



NEUTRON

WHITEPAPER

Conceptualise For A Heterogeneous
Multi-Chain Network

CONTENT

Chapter 1 | Introduction 3

Chapter 2 | Design Concept 5

Chapter 3 | Consensus Mechanism 22

Chapter 4 | Underlying Technology 28

Chapter 5 | Release Notes 34

Chapter 6 | Neutron Partners 37

Chapter 1

Introduction

01

Introduction

Neutron (NTN) has a very unique view of the universe.

Atom is the smallest particle in chemical changes, and also the smallest unit for an element to maintain its chemical properties. However, magnifying atoms to countless times will find another universe. The atomic model is like the solar system. The nucleus is the sun, and the electrons surrounding the nucleus are like planets, revolving in different orbits. Atom is like a system of micro-celestial bodies. As it zooms in, there will be a system of celestial bodies coming out. Decomposing the atom again, it can become neutrons, protons and electrons, and its core element is the neutron.

The micro world is the foundation of the macro world. No matter how small things are, there are microcosms of the universe. Seeing the big in the small, the small part will get to know the trend of development, infinitely, boundlessly. Since neither the cosmic galaxy nor the atomic world is infinite, are there any infinite things in this world?

Yes, the blockchain space. The blockchain world cannot be dominated by a main chain. It must be a world where multiple chains coexist and multiple currencies are in parallel, and must achieve interconnection and mutual access to stimulate the greatest power, just like neutrons, protons and electrons exist individually but relying on each other, one at the outside and the other one stays inside, one is moving and another one stays static, opposing each other and complementing each other. It is Neutron's ultimate ideal to integrate the entire "independent" blockchain world into a complete ecosystem.

Neutron is the core element of all atoms and the most fundamental constitution and characteristic of the underlying technology of blockchain. In Neutron's concept, "coin" can be separated from the "chain". There is a core technology in Neutron that enables tokens to be transferred across blockchains and provides a cross-chain aggregation protocol for free trading of native assets. Through the deployment of "aggregators + asset cross-chain pools" on networks of different public chains mode, to provide users with a "one-click" cross-chain asset exchange experience.

The future blockchain must be interconnected, just as everyone thought that computers must be interconnected when computers were born, and whoever can push the technology forward is the star of tomorrow, and Neutron will take the first step, leading to this unbounded and infinite multidimensional space!

NEUTRON Global Blockchain Research Office

Chapter 2

Design Concept

Overview

The success of an open source ecosystem, decentralized file sharing, and public cryptocurrency has inspired and understood that decentralized Internet protocols can fundamentally improve the social and economic infrastructure. We have already seen a blockchain application with expertise, such as Bitcoin (cryptocurrency), Ethereum's mass smart contract platform, and countless distributed applications based on EVM (Ethereum Virtual Machine), such as Augur (forecast Market) and TheDAO (Investment Club). It can be seen that there has been a blowout of blockchain projects, and more and more companies and developers have joined the wave of blockchain. There are also endless types of chains on the market, extending from finance to health care and supply chains, asset management and traceability, everyone is establishing its own ecosystem, and different public chains may form competitive advantages in local areas through their virtue characteristics.

However, these blockchains have exposed various defects. Most of the on-chain applications or smart contracts cannot cross the boundaries of the current chain, and cannot cooperate with other chains to realize the circulation of value, which limits the scope of the blockchain. Behind the gratifying growth trend, these public chains and their ecosystems are like the civilizations distributed on all continents before the great geographical discovery. There are different civilizations size, limited to their own living space and development direction, and lack of compatibility with other blocks. The ability of chain projects to interconnect and communicate has become an isolated network system, which cannot cooperate with other

Design Concept

Therefore, the ideal solution is to allow multiple parallel blockchains to interoperate while maintaining their security features. Because of this, Neutron was born. Neutron has a new blockchain cross-chain architecture, based on the DEFI atomic ecological open application chain, provides a new blockchain cross-chain architecture, encourages crossing the multi-chain boundary + breaking through the 10,000-coin restriction, and supports the data and asset circulation and contract calls between multiple homogeneous and heterogeneous blockchains. This is a valuable interconnected network with potential, in which assets are issued and controlled by different validators, and can realize seamless transfer and transaction of cross-chain assets without relying on a third party that needs to be trusted. Consensus algorithms, communication protocols, and distributed high-credibility storage mechanisms that support large-scale and high-frequency transactions toward an acyclic graph enable the integration of virtual assets in the entire industry, promote the release of maximum value, and truly achieve integration.

At the same time, through this new multi-chain interactive protocol, Neutron can realize free cross-chain transactions of native assets on different chains, allowing one currency to aggregate another currency to form interactive drainage and circulation, just like a materialized aggregator: let a molecule contact with another molecule, to promote internal atomic reorganization to form macromolecules, to reshape chemical relations and energy, should cover fission reactors, fusion reactors, fission fusion hybrid reactors, and convert virtual assets into fuel to generate new kinetic energy, and then virtual assets of the entire industry carry out the price reactor, build a decentralized technology application layer for impact and dilution, exchange token nuclear anti-applications, and increase the value of virtual assets through the self-sustaining chain nuclear fission process, release digital energy to the extreme, and generate grand financial explosive power. Build ecological consensus, reshape value potential, and truly promote multi-chain ecological applications and interactive flow and re-tracing of value.

Partitioned cross-chain network

Here we present Neutron, a novel blockchain network architecture that addresses all of these problems. Neutron is a network of many independent blockchains, called zones. The zones are powered by Tendermint BFT, which provides a high-performance, consistent, secure PBFT-like consensus engine, where strict fork-accountability guarantees hold over the behaviour of malicious actors. The Tendermint BFT consensus algorithm is well suited for scaling public proof-of-stake blockchains.

