KeetaNet:
Scalable Blockchain Banking

Roy Keene, Tanveer Wahid, Ezra Ripps, Ty Schenk

2025-03-12

Abstract

KeetaNet is a Delegated Proof of Stake (dPoS) blockchain system built from the ground-up to power blockchain banking on a global scale. It was designed as a cloud-scale and cloud-native solution for blockchain systems. Fundamentally, blockchain systems are state machines that high-performance computing and cloud computing have been able to optimize for a long time. KeetaNet has been engineered to produce a system with unprecedented results, scaling linearly from zero to the hardware limits of any cloud provider.

1 Introduction

Keeta

Keeta is an engineering company that builds resilient and scalable solutions for the financial community. It was founded to provide a link between traditional financial institutions and decentralized blockchain-based technologies with distributed ledgers and blockchain technology.

KeetaNet

KeetaNet is a next-generation blockchain system that implements a distributed ledger with built-in multi-token support, an extensible permissions system, and robust capabilities designed to meet the requirements of a regulated ledger for handling complex financial transactions.

KeetaNet builds on the foundation of many other projects, resulting in a state-of-the-art blockchain system capable of achieving breakthroughs in the volume of transaction throughput while preserving the desirable attributes for blockchain systems. Additionally, KeetaNet increases the transaction volume without compromising the transaction latency. This is accomplished through three primary mechanisms, which we will explore in this paper.

- Unique hybrid DAG design: The primary data structure is a directed acyclic graph (DAG) where each account forms the basis of its own graph and inter-account interactions are virtual links between the graphs
- Client directed: Changes to the distributed ledger are validated through a 2-step process directed by the entity requesting the change and are presumed not to be conflicting
- Cloud-scale: Built atop common protocols to take advantage of the enormous scaling capabilities of cloud providers using technologies like serverless processing

Some existing systems have some of these features, but KeetaNet is the only system that has all three.

In addition to being a scalable and fast blockchain system, KeetaNet implements all the features needed to run a reliable digital economy:

- Compliant: Manages information for "know your customer" (KYC) and anti-money laundering (AML) regulations in an efficient way

- Extensible: Permits network participants to define and implement their own policies and regulatory requirements
- Native tokens: To allow users to safely create their own representations of assets on the network
- Atomic swaps: Users can directly settle swaps of user-created tokens without needing third-party intermediaries
- Permissions: Asset issuers control interactions with their tokens to ensure compliance and security requirements are consistently met

With all of these state-of-the-art features – scalability, native token support, atomic swaps, adaptable permissions, and capacity for on-chain regulatory compliance – KeetaNet will fundamentally change the shape of digital currencies.

2 Basic Concepts

2.1 What is a distributed ledger?

A distributed ledger is a specialized form of distributed system designed to maintain a consistent, synchronized ledger among multiple nodes, despite constraints such as network latency, node failures, or the presence of participants lacking inherent trust in one another – known as mutually distrusting parties.

A distributed system is a type of system in which work is spread across multiple interconnected nodes, each of which communicates with other nodes to achieve a common goal. Despite being separate entities, these nodes function together as a unified system. The nodes in these systems can be governed by a single centralized authority or by mutually distrusting parties. In a decentralized system, each node operates with only the information it has–known as information locality–in order to accomplish its goal.

Distributed Ledger Technology (also known as DLT) is a distributed system comprised of three main components: the database (such as a blockchain or other method of persisting the state of the system), a network, and a consensus mechanism.

Although the most common data structure seen in DLT systems is a blockchain, any data structure can be used as long as it maintains a state.

A network in a DLT system consists of multiple nodes, each–in the most simple setup–storing a complete copy of the database. These nodes may be operated independently by mutually distrusting parties from around the world, which communicate with each other using a standard protocol, such as the Gossip protocol used by Bitcoin. Using this shared language, they can convey updates to the state and forward new transactions that are not yet committed to the state.

The consensus mechanism of a DLT system is the process by which network participants agree on which blocks are part of the current state of the system.

2.2 What is a blockchain system?

While a blockchain system can take many forms, the essential component is a sequential data structure composed of records, each containing a cryptographic hash of the previous record. The linkage created by these hashes forms part of a cryptographic proof that validates the integrity and immutability of all preceding records, ensuring they have not been altered.

With this minimal definition, some basic tools utilize a blockchain data structure such as git[1] or fossil[2]; however, a more common interpretation of blockchain systems also requires a mechanism to determine which updates to the records in a given chain are considered "valid" based on application-specific criteria. This is called the "consensus mechanism." Consensus mechanisms can also exist independently of the blockchain data structure, often implemented in distributed key value stores (e.g. MongoDB, DynamoDB, etc.).

Blockchain systems often use Merkle hash-trees[1] to efficiently verify all their content from the most recent entry. This enables quick auditing of the blockchain's history by verifying that the latest change achieved consensus through the consensus

[1] *Is a git repository a blockchain?* https://medium.com/@shemnon/is-a-git-repository-a-blockchain-35cb1cd2c491

[2] *Is Fossil a Blockchain?* https://fossil-scm.org/home/doc/trunk/www/blockchain.md

mechanism. Each change is linked to the preceding change via a cryptographic hash, and this recursive structure ensures that all prior changes are verifiable.

Systems that incorporate both a blockchain data structure and a consensus mechanism are commonly referred to as blockchain systems. The most well-known examples include cryptocurrencies like Bitcoin, Ethereum, XRP, Nano, and others. This is the definition of blockchain systems that we will adopt throughout this paper.

KeetaNet is a blockchain system that implements a novel consensus mechanism and supports a distributed ledger, and thus is also a cryptocurrency system. A main focus of the KeetaNet system is that it is currency-agnostic, supporting tokenization of any asset.

2.3 Why use a blockchain system?

The main advantage of a blockchain system over a conventional database is its ability to facilitate updates from multiple, mutually distrusting participants without the need for a centralized gatekeeper that all parties must implicitly trust. Since the consensus mechanism requires these entities to validate that the constraints of the system are met before any changes to the state of the system are allowed, users can rely on the integrity of the state of the system at any time.

Moreover, because blockchain systems typically involve replication (enabling each participant to validate separate copies of the state and changes), users benefit from increased availability and resilience.

2.4 Importance of scalability

Scalability is critical when evaluating blockchain systems because it directly impacts the practical utility of the system. As the scale of a system grows, so does its capacity to support a wider variety of applications and use cases. The more users and applications a blockchain can effectively support, the greater its practical utility becomes. If a blockchain cannot scale effectively, it significantly limits both its usefulness and the range of potential applications.

The central obstacle to scaling distributed systems, including blockchains, is state management—the data reflecting the current status of the system, such as account balances, transaction records, and interdependent information. Stateless systems scale trivially, but systems maintaining complex state face significant challenges as transaction volumes increase. The more complex the state information becomes, the more coordination and communication are required among nodes, leading to performance bottlenecks and limiting overall scalability.

This scaling challenge is deeply tied to the constraints described by E. Brewer's CAP theorem[2], which states that distributed systems cannot simultaneously guarantee consistency, availability, and partition tolerance. Blockchain systems must inherently maintain partition tolerance to function during network disruptions, forcing them to manage trade-offs between consistency (ensuring all nodes share a unified view of state) and availability (maintaining continuous operation despite node disruptions).

Prioritizing consistency ensures atomic updates across the network at nearly the same time but requires that all nodes be reachable, meaning that the system cannot operate during a partition. In contrast, prioritizing availability allows the system to remain operational even if some nodes are offline, but this approach may lead to temporary inconsistencies during a partition. As a result, transactions or operations that participants believed were successfully completed may be undone or invalidated due to the lack of global consistency in the partitioned system state being not globally consistent.

Thus, managing state effectively within the constraints of the CAP theorem is fundamental to achieving scalability. Striking the right balance between consistency, availability, and transaction throughput enables blockchain systems to scale effectively, supporting diverse and demanding applications in a reliable and performant manner.

2.5 KeetaNet improvements upon existing systems

Although blockchain systems have made significant strides in creating decentralized and secure systems for data storage and transfer, most existing solutions face limitations in multiple areas.

- Extensibility – the capacity to adapt and evolve over time, incorporating new features and improvements without compromising core functionality
- Decentralization – the degree to which control and governance are distributed across the network, rather than being concentrated in a single entity or small group of entities
- Scalability – the ability to handle a large volume of transactions without experiencing a decline in performance
- Features for Global Finance – the range and utility of features offered, particularly with regard to interacting with regulated financial institutions and changing user requirements

2.5.1 Extensibility

Many existing blockchain systems are limited by a lack of extensibility, often due to hard-coded rules in the protocol. This rigidity of the design makes updating some fundamental assumptions a challenging process requiring broad community consensus, impeding long-term adaptability. In contrast, KeetaNet leverages a more dynamic and adaptable framework in several ways:

- Representative authority – the consensus rules in KeetaNet are determined by the nodes on the network providing validation services – called "representatives", allowing for a flexible governance structure that can accommodate changes and improvements
- Block structure versioning – KeetaNet supports versions in its block structure, enabling future enhancements to be smoothly integrated without disrupting existing operations
- Extensible formats – KeetaNet utilizes extensible formats throughout its architecture, offering greater flexibility in how data is processed and stored:
  - Operation types are represented as big integers, providing ample room for a variety of transaction types
  - The system uses object identifiers (OID) for specifying cipher formats, enabling easy adoption of future cryptographic methods
  - No Fixed-Length Encoding – by avoiding fixed-length encoding, KeetaNet provides the flexibility needed for to adapt to unknown changes in the future without being constrained by initial design choices
  - Usage of ASN.1 – by using well-established standards such as ASN.1, KeetaNet allows for flexible arbitrary data storage which can be versioned and encoded/decoded in a familiar way
  - Extensible keys – by supporting an extensible keying format KeetaNet is able to deal with changes that may arise from weaknesses discovered, this enables post-quantum cryptographic support on KeetaNet

By incorporating these features, KeetaNet aims to offer a blockchain system that is not only secure and efficient, but also equipped for long-term scalability and adaptability.

2.5.2 Decentralization

A core tenet of blockchain systems is decentralization, yet many existing platforms compromise on this aspect due to various design constraints or governance models.

KeetaNet ensures a high level of decentralization by allowing users to distribute decision-making authority among what it calls "representatives", which are actors within the system that are given "voting power" to vote on the permissibility of blocks within the blockchain system.

2.5.3 Scaling

Most existing blockchain systems effectively function as secure distributed ledgers and, in some cases, as platforms for running programs (called "smart contracts"[3]) across instances of the distributed ledger. However, they fail to do so in a way that is scalable. These systems fail in a few different ways:

- Validation architectures that are not intrinsically scalable
- Inefficient transaction distribution mechanisms
- Utilization of bespoke protocols that may not be holistically optimized for scalability

Existing blockchain systems currently face scalability issues analogous to those experienced by the early World Wide Web. Typically, blockchains attempt to enhance performance by incorporating multithreading techniques within individual validator nodes. This approach resembles transitioning a monolithic application to use multithreading but remains constrained by the resources and capacity of a single node. A more effective solution would be analogous to migrating from a multithreaded monolithic application to a cloud-based distributed architecture. This approach distributes the workload of a single logical validator across multiple cooperating nodes, significantly enhancing scalability beyond traditional single-node solutions.

In contrast, KeetaNet goes beyond simple multithreading by supporting operation across large numbers of computers, allowing it to scale with traffic. This approach mirrors how the Web has evolved to solve scalability challenges using distributed systems using cloud computing and serverless architectures.

