stored information can only be deleted by those that were given permission. In this way, the file system can act as a revocation service. For example, in the future, a user might be issued a driver's license from the Department of Motor Vehicles (DMV), and both the user and the DMV digitally sign the transaction that puts a hash of it into the ledger. Both have the right to remove the hash of the license. The user can choose to prove to someone that they have a valid license, by giving that person a copy of the license file, so the person can check whether the hash is still stored in the ledger. If the DMV revokes the license, it would also delete the hash, to show the world that the license is no longer valid. If the user tries to store the hash again, without a signature from the DMV, it will be evident that the hash was stored only by the user without DMV cooperation, and would not be considered valid evidence of the user's right to drive.

Files are actually stored as Merkle trees, but we provide Java classes to allow developers to manipulate them.

We give developers Java code to manipulate a Merkle tree as if it were a file system. They see directories, subdirectories and files. And they change file contents and directory names, move things around, and copy and paste. Yet, underneath, it's all being stored as a Merkle tree automatically. This allows us to give proofs that a file is part of the consensus state. Users also can store an entire directory in the Hedera file system.

We not only store Merkle trees, we store Merkle DAGs, which means that if two files have some bytes in common, we might only store one copy of the common bytes.

A file can be accessed by its hash, so people can rely on the fact that it is immutable. But it also has a File ID. Its owner can create a new file, and make the File ID to be associated with the new file instead of the old one. In this way, it is possible for users to always find the latest version of a file. They just access the File ID instead of the hash. So files are both securely immutable and securely non-immutable, at the same time. If a file is accessed by its hash, then it never changes. If it is accessed by its File ID, then the latest version is found.

SMART CONTRACTS

The Hedera ledger can run smart contracts written in Solidity. Currently, large libraries of Solidity smart contract code exist, and they can be run unchanged on Hedera. These allow for distributed applications to be easily built on top of Hedera.

CONSENSUS

The Hedera Consensus Service will offer an efficient alternative to smart contracts and the file system for building distributed applications. Clients submit messages to Hedera for time-stamping and ordering within specific topics. These ordered messages will flow out to mirror nodes or clients of mirror nodes for processing in the consensus order. The consensus service will give distributed applications direct access to the native speed, security, and fair ordering guarantees of the hashgraph consensus algorithm, with the full trust of the Hedera ledger. Full details on the Hedera platform's consensus service, which will not be available initially but added in a subsequent phase of the platform's development, are available in Appendix 2.

Mirror network

The Hedera mirror network is a set of nodes that will maintain all of the same requirements and most of the functionality of the main Hedera network The primary difference in functionality is that mirror nodes do not have the ability to submit transactions to be incorporated into the network's ledger. Mirror nodes will gossip, and will calculate consensus and verify signatures, but because they are unable to create events, mirror nodes have no effect on the hashgraph structure. Therefore they have no ability to submit transactions for consensus and no voting power. Mirror nodes can be thought of as "read-only" nodes in that transactions cannot be submitted to a mirror node via the Hedera API. Mirror nodes are free to develop additional APIs for providing new kinds of services that they create.

The mirror network will provide an efficient way to get the state of the ledger out to many more users and dApps in a short period of time without having a major impact on the performance of the main network. As a result, dApps may choose to host their own mirror nodes to listen to transactions on the main network and respond accordingly. For example, a dApp that deploys a smart contract may choose to host a mirror node, listen for events from its smart contract, filter out other transactions, and respond accordingly.

Sharding

Initially, the Hedera network will be a small number of nodes operated by Governing Council Members, all in a single shard. As the Hedera Governing Council grows and others begin to operate nodes, the network will gain a sufficient number of nodes to justify multiple shards. Sharding can offer performance advantages as every node need not process every transaction. Consensus can consequently proceed in parallel. Shards will trust each other, so one shard will honor requests to move cryptocurrency or to put a hold on various resources made by another shard - as long as those requests can be proven to reflect the consensus of the requesting shard. This will allow the multi-shard ledger as a whole to achieve asynchronous Byzantine fault tolerance, and to prevent double spends or other conflicting states of the ledger, because each individual shard will have those properties, and because all messages between them will contain proofs that they are the consensus of that shard.

The plan is that nodes will be randomly grouped into different shards, within which consensus on transactions will be established as normal. Each shard will be made up of a subset of the nodes, all of which share the same state, which will be a subset of the state of the entire ledger. Transactions will be placed into consensus order within individual shards in the normal manner — all nodes within a shard will contribute only to the consensus for transactions that originate in that shard. The assignment of nodes to shards will be performed randomly by a master shard, which will assign new nodes to a shard once a day, and also will move nodes between shards as necessary to ensure that each shard has a large total amount of hbars being staked, and that no one node within a shard will have a large fraction of that total amount.

Shards will communicate through the exchange of messages between nodes of the different shards.

All such messages are push (rather than pull). Each shard (its nodes) will maintain a queue of outgoing messages to each of the other shards. Each shard will remember the sequence number of the last message it processed from each of the other shards. A message will be sent for transfer from shard Alpha to shard Beta by nodes in Alpha randomly contacting nodes in Beta, along with a proof that it is part of the consensus state of the Alpha shard. They will continue doing this until one of the Beta nodes replies with a proof that the Beta shard shared state includes a sequence number indicating that this message was received and processed. In this manner, transactions that impact addresses in different shards will be appropriately recorded into each shard's state, and so into the entire state of the entire ledger.

More details are given in the Sharding appendix.

Part 3
Cryptoeconomics

Staking and proxy staking

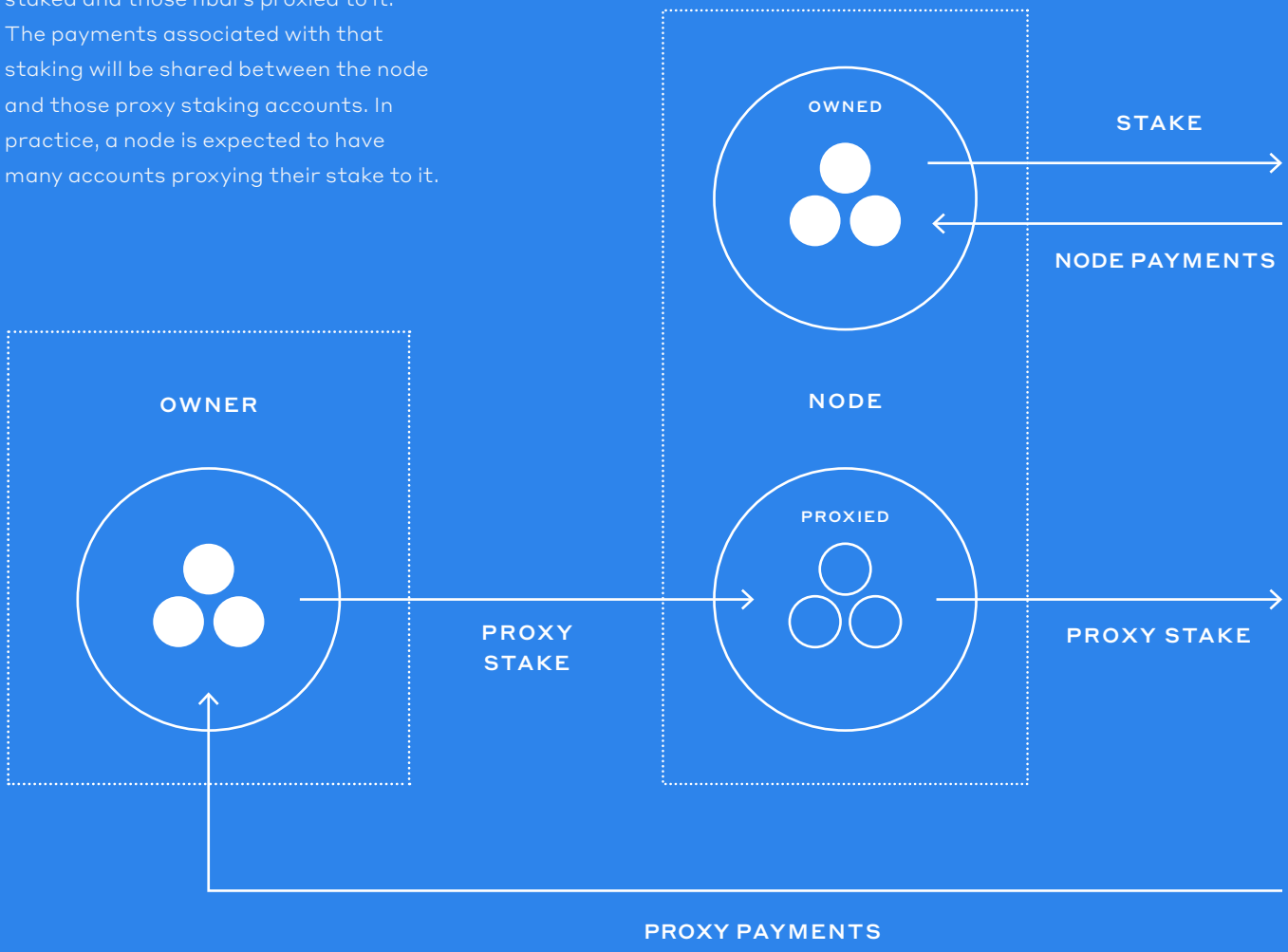
The Hedera ledger uses a proof-of-stake consensus mechanism, in which each node's influence on consensus is proportional to the amount of cryptocurrency it has staked to it. A transaction is validated and placed into consensus after it is validated by nodes representing an aggregate stake of over two-thirds of the network's total number of hbars (the number of hbars is fixed at 50 billion). It is important to ensure that most of the cryptocurrency is actually being staked, so that the network continues to run. In the initial phase of the network, the Hedera Treasury will "proxy-stake" over two-thirds of the total number of hbars to nodes hosted by Council Members.

