Hedera Hashgraph

Hedera Consensus Service

Authored by:

Dr. Leemon Baird
Co-Founder & Chief Scientist
Hedera Hashgraph

Bryan Gross
Principal Product Manager
IBM

Donald Thibeau
Principal Product Manager
Hedera Hashgraph

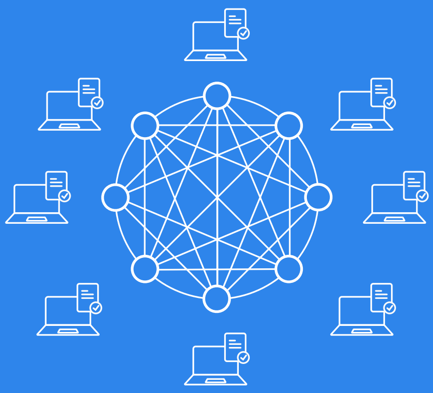
HEDERA 2019

Table of Contents

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 these 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 who 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 the Asynchronous Byzantine fault Tolerance (ABFT) nature of the hashgraph consensus algorithm. 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 is 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 then 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.

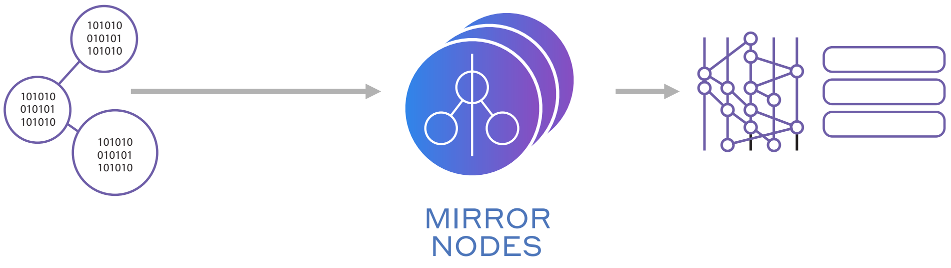
[9] https://www.hedera.com/whitepaper

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 their 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, and has 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 which 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 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, 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.

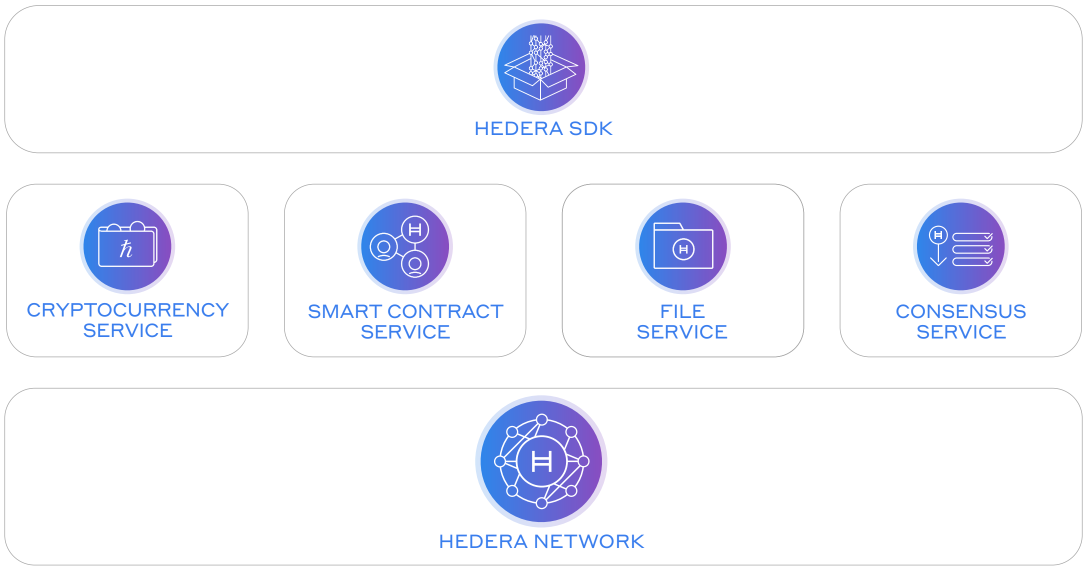


Figure 1A: Public Network

In practice we expect the Consensus Service to be used by a group of mirror node operators and users who are leveraging an application which 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 which they can use to identify transactions relevant to their group. This topic will be attached to messages which the client application will send to the Hedera public network.