Bitcoin network is inherently difficult to scale in its base layer, largely by design. The expectation is that scaling will occur through secondary layers, such as the Lightning network[4]. For example, Bitcoin blocks were originally capped at 1 megabyte in size, and changing this limit has proven challenging, as demonstrated by the "Blocksize War"[5].

In 2017, a soft-fork update to the Bitcoin protocol known as Segregated Witness (SegWit)[6] was introduced to address transaction malleability. SegWit moved digital signatures out of the transaction data and into a segregated witness area, which also allowed more efficient use of block space. Additionally, SegWit introduced the concept of block weight rather than block size. Under this new system, the block weight was set to 4 million units, which corresponds to a maximum block size of 4 megabytes after the SegWit implementation.

Despite the improvements made by SegWit, Bitcoin underwent a hard fork in August 2017 at block height 478559, splitting into Bitcoin and Bitcoin Cash. Although Bitcoin's consensus mechanism ensures that protocol updates are agreed upon by a large majority (usually around 95%) for the long-term benefit of all network participants, this same process makes updates, scaling, and agreement on the best path forward particularly difficult, often leading to conflict.

Moreover, Bitcoin faces other challenges, particularly in connection with its mempool. The mempool acts as a queue for transactions that have been broadcast to the network but have not yet been included in a block by the miners.

Due to limited blockspace, network congestion can occur during sudden spikes in transaction volume or when there is a sharp drop in hash rate, as seen on March 9th, 2023 when nearly 81,000 transactions piled up in Bitcoin's mempool[7]. This congestion led to higher than average transaction fees and longer processing times. As adoption and network usage on the network increases, blockspace will become more scarce and expensive, resulting in increased demand for layer 2 solutions like the Lightning network to handle everyday transactions.

Ethereum faces similar challenges when it comes to scaling. Several proposals such as sharding, rollups, state channels, side chains, and the use of layer 2 protocols have been made to address these shortcomings[8]. Sharding allows for partitioning of the database and allowing subsets of validators to be responsible for individual shards at the expense of more complex consensus logic. However, it was shelved in favor of rollups such as optimistic rollups and zero-knowledge rollups. Rollups allow transactions to be handled off-chain on a different layer and then posted back on the base layer (L1). State channels are

multi-signature smart contracts that allow for transacting off-chain and settling back on-chain similarly to rollups. Side-chains, like Polygon, are separate blockchains running the Ethereum Virtual Machine (EVM) which connect to the Ethereum mainnet using two-way bridges; however, they do not post state changes back to the Ethereum mainnet. While there are several approaches, each comes with their own drawbacks and many often require processing either off-chain or on different blockchains.

Additionally, its mempool is even more challenging due to the network's support of smart contracts. Sometimes referred to as the "Dark Forest,"[9] Ethereum's mempool is a highly adversarial environment where automated bots are continuously attempting to achieve miner extractable value (MEV also known as maximal extractable value post-Merge). Users can be subjected to attacks such as "front-running" resulting in higher than expected costs and delayed transaction settlements [10]. There have been solutions to prevent this, such as Flash-Bots, which submits transactions directly to validators without exposing them in the public mempool, however, this is a separate project[11]. Even after "The merge" where Ethereum migrated to a Proof-of-Stake (PoS) consensus mechanism, newly formed block builders are still incentivized to construct the most lucrative blocks.

2.5.4 Features for Global Finance

When it comes to the features needed for global finance, many existing blockchain systems fall short, especially in terms of support for regulatory compliance. Traditional blockchain systems often lack the infrastructure to handle compliance requirements such as Know Your Customer (KYC) and Anti-Money Laundering (AML) regulations, as well as to deal with the needs of banking systems to be able to administratively control currencies.

KeetaNet addresses these limitations by providing a framework that can be adapted to meet various regulatory needs without sacrificing its core features or decentralization. This makes KeetaNet not only technologically superior but also practically viable for real-world financial applications.

3 Prior Art

Before diving into the specifics of various existing systems, it is crucial to understand the varied landscape in which these systems operate and differ from KeetaNet. Each blockchain protocol is a unique concoction of decisions made to balance trade-offs inherent to distributed systems. These challenges include, but are not limited to, Byzantine Fault Tolerance (BFT), Consistency, Availability, and Partition Tolerance. By unpacking these systems, we aim to shed light on the limitations they exhibit when faced with these complex, interrelated challenges and how KeetaNet seeks to address them more comprehensively.

3.1 Core challenges

The Byzantine Generals problem is a classic feature in distributed systems that strive to achieve broad consensus among distinct but connected participants, some of whom may be unreliable or outright malicious. Another classic distributed system problem involves trying to maximize three competing goals. Consistency, Availability, and Partition Tolerance in a distributed system. Academic papers dating back to the 1970s and 1980s, well before modern blockchain technologies, researched these concepts related to distributed systems. This research resulted in several seminal papers such as ones on the CAP theorem[2] (also called Brewer's theorem after its author), the Byzantine Generals problem[12], and the FLP Impossibility theorem[13].

The importance and usefulness of the Byzantine Generals problem as an allegory for consensus has led to a general design goal referred to as Byzantine Fault Tolerance (BFT). The more general CAP theorem states that three key features of distributed systems cannot all be optimized simultaneously. Consistency refers to the fact that each node in a network has the same view of the state of the system. This feature is key to the network's security. Availability simply refers to the fact that the nodes are online and responsive at all times in a distributed network of computers. This feature deals with node operations failures and communications links of various kinds. Partition Tolerance refers to proper operation and

recovery of the network in the face of communication failures causing one or a subset of operating nodes to become partially isolated from the rest of the network for some period of time.

Thus, design and implementation of consensus seeking distributed network protocols have to make trade-offs across these three functional areas depending on the project goals and priorities'. For example, Consistency among distributed data stores (e.g., ledgers) generally means eventual data consistency in light of network latency, faults, etc. Availability of 100% of nodes being fully operational for all time can never be guaranteed. And partitions will inevitably occur in the presence of node faults, crashes, and connectivity issues.

Finally, the FLP Impossibility theorem asserts that perfect consensus cannot be guaranteed across an inherently asynchronous network in the presence of just a single fault.

Modern blockchains closely resemble distributed databases regardless of how they implement state persistence. Since all the classic challenges and trade-offs of legacy distributed systems remain a core challenge for blockchain systems, every blockchain design must address how to reach and maintain consensus across the network in the face of faults and potentially malicious and colluding participants.

3.2 Existing blockchain systems

Bitcoin The pioneering blockchain platform, Bitcoin, was created by an anonymous person or group of people under the name Satoshi Nakamoto, who first introduced it in a white paper in 2008[14] and launched the Bitcoin network in 2009. It uses a consensus mechanism known as Proof of Work (PoW) to validate transactions and create new blocks. Bitcoin's PoW relies heavily on computational power.

The base transaction layer in Bitcoin does not support complex smart contracts, as seen in Ethereum or Sui. Instead, it primarily focuses on the transfer of Bitcoin, the native digital currency, between "addresses." The simplicity of Bitcoin's scripting language ensures a high level of security, but limits the scope of transactions to sending funds or basic conditions such as time-locked or escrow transactions and multi-signature wallets.

Blocks in the Bitcoin blockchain are created approximately every 10 minutes. Each block includes a collection of transactions, and once a block is added to the blockchain, the transactions within it are considered confirmed. Users of the Bitcoin network generally consider a transaction to be sufficiently "final" after six confirmations, or roughly one hour.

As for scalability, Bitcoin has a theoretical throughput limit of around 7 transactions per second with its base layer, which is much lower than that of newer blockchain systems. To address this, "layer 2" solutions such as the Lightning Network[4] have been developed to handle smaller and more frequent off-chain transactions, thus freeing up the main Bitcoin blockchain for larger and more important transactions.

Unlike Sui and Keeta, Bitcoin's consensus mechanism does not rely on authorities or validators for transaction ordering and confirmation. Instead, it relies on decentralized miners who participate voluntarily and are incentivized by block rewards and transaction fees.

Moreover, Bitcoin's architecture is intentionally designed to be minimalistic and focused, which contributes to its robust security, but also limits its flexibility and scalability. Over the years, various forks and updates have attempted to address these limitations, but the core protocol remains largely unchanged.

Ethereum Originally proposed by Vitalik Buterin in late 2013, Ethereum was formally launched on 30 July 2015 and has evolved significantly over the years. Originally built on a Proof of Work (PoW) consensus mechanism similar to Bitcoin, Ethereum transitioned to a Proof of Stake (PoS) model on 15 September, 2022, an upgrade commonly known as "The Merge"[15].

Unlike Bitcoin's transaction-focused architecture, Ethereum includes a more versatile scripting language that allows for the development of decentralized applications (dApps) and smart contracts. Although this makes the platform more flexible, it also introduces complexities such as varying gas fees

	Consensus Mechanism	Throughput	Finality[3]
KeetaNet	dPoS	10M TPS	Sub-second
Sui	dPoS	300K TPS	Sub-second
Aptos	PoS	160K TPS	Sub-second
Tendermint	PoS	10K TPS	2 seconds
Ethereum (L2)	-	5K TPS	minutes to weeks
Ethereum (L1)	PoS	15 TPS	6.4 minutes
Nano	dPoS	160 TPS	Sub-second
Bitcoin	PoW	7 TPS	60 minutes

Table 1: Comparison of Features Between Blockchain Systems

and slower transaction speeds. Additionally, Turing-complete smart contracts written in languages such as Solidity may lead to security vulnerabilities, as evidenced by the numerous exploits over the years[16].

The transition to PoS marked a major shift in Ethereum's consensus algorithm, yet it did not significantly alter transaction speeds or gas fees for end users. In this new PoS system, Ethereum organizes its blocks into epochs, each lasting about 6.4 minutes. This is roughly similar to the finality time under the former PoW mechanism, which stood at about 7 minutes for 35 confirmations by convention. However, the PoS model introduces an additional layer of security, enhancing the network's overall robustness.

In terms of scalability, Ethereum currently supports a throughput of about 15 transactions per second (TPS) at its base layer. However, layer 2 solutions are designed to enhance the scalability of Ethereum-based applications, offering higher-throughput and increasing the network's overall transaction capacity.

Much like Bitcoin, Ethereum has inspired a multitude of forks and updates aiming to enhance its functionality and performance, although its core features and purpose–enabling smart contracts and dApps–have largely remained unchanged.

Aptos Aptos is the successor to the Diem blockchain project (formerly known as Libra), which Meta discontinued in early 2022. Aptos uses a Proof-of-Stake (PoS) consensus mechanism, DiemBFTv4[17], for global consensus. In this regard, Aptos is most similar to Tendermint and can be thought of as an evolution of leader-based PBFT protocols. However, Aptos distinguishes itself by enabling parallel execution through its Block-STM[18].

In Aptos, validators group transactions into batches. When a client submits a transaction, it is broadcast to all validators, each of which independently processes the transaction and adds it to their batch. These compiled and signed block batches are then sent to all other validators, who store them persistently. This "continuous transaction dissemination", as the name suggests, happens continuously between all network validators. These persistent block stores, or "mempools", serve as batch storage. Transactions are validated locally when the majority of stake-weighted signatures (one-half plus one) from other validators are received.

A leader is temporarily selected from the active validator set via the DiemBFTv4 protocol. The leader's role is to order their block batches and propose a new block, a process known as "block metadata ordering," which closely mirrors Tendermint's voting mechanism. Batch metadata ordering undergoes two rounds of voting, where two-thirds plus one of the validators must agree. Once the block metadata are agreed upon, each validator can independently execute the block batches on their ledger. Periodically, the validators share their ledger status with other validators, ensuring that a globally authenticated ledger propagates throughout the network.