After hbars become more widely distributed (so that no single user or group of users can attain control of one-third of all hbars), the network will allow anyone to host a node (i.e., it will be a permissionless network). At that time, when a node joins the network, it must declare one or more accounts that it can control, and prove that it has the private keys for those accounts. From then on, the amount of hbars in those accounts will be used to weight its votes in the hashgraph virtual voting algorithm. Additionally, it will be paid to serve as a node, with that payment proportional to the amount of hbars in those accounts. It is still free to spend those hbars at any time. Consequently, a potential disincentive of bonded proof-of-stake models — that of nodes unwilling to stake for fear of the associated loss of liquidity — is avoided.

In addition, a mechanism called "proxy staking" will allow a person who owns hbars but does not run a node to nevertheless stake those hbars and earn a small amount of hbars for contributing to network operation by "proxy staking" their account to a node. That means giving another account credit for their hbars and allowing the node to use that stake when it contributes to consensus. The payments for running the node (proportional to the amount staked) are then split between the node and the owner of the hbars being proxy staked. The hbars that are being proxy staked still remain under the control of their owner. The owner will be able to turn off or redirect the proxy staking to another node at any time. They will also be able to spend the proxy staked hbars at any time, though again that will reduce the amount they receive in payment for staking. Note that the node to which the hbars are proxy staked cannot spend those hbars. This proxy-staking feature for network users will not be available initially, and will be added in a subsequent phase of the platform's development.

A node must have at least some hbars in its account for it to be able to influence consensus or to pay fees associated with sending transactions to the ledger.

The proxy staking model is shown below. A node's stake towards consensus will reflect both the hbars it owns and has staked and those hbars proxied to it. The payments associated with that staking will be shared between the node and those proxy staking accounts. In practice, a node is expected to have many accounts proxying their stake to it.



Proxy staking will allow those who do not run nodes to earn a small amount of hbars by contributing the weight of their hbars to a node's vote. By encouraging this practice, the network makes it more difficult for a bad actor to gain influence over a third of the entire stake. And those who do run nodes will be able to increase their revenue.

Payments and fees

Users pay fees to use the Hedera platform for activities like transferring cryptocurrency or adding items to the ledger. Because the Hedera network has high throughput and does not require proof-of-work, we anticipate the fees to be a small fraction of other public DLT platforms in the market today.

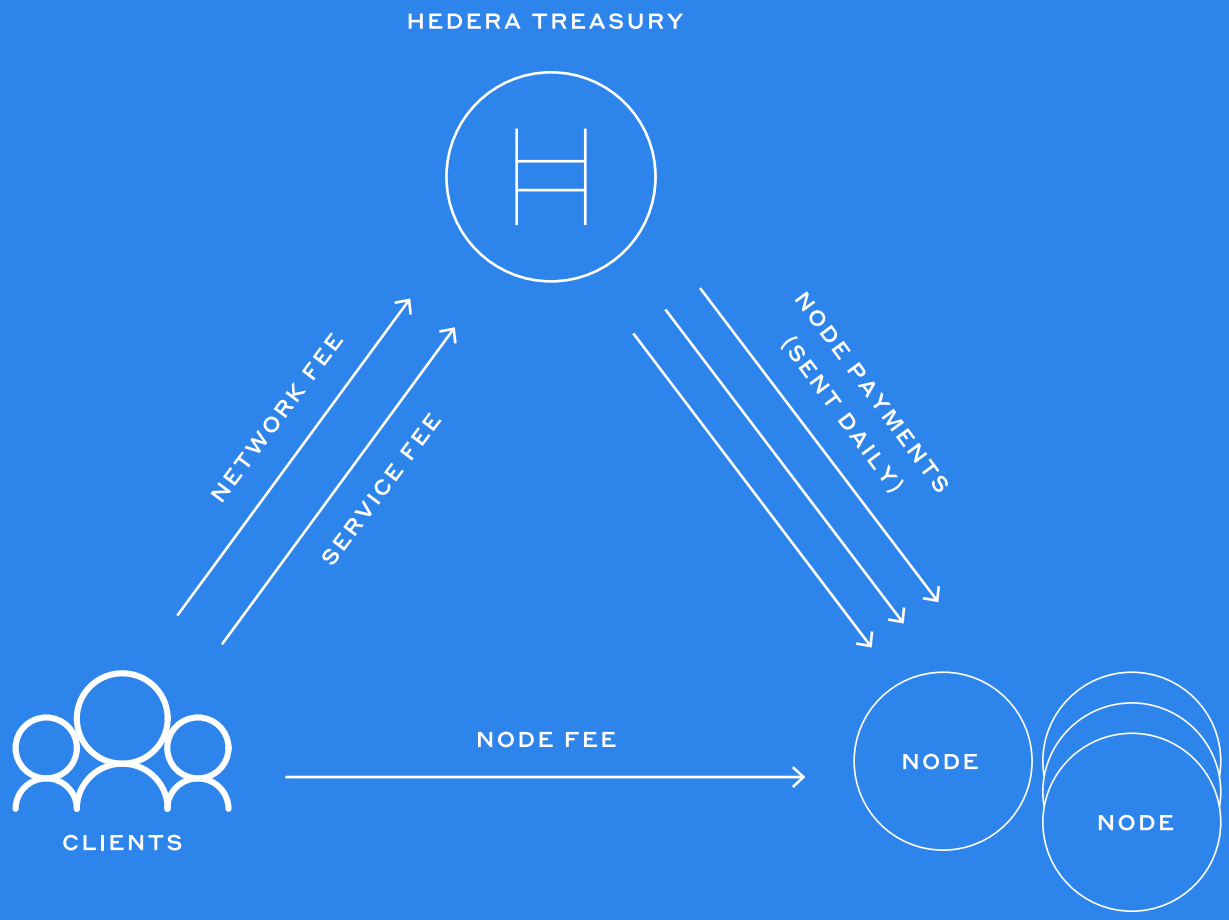
Nodes in the Hedera ledger are compensated for the computing, bandwidth, and storage resources they use in establishing consensus and providing services. There are several types of payments and fees:

1. **NODE FEE** - A user or application seeking to complete an action on the platform will send the corresponding transaction to a single node, which will then submit that transaction to the network. In doing so, the node will expend a small amount of resources and energy. Node Fees compensate nodes for those resources and incentivize nodes to take on this critical role. Initially, the Hedera Governing Council will set the amount of the Node Fees, but Node Fee amounts will eventually be left to each node to determine. Node fees are paid by end users directly to the account of the node that submits the user's transaction to the network.

