NEAR Protocol: Mobile-native blockchain

NEAR Protocol

September 14 2018

1 Executive Summary

While blockchain systems are mostly praised for the trustless environment they bring, they have many other advantages compared to traditional execution environments.

For developers, blockchain provides the infrastructure that currently requires months to set up: partition resilient networking, always available database and compute, ease of GDPR compliance, and reduced need for local regulatory compliance. Additionally, entrepreneurs can come up with entirely new business models for businesses built on top of the blockchain.

A situation in which a single entity can shut down an entire cluster that is responsible for monitoring time-sensitive events [1] is not possible on blockchain.

The adoption of blockchain for modern applications has been slow, however. The primary reason for that is an inherent scalability issue and lack of user-friendliness for both developers and consumers.

Scalability is presently table stakes. No new protocol can be released without delivering finality time on the order of seconds and thousands transactions per second. However, just processing more transactions per second is not enough. To be future proof, the system needs to scale with the size of the network, meaning that computation needs to be carried out across the network in parallel. With growth of adoption, the size of the information that needs to be stored in the system grows rapidly and it is unfeasible for any node in the system to store the entire state after some point in time.

Near Protocol is a sharded decentralized computing platform that targets low-end devices. Both processing of transactions and the state are sharded, making Near Protocol future-proof in terms of both compute and storage. Enabling low-end devices to participate in the operation of the network allows significantly more nodes to participate, bringing higher potential throughput and lower centralization of the system.

Building sharded systems is an extremely hard engineering problem. Even outside of the blockchain world, there are likely only around a dozen production-tested sharding implementations in the world. The NEAR Protocol core team previously built one of these, which included sharding and distributed transactions, at MemSQL[2], a distributed database that has large cluster deployments powering analytics at Uber, Pinterest, Goldman Sachs, Akamai and other large corporations.

2 Why NEAR Protocol

Blockchain so far has found only limited usage. Developers, users and enterprises are now educated about the existence of such a platform but most are not yet ready to use it. Scalability, availability, price and security through decentralization are key dimensions that people typically consider along the path to adoption.

Scalability is a pressing issue that has attracted a lot of attention in existing projects like Bitcoin and Ethereum. Many incumbents try to address the transactions per second constraint with approaches that are just temporary fixes or constant-factor improvements on the status quo. Given the extreme

[1] https://medium.com/@serverpunch/why-you-should-not-use-google-cloud-75ea2aec00de

[2] https://memsql.com/

complexity of the design of such systems, updating an existing ecosystem with a more novel design is extremely hard.

Even though Bitcoin and Ethereum compete on their level of decentralization, they actually suffer from many centralization issues. Bitcoin, for example, has 53% of its mining power controlled by just three pools. On top of that, running mining nodes currently requires expensive hardware, which increases barriers to entry and reduces the incentives for nodes to join over time.

An alternative approach is to use the Proof of Stake consensus, which doesn't require the wasted energy and custom chip design of Proof of Work. Unfortunately, current implementations of this, such as those used by EOS or Steemit, end up even more centralized.

NEAR Protocol brings together the mobile-first nature of the modern world, industry proven sharding design and priority on end user and developer experiences.

By connecting millions of mobile devices into the network and using a new way of selecting witnesses to run consensus (named "Thresholded Proof of Stake"), NEAR enables a much more decentralized blockchain. The protocol, which shards both the state of the network and the processing of the transactions, scales capacity linearly with the number of nodes and provides a future-proof platform for decentralized applications.

Even though there are early adopters currently building decentralized applications on other protocols, it's a very small community. NEAR's go-to-market strategy is to go after developers who are not currently building these applications and who may not even be aware of the potential of blockchain platforms by directly addressing the pain points of current platforms:

1. Business models. Large corporations are monopolizing the market and limiting the business models of new entrants.

2. Platform risk. Twitter and Facebook have closed their platform before. Google Cloud and AWS have declined service or rolled out competing products. Other platforms have been acquired by large corporations and shut down.

3. The liability of data. Regulations like GDPR and the ongoing threat of hacks make custody of customer and user data into a liability.