Unlike Sui or Keeta, which prioritize horizontal scalability through alternative means, Aptos aims to achieve scalability primarily through network sharding. This approach, similar to those used by Ten-

dermint derivatives, divides the network into multiple states to improve throughput. However, sharding introduces additional complexity and can negatively impact the network's liveness–the property that ensures that the network can continue functioning over time.

Sui Sui uses a hybrid consensus model, the base transaction layer of which has been adapted from the original FastPay system developed in 2017[19]. This base layer utilizes a dPoS consensus mechanism similar to Keeta, called Byzantine Consistent Broadcast. However, for shared objects (like smart contracts) which require a defined global order, a different consensus mechanism called Narwhal and Tusk may be used, which is currently under development by the Sui team. Narwhal is designed to handle the high throughput of transaction processing, while Tusk provides an asynchronous Byzantine Fault Tolerant (BFT) consensus mechanism for achieving precise ordering of these shared objects, a process known as Byzantine Agreement[20].

In Sui, transactions are initially signed by the user, and the client sends the transaction to all authorities (analogous to validators or representatives). Upon receiving the transaction, each authority validates the authenticity of the transaction and returns a signature to the client, called a partial certificate. Once the client has collected enough partial certificates, equal to the quorum of that epoch (two-thirds plus one of the authorities), it forms a full transaction certificate.

What happens next depends on the type of transaction. If the transaction involves shared objects (such as smart contract objects), the full transaction certificate is sent to the authorities and a higher-level Byzantine Agreement Protocol, which is responsible for the global ordering of the shared objects. This separate consensus mechanism, which is detailed in the Narwhal and Tusk white paper[20], uses an asynchronous consensus mechanism based on a DAG-based mempool. Authorities then wait for the Byzantine Agreement Protocol to finalize the transaction, ensuring the correct ordering of shared objects before the process proceeds.

If the transaction does not involve shared objects, or if it does involve shared objects that have been ordered by the Byzantine Agreement Protocol, each authority validates the authenticity of the full transaction certificate sent by the client and summarizes the result into a signed effects response. Clients are responsible for collecting each signed effects response from the authorities until a quorum of the signed effects has been collected, equal to two-thirds plus one of the authorities. Clients bundle these signed effects responses into an effects certificate, which provides proof to any entity that the transaction may proceed.

Finally, either the client or the recipient (if the client sent the effects certificate directly) publishes the effects certificate to the authorities, who then independently update their internal ledgers. At this point, the transferred assets are available for use by the recipient.

Unlike Sui, Keeta does not utilize a separate consensus mechanism for the global ordering of shared objects. This greatly reduces the complexity and latency of the system, but at the cost of liveness for the shared objects. For more information on the engineering trade-offs of using two consensus mechanisms, see the reference provided by the Sui team[21].

Like Sui, Keeta is expected to scale linearly with the number of workers, analogous to persistent "machines" in other systems. Unlike persistent machines, however, workers are spawned as needed for each incoming request and terminate when the request is handled. This allows Keeta to scale completely linearly, bound only by available hardware, by increasing the workers available to each representative as network throughput increases.

Unlike KeetaNet, Sui is limited by the number of cores per machine, preventing true linear scaling. Keeta is only dependent on the number of workers it can spawn assuming that no bottlenecks, like internal network bandwidth, impede it. In practice, however, the serverless workers in KeetaNet are provided by either Google Cloud Run or AWS Lambdas and are almost unbounded.

Tendermint Tendermint is the consensus protocol behind several well-known projects, including Cosmos, Terra, Injective, Binance Smart Chain, and

Provenance (the blockchain system powering the USDF Consortium[4]). Tendermint is a leader-based proof of stake (PoS) consensus mechanism. In some cases, such as for Provenance[5], the Tendermint protocol has been modified to support dPoS, but is still leader-based.

Like other PoS systems, the validators in Tendermint are selected based on the bonding of the native governance token (for example, Atom in the Cosmos network) to a validator's account. The more tokens bonded to the account, the greater voting power that validator has on the network. Validators are incentivized to vote honestly by two methods: through fees collected from validated blocks and through a slashing mechanism, where some of the validator's stake is forfeited if other validators detect malicious behavior during the voting process.

To initiate a transaction using the Tendermint protocol, a client first broadcasts their transaction to the network mempool cache with an associated fee. Typically, each validator maintains their own mempool cache. A leader is selected in a round-robin style based on each validator's total network stake, and the leader begins to build a block from the transactions existing in the mempool starting with the highest fee transactions (of which the leader will reap the fees). Since blocks have a maximum size, it is possible that not all transactions in the mempool will be included in the block to be voted on. In this case, those user transactions will be included in subsequent blocks, or in the case of high network congestion, may not be included at all.

Once the leader has assembled the block, it broadcasts the proposed block to the other validators. Validators then perform a two-step voting process: first, they sign a prevote if they deem the block valid and broadcast these prevotes to other validators. Once a validator receives a two-thirds majority of prevotes (i.e., more than 66% of the validators have voted for the block), it signs a pre-commit message and broadcasts this message to the other validators. Finally, once a validator receives a two-thirds majority of pre-commit messages, it commits the block, adding it to the global blockchain and increasing the block height by one.

At this point, all transactions included in the block are considered final and can be spent by the recipient. The process then starts again with a new leader selected for the next round of voting, and the cycle repeats.

One important note about Tendermint is that it enforces absolute global ordering for transactions, meaning that transactions cannot be processed in parallel. Every transaction must wait to be included in a sequential block (the "monolithic block" model[22].). This design choice ensures strong consistency but limits scalability, as it prevents parallel processing of transactions. Global ordering is absolute and transactions cannot be processed in parallel. Transactions must wait to be included in each monolithic block.

Like Nano (discussed below), each vote in Tendermint is broadcast to all other nodes asynchronously. This results in very high network traffic, limiting maximum network throughput. In KeetaNet, non-deterministic network traffic is unicast to the client only, significantly reducing network traffic and allowing for higher network throughput.

Nano Nano sets itself apart in the blockchain ecosystem by prioritizing fee-less and instant transactions. It uses a unique block-lattice architecture, where each account has its own independent blockchain. This structure facilitates high transaction throughput, minimizes conflicts, and ensures that transactions are processed quickly and efficiently. Unlike many other blockchains, Nano focuses exclusively on peer-to-peer transactions, deliberately avoiding complex smart contracts and decentralized applications. By narrowing its focus, Nano optimizes for speed and simplicity, with transactions generally finalizing within seconds and without any associated fees.

Nano's consensus mechanism operates on an Open Representative Voting (ORV) system, a specific form of Delegated Proof of Stake (dPoS). Unlike typical dPoS systems, Nano's ORV doesn't economically incentivize delegates, making it fundamentally different

[4]https://usdfconsortium.com/
[5]https://provenance.io/

in governance. In this network, users delegate their balances to representatives, often run by third-party wallets or exchange services, who are responsible for transaction verification.

Most clients in Nano's network rely on these third-party services because operating a full node is necessary for keeping track of the network's state. The transaction process in Nano involves multiple rounds of voting among these representatives. Each representative initially sends a non-final vote for the transaction block, and once the block receives enough votes to cross a weight threshold—typically 50% of the online voting weight—a final vote is cast. This block is then cemented into the ledger.

The accounting process in Nano is particularly rigorous when it comes to fund transfers. After a "send" operation, the recipient must issue a "receive" block to spend the transferred funds. This dual-action mechanism serves as a form of digital double-entry bookkeeping. Specifically, the "send" block acts as a debit entry and the "receive" block acts as a credit entry.

One notable aspect of Nano's protocol is that votes are not permanently recorded on the ledger. If a network node requires bootstrapping, it must obtain each missing block individually from other nodes and go through a new round of voting by representatives against the "frontier blocks."

Another unique feature of Nano's network is that each transaction must include a small Proof of Work (PoW) nonce before being broadcast. This PoW acts as a deterrent against spam transactions, effectively serving as a "fee" paid for computing power.

Similarly, KeetaNet introduces its own flexible fee mechanism, which allows representatives, who directly handle transaction requests from users, to enforce network constraints at their discretion. This could include refusing service under certain conditions, providing a more customizable approach to transaction validation and fee management.

4 System Architecture

KeetaNet's architecture draws parallels to both contemporary database systems and modern web backend design. At its core, like any blockchain system, KeetaNet functions as a distributed database. However, much like web backends serve as interfaces to databases, KeetaNet models itself as an interface to its underlying ledger.

This dual resemblance is not accidental, it reflects a deliberate design choice to harness the decades of research and development in database technologies and web backend scalability. By combining the strengths of both fields, KeetaNet achieves remarkable scalability, benefiting from the best practices and time-tested methodologies developed in these mature domains.

The KeetaNet reference implementation is written in TypeScript to underscore the fact that KeetaNet's scalability and performance are not the result of excessive hyper-optimization, but rather the product of a thoughtful, and inherently scalable design. It also ensures that the reference implementation is accessible and readable to a broad range of developers.

4.1 Components and their interconnections

The KeetaNet system comprises multiple interconnected components, each contributing to the overall functionality and efficiency of the platform. See Figure 1 on the following page for a graphical representation of these components and their relationships.

4.1.1 Node

A node is a piece of software that participates in the KeetaNet network. Nodes can be either representatives, which actively vote on network decisions, or participants, which only observe the network's operations. Nodes that serve as representatives refer to their own copies of the ledger to ensure that new operations are valid. These nodes are essential for maintaining the security, integrity, and overall functionality of the network.

4.1.2 Ledger

The ledger records everything on the network, including user identities, account balances, operation history, and votes. This is different from the blockchains

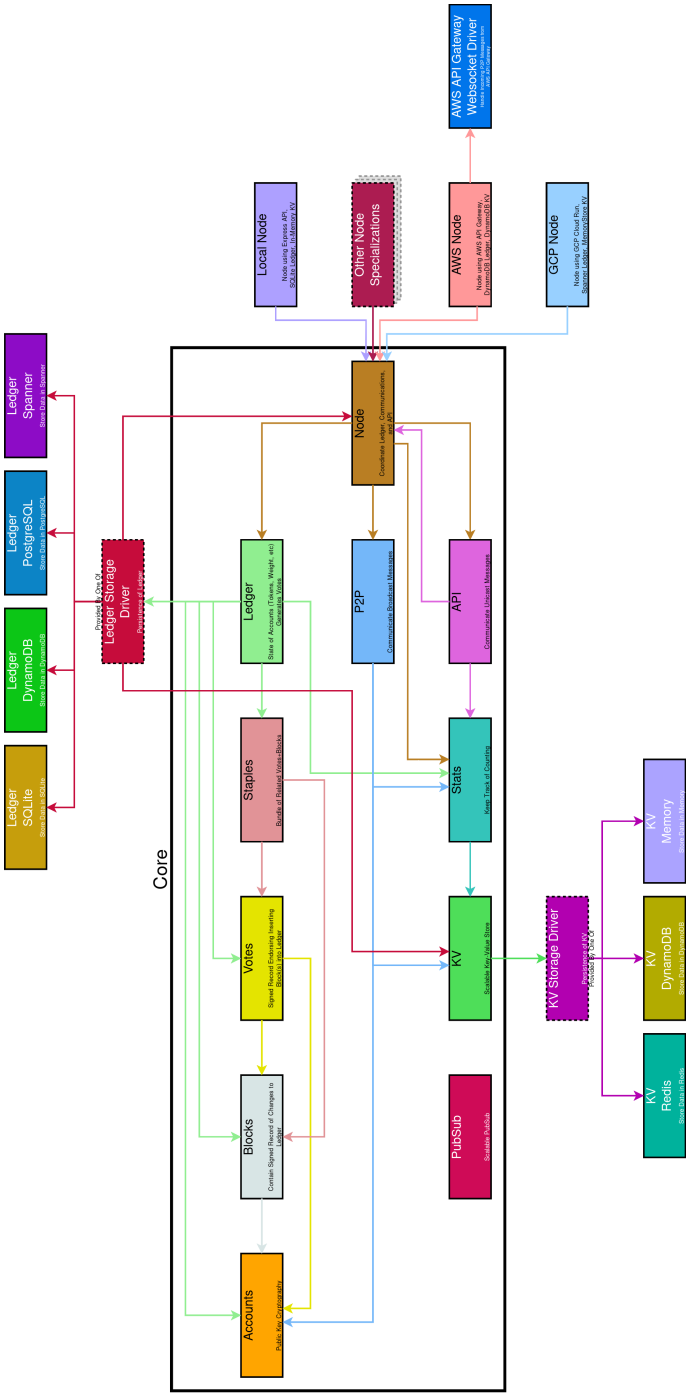

Figure 1: Components and their relationships

that record operations for a specific account. All operations are added to the ledger once they are confirmed by a quorum of representatives through the consensus mechanism and added to the account's blockchain. The ledger tracks the effects of every operation since the inception of the network, providing a reliable reference for nodes to maintain an accurate historical record.