The first zone on Neutron is called the Neutron Hub. The Neutron Hub is a multi-asset proof-of-stake cryptocurrency with a simple governance mechanism which enables the network to adapt and upgrade. In addition, the Neutron Hub can be extended by connecting other zones.

The hub and zones of the Neutron network communicate with each other via an inter-blockchain communication (IBC) protocol, a kind of virtual UDP or TCP for blockchains. Tokens can be transferred from one zone to another securely and quickly without the need for exchange liquidity between zones. Instead, all inter-zone token transfers go through the Neutron Hub, which keeps track of the total amount of tokens held by each zone. The hub isolates each zone from the failure of other zones. Because anyone can connect a new zone to the Neutron Hub, zones allow for future-compatibility with new blockchain innovations.

Validators

In classical Byzantine fault-tolerant (BFT) algorithms, each node has the same weight. In Tendermint, nodes have a non-negative amount of voting power, and nodes that have positive voting power are called validators. Validators participate in the consensus protocol by broadcasting cryptographic signatures, or votes, to agree upon the next block.

Validators' voting powers are determined at genesis, or are changed deterministically by the blockchain, depending on the application. For example, in a proof-of-stake application such as the Neutron Hub, the voting power may be determined by the amount of staking tokens bonded as collateral.

NOTE: Fractions like $\frac{1}{3}$ and $\frac{1}{3}$ refer to fractions of the total voting power, never the total number of validators, unless all the validators have equal weight. $>\frac{2}{3}$ means “more than $\frac{2}{3}$ ”, $\geq\frac{1}{3}$ means “at least $\frac{1}{3}$ ”.

Consensus

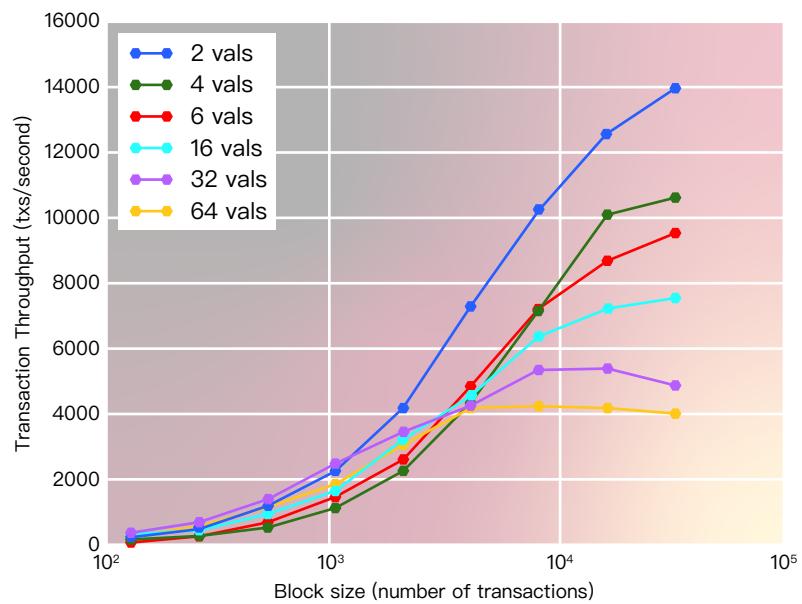
Tendermint is a partially synchronous BFT consensus protocol derived from the DLS consensus algorithm. Tendermint is notable for its simplicity, performance, and fork-accountability. The protocol requires a fixed known set of validators, where each validator is identified by their public key. Validators attempt to come to consensus on one block at a time,

where a block is a list of transactions. Voting for consensus on a block proceeds in rounds. Each round has a round-leader, or proposer, who proposes a block. The validators then vote, in stages, on whether to accept the proposed block or move on to the next round. The proposer for a round is chosen deterministically from the ordered list of validators, in proportion to their voting power.

Tendermint's security derives from its use of optimal Byzantine fault-tolerance via super-majority ($>\frac{2}{3}$) voting and a locking mechanism. Together, they ensure that:

- $\geq \frac{1}{3}$ voting power must be Byzantine to cause a violation of safety, where more than two values are committed.
- if any set of validators ever succeeds in violating safety, or even attempts to do so, they can be identified by the protocol. This includes both voting for conflicting blocks and broadcasting unjustified votes.

Despite its strong guarantees, Tendermint provides exceptional performance. In benchmarks of 64 nodes distributed across 7 datacenters on 5 continents, on commodity cloud instances, Tendermint consensus can process thousands of transactions per second, with commit latencies on the order of one to two seconds. Notably, performance of well over a thousand transactions per second is maintained even in harsh adversarial conditions, with validators crashing or broadcasting maliciously crafted votes. See the figure below for details.



Light Clients

A major benefit of Tendermint's consensus algorithm is simplified light client security, making it an ideal candidate for mobile and internet-of-things use cases. While a Bitcoin light client must sync chains of block headers and find the one with the most proof of work, Tendermint light clients need only to keep up with changes to the validator set, and then verify the $>\frac{2}{3}$ PreCommits in the latest block to determine the latest state. Succinct light client proofs also enable inter-blockchain communication.

ABCI

The Tendermint consensus algorithm is implemented in a program called Tendermint Core. Tendermint BFT is an application-agnostic “consensus engine” that can turn any deterministic blackbox application into a distributedly replicated blockchain. Tendermint BFT connects to blockchain applications via the Application Blockchain Interface (ABCI). ABCI is an interface that defines the boundary between the replication engine (the blockchain), and the state machine (the application). By using a socket protocol, we enable a consensus engine running in one process to manage an application state running in another. Thus, the ABCI allows for blockchain applications to be programmed in any language, not just the programming language that the consensus engine is written in. Additionally, the ABCI makes it possible to easily swap out the consensus layer of any existing blockchain stack.