4. Scaling and availability. Many platforms offer services to scale up applications, but they are still complex in configuration and not transparent in usage.

Historically, the most successful developer platforms have also offered a channel to connect with end users. Current blockchain solutions are very limited in their ability to offer user acquisition channels because they don't run natively where the users spend their time. NEAR Protocol, by virtue of running natively on the mobile phones of end users, allows developers to directly access these users through the platform.

3 Key Features

3.1 Sharding

A future-proof protocol must shard both state and processing. With significant adoption of the platform, no single machine would be capable of storing all the information on the chain (let alone the entire history) or verifying all of the transactions.

A lot of recent sharding research in the blockchain community separates transactions into intra-shard and cross-shard categories, optimizing for the former and providing a much slower solution for the latter. The NEAR Protocol assumes that transactions will touch multiple shards by default, which is the likely behavior for arbitrary smart contracts.

The NEAR Protocol uses an approach similar to MapReduce [5] by effectively executing transactions as a sequence of map/shuffle steps, maps are processed in parallel on many nodes, and shuffle steps are moving transaction between shards. For example, a smart contract that processes a large number of payments would first debit all of the accounts in a single parallel map step, then shuffle the information about credits that need to be applied across the network, and finally apply all the credits in another map step.

Performing effective atomic cross-shard transactions via map, shuffle and reduce operations has been a well-researched topic in the database community, though it has very few actual implementations due to high engineering complexity. MemSQL continuously delivers thousands of atomic cross-shard transactions with shuffles per aggregator node in a production environment. Having three early core MemSQL engineers in the team uniquely positions NEAR to build a practical implementation of atomic cross-shard MapReduce transactions at scale.

On top of the per-shard chains, the NEAR Protocol maintains a checkpoint blockchain. Approximately every minute, 1000 witnesses agree on the previous checkpoint and the recent epoch blocks for a particular shard before multisigning a checkpoint block (see Consensus 3.3 for more details of the process). This chain maintains overall information about management of the network (witnesses, verifiers, etc) and allows any node to quickly jump into any shard or catch up with the network.

3.2 Mobile Friendliness

The NEAR Protocol is designed so that a node operating hardware as low-level as a commodity Android phone can participate in the operation of the network. This is enabled by two core properties of the protocol: (1) Proof-of-Stake consensus, in which no entity needs to perform any expensive computation to produce a block; and (2) Sharding by State, in which nodes only need to store a small portion of that state.

Enabling operation of the network from mobile devices provides multiple advantages:

1. Users participating as nodes in the network already have an application on their phone that is a gateway to the decentralized apps in the network, effectively serving as an extra distribution channel;

2. Enabling mobile devices to fully participate in the network significantly increases the number of potential nodes, making it possible to scale the network to millions or even billions of devices, significantly reducing centralization and increasing the throughput of the network;

3. Using a decentralized application on a phone that also runs an operating node reduces the latency of that app and improves user experience.

Besides the preliminaries of Proof-of-Stake consensus and Sharding by State, few other components need to be redesigned to make the system run on phones.

Phones are not always available. Running complex operations that require continuous network communications would both deplete the battery and saturate expensive network access plans. The system needs to tolerate a large majority of nodes which only actively participate at times of their choice (e.g. when the phone is connected to WiFi and is plugged into a charger). We assume, however, that the majority of participants can perform computations for a short period of time with low network overhead at least a few times a day.

Phones run on slower network. That means that the low-end devices cannot participate in approving latency-critical transactions. In most places the latency of 3G network is within 500ms, and the latency of less widespread 4G network is within 100ms. Given that at least a few network round trips are necessary for a node to confirm a transaction, the participants of the network will be able to opt for either faster finality times by only waiting for large nodes with fast network to confirm transactions or higher security by waiting for a few seconds until a sufficient number of nodes running on low-end devices send their confirmations.