The KeetaNet ledger maintains the state of the system and contains an up-to-date state for the following kinds of data:

Voting Power When an account balance is delegated to a representative through the "Set Representative" operation, the representative gains voting power. This voting power is currently determined by the total balance of the base token across all accounts that have delegated their funds to that representative.

Balances Within the ledger, balances are maintained on a per-token basis on each account and are maintained as an arbitrary large big integer.

Tokens Tokens may be created with the "Create Identifier" operation and are stored within the KeetaNet ledger. Each token has the supply (total number of minted tokens) and outstanding balance of that token (tokens which have been issued to accounts) maintained as state within the ledger.

Certificates Certificates are a specific type of metadata within the KeetaNet ledger which can be used to identify the user associated with an account. The certificate for an account is validated by the network validators to share the same public key as the account.

It is possible for permissions to reference a certificate authority as a way of permitting actors, such as regulated entities, to make decision based on the fact that a trusted party has verified certain properties of the associated user.

Metadata As a result of the "Set Info" operation, some basic user information may be set on an address for informational purposes.

Permissions KeetaNet has an extensive permission system, allowing for fine-grained control over accounts, tokens, and the network.

Blocks In order to maintain ordering, the blocks are recorded in the ledger. The blocks contain all the updates made to the ledger, as well as which accounts are being updated.

Votes Issued by ledgers as a side-effect of the ledger contents, votes are endorsements by a particular representative to insert a given block or set of blocks into their ledger. Votes come in two kinds: permanent and temporary. Only permanent votes can end up on the "main" ledger, but a representative must maintain all votes it issues, and so unpublished votes (whether permanent or temporary) will be stored in the "side ledger."

A compliant node will never issue two contradictory votes–that is, two votes that endorse separate sets of blocks to succeed other blocks on a chain that have overlapping lifetimes.

Votes are also uniquely identified by the tuple of their issuer and the vote serial number, ensuring that each vote can be traced and is distinguishable.

Side ledger The side ledger is a secondary area where unpublished votes and blocks are stored. All temporary votes will be unpublished, but the representative that issued them must keep track of them throughout the lifetime of the vote to avoid violating one of the core principles of the protocol – that representatives never issue competing valid votes.

A vote transitions from the "side ledger" to the "main ledger" when it is received over the peer-to-peer network and has enough permanent votes within its vote staple to reach consensus. Votes, and blocks which are referenced by those votes, on the side ledger are consulting the ledger to check for conflicting blocks, but only the main ledger is checked when validating that a block has a valid predecessor.

4.1.3 Communications

Voting Methodology Modeled after common opportunistic concurrency control (OCC13[23]) systems, KeetaNet's voting is "directed" by the client (as shown in Figure 2 on the next page). To publish a block or set of blocks, a client first requests temporary votes from a quorum of representatives, weighted by voting power; once these temporary votes are received, the client then requests a permanent vote from the subset of representatives who provided the temporary votes.

As noted in section 2.5.2, the voting power required for a quorum is adaptive. If a single representative holds a disproportionate amount of voting power, the quorum threshold increases, ensuring that more than one representative is needed to reach consensus.

This communication is achieved by connecting directly to each representative's API endpoints over HTTP for generating votes.

Distribution Once a set of blocks has been voted on and received permanent votes from a quorum of representatives, the resulting staple can be published to the entire network. To reduce spam, only staples may be published to the peer-to-peer network which distributes new information to all connected nodes. This communication happens over a WebSocket-based protocol.

A compliant node will only forward messages if it can incorporate them into its ledger (in the case of a representative) or if the message contains a staple with votes that meet the required voting power.

5 Core Algorithms and Protocols

5.1 Consensus algorithm

The KeetaNet system utilizes an enhanced delegated proof of stake (dPoS) consensus to ensure a balanced and healthy democratic voting system. Each account, including representatives, is associated with an X.509 certificate that verifies their identity, supports network security, and, when necessary, ensures compliance with KYC regulations. Importantly, no personally identifiable information (PII) is included in the certificate.

Account holders have the option to delegate their entire balance of the network's base token to a voting representative of their choice. The sum of all the delegated balances a representative has at a given time is referred to as their "weight." Representatives with more weight have more "voting power." The balance delegated to a representative remains in the delegator's account and can be moved freely. However, moving tokens may change the representative's weight.

5.2 Transaction validation

In KeetaNet, transaction validation is in the hands of its representatives. Acting as gatekeepers of what is permitted in the ledger, representatives ensure that all transactions comply with the defined rules and constraints of the ledger. Their role is not only passive validation, but active enforcement of the rules of the ledger as well as the overall health of the network.

Within KeetaNet, representatives collectively define the rules by which all participants can update the ledger.

Two-Phase Voting A unique feature of KeetaNet's transaction validation process is its two-phase voting mechanism. This process ensures a robust vetting mechanism, adding depth and rigor to the validation process. Not only does it filter out invalid transactions, but also it offers a buffer against malicious or contentious activities, ensuring that the KeetaNet blockchain system remains reliable and trustworthy. The two steps of the process, represented in Figure 2 on the following page, are as follows:

Temporary Votes When a transaction is initiated, it is not broadcast outright. Instead, the details of the transaction are conveyed to representatives over secure HTTPS unidirectional channels. In response, representatives issue temporary votes for the provided blocks based on their validity at the time they were received. This phase is crucial because it establishes a primary consensus on the transaction's

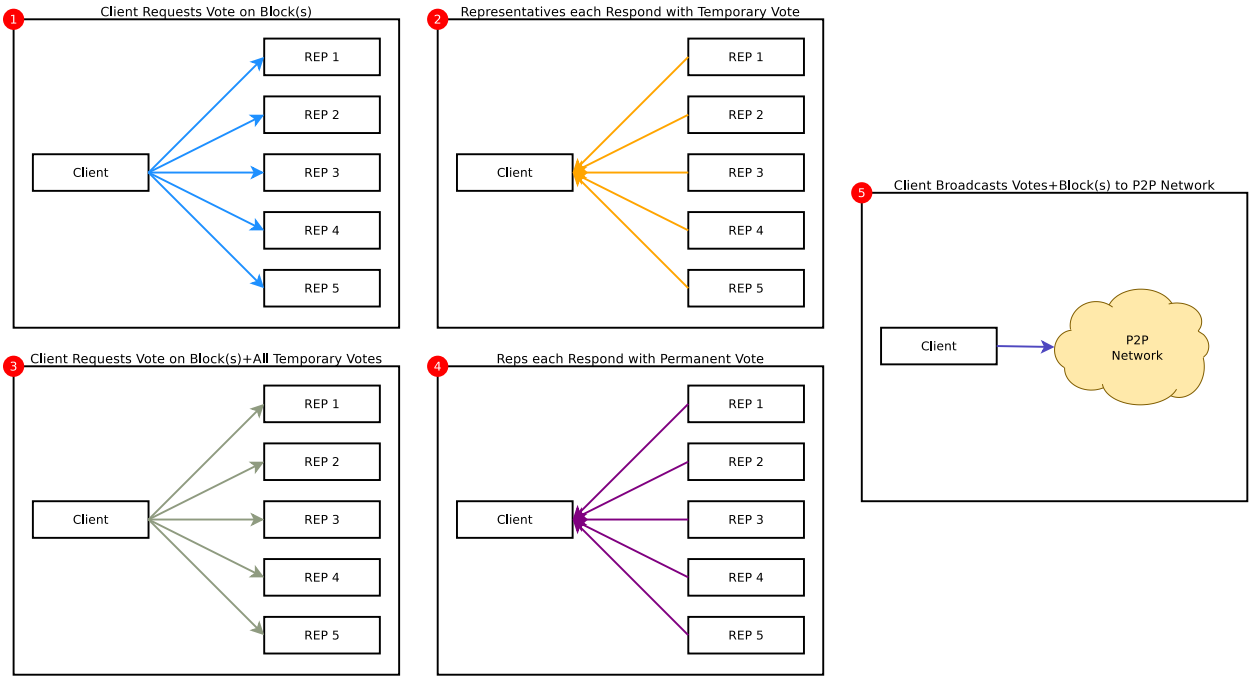

Figure 2: KeetaNet Voting Process

legitimacy. The main goal of this phase is to determine and establish the quorum.

Permanent Votes Upon obtaining a quorum of temporary votes, representatives that provided those initial endorsements can issue permanent votes for the transaction. The differentiation between temporary and permanent votes lies primarily in their duration. The key here is to ensure that a permanent vote is only cast after the quorum is determined through the temporary votes. This protects against premature and potentially erroneous approval of transactions. In particular, a compliant node will strictly avoid issuing conflicting votes (of any lifetime) for competing chain successors.

Publishing Once validated, the votes and blocks are broadcast to the peer-to-peer network as a consolidated package, called a vote staple. This gives the entire network a synchronized and transparent record. Nodes will only forward staples over the peer-to-peer network that have a quorum of permanent votes attached and that can be inserted into their own ledgers if appropriate.

5.2.1 Voting power

The weight assigned to a representative translates into their "voting power" within the network. This voting power is critical during consensus formation. The more voting power a representative has, the more influence they wield in network decisions, including transaction validation and system rule changes.

5.3 Data storage and retrieval

KeetaNet employs a sophisticated data storage model designed for high-performance read and write operations. This section provides details on how the system maintains data integrity while supporting concurrent operations.

5.3.1 Concurrency control

Concurrency control is the process of coordinating simultaneous access to the same data, and in the context of distributed systems or blockchain computing, it presents a multifaceted challenge. In these systems, where data is distributed across multiple nodes or servers, ensuring data consistency and preventing conflicts between simultaneous operations is critical.

Optimistic concurrency control Concurrency control in KeetaNet is managed through an Optimistic Concurrency Control (OCC) mechanism. Using this approach, transactions are initially assumed to be non-conflicting, permitting them to proceed without locks. Conflicts are detected and addressed after the fact, which reduces latency and increases throughput for the common case where no conflicts occurs.

Clients are responsible for managing conflicts and resubmitting transactions if necessary.

5.3.2 Consistency

Having consistent data when required is one of the most important aspects of any distributed system. Without this, it is impossible to prove integrity and trustworthiness. KeetaNet is able to provide different levels of consistency for different aspects of the network, ensuring maximum performance and integrity.

Fully Consistent Writes Writes in KeetaNet are fully consistent, achieved by obtaining a quorum of representatives based on voting power. This ensures that data is consistently replicated across the network.