2. **NETWORK FEE** - After a transaction is submitted to the network, it is communicated to nodes that validate any digital signatures. The transaction is then further communicated to other nodes, which temporarily store it in their memory while the network reaches consensus. Users pay a Network Fee that compensates all participating nodes for this activity of calculating consensus on the transaction. The resources required to validate a transaction can vary based on the particulars of the transaction, but generally depend on the transaction's file size and number of digital signatures. Network Fees are paid by users into a Hedera Treasury account and a portion of such collected amounts are subsequently distributed daily to participating nodes as Node Reward Payments.

3. **SERVICE FEE** - Service Fees compensate nodes for the ongoing burden of maintaining or supporting the transaction. As an example, for a file storage transaction, all nodes will store the file on their hard drives for a specified period of time and the Service Fee for that transaction will reflect the size of the file and duration of time it's stored. For a transaction that requests a smart contract function be executed, the Service Fee will be based on the processing power required by network nodes to perform that computation and any associated storage burden, e.g., storing the results of the contract execution. Service Fees are paid by users into a Hedera Treasury account, and a portion of such collected amounts are subsequently distributed daily to participating nodes as Node Reward Payments.

The three different fee types are shown in the diagram below, indicating from which account they are taken, and to which they are added. Clients pay nodes fees directly to the node that they requested to process their transaction. Clients pay network fees to the same node, but those fees will be passed on to the Hedera Treasury. Clients pay service fees directly to Hedera. All fee payments are made through the network coming to consensus on a gossiped transaction that authorized the payment.



Hedera collects services and transaction fees on behalf of all the nodes processing the transactions and performing the services. Hedera uses those collected fees to fund incentive payments to nodes:

NODE REWARD PAYMENT – Once a day, payments are made from the Hedera account to nodes, to incentivize them to serve as nodes. To be paid, a node must have been online for the full day, according to thresholds defined by the Hedera Governing Council (e.g., requiring that the node contribute at least one event each to at least 90% of the rounds during that 24 hour period). A node will be paid proportional to the amount of cryptocurrency it is staking (both owned by itself and proxy staked to it by others).

The fee model is designed to allocate costs and risks appropriately.

The biggest resource costs are paid for by service fees, and those resource costs (e.g., storing a large file) are never incurred until proper payment has been made by the client. As an example, a client must pay for the storage of a file for 30 days in advance at the time of the transaction that creates the file on the network.

The smaller resource costs of gossiping and reaching consensus on the transactions themselves are paid for by the network fees. The node that submits a transaction will first verify that the account paying the fees has sufficient funds, but there is a small risk that the balance will drop below this level in the short period of time it takes for the transaction to be processed Into consensus. In this case, the network will have spent resources but will not be paid for that effort.

So at each level of the system, costs are paid for, and economic incentives are aligned.

When a client contacts a node for help submitting a transaction to the network to perform some service for the client, the client stipulates on the transaction a transaction fee parameter – this is the maximum amount the client is willing to pay for the requested service.

The node performs a precheck by analyzing the transaction to determine the amounts of the various network resources (bandwidth, CPU, storage, etc.) the associated service will require and multiplies those amounts by coefficients in a published fee schedule to determine the total transaction fee. The node compares this calculated transaction fee to the client's stipulated maximum transaction fee as well as to the client's balance to determine if the transaction should be submitted to the network.

Once the transaction is submitted, all nodes process it into consensus order, calculate the transaction fee, determine if the client account still has a sufficient balance and, if so, applies the transaction to the consensus state, and finally process fees as described above.

The fees incentivize the node to submit a transaction correctly as it will not be paid if it fails to do so.

The service fee is only paid if the service is performed so the risk to the client is minimal. And because the service is only performed if the payment occurs, there is not a risk to the Hedera network.

The Roadmap to Scale

The following are the phases by which we expect the network will grow from concentrated nodes and stake to widespread nodes and stake. These phases will not be distinct but represent the expected evolution of the network and its currency.

PHASE 1 The Hedera Treasury holds most of the coins, and the Treasury will begin to "proxy stake" coins to the nodes managed by the Hedera Council Members. Some coins from the Hedera Treasury will also be distributed to the general population for use on the network. Individuals that use the Hedera-provided wallet software may also proxy their coins to the Council Member-managed nodes.

PHASE 2 In addition to Hedera Governing Council Members, additional trusted parties will become able to stand up nodes. The Hedera Treasury and Hedera-provided wallet software used by hbar holders (by default) will proxy stake coins to both the Council Member nodes and these additional trusted node operators. Over time the distribution of coins will become more widely spread out across a greater number of nodes. More coins continue to be distributed from the Hedera Treasury to the general population of network users.

PHASE 3 Individuals who are interested may go through a Know-Your-Customer process and then also stand up nodes and receive staking of coins from the Hedera Treasury and the default Hedera wallet software. More coins will continue to be distributed out of the Hedera Treasury to the general population. Anonymous individuals can begin to run nodes and receive proxy-staked coins. The Hedera Treasury and Hedera wallet software will not proxy coins to those anonymous nodes, but those nodes may be able to receive proxied coins from 3rd-party wallet software.

PHASE 4 As the coins are distributed widely and competing wallet software programs arise, there will be a market for proxying coins that is independent of the Hedera Governing Council. Eventually all of the coins are widely distributed, there is a market of wallet software, and a market of nodes competing for proxy staking.

In this way, all of the coins start in the Hedera Treasury account, and the initial Hedera wallet software defaults to proxying just to members of the Hedera Governing Council. Over time, however, both the coins and the proxying gain wider and wider distribution until they are distributed across millions of nodes and accounts.

Acknowledgements

We gratefully acknowledge the contributions and help from our advisors, Natalie Furman, Tom Trowbridge, Edgar Seah, Jordan Fried, Christian Hasker, Arlan Harris, Paul Bugeja, Alex Godwin, Ken Anderson, Patrick Harding, Zenobia Godschalk, George Samman, Sam Brylski, Tom Sylvester, and Rachel Epstein.

Appendices

Appendix 1: Team

DR. LEEMON BAIRD, CO-FOUNDER

Leemon is the inventor of the hashgraph distributed consensus algorithm, and is the Co-Founder and Chief Scientist of Hedera. With over 20 years of technology and startup experience, he has held positions as a Professor of Computer Science at the US Air Force Academy and as a senior scientist in several labs. He has been the Co-Founder of several startups, including two identity-related startups, both of which were acquired. Leemon received his PhD in Computer Science from Carnegie Mellon University and has multiple patents and publications in peer-reviewed journals and conferences in computer security, machine learning, and mathematics.

MANCE HARMON, CO-FOUNDER

Mance is an experienced technology executive and entrepreneur with more than 20 years of strategic leadership experience in multi-national corporations, government agencies, and high-tech startups, and is Co-Founder and CEO of Hedera. His prior experience includes serving as the Head of Architecture and Labs at Ping Identity, Founder and CEO of two tech startups, the senior executive for product security at a $1.7B revenue organization, Program Manager for a very-large scale software program for the Missile Defense Agency, the Course Director for Cybersecurity at the US Air Force Academy, and research scientist in Machine Learning at Wright Laboratory. Mance received a MS in Computer Science from the University of Massachusetts and a BS in Computer Science from Mississippi State University.

Note that as part of the Hedera Governing Council's decisions to enable broader market participation on the Hedera network, Mance Harmon and Dr. Leemon Baird have left their roles as CEO and Chief Scientist of Hedera Hashgraph, LLC, respectively, and have assumed new roles as co-CEOs of Swirlds Labs, effective as of May 1, 2022.