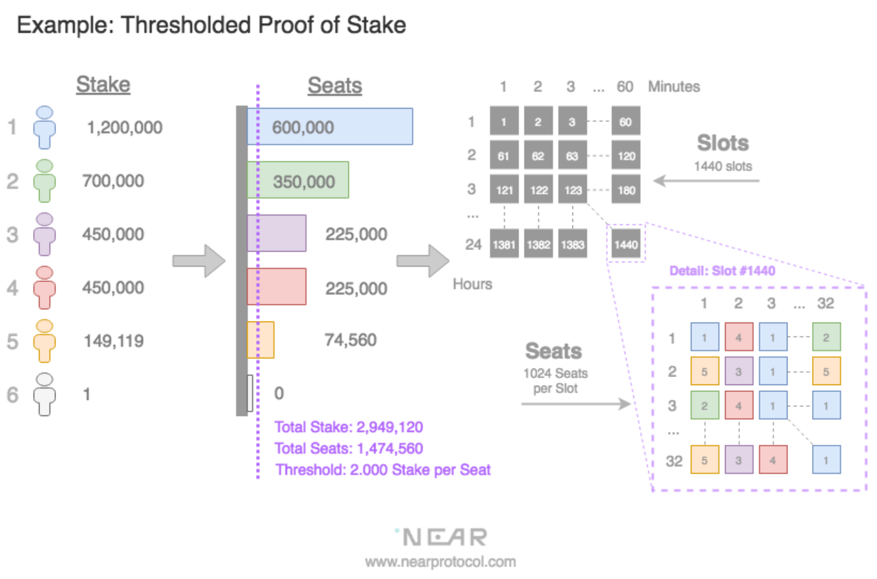
3.3 Consensus

Many proposed sharding implementations where blocks are created independently in each shard suffer from reduced security since the number of participants working on each block is divided by the number of shards. This means that fewer nodes need to be corrupted to undo, censor, reorder or otherwise tamper with transactions.

The NEAR Protocol addresses this by decoupling transaction verification and block signing activities. Participants that verify transactions (verifiers) are split into shards, and each verifier only executes and validates transactions in their shard.

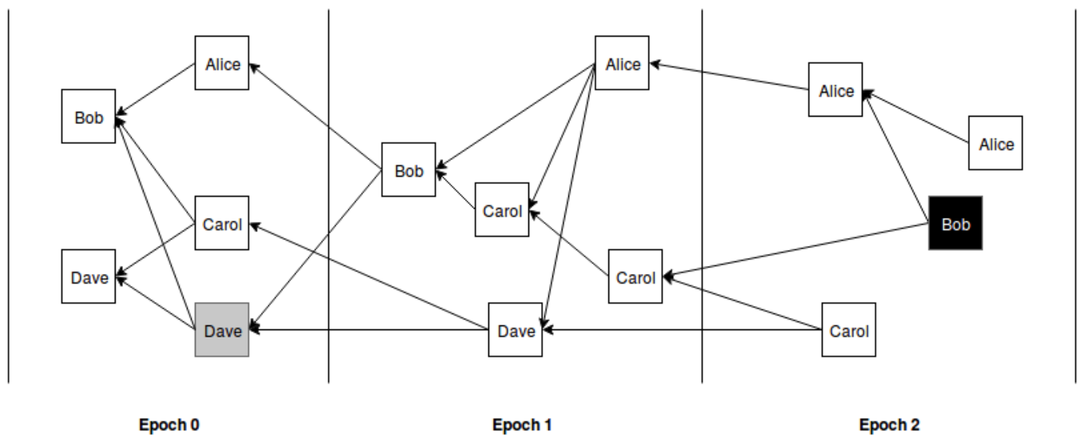
Participants who sign blocks (witnesses) create global blocks that encompass transactions from all the shards. Once a block is created, it is irreversible. Thus, in order to corrupt a block, an attacker needs to corrupt all witnesses not just those assigned to a certain shard.

The NEAR Protocol creates one block roughly every minute and that block is signed by 1,024 witnesses. These witnesses are rotated every minute, resulting in the need to fill 1,474,560 different witness seats over the course of a day. Witnesses are not necessarily different entities. If one entity stakes 10x more than another, it will have 10x more instances throughout the day to participate as a witness, some of which might end up in the same block.