Eventually Consistent Reads KeetaNet allows for eventually consistent reads by permitting reads from any representative. This optimizes for latency and is useful for non-critical data reads.

Fully Consistent Reads For critical operations, KeetaNet offers fully consistent reads by obtaining confirmation from a quorum of representatives based on their voting power. This ensures data integrity and consistency.

5.3.3 Ordering

KeetaNet implements partial ordering of transactions, meaning transactions are ordered only in relation to those required by the constraints of the ledger. This contrasts with most blockchain systems, which enforce total global ordering, where each operation is assigned a linear position relative to all others.

Partial ordering enables many transactions to be processed in parallel without conflicts, improving efficiency.

5.4 Data formats

5.4.1 Accounts

Accounts in KeetaNet refer to either the public key of a key-pair or a deterministically generated account. Every account is given an address, which is a representation of their public key or some other uniquely identifiable information. Each account has a separate ordered blockchain within the DAG to store that account's blocks.

Keyed accounts Keyed accounts are comprised of a private and public key-pair. They can digitally sign a block and are generally the only kind of account on other blockchains. On KeetaNet, these are the only accounts that can sign votes or blocks.

Generated accounts Generated accounts are special-purpose addresses generated deterministically from publicly available data. Unlike keyed accounts, they do not have the ability to sign transactions but serve other specific roles within the network. Generated accounts have an address which is deterministically derived from some fixed input, either from an operation within a block or from well-known information such as the network ID. Ownership permissions will automatically be granted to the creator over the generated account. To publish a block to the chain of a generated account, a keyed account with proper permissions must sign a block for the generated account. There are three types of generated accounts:

Network accounts Each network has exactly one Network Account. This account is generated from the unique numeric representation of that network. Network accounts are used to assign network-wide permissions. For example, to create a new token on the network, the creator must have that permission assigned to them on the network account. The base token for the network is also generated from this address.

Storage accounts Storage accounts are a versatile account type that can hold balances and are generally meant to be used as holding accounts for funds. They may be jointly owned or controlled by multiple accounts by setting the appropriate access control list (ACL) entries.

Token accounts Tokens act as identifiers for different transferrable currencies on the network. Administrators and owners of these accounts have the ability to modify the total supply, modify an entity's balance of the token, and grant/revoke a specific user's ability to use the token.

5.4.2 Operations

Operations in KeetaNet are high-level instructions to modify the state of the ledger. Operations are limited to a predefined set of instructions. For example, a "Send" operation is used to send a token from one account to another and a "Set Rep" operation is used to specify a delegated representative to vote on behalf of the account. These instructions can be expanded in the future to support additional instructions for other use cases.

Effects With any set of instructions, there is a desired result. When a node on the KeetaNet network validates or adds a set of blocks to its ledger, it first computes the results of the operations contained within those blocks. An effect represents either an update to or a requirement from the data in the ledger. For example, a "Send" operation produces two effects: it decrements the sender's balance and increments the recipient's balance for a given token.

Furthermore, the operation imposes certain requirements on the ledger data, such as ensuring that both parties have permission to use the token and verifying that the sender has sufficient balance to complete the transaction.

Defined operations KeetaNet has a set of predefined operations for an address to use. All operations are self-contained and do not have references to one another within a block.

See table 2 on the next page for a list of operations and a brief description of the operation as well as what parameters they accept.

5.4.3 Blocks

Blocks in KeetaNet are the fundamental mechanism by which the ledger is updated. Each block contains an ordered set of operations performed by a specific account. A block can include multiple operations from the same account, and its size may vary depending on the number of operations it holds.

Structure Blocks are encoded in ASN.1 DER[24] which is a flexible container as well as versioned – both of these features enable the block to be extensible in the future since the container format will support a wide variety of types and can represent any backwards-incompatible changes with a version change. The block is dynamic in size, so it can be as large as the representatives will accept.

See table 3 on page 20 for a breakdown of what is in a block as well as Listing 1 on page 33 for the full ASN.1 schema of blocks.

5.4.4 Votes

Votes in KeetaNet are encoded as X.509[25] certificates. X.509 was chosen because it is a widely-used and well-known standard (RFC5280) and contains a flexible data structure using ASN.1. Representatives use a vote as a means to communicate their intention to add a set of blocks to their ledger. Depending on the timeframe for which an issued vote is valid, it is either considered a temporary or a permanent vote.

Contents Each vote contains the following:

- An issuer - The account that signed the vote
- Serial - An arbitrary integer defined by the issuer which is suggested to be incrementing, but can be any unique value. Two separate votes with the same issuer and serial will not be accepted by the network
- Blocks - A set of block hashes for which the issuer vouches the validity
- Starting Time - The timestamp when the vote was issued
- Ending/Expiry Time - The timestamp of when the vote expires and should not be considered in quorum anymore
- Signature - The issuer's cryptographic signature on the data, confirming that they were the one who issued the vote

Validity A vote is considered valid for a representative in the voting process if it is not expired, the serial has not been seen before, and the set of blocks is valid.

5.4.5 Staples

Inspired by OCSP Stapling[26, § 8], Staples–also known as Vote Staples–create a permanent voting context by bundling multiple blocks together for simultaneous voting and publication to the network. This process enhances the efficiency of the voting and validation stages, resulting in higher throughput.

Staples are the fundamental unit of exchange within KeetaNet, functioning as journaled updates to the system. If any block within a staple is found to be invalid, the entire staple is rejected and must go through the voting process again, excluding the invalid operation(s).

By grouping blocks and votes together, stapling increases efficiency and reduces the need for the network to locate multiple pieces of data for the same operation. A staple is constructed from at least one

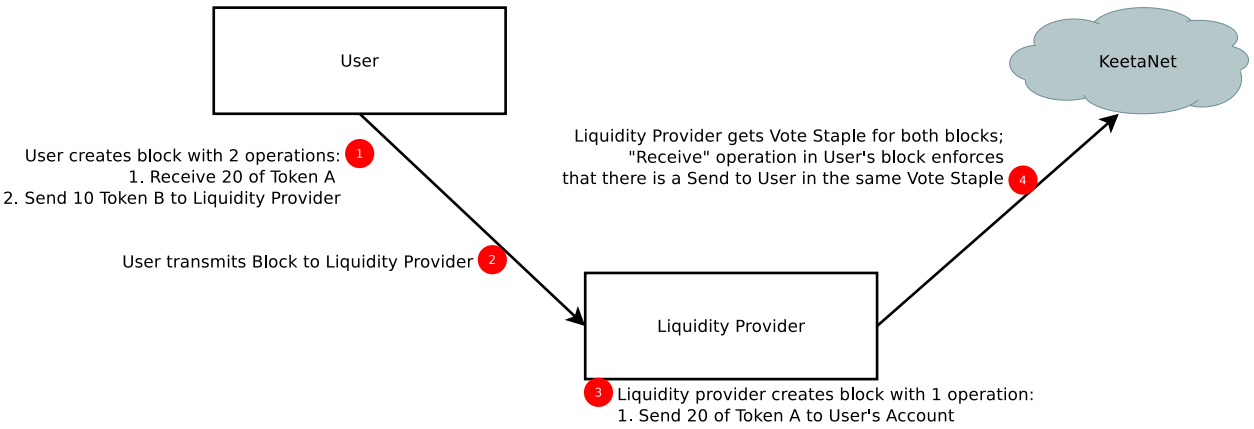

Figure 3: Native Exchange Visualized

Name	Arguments	Description
SEND	to, amount, token	Transfers “amount” of “token” to the recipient (“to”)
SET REP	to	This operation takes in a representative address and delegates the sender’s weight to that representative.
SET INFO	name, description, metadata, defaultPermissions	SET INFO updates an address’s name, description, metadata, and default permissions (if it is from a generated identifier account).
MODIFY PERMISSIONS	principal, target, permissions	Modifies permissions of “principal” with the entity being the sender
CREATE IDENTIFIER	type	Creates a new identifier
TOKEN ADMIN SUPPLY	method, amount	Modifies the supply of a token
TOKEN ADMIN MODIFY BALANCE	token, amount, method	Modifies the balance of a user for the specified token
RECEIVE	from, amount, token, forward	Requires a certain amount received previously within the Vote Staple to be valid

Table 2: KeetaNet Operations

Field	JSON Type	ASN.1 Tag Type	Description
version	integer	INTEGER	The block specification version (version 1 described here)
date	string	GENERALIZED TIME	ISO-8601 UTC formatted date for the block
previous	256 bit hex-char string	OCTET STRING	Previous block hash for the (account's) chain
account	string	OCTET STRING or NULL	String representation of the account public key + key type
signer	string	OCTET STRING	String representation of the signer public key + key type
signature	512 bit hex-char string	INTEGER	ECDSA or Ed25519 512-bit signature
network	string	INTEGER	ID of the network
subnet	string	INTEGER or NULL	ID of the subnet
operations	array	SEQUENCE	Array list of operations and their contents

Table 3: KeetaNet Block Structure

block and at least one vote, and all votes within a single staple must correspond to the same blocks, in the same order, as the staple was constructed. Furthermore, a staple cannot mix temporary and permanent votes.

To maximize efficiency, the network uses compressed staples. The more similar the contents of the included blocks and votes, the more the staple can be compressed.

5.5 Auxiliary functions

5.5.1 Bootstrapping

The bootstrapping protocol exists to help a new node or a node that has potentially fallen behind the rest of the network bulk-fetch changes to the ledger that it may have missed (for instance, if it was offline). It works on the HTTP protocol by talking to representatives and requesting a set of staples that occurred after a given timestamp. This timestamp is compared to the votes within the staple for reference. In addition, a Bloom filter[27] is used to prevent staples that have already been sent to the requester from being included.

5.5.2 Permissions

Two types of permissions are used in KeetaNet: "base permissions" and "external permissions." Base permissions are represented by a symbolic name that corresponds to a specific value. External permissions are not managed by KeetaNet and can hold arbitrary flags (offsets) managed by an external party.

Encoding Both base and external permissions are encoded using a bit-field of offsets, enabling fast, compact bitwise operations for combining and checking permissions. This approach also makes subsequent modifications highly efficient.

Access Control List On KeetaNet, permissions are stored and represented alongside the information that describes their use. An access control list (ACL) entry will contain the following fields.

- Principal - The address identifying the actor on the network who is accessing an entity
- Entity - The network address being granted/denied access and the address of the chain where ACL modifications are applied
- Target - An optional address that further narrows the scope of the permission
- Permissions - The list of base and/or external permissions granted, both represented as bit fields

Permission Hierarchy Permissions on KeetaNet are always read from most to least specific, with the default if none are set being empty. Permissions do not inherit from level to level. If one is defined, it will override the less specific entry.

1. ACL entry that matches the principal + entity + target exactly
2. ACL entry that matches the principal + entity exactly, but does not include a specific target
3. (If the entity is able to) The default permission set by the entity
4. If none of the above are found, the permissions are assumed to be empty.

For example, if a storage address grants the ability for one user "SEND_ON_BEHALF" with no target, that user will be able to send any token from the storage account. If the storage account then creates an ACL entry for the same user with a specific token as a target not including "SEND_ON_BEHALF" that user will still be able to send any token, just excluding the specific token from which permissions were removed.

Base Permissions The base permissions each have a symbolic name and use case defined by KeetaNet. Each symbolic name is tied to a specific offset to be used in the bit field and a specific use on the network. Each type of base permission is listed in Table 4 on the following page, along with a brief description, what type of addresses they can apply to, and if they can be used as a default permission.

External Permissions External permissions are not managed by KeetaNet, but can store arbitrary flags controlled by an external party. External parties can set these to arbitrary values using the same bit-field format as their counterpart. Within the network, each offset is not associated with a symbolic name and is always represented as a bit field. If an external party wishes to use symbolic names for these permissions, a client-side method is available to facilitate this representation.