We draw an analogy with the well-known cryptocurrency Bitcoin. Bitcoin is a cryptocurrency blockchain where each node maintains a fully audited Unspent Transaction Output (UTXO) database. If one wanted to create a Bitcoin-like system on top of ABCI, Tendermint BFT would be responsible for:

- Sharing blocks and transactions between nodes
- Establishing a canonical/immutable order of transactions (the blockchain)

Meanwhile, the ABCI application would be responsible for:

- Maintaining the UTXO database
- Validating cryptographic signatures of transactions
- Preventing transactions from spending non-existent funds
- Allowing clients to query the UTXO database

Zone Explanation

The first blockchain in this network will be the Neutron Hub. The Neutron Hub connects to many other blockchains (or zones) via a novel inter-blockchain communication protocol. The Neutron Hub tracks numerous token types and keeps record of the total number of tokens in each connected zone. Tokens can be transferred from one zone to another securely and quickly without the need for a liquid exchange between zones, because all inter-zone coin transfers go through the Neutron Hub.

This architecture solves many problems that the blockchain space faces today, such as application interoperability, scalability, and seamless upgradability. For example, zones derived from Bitcoind, Go-Ethereum, CryptoNote, ZCash, or any blockchain system can be plugged into the Neutron Hub. These zones allow Neutron to scale infinitely to meet global transaction demand. Zones are also a great fit for a distributed exchange, which will be supported as well.

Neutron is not just a single distributed ledger, and the Neutron Hub isn’t a walled garden or the center of its universe. We are designing a protocol for an open network of distributed ledgers that can serve as a new foundation for future financial systems, based on principles of cryptography, sound economics, consensus theory, transparency, and accountability.

Tendermint BFT

The Neutron Hub is the first public blockchain in the Neutron Network, powered by the Tendermint BFT consensus algorithm. The Tendermint open-source project was born in 2014 to address the speed, scalability, and environmental issues of Bitcoin's proof-of-work consensus algorithm. By using and improving upon proven BFT algorithms developed at MIT in 1988, the Tendermint team was the first to conceptually demonstrate a proof-of-stake cryptocurrency that addresses the nothing-at-stake problem suffered by first-generation proof-of-stake cryptocurrencies such as NXT and BitShares1.0.

Today, practically all Bitcoin mobile wallets use trusted servers to provide them with transaction verification. This is because proof-of-work requires waiting for many confirmations before a transaction can be considered irreversibly committed. Double-spend attacks have already been demonstrated on services like CoinBase.

Unlike other blockchain consensus systems, Tendermint offers instant and provably secure mobile-client payment verification. Since the Tendermint is designed to never fork at all, mobile wallets can receive instant transaction confirmation, which makes trustless and practical payments a reality on smartphones. This has significant ramifications for Internet of Things applications as well.

Validators in Neutron have a similar role to Bitcoin miners, but instead use cryptographic signatures to vote. Validators are secure, dedicated machines that are responsible for committing blocks. Non-validators can delegate their staking tokens (called "NTNs") to any validator to earn a portion of block fees and NTN rewards, but they incur the risk of getting punished (slashed) if the delegate validator gets hacked or violates the protocol. The proven safety guarantees of Tendermint BFT consensus, and the collateral deposit of stakeholders—validators and delegators—provide provable, quantifiable security for nodes and light clients.

Governance

Distributed public ledgers should have a constitution and a governance system. Bitcoin relies on the Bitcoin Foundation and mining to coordinate upgrades, but this is a slow process. Ethereum split into ETH and ETC after hard-forking to address TheDAO hack, largely because there was no prior social contract nor mechanism for making such decisions.

Validators and delegators on the Neutron Hub can vote on proposals that can change preset parameters of the system automatically (such as the block gas limit), coordinate upgrades, as well as vote on amendments to the human-readable constitution that govern the policies of the Neutron Hub. The constitution allows for cohesion among the stakeholders on issues such as theft and bugs (such as TheDAO incident), allowing for quicker and cleaner resolution.

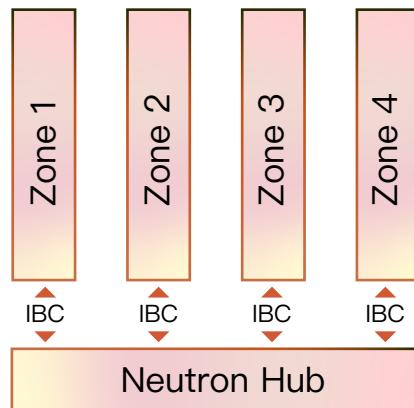
Each zone can also have their own constitution and governance mechanism as well. For example, the Neutron Hub could have a constitution that enforces immutability at the Hub (no roll-backs, save for bugs of the Neutron Hub node implementation), while each zone can set their own policies regarding roll-backs.

By enabling interoperability among differing policy zones, the Neutron network gives its users ultimate freedom and potential for permission less experimentation.

The Hub and Zones

Here we describe a novel model of decentralization and scalability. Neutron is a network of many blockchains powered by Tendermint. While existing proposals aim to create a “single blockchain” with total global transaction ordering, Neutron permits many blockchains to run concurrently with one another while retaining interoperability.

At the basis, the Neutron Hub manages many independent blockchains called “zones” (sometimes referred to as “shards”, in reference to the database scaling technique known as “sharding”). A constant stream of recent block commits from zones posted on the Hub allows the Hub to keep up with the state of each zone. Likewise, each zone keeps up with the state of the Hub (but zones do not keep up with each other except indirectly through the Hub). Packets of information are then communicated from one zone to another by posting Merkle-proofs as evidence that the information was sent and received. This mechanism is called inter-blockchain communication or IBC for short.