To sign the block and make it irreversible, the witnesses need to reach a byzantine consensus on it. Many algorithms have been proposed for this task, though none has yet been production tested on a permissionless system yet. The NEAR Protocol uses its own algorithm called TxFlow both in each shard by verifiers and across shards by witnesses, which leaves a record of all the interactions between verifiers, and allows transaction validation to be postponed.

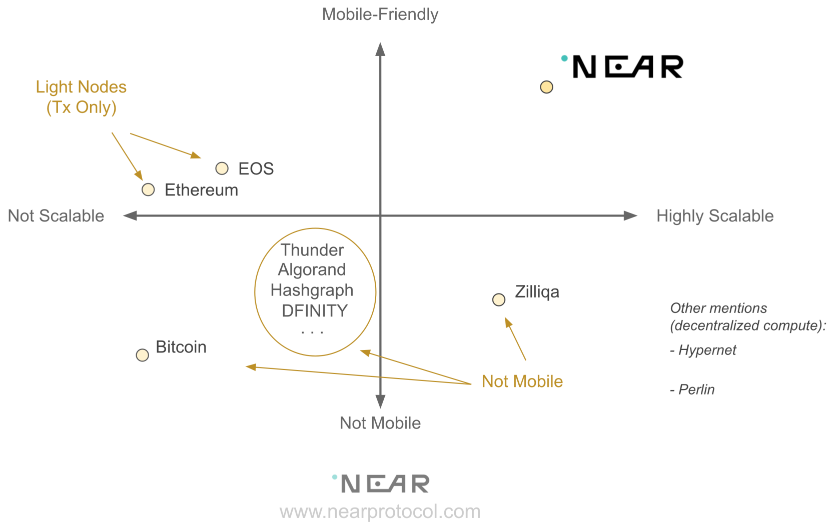
In essence, TxFlow maintains a Direct Acyclic Graph (DAG) of messages built by verifiers. A verifier first receives some number of transactions from other verifiers. They then post a message that includes those transactions plus a list of the edges between that message and all of the messages the given verifier previously received from other verifiers. A directed edge from message `A` to message `B` indicates that `A` directly approves `B`. `A` also indirectly approves all messages that are reachable from messages it directly approves.



Messages are split into epochs. The message `B` published by a verifier `w` has epoch `X` if and only if it approves the message with epoch `X - 1` from this verifier as well as messages with epoch `X - 1` from more than 2/3 of all the verifiers. For each epoch, we assign one of the verifiers to be responsible to indicate when it ends (e.g. when they publish a message with the next epoch). This message is then used by everybody to construct the set of transactions for given epoch that are then passed for verification and processing by all nodes in the given shard.

Verifiers are ordered with deterministic random order and that order is used to select which witness is responsible for given epoch. If a given witness is late or adversarial, the transactions for this epoch will be automatically rolled into the next epoch when a subsequent witness will publish the next epoch message.

4 Comparison



Ethereum [3], the most popular distributed application platform, was a revolution in distributed computing. Unfortunately, the platform can only process 14 transactions per second and it takes around 12 seconds for a transaction to appear on the chain. Ethereum has proposals to address the scalability[3] problem by introducing sharding and checkpoints with byzantine agreement among witness nodes but these proposals are years from implementation.

Numerous other approaches address some of the core scalability problems of first-generation blockchains. EOS [2], Algorand [6] and Thunder [10] all have a higher rate of transaction throughput and a lower

[3]https://notes.ethereum.org/SCIg8AH5SA-O4C1G1LYZHQ

time before each transaction is included on the chain, but none of them has linear scalability, which only delays the moment that they can no longer sustain the load. EOS and Thunder also rely on a very small subset of nodes to carry out the operation of the network (21 participants that know each other in EOS and just one participant in Thunder), significantly reducing the decentralization of the system.