Ownership The "OWNER" base permission represents the ownership of an address and is only applicable to generated account identifiers. For other accounts, ownership is tied to the holder of the private key. Upon account creation, the creator is automatically granted the "OWNER" permission. From that point on, there must always be exactly one address with the "OWNER" permission. Any modifications to ownership must occur within the same vote staple to ensure the count remains equal to one. Here are some examples of valid and invalid ownership modifications for a storage account:

- Invalid Ownership Modification
  - The owner of a storage account signs a block granting a different address the OWNER permission, leaving two owners
  - The owner of a storage account signs a block lowering their own permissions to ADMIN, leaving no owners
  - The owner of a storage account performs both the correct addition and removal, but performs the action in two vote staples, making each one invalid alone
- Valid Ownership Modification
  - The owner of a storage account signs a block in which the new owner is granted the OWNER permission and the previous owner is re-assigned a bit field not including OWNER.

Base Flag Name	Description	Entity	Principal	Target	Target Description	Can Be Default
ACCESS	A generic flag, if any other flag is set, this one must also be true	Any	Any	Any		Yes
OWNER	Full control over an account	Generated Account	Keyed Account	Never		No
ADMIN	Can do everything owner can do, except modify the owner	Any	Keyed Account	Never		No
PERMISSION DELEGATE ADD	Can add a subset of its permissions to other accounts	Any	Keyed Account	Any	Address it can modify permissions for	No
PERMISSION DELEGATE REMOVE	Can remove a subset of permissions to other accounts	Any	Keyed Account	Any	Address it can modify permissions for	No
SEND ON BEHALF	Can SEND on behalf of entity	Any	Keyed Account	Token		No
STORAGE CAN HOLD	Storage can hold targeted token	Storage	Token	Never		Yes
STORAGE CREATE	Can create storage identifiers	Network	Keyed Account	Never		Yes
STORAGE DEPOSIT	Can deposit into storage account	Storage	Any	Token	Which token can be deposited	Yes
TOKEN ADMIN CREATE	Can create token identifiers	Network	Keyed Account	Never		Yes
TOKEN ADMIN MODIFY BALANCE	Can modify the balance of the token for another address	Token	Keyed Account	Any	Address it can modify balance for	No
TOKEN ADMIN SUPPLY	Can increase/decrease the token supply	Token	Keyed Account	Never		No
UPDATE INFO	Ability to use SET_INFO on behalf of entity	Any	Keyed Account	Never		No
MANAGE CERTIFICATE	Ability to manage certificates on behalf of entity		-	-	-	-

Table 4: KeetaNet Permissions

Delegation On KeetaNet, the "PERMISSION_DELEGATE_ADD" and "PERMISSION_DELEGATE_REMOVE" flags grant the principal the ability to delegate permissions to other addresses for the entity from which the permissions were originally granted. Both flags function similarly, but each represents a different "AdjustMethod" that is being used in the "MODIFY_PERMISSIONS" block operation. A principal with either of these flags can only add or remove a subset of the permissions they hold on the same entity. They cannot re-grant delegation permissions (i.e "PERMISSION_DELEGATE_ADD" and "PERMISSION_DELEGATE_REMOVE") unless they are an admin of the entity.

5.5.3 Account certificates

Certificates allow the network to be in regulatory compliance by providing a transparent and standardized identification of participants.

Certificates are implemented in the Keeta network to authenticate and verify the identity of a node or account, but no PII (Personally Identifiable Information) is stored on the ledger. The Keeta network utilizes X.509 certificates. Certificates have a public key to verify that they exclusively belong to a specific account and are digitally signed by a Certificate Authority (CA) to verify that the certificate is legitimate. Nodes add their root certificate as part of their configuration. The signature on a certificate is similar to the signature on a block. X.509 certificates can be signed and verified using the same algorithms as supported accounts (Ed25519 and ECDSA).

Accounts do not have certificates by default, but may be required to have a certificate to complete certain operations.

5.6 Network Initialization

An "initial trusted account" is configured on the network to securely initialize the ledger. It has exclusive permission to create valid votes while there is no delegated weight on the network. The role of this account is crucial for the network's initialization, as it provides initial account funding and delegates weight to representatives. Any vote staples published while there is zero weight must include a vote by this account. Additionally, this account bypasses permission requirements for the opening blocks on the network and the base token's chain. This exception allows the necessary permissions to be granted for these chains, ensuring a smooth network startup.

6 Security Measures

Security in a blockchain system spans multiple factors, from how data is encrypted and stored to the safeguards against malicious attacks. Securing a decentralized network involves addressing a unique set of challenges. To meet these demands, KeetaNet employs a robust security strategy that combines proven protocols, advanced cryptographic techniques, data integrity measures, and protections against common attack vectors.

6.1 Cryptographic methods used

Digital Signatures KeetaNet currently supports 3 different cryptographic algorithms for performing digital signatures, but is extensible to support additional algorithms, as well as deprecating algorithms in the future should the need arise.

The currently supported algorithms are:

- ECDSA[28] with secp256k1[29, § 2.4.1]
- ECDSA[28] with secp256r1[29, § 2.4.2]
- Ed25519[28]

Hashing When a cryptographic hashing algorithm is employed SHA3-256[30] is used.

6.1.1 Use of cryptosystems

KeetaNet uses digital signatures for digitally signing blocks and votes, and cryptographic hashing for referencing blocks.

6.1.2 Post-Quantum Cryptography

KeetaNet is extensible to support additional cryptographic algorithms and can be migrated to fully support post-quantum cryptography (PQC), including deprecating all algorithms which are not post-quantum cryptography.

6.2 Data integrity

Ensuring data integrity is fundamental to the operation and trustworthiness of any blockchain system. In the context of KeetaNet, where transactions are validated through a meticulous two-phase voting process, the preservation of untampered data becomes even more vital.

Append-only Ledger Once a transaction has been validated and appended to the blockchain, it becomes an immutable record. This means that it cannot be altered or deleted without a consensus from a quorum of the network's representatives. This feature ensures that historical data remains consistent.

Cryptographic Hashing KeetaNet employs SHA3-256 hashing for its records. This cryptographic hash function ensures that any alteration to a record will result in a change to its hash, making tampering easily detectable.

Chain Consistency Each block in the KeetaNet blockchain includes a reference to the previous block via its cryptographic hash. This chaining mechanism guarantees that blocks are correctly ordered. Any attempt to modify a block would alter its hash and propagate the change to all subsequent blocks, making unauthorized modifications immediately apparent.

Use of TLS for Authentication and Integrity KeetaNet leverages Transport Layer Security (TLS) not only for encryption of data in transit but also for authentication and message integrity. By enforcing HTTPS for all communications, the platform ensures that messages exchanged between participants are securely delivered and reliably authenticated, confirming that the sender and receiver are indeed who they claim to be, and that the transmitted data has not been altered in transit.

By embedding strong authentication and integrity assurances at the communication layer, KeetaNet establishes a trustworthy foundation critical to its overall security model and essential for the platform's widespread adoption.

6.3 Measures against common attack vectors

6.3.1 Sybil attack

One of the predominant concerns in decentralized systems is the Sybil attack, where a single adversary controls multiple nodes on the network, effectively trying to subvert the network's functionality. Such an attack can disrupt honest nodes from achieving consensus or facilitate malicious activities.

To counteract this, KeetaNet supports the use of X.509 certificates for certifying endpoints for representatives. This certification process ensures that each representative on the network can be authenticated and can be trusted. X.509 certificates provide a standardized way of verifying the identity of participants and tying them to an identity verified by the public key infrastructure (PKI), making it logistically challenging for an attacker to create a significant number of Sybil nodes since it would need to subvert the certification process.

By relying on the trusted certification process, not only does KeetaNet dramatically reduce the potential for Sybil attacks, but it also establishes an added layer of trust among participants. This approach ensures that network nodes represent unique, authenticated entities, effectively fortifying the network's resilience against such threats.

6.3.2 Spam attack

In the decentralized environment of blockchain systems, spam attacks often manifest as an influx of legitimate yet superfluous transactions. These transac-

tions, while valid in their structure, are intentionally designed to flood the network, causing bottlenecks, delays, and inefficiencies.

KeetaNet employs a strategic approach to counter such spamming tactics. Firstly, the network's design incorporates a two-phase voting process (see Section 5.2), which acts as an initial filter to mitigate the volume of these transactions. However, in situations where an actor is persistent in dispatching a large number of genuine but unnecessary transactions, representatives on KeetaNet have the agency to respond.

Representatives can observe transaction patterns and, upon identifying an attempt to spam the network, have the discretion to adjust transaction fees. By ratcheting up these fees in response to abnormal transactional activity, KeetaNet introduces a financial deterrent. This increased cost makes it prohibitively expensive for malicious entities to continue their spamming efforts. Furthermore, representatives can also choose to decline voting for these transactions entirely, effectively blocking them from being added to the blockchain. This dual-layered approach ensures that KeetaNet remains resilient against transactional spam, ensuring smooth operations and preserving network integrity.

6.3.3 Denial of service attack

A Denial of Service attack aims to render a service unavailable by overwhelming it with traffic or exploiting specific vulnerabilities. In blockchain contexts, DoS attacks can severely hamper network operations, affecting all users connected to the network.

KeetaNet's design incorporates preemptive measures against DoS attacks. The aforementioned two-phase voting process not only helps against spam attacks but is also effective in mitigating the impact of DoS attacks. By utilizing HTTPS in the voting process, KeetaNet can employ existing DDoS and DoS prevention mechanisms to safeguard the network such as cloud-based anti-DDoS solutions.

Additionally, representatives observing abnormal traffic or suspicious patterns indicative of a DoS attack can start imposing fees on suspected malicious actors. This proactive stance not only helps to minimize the impact of DoS attacks but also empowers the representatives to maintain the network's integrity actively.

7 Scalability

7.1 Problem areas

Validation architectures: Chains versus DAGs One of the key areas where many blockchains face challenges is their underlying validation architecture. Blockchains like Bitcoin and Ethereum use chain structures where each block is added to a single, linear chain. This structure, while secure and easy to validate, does not scale well due to the serialized nature of block addition.

Blockchain systems based on directed acyclic graphs (DAGs), such as KeetaNet, on the other hand, allow for more flexibility and parallelism. Each new transaction only needs to be ordered with respect to the transactions it references–usually just a single predecessor transaction. This is a more scalable solution but introduces complexities around transaction ordering and finality.

Transaction distribution: The "mempool" bottleneck In many traditional blockchain systems, message distribution mechanisms, such as the mempool, present challenges that hinder the scalability and determinism of the network. These systems often rely on the "waiting room" concept, where transactions await confirmation and inclusion in a block. However, this approach complicates the design of scalable, event-driven architectures that rely on a deterministic state machine model.

For example, in widely used blockchain systems like Bitcoin and Ethereum, the mempool serves as a temporary holding area for transactions that have been broadcast to the network but not yet added to a block. While this mechanism allows nodes to collect and disseminate transactions before they are written into the blockchain, it introduces a significant problem.

The key drawback of using a mempool is its probabilistic nature. Transactions in the mempool can be evicted by nodes before being included in a block,

leading to uncertainty about whether a transaction will ever be confirmed. This introduces additional complexity and reduces the system's determinism, making it more difficult to build scalable and efficient software for blockchain networks.