Appendix 2: Sharding

Initially, the network will consist of a relatively small number of nodes in a single shard. As the network grows, it will gain sufficient nodes to support multiple shards. Those shards will work in the following way.

A transaction is always submitted to a specific shard. Within a shard, every node receives all of that shard's transactions, and every node maintains an identical shared state. Each shard can store both cryptocurrency accounts and files. Every shard can run smart contracts.

A shard uses the hashgraph consensus algorithm to reach a consensus order for its transactions. Each shard must be able to trust the consensus decision of each of the other shards. Therefore, each shard must be composed of randomly chosen nodes, and must be large enough so that it can be trusted to never have 1/3 of its staked cryptocurrency being owned by malicious nodes.

If a transaction involves only resources within a given shard, then when that point in the consensus order is reached, the transaction performs its effect. For example, a transaction might move cryptocurrency between two accounts within the same shard. Or it might save a file within that shard and pay for it with an account in that shard. In those cases, the cryptocurrency transfers or the file is stored immediately, at the point where the transaction occurs in the consensus order.

If a transaction involves resources in different shards, then it will trigger inter-shard messages. For example, if the cryptocurrency account Alice is in shard Alpha, and account Bob is in shard Beta, then Alice creates and signs a transaction to move cryptocurrency from Alice to Bob. She submits that transaction to a node in the Alpha shard, and all of the nodes in Alpha reach consensus on its order. At the point where this event occurs in the consensus order, Alice's account balance is decreased by the amount being sent, and a message to the Beta shard is generated. Each shard maintains a queue of outgoing messages to be sent to each of the other shards. So this new message is added to the queue that Alpha maintains for messages to send to Beta. Each message in a given queue has a 64-bit sequence number, which starts at zero when the network is first created and then increments with each new message sent.

Each member of Alpha will, at random intervals, check to see if there are any messages in any outgoing queues and attempt to send one of the queues. When they see that there are messages intended for Beta, they will call a random member of Beta and give them all messages in the queue, along with the proof that this queue is part of the current signed state for Alpha.

When a member of Beta receives such a list of messages from the member of Alpha, the Beta member submits a transaction to Beta that has the messages and the proof that they are part of the signed state. If they see that a message has already been submitted, then they won't submit it again, although sometimes the same message may be submitted twice at the same time. In that case, the sequence numbers will match, so the duplicate will be ignored, and no harm is done.

All messages between two particular shards will be processed in order of sequence number. If Alpha sends a message to Beta, and it is added to a transaction within Beta, the sequence will be checked when the transaction reaches consensus. If it is the next message in sequence, then its effect is performed immediately. If the sequence number shows that one or more other messages have been skipped, then it has no effect and is ignored. In that case, the other messages will eventually reach consensus, and then the skipped message will be submitted again, and will have an effect.

When Beta processes a message with Alpha with the expected next sequence number from Alpha, then it increments the count of the number of messages from Alpha that have been processed. So each shard maintains a single number for each of the other shards, which is the latest sequence number from that other shard that has been processed.

After Alpha has sent to the message to Beta, that message remains in the outgoing queue, and attempts will repeatedly be made to send it. Eventually, a member of Apha will contact a member of Beta to send that message, but will receive back a proof that the message has already been processed. That proof shows that the signed state contains Beta's sequence count for messages received from Alpha, and that the count is now higher than the message in the queue. At that point, the member of Alpha wraps the proof in a transaction and gossips it out to Alpha. When it reaches consensus order, at that point the message is deleted from the outgoing queue in the shared state.

For the example of transferring cryptocurrency from Alice to Bob, we could say that there is "finality" when we know that the transfer is valid, that Alice had sufficient funds, and that Bob will certainly receive the funds. If this transfer is for Alice to buy a product from Bob, then finality is a point in time where it is safe for Bob to give the product to Alice. The time to finality is actually as short as the consensus time for a single shard. Once consensus has been reached on that initial transaction, it is certain that Alpha will send a message to Beta, and that Beta will process it, and that Bob's account will receive the transfer. So finality is as fast as consensus.

If a transfer is from one source account to two destination accounts, finality is still just as fast. As soon as the initial transaction reaches consensus, it will be known whether it had sufficient funds and whether the two messages will be sent.

However, if a single transaction is to transfer from two source accounts to one destination account, and the source accounts are in different shards, then finality will be slower. Because it will have to involve another type of message: a "hold," which is later followed by a "release."

For example, suppose a transaction is created to transfer 2 coins from Alice in Alpha shard and 3 coins from Bob in Beta shard, with the 5 coins being transferred to Gina in Gamma shard. This is intended to be atomic so that nothing will happen unless Alice and Bob both have sufficient funds for the transfer.

To achieve this, the transaction must be signed by both Alice and Bob, and must be submitted to the Alpha shard. When it reaches consensus, it causes a "hold" to be put on 2 coins in Alice's account. This means that 2 coin's worth of the account is temporarily frozen. While it is frozen, Alice is still free to receive funds and to transfer out funds, but can't do any transfer that would decrease her balance to less than 2 coins.

At the same time that Alpha puts a hold on 2 coins for Alice, it also sends a message to Beta requesting a hold of 3 coins for Bob. This message does not need to be signed by Bob, because it is coming from Alpha, and Alpha has already checked that Bob had signed the transaction.

When the message is received and reaches consensus, Beta will attempt to put a hold on Bob's account for 3 coins. If he has sufficient funds, it succeeds. If he has less than 3 coins, then it fails, and no hold is put on him at all. Beta then sends back to Alpha a message saying whether the hold was successful.

When Alpha receives a reply that the hold was successful, Alpha then decrements Alice's account by 2 coins (which also removes the hold), sends a message to Beta saying to decrement Bob's account by 3 coins (removing his hold) and sends a message to Gamma to increment Gina's account by 5 coins.

On the other hand, if Beta's message said the hold failed because Bob did not have sufficient coins, then Alpha simply released the hold on Alice's account and considers the entire transaction to have failed. None of the three balances change.

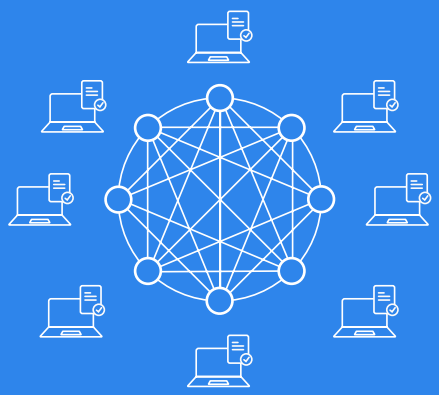
Note that when this all started with Alpha processing the initial transaction, it was possible for Alpha to calculate how many messages would be involved in the entire process, 4 messages in this example. Alpha will therefore check that the transaction included authorization of a service fee that included the fee for the service of sending those 4 messages. Hedera then automatically makes payments to the nodes that created each of the message transactions that were handled (and not ignored as duplicates). This acts as incentive for nodes to do the work of sending messages to other shards, receiving confirmations of receipt from them, and creating the transactions that contain those messages and confirmations.

Appendix 3: Consensus Service

Abstract

The Hedera Consensus Service synchronizes the fair order of messages for distributed systems without relying on a centralized clock.

The Consensus Service proof-of-concept use case is providing custom Hyperledger Fabric networks with decentralized consensus on the validity and order of blockchain transactions without the need to configure a RAFT[1] or Kafka[2] ordering service. Additional use cases include, but are not limited to, financial markets, matching engines (like those used in Uber or AirBnb), or supply chain negotiations (e.g., several competing factories bidding on parts from several competing parts suppliers).



ON A DISTRIBUTED LEDGER, THE ENTIRE NETWORK RECORDS AND VALIDATES EACH TRANSACTION.