Zilliqa [12], another scaling project, uses "network sharding", i.e. sharding of transaction processing. By assigning each smart contract to a separate shard, their network can process different smart contracts on separate shards in parallel. This offers linear scaling of processed transactions in the system overall but suffers from problems with large smart contracts. Most of today's platforms are dominated by 5-10 massive applications (Facebook, Google, YouTube, etc) that use up the vast majority of the network resources. Zilliqa's approach to sharding means that each smart contract will be limited by the throughput of a single shard. Additionally, their approach doesn't address state sharding and requires periodic proof of work, which leads to high hardware requirements for network participation and actively disincentivizes low-end nodes.

Perlin [8] and HyperNet [7] are systems designed for distributed computation. They use the blockchain to schedule and pay for computation jobs. The computation itself happens off chain. This fits use cases like machine learning or scientific computing which require only minimal validation of the results from trustless nodes to be effective. For smart contracts or money transfers which require a more complete level of security and validation for every transaction, this is unsuitable.

4.1 Consensus

Prior to designing TxFlow, we implemented and tested other readily available protocols and concluded that none of them would work in a production environment without modifications. Here we provide a very brief overview.

1. **Snowball / Avalanche** [11] is a fair probabilistic consensus algorithm based on participants building a DAG of transactions. When conflicting transactions are observed, each node queries a sample of other participants to decide on which transaction to choose. This method is not applicable when transactions execute arbitrary Turing-complete program, as there is no clear definition for conflicting transactions. While the paper provides certain proofs for a synchronous setting, it is unclear if Avalanche still maintains its guarantees in an asynchronous setting. Also, Avalanche leaves no trace of the process of achieving the consensus, making it hard to identify malicious nodes.

2. **Hashgraph** [1] is a fair gossip-based consensus algorithm that leaves a full trace of all inter-node communication, which makes it trivial to reveal malicious behavior and provides a higher level of certainty that the consensus was reached. However, *strongly seeing*, the core concept of Hashgraph, requires $O(n^2)$ time to compute per hashgraph event (where n is the number of participants). This effectively makes the complexity of simply running the algorithm, ignoring the network and transactions validation (and assuming the lower bound on the number of events in the Hashgraph is $O(n \log n)$) to be at least $O(n^3 \log n)$, which for 1000 witnesses means several seconds of processing is spent just to reach the consensus. Again, this ignores network, the actual work of validating transactions and signing the block, and other steps of the Hashgraph, such as computing famous witnesses and the order, which have similar complexities.

3. **PBFT** [4] is an unfair algorithm originally designed for a small group of participants reaching the consensus. In the permissionless setting it suffers from both too much control in the hands of the leader, and poor scalability. ByzCoin and OmniLedger [9] propose an extension that provides better scalability in an optimistic case, but still resort to the slow path in case a critical mass of participants goes offline during the consensus. Moreover, optimizations provided in ByzCoin and OmniLedger incentivize the leader to start the consensus with the smallest number of witnesses sufficient to reach a byzantine consensus in order to reduce the chance of any of them going offline, which effectively promotes censorship.

5 Team

The NEAR Protocol core team comprises three ex-MemSQL engineers (including engineers #1 and #4) who built sharding and distributed transactions for the database, and three ex-Google engineers. The engineering team has five finalists of ACM ICPC (the largest world-level collegiate programming competition), including three gold medalists and one two-time champion. Only six people in the World have won ACM ICPC twice.

Alexander Skidanov Alex started his professional career at Microsoft in 2009, then joined MemSQL in 2011 as Engineer #1, where he worked for 5 years as Architect and Director of Engineering. He is a 2x ICPC medalist (gold in 2008 and bronze in 2005).

Illia Polosukhin Illia is an ICPC finalist and has more than 10 years of industry experience, including 3 years at Google where he was a major TensorFlow contributor and a manager of the team building question answering capabilities for the core Google search.

Evgeny Kuzyakov Evgeny is an ICPC gold medalist (2008) who began his professional career at Google in 2011, started at Facebook in 2015 and left in 2017 to work on Machine Learning and Blockchain research.