Protocols: Bespoke approaches Existing blockchain systems generally use bespoke protocols to carry message traffic between users and nodes and between nodes. While these bespoke protocols are often designed to optimize node efficiency, they limit the ability to integrate with existing third-party software or existing tools that are not specifically built to support them. This issue is especially pronounced in cases where the protocol is built on lower-level, less common transport layers such as UDP, which can further complicate compatibility and interoperability. KeetaNet builds upon HTTP and can take full advantage of many HTTP scaling solutions.

7.2 Scalability solutions

Validation architecture The KeetaNet blockchain design and implementation is a hybrid approach that incorporates elements of several different consensus and distributed state machine designs. One of the key distinguishing features of KeetaNet is that there is no single, monolithic blockchain. Each account owner has their own, essentially isolated blockchain. Due to this feature, KeetaNet has similarities to Directed Acyclic Graph (DAG) based blockchains, but it is not implemented as a DAG. It is a "virtual DAG" design. Links across individual account blockchains are not necessarily explicit, as would occur in a true DAG design, rather they are implicit by the atomic transaction based persistence model used for adding blocks and adjusting balances.

Transaction distribution The KeetaNet blockchain system uses "client directed" transaction distribution, where blocks that have not reached quorum are not exposed to the wider network but are limited to the set of nodes which provide voting services on the network. Because the client requesting votes talks directly to the representatives providing votes this forms a single pipeline where event processing is linear instead of happening at some unspecified time in the future which could lead to liveness faults as well as user frustration.

Additionally, because only fully accepted blocks are broadcast this reduces the amount of traffic on the most vulnerable part of the distribution system– the peer-to-peer (P2P) network.

Protocols KeetaNet uses HTTP[31] as the primary mechanism for performing consensus-related actions. Regular HTTP "POST" messages are used to request votes (see section 5.1 on page 14 for more details on voting), and WebSockets[32] are used for peer-to-peer traffic. HTTP is a very common protocol, being the core protocol for the Web and has broad support for creating scalable applications using cloud computing providers.

8 Use Cases

The KeetaNet blockchain provides an excellent base layer for a future cryptocurrency economy. Designed to meet the demands of scalability, security, flexibility, and adaptability to diverse regulatory environments, KeetaNet's core attributes are optimized for a sustainable, civilization-scale solution for representing and transferring value over the coming decades.

As a cryptocurrency system, KeetaNet provides out-of-the-box, reliable and foundational tools, such as native tokens and administration tools, which would typically be added as afterthoughts in other systems. With KeetaNet, these core capabilities are integrated from the start, ensuring reliability and efficiency.

Additionally, businesses and government regulators are increasingly requiring controls to be implemented to safely interact and do business with blockchain technologies. Here too, KeetaNet is well-equipped to meet these demands, offering scalable solutions that cater to global regulatory regimes and evolving regulatory landscapes.

8.1 Real-world scenarios where KeetaNet can be applied

8.1.1 Cryptocurrency Central

KeetaNet's native support for tokenization (as discussed in section 4.1.2 on page 13) with built-in support for atomic swap between tokens (as briefly discussed in section 5.4.2 on page 18) as well as providing for arbitrary metadata (as discussed in section 4.1.2 on page 13) positions KeetaNet as a versatile platform for interconnected services.

Since KeetaNet represents the highest throughput blockchain system, and supports the features discussed in this paper, it is an ideal candidate to act as a "layer 2" for many other cryptocurrencies. KeetaNet allows for high throughput, low-latency transfers of wrapped tokens with a high degree of certainty due to non-repudiation mechanisms (such as certificates, as discussed in section 4.1.2 on page 13). Furthermore, by "wrapping" other currencies as KeetaNet tokens, and advertising availability for atomic swap as well as compliance-based on-boarding and off-boarding using metadata, KeetaNet is an ideal "network of networks."

8.1.2 Blockchain Banking

Due to its support for the primitives needed to operate in a highly regulated environment while preserving privacy, KeetaNet is an ideal candidate for a future blockchain-based banking system.

KeetaNet's permissioning model enables the enforcement of rules such as limiting transactions to users with appropriately issued certificates, making it possible to regulate who can transact in a given token. By leveraging KeetaNet's permissioning system, it is possible to create stablecoins that represent fiat currencies, backed and operated by regulated financial institutions.

This functionality paves the way for the emergence of blockchain-based banking systems, fostering greater adoption of cryptocurrencies and positioning KeetaNet as a key player in the evolution of the global financial ecosystem.

8.1.3 Digital Identity

As a consequence of supporting identity verification using PKI, KeetaNet enables decentralized, user-controlled verification of individual attributes without requiring a single centralized authority. This decentralized approach clearly defines trust boundaries: trust is established directly between users and certificate issuers, with users maintaining control over which private information they disclose.

By using a combination of public-key encryption with ECIES[33] and salting-based hashing, it is possible to encode the user's private attributes into their certificate. These attributes can then be proven to have been verified by the verifying certificate authority (e.g. any number of selected or approved KYC providers), without revealing any private information about the user to the public.

This also enables non-financial participants to use KeetaNet to prove specific details about the user - for example, that their email address has been validated. This can be used to build systems which require some identity management without the cost of each service needing to perform their own validation, and avoiding the all-too-common dangers posed by reliance on centralized repositories of personal information.

Certificates associated with an account must share the same private key as certified in the certificate. However, KeetaNet does not impose restrictions on who can issue certificates nor does it define specific trust anchors within the PKI ecosystem. Establishing trust anchors and managing issuer validation are explicitly left to higher-level protocols and user-defined policies, ensuring transparency about trust boundaries within the system.

8.2 Comparative advantage of KeetaNet

KeetaNet offers a range of advantages that set it apart from not only blockchain systems but a wide array of technology platforms, making it well-suited for diverse use cases. As a global ledger with integrated permissions and a robust capabilities model, KeetaNet's modular and flexible architecture provides the following benefits:

- Scalability: Leveraging an enhanced Delegated

Proof of Stake (dPoS) mechanism, KeetaNet is highly scalable, ensuring efficient performance even as the network grows.

- Security: With digital identity features and fully consistent writes, KeetaNet ensures secure transactions.
- Flexibility: Multi-token support and robust ability to be updated make it adaptable for various applications.
- Global Governance: Its decentralized nature allows for global participation while still adhering to local laws through token-level governance.
- Cost-Efficiency: By leveraging existing cloud infrastructure and providing options for transaction fees in various tokens, it can be more cost-effective and cost-predictable.
- Interoperability: The ability to partition the network into distinct, interoperable subnets allows for greater flexibility and collaboration between diverse parties.
- Regulatory Compliance: Built-in support for digital identity and sanctioning mechanisms make it easier to comply with laws and regulations.

By addressing these critical needs, KeetaNet is positioned to not only revolutionize digital currencies, but has the potential to transform any domain requiring secure, scalable, and governed data management.

9 Benchmarks and Performance Metrics

9.1 Tests conducted

Maximum TPS Throughput Test

This test is designed to measure the best-case throughput of KeetaNet and validate that the system can scale linearly with hardware utilization. It employs a network topology in which the sending node generates permanent votes locally, with only a single voting representative. Each block contains 1,000 simple "send" transactions. The goal of this test is to ensure that KeetaNet's design and implementation can handle increased workloads efficiently while maintaining linear scalability.

It has been conducted multiple times with different node configurations and the results have been compiled into Table 5 on the following page.

9.2 Comparison with existing technologies

As noted in table 1 on page 8, KeetaNet significantly outperforms other blockchain systems in terms of throughput and does not sacrifice latency while doing so.

9.3 Future tests

In the future, we plan to perform benchmarking in various other additional configurations. They are described in Table 6 on the following page.

10 Conclusion

In the evolving digital landscape, KeetaNet represents a significant advancement in blockchain technology, designed to address key challenges that have hindered the scalability, security, and flexibility of traditional blockchain systems. By incorporating cutting-edge features such as high throughput, advanced cryptographic methods, and robust digital identity management, KeetaNet provides a highly scalable and secure foundation for a range of applications, from cryptocurrencies to decentralized finance (DeFi) and identity verification.

What sets KeetaNet apart is its focus on modularity and adaptability. The system's support for multi-token capabilities, permissioned governance, and interoperable subnets allows for seamless integration with existing infrastructure while also enabling new, innovative use cases. This flexibility makes it an ideal platform not only for blockchain applications, but also for industries requiring secure, scalable, and compliant data management solutions.

Date	Cloud Provider	Ledger Database	Max TPS
06/10/2022	AWS	DynamoDB	2M
06/20/2022	AWS	DynamoDB	3.5M
07/12/2022	GCP	Spanner	13M

Table 5: Maximum TPS Throughput Testing Results

Name	Representatives	Other Nodes	Transactions per Block	Blocks per Staple
Basic	5	0	1000	1
Real World	5	10	1 to 10	1 to 2
Large Network	5	30	1	1

Table 6: Future Testing Configurations

Furthermore, KeetaNet's design ensures resilience against centralization risks, with its adaptive voting mechanisms, robust security protocols, and customizable metadata and certificates. The network's ability to support regulatory compliance and provide verifiable methods for cryptographically secure identity verification opens the door to new opportunities for blockchain-based banking, secure digital identities, and compliance with varying global regulations.

10.1 Future work and extensions

Going forward, there are several exciting avenues for the development and deployment of KeetaNet:

- Enhanced Consensus to deal with skewed network voting power distributions
- Formal verification of the protocol and documentation of the state machine
- Additional support for smart contracts so that applications can be built to run directly on KeetaNet
- Post-Quantum Cryptography support as it becomes available
- More efficient bootstrapping and checkpointing systems
- Improvements to the peer-to-peer protocol
- Publish the "KeetaNet Anchor" specification, which:
  - Allows for interoperability between clients on KeetaNet to provide and consume common services (Banking, FX, Inbound, Outbound, and Cards)
  - Documents the X.509v3 Certificate Extensions used for identifying users securely and privately

A References

A.1 Endnotes

[1] R. C. Merkle, "A digital signature based on a conventional encryption function," in *Advances in Cryptology — CRYPTO '87*, C. Pomerance, Ed., Berlin, Heidelberg: Springer Berlin Heidelberg, 1988, pp. 369–378, ISBN: 978-3-540-48184-3.

[2] E. A. Brewer, "Towards robust distributed systems," in *Proceedings of the Nineteenth Annual ACM Symposium on Principles of Distributed Computing*, ser. PODC '00, Portland, Oregon, USA: Association for Computing Machinery, 2000, p. 7, ISBN: 1581131836. DOI: 10.1145/343477.343502.

[3] N. Szabo. "The idea of smart contracts." (1997), [Online]. Available: https://nakamotoinstitute.org/the-idea-of-smart-contracts/.

[4] J. Poon and T. Dryja. "The Bitcoin lightning network: Scalable off-chain instant payments." (Jan. 2016), [Online]. Available: https://lightning.network/lightning-network-paper.pdf (visited on 09/01/2023).

[5] J. Bier, *The Blocksize War: The Battle Over Who Controls Bitcoin's Protocol Rules*. Independently published, 2021, ISBN: 979-8721895609.

[6] E. Lombrozo, J. Lau, and P. Wuille. "Bip 141: Segregated witness (consensus layer)." (Dec. 2015), [Online]. Available: https://github.com/bitcoin/bips/blob/master/bip-0141.mediawiki.

[7] S. Sinclair and D. Canellis. "Bitcoin mempool surges point to new life for world's first blockchain." (Mar. 2023), [Online]. Available: https://blockworks.co/news/bitcoin-mempool-surges-new-life-for-blockchain.

[8] "Scaling." (Apr. 2023), [Online]. Available: https://ethereum.org/en/developers/docs/scaling/.

[9] D. Robinson and G. Konstantopoulos. "Ethereum is a dark forest." (Aug. 2020), [Online]. Available: https://www.paradigm.xyz/2020/08/ethereum-is-a-dark-forest.