Any of the zones can themselves be hubs to form an acyclic graph, but for the sake of clarity we will only describe the simple configuration where there is only one hub, and many non-hub zones.

The Hub

The Neutron Hub is a blockchain that hosts a multi-asset distributed ledger, where tokens can be held by individual users or by zones themselves. These tokens can be moved from one zone to another in a special IBC packet called a “coin packet”. The hub is responsible for preserving the global invariance of the total amount of each token across the zones. IBC coin packet transactions must be committed by the sender, hub, and receiver blockchains.

Since the Neutron Hub acts as the central ledger for the whole system, the security of the Hub is of paramount importance. While each zone may be a Tendermint blockchain that is secured by as few as 4 (or even less if BFT consensus is not needed), the Hub must be secured by a globally decentralized set of validators that can withstand the most severe attack scenarios, such as a continental network partition or a nation-state sponsored attack.

The Zones

A Neutron zone is an independent blockchain that exchanges IBC messages with the Hub. From the Hub's perspective, a zone is a multi-asset dynamic-membership multi-signature account that can send and receive tokens using IBC packets. Like a cryptocurrency account, a zone cannot transfer more tokens than it has, but can receive tokens from others who have them. A zone may be designated as an "source" of one or more token types, granting it the power to inflate that token supply.

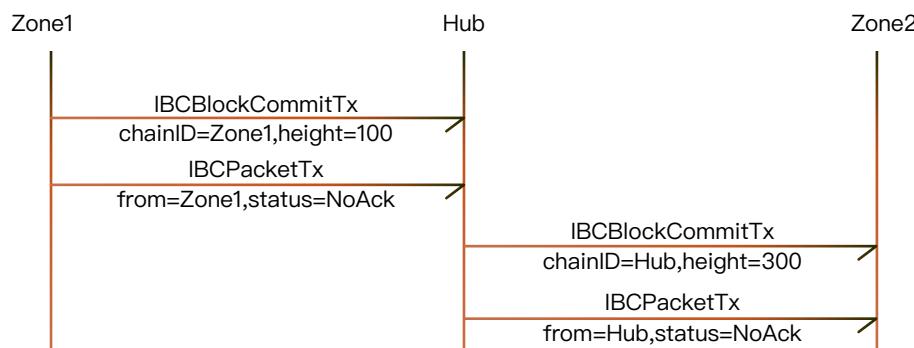
NTNs of the Neutron Hub may be staked by validators of a zone connected to the Hub. While double-spend attacks on these zones would result in the slashing of NTNs with Tendermint's fork-accountability, a zone where $>\frac{2}{3}$ of the voting power are Byzantine can commit invalid state. The Neutron Hub does not verify or execute transactions committed on other zones, so it is the responsibility of users to send tokens to zones that they trust. In the future, the Neutron Hub's governance system may pass Hub improvement proposals that account for zone failures. For example, outbound token

Inter-blockchain Communication (IBC)

Now we look at how the Hub and zones communicate with each other. For example, if there are three blockchains, "Zone1", "Zone2", and "Hub", and we wish for "Zone1" to produce a packet destined for "Zone2" going through "Hub". To move a packet from one blockchain to another, a proof is posted on the receiving chain. The proof states that the sending chain published a packet for the alleged destination. For the receiving chain to check this proof, it must be able keep up with the sender's block headers. This mechanism is similar to that used by sidechains, which requires two interacting chains to be aware of one another via a bidirectional stream of proof-of-existence datagrams (transactions).

The IBC protocol can naturally be defined using two types of transactions: an `IBCBlockCommitTx` transaction, which allows a blockchain to prove to any observer of its most recent block-hash, and an `IBCPacketTx` transaction, which allows a blockchain to prove to any observer that the given packet was indeed published by the sender's application, via a Merkle-proof to the recent block-hash.

By splitting the IBC mechanics into two separate transactions, we allow the native fee market-mechanism of the receiving chain to determine which packets get committed (i.e. acknowledged), while allowing for complete freedom on the sending chain as to how many outbound packets are allowed.



In the example above, in order to update the block-hash of "Zone1" on "Hub" (or of "Hub" on "Zone2"), an `IBCBlockCommitTx` transaction must be posted on "Hub" with the block-hash of "Zone1" (or on "Zone2" with the block-hash of "Hub").

Use Cases

Distributed Exchange

In the same way that Bitcoin is more secure by being a distributed, mass-replicated ledger, we can make exchanges less vulnerable to external and internal hacks by running it on the blockchain. We call this a distributed exchange.

What the cryptocurrency community calls a decentralized exchange today are based on something called “NTNic cross-chain” (AXC) transactions. With an AXC transaction, two users on two different chains can make two transfer transactions that are committed together on both ledgers, or none at all (i.e. NTNicely). For example, two users can trade bitcoins for ether (or any two tokens on two different ledgers) using AXC transactions, even though Bitcoin and Ethereum are not connected to each other. The benefit of running an exchange on AXC transactions is that neither users need to trust each other or the trade-matching service. The downside is that both parties need to be online for the trade to occur.

Another type of decentralized exchange is a mass-replicated distributed exchange that runs on its own blockchain. Users on this kind of exchange can submit a limit order and turn their computer off, and the trade can execute without the user being online. The blockchain matches and completes the trade on behalf of the trader.

A centralized exchange can create a deep orderbook of limit orders and thereby attract more traders. Liquidity begets more liquidity in the exchange world, and so there is a strong network effect (or at least a winner-take-most effect) in the exchange business. The current leader for cryptocurrency exchanges today is Poloniex with a 24-hour volume of \$20M, and in second place is Bitfinex with a 24-hour volume of \$5M. Given such strong network effects, it is unlikely for AXC-based decentralized exchanges to win volume over the centralized exchanges. For a decentralized exchange to compete with a centralized exchange, it would need to support deep orderbooks with limit orders. Only a distributed exchange on a blockchain can provide that.