[1] "Configuring and Operating a RAFT Ordering Service," https://hyperledger-fabric.readthedocs.io/en/release-1.4/raft_configuration.html
[2] "Bringing up a Kafka Ordering Service," https://hyperledger-fabric.readthedocs.io/en/release-1.4/kafka.html

Introduction

The Hedera Consensus Service is the first Distributed Ledger Technology (DLT) network service to provide solely the validity and order of events and transparency into the history of events over time without requiring a persistent history of transactions. As a result, the Hedera Consensus Service provides the benefits of a fast, fair, and secure consensus at a lower cost than any other public distributed ledger network.

Whether in financial services, IoT, or supply chain - the timing and order of events dictates everything from financial transactions to meaningful asset provenance. The applications must execute logic based on events which occur at a specific time, in a specific order. In many cases the users of these applications need to look back in time to the history of that order for everything from audit to reconciliation.

Today, these applications rely on moderation, matching, and ordering performed by single entities. This makes them prone to network outage[3], at risk of collusion by a small number of parties[4], and subject to the cost model of centralized infrastructure providers[5]. Even private distributed ledger networks rely on nodes operated by one or few parties to provide consensus to the rest of the network[6]. Each approach poses economic risk due to cost and operational risk due to unintentional outage or intentional manipulation of a service.

In the distributed ledger space, protocols aim to solve this problem through the provision of two features:

1. Decentralized consensus on the validity and order of events.
2. Transparency into the history of events over time.

The first value relies on the existence of nodes to come to agreement on the time an event occurred, ultimately producing a consensus order for events over time. Distributed ledgers such as R3's Corda, Hyperledger Fabric, or enterprise version of Ethereum either deploy known and trusted nodes operated by institutions, trust a single party to provide the order, or rely on slow and expensive public ledgers like Bitcoin or Ethereum to select a block producer through proof of work[7]. Applications can be forced to wait minutes or even hours for confirmation on finality of a transaction. There is a market need for distributed and fast consensus without the need to centralize the consensus process.

[3]Gps Error Caused '12 Hours Of Problems' For Companies
Chris Baraniuk - https://www.bbc.com/news/technology-35491962
[4]Corrupt Governance? What We Know About Recent Eos Scandal
Stephen O'Neal - https://cointelegraph.com/news/corrupt-governance-what-we-know-about-recent-eos-scandal
[5]Cloud Pub/sub | Google Cloud
https://cloud.google.com/pubsub/
[6]The Ordering Service
https://hyperledger-fabric.readthedocs.io/en/release-1.4/orderer/ordering_service.html and Notaries¶
[7]"Recent studies hint that the performance of PoW based blockchains cannot be enhanced without impacting their security" - https://eprint.iacr.org/2016/555.pdf

The second value is the ability of any individual or entity to independently verify whether and when an event occurred. This is most frequently used to track account balances of tokens, or to verify the provenance of an asset. Traditional blockchains rely on storage of all events across all members of the network. This model allows for simple querying of account balances but limits performance to 10-20 tps and means the ledger will continue to grow (bitcoin alone is over 200 GB as of the writing of this paper).[8]

Hedera provides a unique solution to deliver optimized performance of decentralized consensus without the need to persist a history of transactions over time through the provision of the Hedera Consensus Service. The Hedera Consensus Service will use the Hedera public network and underlying hashgraph consensus algorithm for fast, fair, and secure consensus while offloading the validation and storage requirements for distinct applications to computers using the Mirror Network.

[8]Bitcoin Blockchain Size 2010-2019 | Statistic
https://www.statista.com/statistics/647523/worldwide-bitcoin-blockchain-size/

Hedera Governance and the Path to Decentralization

The Hedera public network is built from the ground up to deliver decentralized services at a scale needed for enterprise and consumer applications. This includes full decentralization of network operations, high performance, and guaranteed finality.

Decentralization

The Hedera public network will be governed by a Council of 39 term-limited, multi-industry and multi-geo large enterprises to ensure the stability and growth of the network. They act as the initial node operators before node operation becomes public long term. This ensures that the network itself and services it provides are not prone to collusion or manipulation at the desire of a small group of entities or miners. This acts as a multi-cloud/multi-data center service with inherent disaster recovery and high availability that can be used by any application. The Hedera Consensus Service provides transaction ordering on a decentralized network rather than relying on a single cloud provider or small group of private node operators.

Performance

Throughput continues to bottleneck the majority of decentralized public networks today. Hashgraph enables faster consensus with 100% finality and without the need to elect leaders, trust a small subset of nodes, or in any way compromise the security of the network. The Hedera Consensus Service will extend this benefit for any transaction type submitted by an application. Consensus occurs in a matter of seconds while processing tens to hundreds of thousands of transactions per second. This level of performance is required for any scaled consumer or enterprise application.

Finality

Finally, the order that is created by the mainnet is final and verifiable. Consensus timestamps on the Hedera mainnet are 100% final once they are created due to its Asynchronous Byzantine fault Tolerance (ABFT) nature. This means that the timestamp of a given transaction is final and cannot change. Any client application can query the network for the record (created automatically by the nodes to capture key information about the transaction being added to consensus) and optionally ask for a corresponding state proof (a cryptographically secure and persistent assertion from the network as to the accuracy of the record) to confirm that timestamp. Additionally, Consensus Service transactions will be stored on mirror nodes running additional software. Users can run mirror nodes themselves, query a mirror node for state proofs, or validate records between multiple mirror nodes to verify the complete order without needing to trust a single node operator.

The Fair Order of Transactions

Hashgraph's primary function is to calculate a fair order of transactions in a decentralized environment. One of the major differentiators is the degree to which individuals or small groups are prevented from manipulating the order, ensuring fairness.[9]

The Hedera public ledger uses the hashgraph consensus algorithm and its HBAR cryptocurrency to initially provide three services: Cryptocurrency, Smart Contract, and File Service. Hashgraph uses gossip about gossip and virtual voting in order to bring the network to consensus on the timestamp of any event with efficiency of bandwidth usage without centralizing around any entity or group of entities. Hbars are the network coin, which enable any holder of the coin to pay for utility provided by the network, and also ensures security of the network through the process of staking (tying influence within virtual voting to the amount of coin held).

Gossip between the Hedera nodes takes place at the same speed regardless of which node submitted the transaction and cannot be increased by paying more for a given transaction. This differs from other public network models, which allow applications to pay more for their transactions to be processed first. Similarly, because there is no concept of leaders in the consensus, no small subset of nodes can collude to unduly influence the consensus order in their own favor.

Transactions are propagated to the network and come to final consensus in a matter of seconds. If an application is worried about a single node holding back from sending the transaction to the rest of the network then they can submit to multiple nodes. In this scenario only the first transaction to reach consensus would be kept and the others would be ignored.

Hedera's Other Services

Hedera built three initial services to expose the value of decentralization to any application builder:

1. Cryptocurrency: access a low cost and fast method of transferring value between accounts without relying on intermediaries. The Cryptocurrency Service can be used for payment applications, data purchases, and many other use cases relying on fast value transfer.

2. Smart Contracts: execute code deterministically without needing to trust an application operator. Build fair markets, issue tokens, and program business logic in Solidity and deploy it on Hedera to benefit from trusted security and fair ordering.

3. File Service: store data across the network for any node or user to access or store obfuscated data to benefit from a consensus timestamp of the state of data at a point in time. Create decentralized registries, records, and other public data on the File Service.

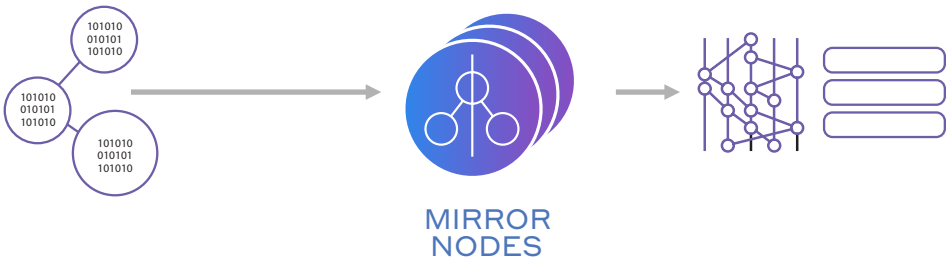
[9] https://www.hedera.com/whitepaper

These services enable large enterprises, independent developers, and consumers to build or use applications across industries and geographies. They are consumable through SDKs in common programming languages and are intended to support applications of any type.