[10] E. Mikalauskas. "280 usd million stolen per month from crypto transactions." (Feb. 2023), [Online]. Available: https://cybernews.com/crypto/flash-boys-2-0-front-runners-draining-280-million-per-month-from-crypto-transactions.

[11] P. Wackerow. "Miner extractable value (mev)." (Jun. 2023), [Online]. Available: https://ethereum.org/en/developers/docs/mev.

[12] L. Lamport, R. Shostak, and M. Pease, "The Byzantine generals problem," *ACM Transactions on Programming Languages and Systems*, vol. 4, no. 3, pp. 382–401, 1982. DOI: 10.1145/357172.357176.

[13] M. J. Fischer, N. A. Lynch, and M. S. Paterson, "Impossibility of distributed consensus with one faulty process," *ACM*, vol. 32, no. 2, pp. 374–382, Apr. 1985, ISSN: 0004-5411. DOI: 10.1145/3149.214121.

[14] S. Nakamoto, *Bitcoin: A peer-to-peer electronic cash system*, 2008. [Online]. Available: https://bitcoin.org/bitcoin.pdf.

[15] "The merge." (Aug. 2023), [Online]. Available: https://ethereum.org/en/roadmap/merge/ (visited on 09/01/2023).

[16] "Hackers are stealing more cryptocurrency from DeFi platforms than ever before." (Apr. 2022), [Online]. Available: https://www.chainalysis.com/blog/2022-defi-hacks/.

[17] Diem Team, "DiemBFT v4: State machine replication in the Diem blockchain," Diem Association, Tech. Rep., Aug. 2021. [Online]. Available: https://developers.diem.com/papers/diem-consensus-state-machine-replication-in-the-diem-blockchain/2021-08-17.pdf.

[18] R. Gelashvili *et al.*, *Block-STM: Scaling blockchain execution by turning ordering curse to a performance blessing*, Aug. 2022. arXiv: 2203.06871 [cs.DC].

[19] M. Baudet, G. Danezis, and A. Sonnino, *FastPay: High-performance byzantine fault tolerant settlement*, Nov. 2020. arXiv: 2003.11506 [cs.CR].

[20] G. Danezis, E. K. Kogias, A. Sonnino, and A. Spiegelman, *Narwhal and Tusk: A DAG-based mempool and efficient BFT consensus*, Mar. 2022. arXiv: 2105.11827 [cs.CR].

[21] "Sui FAQ: Design complexity." (Jul. 2023), [Online]. Available: https://docs.sui.io/learn/sui-compared#design-complexity (visited on 09/01/2023).

[22] "What is Tendermint." (May 2021), [Online]. Available: https://docs.tendermint.com/v0.34/introduction/what-is-tendermint.html (visited on 09/01/2023).

[23] H. T. Kung and J. T. Robinson, "On optimistic methods for concurrency control," *ACM Trans. Database Syst.*, vol. 6, no. 2, pp. 213–226, Jun. 1981, ISSN: 0362-5915. DOI: 10.1145/319566.319567. [Online]. Available: https://doi.org/10.1145/319566.319567.

[24] ITU-T, "ASN.1 Encoding Rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER)," ITU-T, Standard, 2002. [Online]. Available: http://www.itu.int/ITU-T/studygroups/com17/languages/X.690-0207.pdf.

[25] ITU-T, "Information technology – Open Systems Interconnection – The Directory: Public-key and attribute certificate frameworks," ITU-T, Standard, 2020. [Online]. Available: http://www.itu.int/rec/T-REC-X.509.

[26] D. E. Eastlake 3rd, *Transport Layer Security (TLS) Extensions: Extension Definitions*, RFC 6066, Jan. 2011. DOI: 10.17487/RFC6066.

[27] B. H. Bloom, "Space/time trade-offs in hash coding with allowable errors," *Commun. ACM*, vol. 13, no. 7, pp. 422–426, Jul. 1970, ISSN: 0001-0782. DOI: 10.1145/362686.362692.

[28] L. Chen, D. Moody, A. Regenscheid, and A. Robinson, "Digital signature standard (DSS)," en, NIST, Standard, Feb. 2023. DOI: 10.6028/NIST.FIPS.186-5.

[29] D. R. L. Brown, "SEC 2. standards for efficient cryptography group: Recommended elliptic curve domain parameters," SECG, Tech. Rep., Jan. 2010. [Online]. Available: http://www.secg.org/sec2-v2.pdf.

[30] M. Dworkin, *SHA-3 standard: Permutation-based hash and extendable-output functions*, Aug. 2015. DOI: 10.6028/NIST.FIPS.202.

[31] H. Nielsen *et al.*, *Hypertext Transfer Protocol – HTTP/1.1*, RFC 2616, Jun. 1999. DOI: 10.17487/RFC2616.

[32] A. Melnikov and I. Fette, *The WebSocket Protocol*, RFC 6455, Dec. 2011. DOI: 10.17487/RFC6455.

[33] "Ieee standard specifications for public-key cryptography," *IEEE Std 1363-2000*, pp. 1–228, 2000. DOI: 10.1109/IEEESTD.2000.92292.

A.2 Nomenclature

AML	Anti-Money Laundering – A set of regulatory measures and processes that financial institutions and other regulated entities implement to detect and prevent money laundering activities
ASN.1	Abstract Syntax Notation One – Specification for defining schemas, defined by ITU X.690.
AWS	Amazon Web Services – A cloud computing provider owned by Amazon
BFT	Byzantine Fault Tolerance – The property of a distributed system that allows it to continue functioning correctly, even if some components (or nodes) fail or act maliciously, as long as the number of faulty nodes is below a certain threshold
CAP	Consistency, Availability, and Partition Tolerance – an "iron triangle" for which a distributed system must optimize between
DAG	Directed Acyclic Graph – A type of graph in which edges are directed but never form any closed loops
DER	Distinguished Encoding Rules – Rules for encoding ASN.1 data into bytes where each encoded element has nominally only a single way of being represented
DLT	Distributed Ledger Technology – A type of ledger built upon distributed systems
dPoS	Delegated Proof of Stake
EcDSA	Elliptic Curve Digital Signature Algorithm – a mechanism for creating digital signatures using very efficient key sizes
Ed25519	Elliptical Curve-based digital signature algorithm using the Edwards-curve and specifc curve parameters
FLP	Fischer, Lynch, and Paterson – Authors of the FLP Impossibility theorem
HTTP	Hyper Text Transport Protocol – a very common protocol for transporting messages over TCP/IP
KYC	Know Your Customer – A regulatory principle that a regulated entity is responsible for understanding who they are doing business with
OCC	Opportunistic Concurrency Control; As opposed to a system like multiversion concurrency control, provides transactional semantics but by default assumes every transaction is conflict free
OID	Object Identifier – An ASN.1 type that supports a hierarchical approach to assigning identifiers based on a federated model
ORV	Open Representative Voting – Nano's name for its style of dPoS consensus where users can delegate any representative to vote on their behalf
PII	Personally identifiable information – sensitive information which can be used to identify a specific person
PKI	Public Key Infrastructure; The infrastructure required to validate the identity of entities using X.509 certificates

PQC — Post-quantum cryptography – a set of cryptographic algorithms that are expected to be safe from being broken given sufficiently powerful quantum computer systems

secp256k1 — Specific parameters for EcDSA, defined in SEC 2

secp256r1 — Specific parameters for EcDSA, defined in SEC 2; Also known as NIST P-256 or prime256v1

TCP — Transmission Control Protocol – The most common Internet Protocol; Stream oriented and used by many other protocols; Known for reliable and ordered delivery of bytes within a stream

UDP — User Datagram Protocol – A common Internet Protocol; Packet oriented and largely offloads the semantics to the application; Compare with TCP

XRP — Ripple's cryptocurrency

A.3 Schemas

Listing 1: KeetaNet Block Schema

```
Keeta DEFINITIONS ::= BEGIN
  AdjustMethod ::= INTEGER {
    add(0),
    remove(1),
    set(2)
  }

  BlockOperation ::= CHOICE {
    -- Send operation
    send [0] SEQUENCE {
      -- Destination account to send to
      to          OCTET STRING,
      -- Amount of the token to send
      amount      INTEGER,
      -- Token ID to send
      token       OCTET STRING,
      -- External reference field (optional)
      external    UTF8String OPTIONAL
    },

    -- SET_REP operation
    setrep [1] SEQUENCE {
      -- Representative to delegate to
      to          OCTET STRING
    },

    -- SET_INFO operation
    setinfo [2] SEQUENCE {
      -- Name to specify for this account
      name              UTF8String,
      -- Description to specify for this account
      description       UTF8String,
      -- Metadata to specify for this account
      metadata          UTF8String,
      -- Default permission to specify for this account (optional)
      defaultPermission SEQUENCE {
```

```
      -- Base permissions, which the network verifies
      base        INTEGER,
      -- External permissions, which the network
      -- does not verify (for example, for use in
      -- smart contracts)
      external    INTEGER
    } OPTIONAL
  },

  -- MODIFY_PERMISSIONS operation
  modifypermissions [3] SEQUENCE {
    -- Principal to modify permissions for
    principal    OCTET STRING,
    -- Method to modify permissions for
    method       AdjustMethod,
    -- Permissions to modify, as a bitfield
    permissions  CHOICE {
      -- Permissions to set
      value SEQUENCE {
        base        INTEGER,
        external    INTEGER
      },
      -- If no permissions are required
      none NULL
    },
    -- Target to modify permissions for
    target      OCTET STRING OPTIONAL
  },

  -- CREATE_IDENTIFIER operation
  createidentifier [4] SEQUENCE {
    -- Identifier to create, this must match
    -- the deterministic identifier which is
    -- generated from the account, blockhash,
    -- and operation index
    identifier  OCTET STRING
  },

  -- TOKEN_ADMIN_SUPPLY operation
  tokenadminsupply [5] SEQUENCE {
    -- Amount of chagne to the supply
    amount      INTEGER,
    -- Method to modify the supply
    method      AdjustMethod
  },

  -- TOKEN_MODIFY_BALANCE operation
  tokenmodifybalance [6] SEQUENCE {
    -- Token to modify the balance of
    token       OCTET STRING,
    -- Amount to modify the balance by
    amount      INTEGER,
    -- Method to modify the balance
    method      AdjustMethod
  },

  -- RECEIVE operation
```

```
    receive [7] SEQUENCE {
      -- Amount to receive
      amount      INTEGER,
      -- Token to receive
      token       OCTET STRING,
      -- Sender from which to receive
      from        OCTET STRING,
      -- Whether the received amount must match
      -- exactly (true) or just be greater than or
      -- equal to the amount (false)
      exact       BOOLEAN,
      -- Forward the received amount to another
      -- account (optional)
      forward     OCTET STRING OPTIONAL
    }
  }

  Block ::= SEQUENCE {
    -- Version of this block
    version        INTEGER { v1(0) },
    -- Network ID
    network        INTEGER,
    -- Subnet ID
    subnet         CHOICE {
      -- Subnet ID
      subnetID      INTEGER,
      -- Null if no subnet (mainnet)
      null          NULL
    },
    -- Date of the block
    date           GeneralizedTime,
    -- Signer's public key
    signer         OCTET STRING,
    -- Account ID
    account        CHOICE {
      -- Account ID
      accountID     OCTET STRING,
      -- Null if account is signer
      null          NULL
    },
    -- Previous block hash
    previous       OCTET STRING,
    -- Operations in this block
    operations     SEQUENCE OF BlockOperation,
    -- Signature of the block
    signature      OCTET STRING
  }
END
```

Version 513253dbef757c4ea22c7cb0e2a3db21449a569e