Tendermint provides additional benefits of faster transaction commits. By prioritizing fast finality without sacrificing consistency, zones in Neutron can finalize transactions fast — for both exchange order transactions as well as IBC token transfers to and from other zones.

Given the state of cryptocurrency exchanges today, a great application for Neutron is the distributed exchange (aka the Neutron DEX). The transaction throughput capacity as well as commit latency can be comparable to those of centralized exchanges. Traders can submit limit orders that can be executed without both parties having to be online. And with Tendermint, the Neutron hub, and IBC, traders can move funds in and out of the exchange to and from other zones with speed.

Bridging To Other Cryptocurrencies

A privileged zone can act as the source of a bridged token of another cryptocurrency. A bridge is similar to the relationship between a Neutron hub and zone; both must keep up with the latest blocks of the other in order to verify proofs that tokens have moved from one to the other. A “bridge-zone” on the Neutron network keeps up with the Hub as well as the other cryptocurrency. The indirection through the bridge-zone allows the logic of the Hub to remain simple and agnostic to other blockchain consensus strategies such as Bitcoin’s proof-of-work mining.

Sending Tokens to the Neutron Hub

Each bridge-zone validator would run a Tendermint-powered blockchain with a special ABCI bridge-app, but also a full-node of the “origin” blockchain.

When new blocks are mined on the origin, the bridge-zone validators will come to agreement on committed blocks by signing and sharing their respective local view of the origin’s blockchain tip. When a bridge-zone receives payment on the origin (and sufficient confirmations were agreed to have been seen in the case of a PoW chain such as Ethereum or Bitcoin), a corresponding account is created on the bridge-zone with that balance.

In the case of Ethereum, the bridge-zone can share the same validator-set as the Neutron Hub. On the Ethereum side (the origin), a bridge-contract would allow ether holders to send ether to the bridge-zone by sending it to the bridge-contract on Ethereum. Once ether is received by the bridge-contract, the ether cannot be withdrawn unless an appropriate IBC packet is received by the bridge-contract from the bridge-zone. The bridge-contract tracks the validator-set of the bridge-zone, which may be identical to the Neutron Hub’s validator-set.

In the case of Bitcoin, the concept is similar except that instead of a single bridge-contract, each UTXO would be controlled by a threshold multisignature P2SH subscript. Due to the limitations of the P2SH system, the signers cannot be identical to the Neutron Hub validator-set.

Withdrawing Tokens from Neutron Hub

Ether on the bridge-zone (“bridged-ether”) can be transferred to and from the Hub, and later be destroyed with a transaction that sends it to a particular withdrawal address on Ethereum. An IBC packet proving that the transaction occurred on the bridge-zone can be posted to the Ethereum bridge-contract to allow the ether to be withdrawn.

Ethereum Scaling

Solving the scaling problem is an open issue for Ethereum. Currently, Ethereum nodes process every single transaction and also store all the states.

Since Tendermint can commit blocks much faster than Ethereum’s proof-of-work, EVM zones powered by Tendermint consensus and operating on bridged-ether can provide higher performance to Ethereum blockchains. Additionally, though the Neutron Hub and IBC packet mechanics does not allow for arbitrary contract logic execution per se, it can be used to coordinate token movements between Ethereum contracts running on different zones, providing a foundation for token-centric Ethereum scaling via sharding.

Total Accountability of Bridge Zones

The risk of such a bridgegging contract is a rogue validator set. $\geq \frac{1}{3}$ Byzantine voting power could cause a fork, withdrawing ether from the bridge-contract on Ethereum while keeping the bridged-ether on the bridge-zone. Worse, $> \frac{2}{3}$ Byzantine voting power can steal ether outright from those who sent it to the bridge-contract by deviating from the original bridgegging logic of the bridge-zone.

It is possible to address these issues by designing the bridge to be totally accountable. For example, all IBC packets, from the hub and the origin, might require acknowledgement by the bridge-zone in such a way that all state transitions of the bridge-zone can be efficiently challenged and verified by either the hub or the origin's bridge-contract. The Hub and the origin should allow the bridge-zone validators to post collateral, and token transfers out of the bridge-contract should be delayed (and collateral unbonding period sufficiently long) to allow for any challenges to be made by independent auditors. We leave the design of the specification and implementation of this system open as a future Neutron improvement proposal, to be passed by the Neutron Hub's governance system.

Multi-Application Integration

Neutron zones run arbitrary application logic, which is defined at the beginning of the zone's life and can potentially be updated over time by governance. Such flexibility allows Neutron zones to act as bridges to other cryptocurrencies such as Ethereum or Bitcoin, and it also permits derivatives of those blockchains, utilizing the same codebase but with a different validator set and initial distribution. This allows many existing cryptocurrency frameworks, such as those of Ethereum, Zerocash, Bitcoin, CryptoNote and so on, to be used with Tendermint BFT, which is a higher performance consensus engine, on a common network, opening tremendous opportunity for interoperability across platforms. Furthermore, as a multi-asset blockchain, a single transaction may contain multiple inputs and outputs, where each input can be any token type, enabling Neutron to serve directly as a platform for decentralized exchange, though orders are assumed to be matched via other platforms. Alternatively, a zone can serve as a distributed fault-tolerant exchange (with orderbooks), which can be a strict improvement over existing centralized cryptocurrency exchanges which tend to get hacked over time.