Extending to the Consensus Service

> **Hedera extends these network services with a fourth service, which brings the fairness provided by the hashgraph consensus algorithm to the Hedera Consensus Service. This service receives messages, and assigns them consensus timestamps and a consensus order.**

The Consensus Service relies on another feature to propagate the full history of transactions and its results to many participants: the mirror network. Messages ordered by the Consensus Service will be received by the mirror nodes. Developers can choose to implement additional software on each mirror node. A mirror node could store all the messages for certain topics (identifiers used to associate a message with a specific application or network). It could store all messages (strings of bytes representing a given transaction). Or it could even store the records of all transactions that reached consensus on the ledger. If everything is stored, it can form something that resembles a traditional blockchain, even though the Hedera ledger never keeps that history.



The Hedera mirror network (mirrornet) is a parallel network dedicated to propagating the state of the Hedera main network (mainnet). This propagation is accomplished without adding unnecessary strain to the mainnet. And while anyone will be able to host a mainnet node in the future, mirror nodes allow businesses to extend the functionality of Hedera without a serious impact on the mainnet.

The mirrornet is a set of nodes that maintain all of the same requirements and most of the functionality of the mainnet. The primary difference in functionality is that mirror nodes do not participate in consensus. They receive information from the mainnet, but do not send information to it. Mirror nodes continue to gossip with other mirror nodes, and will calculate consensus and verify signatures, but they have no effect on the mainnet. Therefore they have no ability to submit transactions for consensus and no voting power. Mirror nodes can be thought of as read-only nodes in that transactions cannot be submitted to a mirror node via the Hedera API. Mirror nodes operators

are free to develop additional APIs for providing new kinds of services that they develop. The beta version of the mirror nodes was completed in May 2019 with greater latency (e.g., a minute), but the full mirror nodes will have a latency of seconds.

Mirror nodes operated by individuals or private networks leveraging the Consensus Service will be able to filter and receive events and records for transactions that have a specified topic. They will then be able to store the full history of events relevant to their application.

Architecture

This section will describe the architecture of the Consensus Service on the Hedera public network along with an overview of how the Consensus Service can enable interoperability with and between any Hyperledger Fabric based network.

Hedera Consensus Service Architecture

The Hedera Consensus Service is the fourth core service provided by Hedera. Like the other services, the Consensus Service will be exposed via a diverse number of SDKs in common programming languages, as well as the Hedera API (HAPI) using protobufs. This allows applications to access the network services using both the SDK abstractions as well as the lower level APIs.

The client application would submit a message (a string of bytes) and give it a topic (an ID number). The message would include the relevant details of a transaction such as a bid on a financial asset, or even just the hash of data stored elsewhere. The topic will allow messages with the same topic to be classified together. The client application would pay a transaction fee, denominated in hbars, for the use of the Consensus Service.

The Hedera public ledger will return a record which says that consensus has been reached, the timestamp it was reached and the sequence number of the event for the given topic. The sequence number will allow the application to interpret the order of the message relative to the other messages with the same topic. The result will also include a running hash of all the messages so far for that topic. A running hash is a few bytes that act as a fingerprint of all the messages so far for that topic.

Topics are created by executing a transaction defined by the HAPI, which allows the topic to be created, the keys of the owner to be specified, the keys of who is allowed to post to or delete the topic to be specified, and which will return the ID number of the topic.

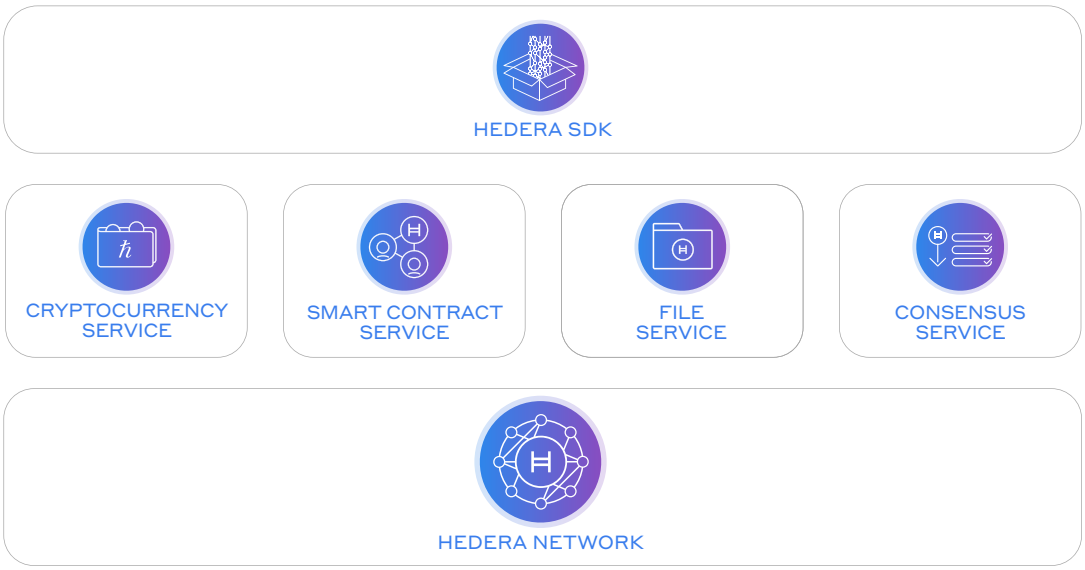


Figure 1A: Public Network

In practice we expect the Consensus Service to be used by a group of mirror node operators and users who are leveraging an application that handles private or proprietary data, but benefits from the fast ordering, decentralized trust, and immutable record of a public ordering service.

To set up the network, the organizations would configure one or multiple mirror nodes, program client applications on them, and configure one or multiple keys that allow those who have the keys to see what the group is doing. The group would also define a topic that they can use to identify transactions relevant to their group. This topic will be attached to messages that the client application will send to the Hedera public network.

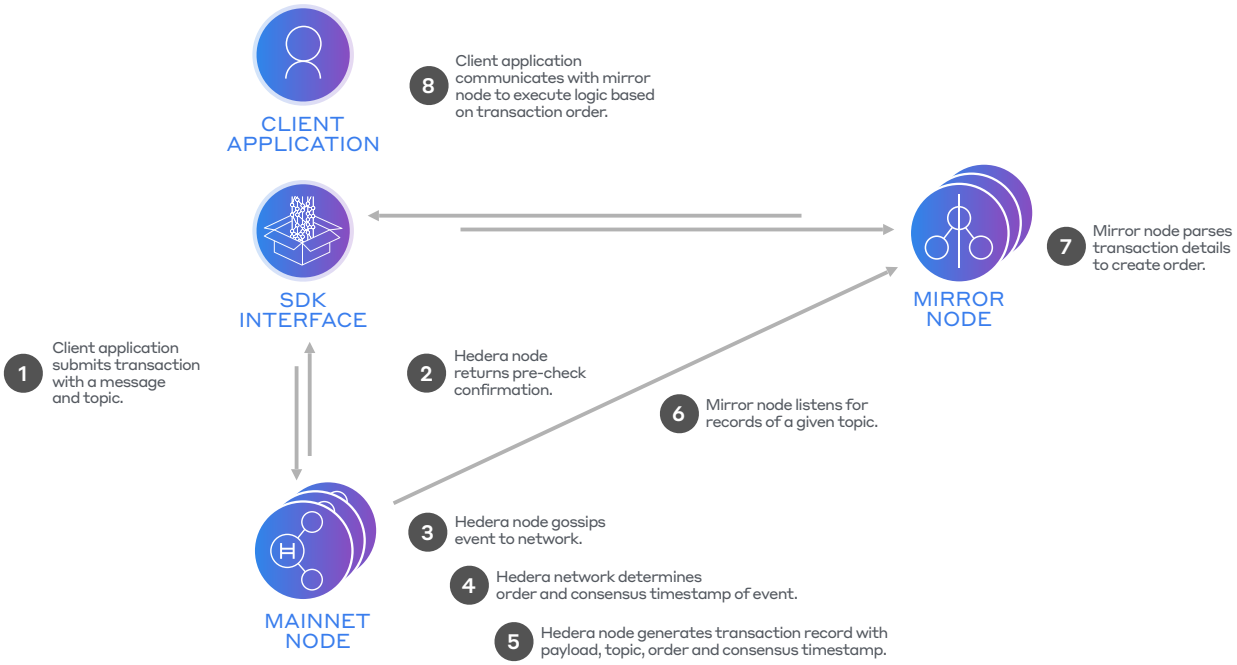


Figure 1B: Transaction Ordering Process

The figure above outlines the process for sending a transaction to the Hedera Consensus Service. The client application will create a transaction using the Hedera SDK which allows it to include a message and topic. The message could describe some action, or contain just a hash, or be any other byte array relevant to the client application. Each application will need to use one or more topics.