Mikhail Kever Mikhail was a senior software engineer at MemSQL for more than 4 years. He is also one of only 6 people to become a two-time ICPC World champion.

Dr. Maksym Zavershynskyi Maksym has more than 10 years of industry experience, most recently at Google where he worked on applied Machine Learning.

David Stolp David is an ICPC finalist and early MemSQL engineer. He worked at MemSQL for five years and left in 2017 to concentrate on Blockchain opportunities.

Bowen Wang Bowen graduated from University of Chicago with a Masters in Programming Languages and Machine Learning.

Erik Trautman Erik is an entrepreneur who started 5 years on Wall Street in 2007 and left to create an open source community which grew to over 100,000 members. He led Viking Education as CEO until its acquisition in 2017.

Aliaksandr Hudzilin Aliaksandr has a business degree from UC Berkeley. In industry, he worked in investment banking under Frank Quattrone and spent 4 years selling and marketing MuleSoft and MemSQL products.

6 Roadmap

The latest roadmap of the project can be found at `https://nearprotocol.com/#roadmap`.

7 Future Research and Development

Success in blockchain protocol development hinges on having the fastest rate of innovation and pace of execution in the industry. NEAR is committed to including the most successful approaches from across the space and actively pursuing additional areas of research as part of the development process. We will track research proposals and their progress after TestNet launch on our website at `https://nearprotocol.com`.

References

[1] Leemon Baird. The swirlds hashgraph consensus algorithm: Fair, fast, byzantine fault tolerance," https://www.swirlds.com/downloads/swirlds-tr-2016-01.pdf.

[2] block.one. Eos.io technical white paper v2. `https://github.com/EOSIO/Documentation/blob/master/TechnicalWhitePaper.md`.

[3] Vitalik Buterin. Ethereum: A next-generation smart contract and decentralized application platform. `https://github.com/ethereum/wiki/wiki/White-Paper`, 2013.

[4] Miguel Castro and Barbara Liskov. Practical byzantine fault tolerance. In *Proceedings of the Third Symposium on Operating Systems Design and Implementation*, OSDI '99, pages 173–186, Berkeley, CA, USA, 1999. USENIX Association.

[5] Jeffrey Dean and Sanjay Ghemawat. Mapreduce: Simplified data processing on large clusters. In *Proceedings of the 6th Conference on Symposium on Opearting Systems Design & Implementation - Volume 6*, OSDI'04, pages 10–10, Berkeley, CA, USA, 2004. USENIX Association.

[6] Yossi Gilad, Rotem Hemo, Silvio Micali, Georgios Vlachos, and Nickolai Zeldovich. Algorand: Scaling byzantine agreements for cryptocurrencies. In *Proceedings of the 26th Symposium on Operating Systems Principles*, SOSP '17, pages 51–68, New York, NY, USA, 2017. ACM.

[7] HyperNet. Hypernet protocol: Distributed high-performance computing powered by blockchain technology," https://hypernetwork.io/hypernetwhitepaper_v1.1.pdf.

[8] Kenta Iwasaki. Perlin: Scalable dag-based distributed ledger protocol using avalanche consensus," https://ipfs.io/ipfs/qmxnjxntxfdg5vwbixudsugpfatakock1qjzqknujzrp37.

[9] Eleftherios Kokoris-Kogias, Philipp Jovanovic, Linus Gasser, Nicolas Gailly, Ewa Syta, and Bryan Ford. Omniledger: A secure, scale-out, decentralized ledger via sharding. Cryptology ePrint Archive, Report 2017/406, 2017. `https://eprint.iacr.org/2017/406`.

[10] Elaine Shi Rafael Pass. The thunder protocol. `https://docs.thundercore.com/thunder-whitepaper.pdf`.

[11] Team Rocket. Snowflake to avalanche: A novel metastable consensus protocol family for cryptocurrencies," https://ipfs.io/ipfs/qmuy4jh5mgnzvlkjies1rwm4yuvjh5o2fyopnpvywrrvgv.

[12] The Zilliqa Team. The zilliqa technical whitepaper" https://docs.zilliqa.com/whitepaper.pdf.