Zones can also serve as blockchain-backed versions of enterprise and government systems, where pieces of a particular service that are traditionally run by an organization or group of organizations are instead run as a ABCI application on a certain zone, which allows it to inherit the security and interoperability of the public Neutron network without sacrificing control over the underlying service. Thus, Neutron may offer the best of both worlds for organizations looking to utilize blockchain technology but who are wary of relinquishing control completely to a distributed third party.

Network Partition Mitigation

Some claim that a major problem with consistency-favouring consensus algorithms like Tendermint is that any network partition which causes there to be no single partition with $> \frac{2}{3}$ voting power (e.g. $\geq \frac{1}{3}$ going offline) will halt consensus altogether. The Neutron architecture can help mitigate this problem by using a global hub with regional autonomous zones, where voting power for each zone are distributed based on a common geographic region. For instance, a common paradigm may be for individual cities, or regions, to operate their own zones while sharing a common hub (e.g. the Neutron Hub), enabling municipal activity to persist in the event that the hub halts due to a temporary network partition. Note that this allows real geological, political, and network-topological features to be considered in designing robust federated fault-tolerant systems.

Federated Name Resolution System

NameCoin was one of the first blockchains to attempt to solve the name-resolution problem by adapting the Bitcoin blockchain. Unfortunately there have been several issues with this approach.

With Namecoin, we can verify that, for example, @satoshi was registered with a particular public key at some point in the past, but we wouldn't know whether the public key had since been updated recently unless we download all the blocks since the last update of that name. This is due to the limitation of Bitcoin's UTXO transaction Merkle-ization model, where only the transactions (but not mutable application state) are Merkle-ized into the block-hash. This lets us prove existence, but not the non-existence of later updates to a name. Thus, we can't know for certain the most recent value of a name without trusting a full node, or incurring significant costs in bandwidth by downloading the whole blockchain.

Even if a Merkle-ized search tree were implemented in NameCoin, its dependency on proof-of-work makes light client verification problematic. Light clients must download a complete copy of the headers for all blocks in the entire blockchain (or at least all the headers since the last update to a name). This means that the bandwidth requirements scale linearly with the amount of time. In addition, name-changes on a proof-of-work blockchain requires waiting for additional proof-of-work confirmation blocks, which can take up to an hour on Bitcoin.

With Tendermint, all we need is the most recent block-hash signed by a quorum of validators (by voting power), and a Merkle proof to the current value associated with the name. This makes it possible to have a succinct, quick, and secure light-client verification of name values.

In Neutron, we can take this concept and extend it further. Each name-registration zone in Neutron can have an associated top-level-domain (TLD) name such as ".com" or ".org", and each name-registration zone can have its own governance and registration rules.

Limitations on The Number Of Validators

Unlike Bitcoin or other proof-of-work blockchains, a Tendermint blockchain gets slower with more validators due to the increased communication complexity. Fortunately, we can support enough validators to make for a robust globally distributed blockchain with very fast transaction confirmation times, and, as bandwidth, storage, and parallel compute capacity increases, we will be able to support more validators in the future.

On genesis day, the maximum number of validators will be set to 100, and this number will increase at a rate of 13% for 10 years, and settle at 300 validators.

Year 0: 100

Year 1: 113

Year 2: 127

Year 3: 144

Year 4: 163

Year 5: 184

Year 6: 208

Year 7: 235

Year 8: 265

Year 9: 300

Year 10: 300

...

Penalties for Validators

There must be some penalty imposed on the validators for any intentional or unintentional deviation from the sanctioned protocol. Some evidence is immediately admissible, such as a double-sign at the same height and round, or a violation of “prevote-the-lock” (a rule of the Tendermint consensus protocol). Such evidence will result in the validator losing its good standing and its bonded NTNs as well its proportionate share of tokens in the reserve pool — collectively called its “stake” — will get slashed.

Sometimes, validators will not be available, either due to regional network disruptions, power failure, or other reasons. If, at any point in the past ValidatorTimeoutWindow blocks, a validator’s commit vote is not included in the blockchain more than ValidatorTimeoutMaxAbsent times, that validator will become inactive, and lose ValidatorTimeoutPenalty (DEFAULT 1%) of its stake.

Some “malicious” behavior does not produce obviously discernable evidence on the blockchain. In these cases, the validators can coordinate out of band to force the timeout of these malicious validators, if there is a supermajority consensus.

In situations where the Neutron Hub halts due to a $\geq \frac{1}{3}$ coalition of voting power going offline, or in situations where a $\geq \frac{1}{3}$ coalition of voting power censor evidence of malicious behavior from entering the blockchain, the hub must recover with a hard-fork reorg-proposal.

Transaction Fees

Neutron Hub validators can accept any token type or combination of types as fees for processing a transaction. Each validator can subjectively set whatever exchange rate it wants, and choose whatever transactions it wants, as long as the BlockGasLimit is not exceeded. The collected fees, minus any taxes specified below, are redistributed to the bonded stakeholders in proportion to their bonded NTNs, every ValidatorPayoutPeriod (DEFAULT 1 hour).

Of the collected transaction fees, ReserveTax (DEFAULT 2%) will go toward the reserve pool to increase the reserve pool and increase the security and value of the Neutron network. These funds can also be distributed in accordance with the decisions made by the governance system.

NTN holders who delegate their voting power to other validators pay a commission to the delegated validator. The commission can be set by each validator.

Governance Specification

The Neutron Hub is operated by a distributed organization that requires a well-defined governance mechanism in order to coordinate various changes to the blockchain, such as the variable parameters of the system, as well as software upgrades and constitutional amendments.

All validators are responsible for voting on all proposals. Failing to vote on a proposal in a timely manner will result in the validator being deactivated automatically for a period of time called the AbsenteeismPenaltyPeriod (DEFAULT 1 week).

Delegators automatically inherit the vote of the delegated validator. This vote may be overridden manually. Unbonded NTNs get no vote.