Like the other Hedera network services, the transaction can be sent to a single or multiple mainnet nodes. The mainnet node will check that the transaction has the necessary information (signature(s), payment, inputs) and return an acknowledgment to the client application that the transaction has met precheck. A sample transaction is shown below:

```
{
  "message":"7a6a7c5ce8c7ba78c823faf29b32456b001ccef8d5810c05a6624276a0ac7866f2f331a74c06a5faa0daa7797868454",
  "topic":"0.0.1234"
}
```

The mainnet node will gossip the event to the rest of the network, allowing the network to determine a consensus timestamp for the event using the hashgraph consensus algorithm. A record will then be generated which includes the message, the topic, the order, the running hash, and the consensus timestamp. The consensus timestamp is 100% final once reached, and typically is reached in a matter of seconds.

The mirror node receives all information from the mainnet, and therefore learns of the transaction and its consensus order, with consensus timestamps, tied together with a running hash. It can also construct state proofs that can prove to a third party the exact list of messages received for the topic, and in what order, and with what timestamps.

The mirror node runs software that implements the application's business logic. It would take the results of an ordered transaction and return results to the application such as matching bids and asks in a stock market, transferring security tokens between account holders, or updating the status of a good for a shipping and logistics provider.

This feature would have a performance and cost profile similar to using the Cryptocurrency Service (<$0.001 per transaction). Finality would be achieved within a matter of seconds.

The application benefits from the distribution of both the mirror network and Hedera public network. Any user can get records from one or multiple mirror nodes and check state proofs[10] to confirm that the mainnet agreed upon that consensus timestamp and order of a transaction. This enables a real-time audit of the mirror node to verify it did the right thing. Any user can also run a mirror node and would immediately know the truth about ordering and correct conclusions.

Hyperledger Fabric Interoperability

The Hedera Consensus Service proof-of-concept use case provides custom Hyperledger Fabric networks with decentralized consensus on the validity and order of blockchain transactions without the need to configure a RAFT[11] or Kafka[12] ordering service.

Hyperledger Fabric features a kind of a node called an **orderer** (it's also known as an "ordering node") that does this transaction ordering, which along with other nodes forms an **ordering service**. Because Fabric's design relies on **deterministic** consensus algorithms, any block a peer validates as generated by the ordering service is guaranteed to be final and correct.[13] In addition to promoting finality, separating the endorsement of chaincode execution (which happens at the peers) from ordering gives Fabric advantages in performance and scalability, eliminating bottlenecks which can occur when execution and ordering are performed by the same nodes.

New as of Hyperledger Fabric v1.4.1, Raft is a crash fault tolerant (CFT) ordering service based on an implementation of Raft protocol in etcd. Raft follows a "leader and follower" model, where a leader node is elected (per channel) and its decisions are replicated by the followers.[14]

[10] State Proof: A cryptographically secure, portable assertion from a majority of the network nodes as to some fact about a transaction entered into consensus or the state that resulted

[11] Configuring and Operating a Raft Ordering Service
https://hyperledger-fabric.readthedocs.io/en/release-1.4/raft_configuration.html

[12] Bringing Up a Kafka-based Ordering Service
https://hyperledger-fabric.readthedocs.io/en/release-1.4/kafka.html

[13] The Ordering Service
https://hyperledger-fabric.readthedocs.io/en/release-1.4/orderer/ordering_service.html

[14] The Ordering Service
https://hyperledger-fabric.readthedocs.io/en/release-1.4/orderer/ordering_service.html

While RAFT is easier to set up and manage than Kafka-based ordering services (another option in Hyperledger Fabric), it still has two drawbacks:

1 Configuration Complexity: There are four interrelated steps in the process of bootstrapping a Hyperledger Fabric ordering node[15] and six steps to add a new node to a Raft cluster.[16]

2 Byzantine Fault Tolerance: A Byzantine fault is a condition where components may act in a malicious way. It even includes the situation where the network itself may be controlled by an attacker.[17] Raft is the first step toward Fabric's development of a Byzantine fault tolerant (BFT) ordering service, but it isn't Byzantine fault tolerant today.[18]

The Hedera Consensus Service will make a global, fault tolerant, and cost-effective ordering service available to any Hyperledger Fabric network built today.[19]

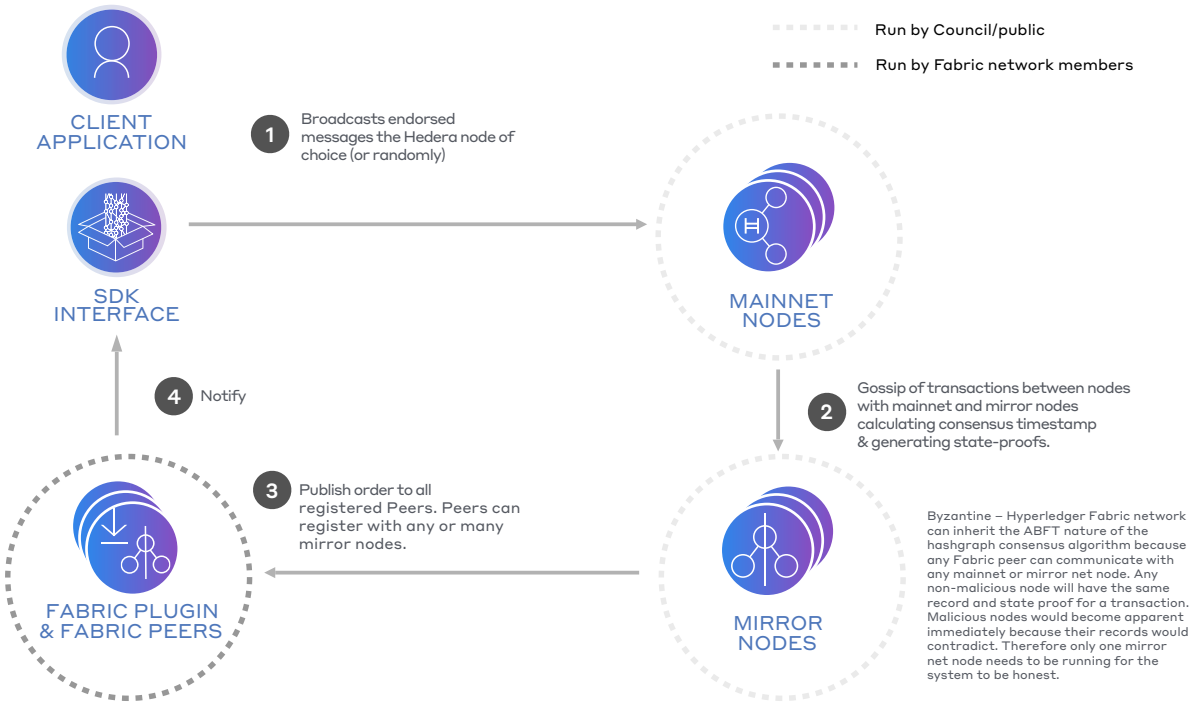


Figure 1C: Hyperledger Fabric/ Hedera Consensus Service Interoperability

[15] Setting Up an Ordering Node
https://hyperledger-fabric.readthedocs.io/en/release-1.4/orderer_deploy.html
[16] Configuring and Operating a Raft Ordering Service
https://hyperledger-fabric.readthedocs.io/en/release-1.4/raft_configuration.html
[17] Byzantine Fault
https://en.wikipedia.org/wiki/Byzantine_fault
[18] The Ordering Service
https://hyperledger-fabric.readthedocs.io/en/release-1.4/orderer/ordering_service.html
[19] Introduction
https://hyperledger-fabric.readthedocs.io/en/release-1.2/whatis.html

The diagram on the previous page demonstrates the architecture for enabling any Hyperledger Fabric network to use the Hedera Consensus Service. In doing so the Hyperledger Fabric network can inherit the Byzantine nature of the Hedera public network.