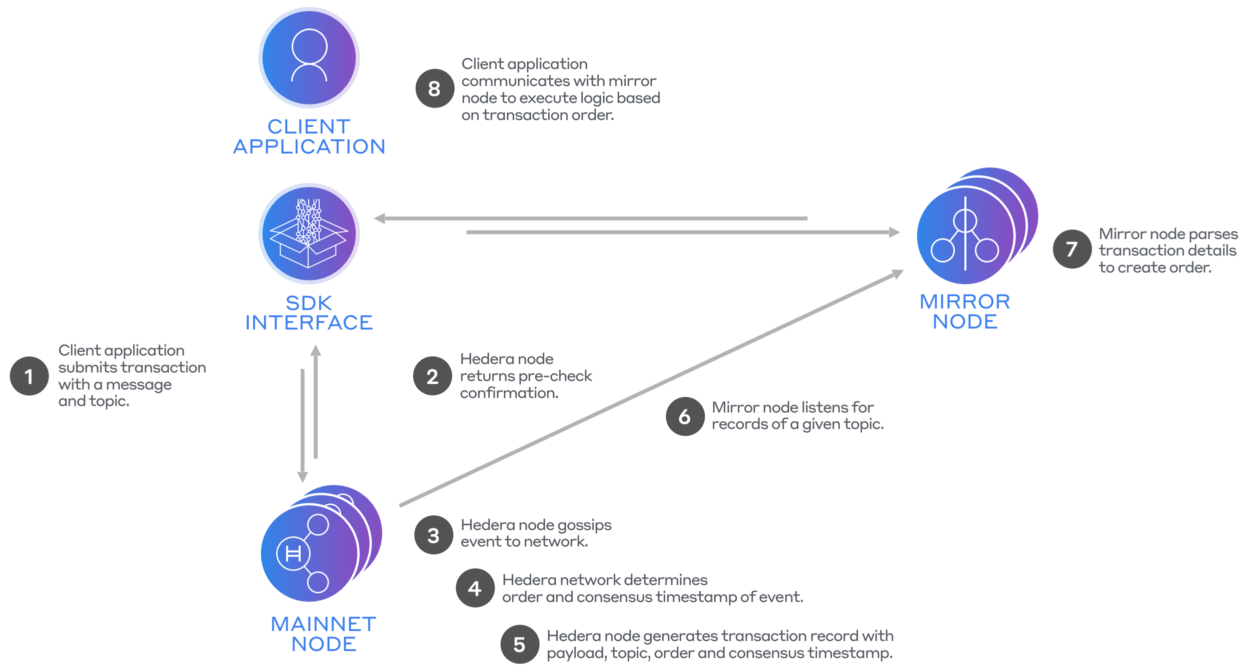

CLIENT APPLICATION
8 Client application communicates with mirror node to execute logic based on transaction order.
SDK INTERFACE
MIRROR NODE
7 Mirror node parses transaction details to create order.
1 Client application submits transaction with a message and topic.
2 Hedera node returns pre-check confirmation.
6 Mirror node listens for records of a given topic.
3 Hedera node gossips event to network.
4 Hedera network determines order and consensus timestamp of event.
5 Hedera node generates transaction record with payload, topic, order and consensus timestamp.
MAINNET NODE

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 is providing 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]

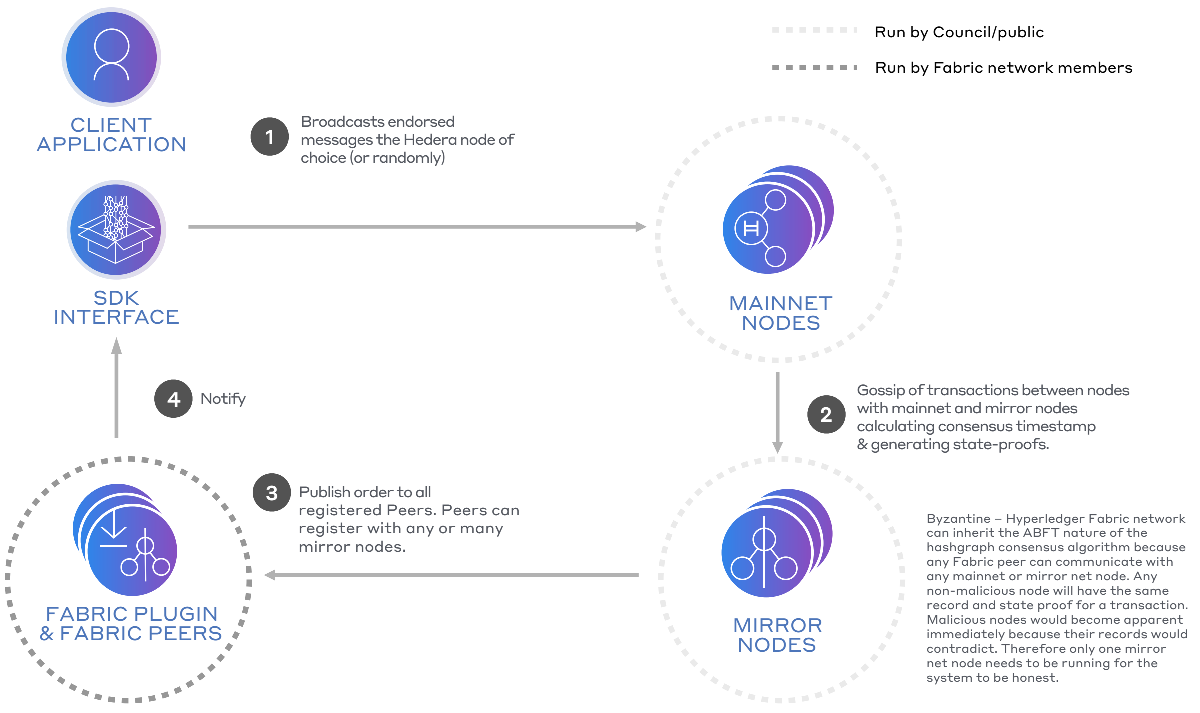

Run by Council/public
Run by Fabric network members
CLIENT APPLICATION
SDK INTERFACE
1 Broadcasts endorsed messages the Hedera node of choice (or randomly)
MAINNET NODES
2 Gossip of transactions between nodes with mainnet and mirror nodes calculating consensus timestamp & generating state-proofs.
MIRROR NODES
3 Publish order to all registered Peers. Peers can register with any or many mirror nodes.
FABRIC PLUGIN & FABRIC PEERS
4 Notify
Byzantine – Hyperledger Fabric network can inherit the ABFT nature of the hashgraph consensus algorithm because any Fabric peer can communicate with any mainnet or mirror net node. Any non-malicious node will have the same record and state proof for a transaction. Malicious nodes would become apparent immediately because their records would contradict. Therefore only one mirror net node needs to be running for the system to be honest.

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 in order 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 which allows 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 which 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 they 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, they 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 they 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 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 manufactures 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 ledger 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 over 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.

Hedera™ Hashgraph

What future will you build?