Each proposal requires a deposit of MinimumProposalDeposit tokens, which may be a combination of one or more tokens including NTNs. For each proposal, the voters may vote to take the deposit. If more than half of the voters choose to take the deposit (e.g. because the proposal was spam), the deposit goes to the reserve pool, except any NTNs which are burned.

For each proposal, voters may vote with the following options:

- Yea
- YeaWithForce
- Nay
- Abstain

A strict majority of Yea or YeaWithForce votes (or Nay or NayWithForce votes) is required for the proposal to be decided as passed (or decided as failed), but 1/3+ can veto the majority decision by voting “with force”. When a strict majority is vetoed, everyone gets punished by losing VetoPenaltyFeeBlocks (DEFAULT 1 day’s worth of blocks) worth of fees (except taxes which will not be affected), and the party that vetoed the majority decision will be additionally punished by losing VetoPenaltyNTNs (DEFAULT 0.1%) of its NTNs.

Incentivizing Hackers

The security of the Neutron Hub is a function of the security of the underlying validators and the choice of delegation by delegators. In order to encourage the discovery and early reporting of found vulnerabilities, the Neutron Hub encourages hackers to publish successful exploits via a ReportHackTx transaction that says, “This validator got hacked. Please send bounty to this address”. Upon such an exploit, the validator and delegators will become inactive, HackPunishmentRatio (default 5%) of everyone’s NTNs will get slashed, and HackRewardRatio (default 5%) of everyone’s NTNs will get rewarded to the hacker’s bounty address. The validator must recover the remaining NTNs by using their backup key.

In order to prevent this feature from being abused to transfer unvested NTNs, the portion of vested vs unvested NTNs of validators and delegators before and after the ReportHackTx will remain the same, and the hacker bounty will include some unvested NTNs, if any.

Consensus Systems Classic Byzantine Fault Tolerance

Consensus in the presence of malicious participants is a problem dating back to the early 1980s, when Leslie Lamport coined the phrase “Byzantine fault” to refer to arbitrary process behavior that deviates from the intended behavior, in contrast to a “crash fault”, wherein a process simply crashes. Early solutions were discovered for synchronous networks where there is an upper bound on message latency, though practical use was limited to highly controlled environments such as airplane controllers and datacenters synchronized via NTNic clocks. It was not until the late 90s that Practical Byzantine

Fault Tolerance (PBFT) was introduced as an efficient partially synchronous consensus algorithm able to tolerate up to $\frac{1}{3}$ of processes behaving arbitrarily. PBFT became the standard algorithm, spawning many variations, including most recently one created by IBM as part of their contribution to Hyperledger.

The main benefit of Tendermint consensus over PBFT is that Tendermint has an improved and simplified underlying structure, some of which is a result of embracing the blockchain paradigm. Tendermint blocks must commit in order, which obviates the complexity and communication overhead associated with PBFT's view-changes. In Neutron and many cryptocurrencies, there is no need to allow for block $N+i$ where $i \geq 1$ to commit, when block N itself hasn't yet committed. If bandwidth is the reason why block N hasn't committed in a Neutron zone, then it doesn't help to use bandwidth sharing votes for blocks $N+i$. If a network partition or offline nodes is the reason why block N hasn't committed, then $N+i$ won't commit anyway.

This allows for faster provable transaction commits for light-clients and faster inter-blockchain communication.

Horizontal Scaling Interledger Protocol

The Interledger Protocol is not strictly a scalability solution. It provides an ad hoc interoperability between different ledger systems through a loosely coupled bilateral relationship network. Like the Lightning Network, the purpose of ILP is to facilitate payments, but it specifically focuses on payments across disparate ledger types, and extends the NTNic transaction mechanism to include not only hash-locks, but also a quorum of notaries (called the NTNic Transport Protocol). The latter mechanism for enforcing NTNicity in inter-ledger transactions is similar to Tendermint's light-client SPV mechanism, so an illustration of the distinction between ILP and Neutron/IBC is warranted, and provided below.

1. The notaries of a connector in ILP do not support membership changes, and do not allow for flexible weighting between notaries. On the other hand, IBC is designed specifically for blockchains, where validators can have different weights, and where membership can change over the course of the blockchain.
2. As in the Lightning Network, the receiver of payment in ILP must be online to send a confirmation back to the sender. In a token transfer over IBC, the validator-set of the receiver's blockchain is responsible for providing confirmation, not the receiving user.
3. The most striking difference is that ILP's connectors are not responsible for keeping authoritative state about payments, whereas in Neutron, the validators of a hub are the authority of the state of IBC token transfers as well as the authority of the amount of tokens held by each zone (but not the amount of tokens held by each account within a zone). This is the fundamental innovation that allows for secure asymmetric transfer of tokens from zone to zone; the analog to ILP's connector in Neutron is a persistent and maximally secure blockchain ledger, the Neutron Hub.
4. The inter-ledger payments in ILP need to be backed by an exchange orderbook, as there is no asymmetric transfer of coins from one ledger to another, only the transfer of value or market equivalents.

Sidechains

Sidechains are a proposed mechanism for scaling the Bitcoin network via alternative blockchains that are “two-way pegged” to the Bitcoin blockchain. (Two-way pegging is equivalent to bridging. In Neutron we say “bridging” to distinguish from market-pegging). Sidechains allow bitcoins to effectively move from the Bitcoin blockchain to the sidechain and back, and allow for experimentation in new features on the sidechain. As in the Neutron Hub, the sidechain and Bitcoin serve as light-clients of each other, using SPV proofs to determine when coins should be transferred to the sidechain and back. Of course, since Bitcoin uses proof-of-work, sidechains centered around Bitcoin suffer from the many problems and risks of proof-of-work as a consensus mechanism. Furthermore, this is a Bitcoin-maximalist solution that doesn’t natively support a variety of tokens and inter-zone network topology as Neutron does. That said, the core mechanism of the two-way peg is in principle the same as that employed by the Neutron network.