In step 1 the client application will broadcast an endorsed message to the Hedera network. The transaction will have been endorsed by the Hyperledger Fabric peers using the endorsement policy. The client application can submit the transaction to any of the mainnet nodes, or even multiple if it would like a higher degree of confidence that the transaction is submitted to the network.

The transaction could be passed as any byte array (hash of the transaction, unique transaction id, etc.) and would include a topic which identifies the transaction as belonging to the specific Fabric network. The transaction would then get a consensus timestamp from the Hedera network, preparing it to be ordered.

Mirror nodes would also receive gossip from the mainnet nodes to calculate the consensus timestamp and generate a state proof themselves. One or many mirror nodes would then publish the order to a registered set of Hyperledger Fabric Peers. These transactions would then be structured and stored using a running hash to create a tamper proof chain of ordered transactions relevant to the Fabric network.

Any Fabric peer can communicate with any mainnet or mirror net node. Any non-malicious node will have the same record and state proof of a transaction. Malicious nodes would become apparent immediately because they would be unable to provide valid state proofs.

The Hedera Consensus Service provides any Fabric network the ability to order transactions with high throughput using a global network of nodes that do not need to be operated or individually trusted by the members of the Fabric network. This will reduce operational cost of Fabric-based solutions, improve resiliency to data center outages, and alleviate the need to determine who operates private Fabric ordering services per network.

Use Case

We will explore the example of a Fabric network security token followed by a decentralized stock market in this paper to demonstrate the use of the Hedera Consensus Service in operation.

Private Network Token Issuance

A private network based on Hyperledger Fabric could issue a token for securities trading between regulated industries. This could be a fungible token representing fractional ownership in a specific property where only permissioned investors can purchase or trade the asset.

The members or network administrator would create a topic and communciate the topic ID to all members of the network, so they can recognize both the network and token for the associated transaction.[20]

When user A transfers a token to user B, the client application would automatically hash the transaction ID and submit it in the message payload of a transaction while specifying the correct topic. The transaction would be signed by the user's client application and sent to the Hedera public network.

Once the record is returned with an order of the message, the client application would complete the transfer between users, now having a fair and final consensus timestamp on when the transaction occurred. At scale the application would be able to determine which transfers come first, and which may be invalid depending on their timestamp. Users would be able to query the mirror node or the mainnet directly to confirm the record for a given transaction, or use mirror nodes to look further back in time to ensure the application is decentralized correctly.

The same network could also support atomic swaps of security tokens between networks. Say user A came to an agreement to trade token 1 for token 2, a token issued and traded in another regulated network and currently owned by user C.

In order to exchange the tokens, each user would be a participant in each network. In this use case this may be required because both networks act as regulated financial markets where the investors are verified.

Each token would then be locked up in a smart contract deployed in each network that requires signatures from both users and a timestamp from a transaction on the Hedera public network. Each user will agree to unlock (transfer) the tokens to the new owner simultaneously when triggered by a transaction sent to the Hedera Consensus Service and returned with a consensus timestamp.

[20] Networks have different privacy requirements. Certain networks could choose to use rotating or more anonymized topics to make their transactions harder to identify.

The Fabric-Hedera architecture enables the benefits of permissioned asset trading in a private network and the decentralized trust and immutability of the Hedera public network. Users do not have to worry about manipulation of the transaction order by a centralized party and have confidence that the service can sustain downtime from individual nodes.

Ordering for a Stock Market

Stock markets typically foster behavior causing financial firms to spend millions to get a millisecond advantage in the amount of time it takes them to communicate their bid or ask to the stock market.[21] This fosters malicious behavior where certain firms front run others to achieve a financial advance.

Stock markets built on Hedera will deliver fairness to all market participants.

A stock market could be built as an application in either a private network, similar to the token trading use case above, or directly on top of a mirror node or series of mirror nodes. These mirror nodes would run software allowing them to receive messages from the mainnet, including the consensus timestamp and order. They may only listen to messages sent to the topic related to the stock market built on top.

A user of the application would submit their bid or ask to the Hedera Consensus Service either in an anonymized manner using a random transaction ID, or as a plaintext bid or ask. The message would include a topic that identifies it as belonging to either a specific market or even asset class.

The message would receive a consensus order and timestamp and be returned to the single or multiple mirror nodes running the stock market. The mirror nodes would only be listening for messages which have the correct topic to reduce storage burden. A local database would be structured with the ordered messages.

The application would be able to use this database to match bids and asks based on their consensus timestamps to operate an efficient and fair stock market.

If a user doesn't trust a mainnet node, that used can submit to one or multiple other nodes rather than be bottlenecked by a single source of truth. If a user doesn't trust one mirror node, that user would be able to ask it for a state proof, or ask any other mirror node for the records of transactions and even ask the mainnet for a state proof if users choose. The users may even be able to run the mirror node themselves to additionally verify the outcomes of the stock market.

Any honest node would be able to cryptographically prove they are right and the other is wrong.

[21] On A 'rigged' Wall Street, Milliseconds Make All The Difference
https://www.npr.org/2014/04/01/297686724/on-a-rigged-wall-street-milliseconds-make-all-the-difference

Liars in this use case (users, mirror nodes) would be shown immediately because two node's results will conflict with each other. The barrier for these malicious users to impact the overall consensus process is also much higher through the use of a public network. A single entity (or an aligned group) would need to attain at least 1/3 of all the hbars in existence to materially impact consensus. In smaller private networks this barrier is lower due to both the fewer number of participants and lack of proof-of-stake security mechanism.

It is most likely that stock market applications will add privacy to the above architecture to keep certain information confidential. The application in this case could encrypt the message and submit it to Hedera. Only the appropriate parties would be able to read the message while Hedera would only know that some message was processed.

The application would then decrypt the message using its key before comparing the bids and asks. This allows for true privacy for the stock market.

Additional Opportunities

The previous two use cases only scratch the surface of potential use cases of the Hedera Consensus Service. Ride hailing applications could use the service to match supply with demand for a taxi in real time. Supply chains could use the service to get an accurate and fair timestamp for asset transfers across a supply chain. Parts manufacturers could use the service for real-time actions of goods. IoT manufacturers could use the service to get a consensus timestamp on data read outs from sensors across the globe.

The Hedera Consensus Service provides another tool in the box of application builders for leveraging the power of decentralization.

Conclusion

The Hedera Consensus Service brings the value of fast, fair, secure, and decentralized consensus to any application – private ledger-based or not. Organizations and individuals can issue and trade tokens between private ledgers by using the fair global ordering of transactions for any application.

The Hedera Consensus Service reduces the cost of operating private networks, enables both privacy and scalability, and improves the trust model for both private ledgers and centralized servers.

As with the other Hedera network services, the value of the Consensus Service will be realized most by the diverse applications built on the network by developers from organizations of any size or focus. Long term this will enable a network of interconnected applications leveraging a common service for the ordering of transactions within and between their user bases.