Ethereum Scalability Efforts

Ethereum is currently researching a number of different strategies to shard the state of the Ethereum blockchain to address scalability needs. These efforts have the goal of maintaining the abstraction layer offered by the current Ethereum Virtual Machine across the shared state space. Multiple research efforts are underway at this time.

Neutron vs Ethereum 2.0 Mauve

Neutron and Ethereum 2.0 Mauve have different design goals.

- Neutron is specifically about tokens. Mauve is about scaling general computation.
- Neutron is not bound to the EVM, so even different VMs can interoperate.
- Neutron lets the zone creator determine who validates the zone.
- Anyone can start a new zone in Neutron (unless governance decides otherwise).
- The hub isolates zone failures so global token invariants are preserved.

General Scaling Lightning Network

The Lightning Network is a proposed token transfer network operating at a layer above the Bitcoin blockchain (and other public blockchains), enabling improvement of many orders of magnitude in transaction throughput by moving the majority of transactions outside of the consensus ledger into so-called “payment channels”. This is made possible by on-chain cryptocurrency scripts, which enable parties to enter into bilateral stateful contracts where the state can be updated by sharing digital signatures, and contracts can be closed by finally publishing evidence onto the blockchain, a mechanism first popularized by cross-chain NTNic swaps. By opening payment channels with many parties, participants in the Lightning Network can become focal points for routing the payments of others, leading to a fully connected payment channel network, at the cost of capital being tied up on payment channels.

While the Lightning Network can also easily extend across multiple independent blockchains to allow for the transfer of value via an exchange market, it cannot be used to asymmetrically transfer tokens from one blockchain to another. The main benefit of the Neutron network described here is to enable such direct token transfers. That said, we expect payment

Segregated Witness

Segregated Witness is a Bitcoin improvement proposal link that aims to increase the per-block transaction throughput 2X or 3X, while simultaneously making block syncing faster for new nodes. The brilliance of this solution is in how it works within the limitations of Bitcoin's current protocol and allows for a soft-fork upgrade (i.e. clients with older versions of the software will continue to function after the upgrade). Tendermint, being a new protocol, has no design restrictions, so it has a different scaling priorities. Primarily, Tendermint uses a BFT round-robin algorithm based on cryptographic signatures instead of mining, which trivially allows horizontal scaling through multiple parallel blockchains, while regular, more frequent block commits allow for vertical scaling as well.

Chapter 3

Consensus Mechanism

03

Fork Accountability

A well designed consensus protocol should provide some guarantees in the event that the tolerance capacity is exceeded and the consensus fails. This is especially necessary in economic systems, where Byzantine behaviour can have substantial financial reward. The most important such guarantee is a form of fork–accountability, where the processes that caused the consensus to fail (ie. caused clients of the protocol to accept different values – a fork) can be identified and punished according to the rules of the protocol, or, possibly, the legal system. When the legal system is unreliable or excessively expensive to invoke, validators can be forced to make security deposits in order to participate, and those deposits can be revoked, or slashed, when malicious behaviour is detected.

Tendermint Consensus

We call the voting stages PreVote and PreCommit. A vote can be for a particular block or for Nil. We call a collection of $>\frac{2}{3}$ PreVotes for a single block in the same round a Polka, and a collection of $>\frac{2}{3}$ PreCommits for a single block in the same round a Commit. If $>\frac{2}{3}$ PreCommit for Nil in the same round, they move to the next round.

Note that strict determinism in the protocol incurs a weak synchrony assumption as faulty leaders must be detected and skipped. Thus, validators wait some amount of time, TimeoutPropose, before they Prevote Nil, and the value of TimeoutPropose increases with each round. Progression through the rest of a round is fully asynchronous, in that progress is only made once a validator hears from $>\frac{2}{3}$ of the network. In practice, it would take an extremely strong adversary to indefinitely thwart the weak synchrony assumption (causing the consensus to fail to ever commit a block), and doing so can be made even more difficult by using randomized values of TimeoutPropose on each validator.

An additional set of constraints, or Locking Rules, ensure that the network will eventually commit just one block at each height. Any malicious attempt to cause more than one block to be committed at a given height can be identified. First, a PreCommit for a block must come with justification, in the form of a Polka for that block. If the validator has already PreCommit a block at round R₁, we say they are _locked on that block, and the Polka used to justify the new PreCommit at round R₂ must come in a round R_{_polka} where R_{_1} < R_{_polka} <= R_{_2}. Second, validators must Propose and/or PreVote the block they are locked on. Together, these conditions ensure that a validator does not PreCommit without sufficient evidence as justification, and that validators which have already PreCommit cannot contribute to evidence to PreCommit something else. This ensures both safety and liveness of the consensus algorithm.

Preventing Long Range Attacks

Assuming a sufficiently resilient collection of broadcast networks and a static validator set, any fork in the blockchain can be detected and the deposits of the offending validators slashed. This innovation, first suggested by Vitalik Buterin in early 2014, solves the nothing-at-stake problem of other proof-of-stake cryptocurrencies. However, since validator sets must

be able to change, over a long range of time the original validators may all become unbonded, and hence would be free to create a new chain from the genesis block, incurring no cost as they no longer have deposits locked up. This attack came to be known as the Long Range Attack (LRA), in contrast to a Short Range Attack, where validators who are currently bonded cause a fork and are hence punishable (assuming a fork-accountable BFT algorithm like Tendermint consensus). Long Range Attacks are often thought to be a critical blow to proof-of-stake.

Fortunately, the LRA can be mitigated as follows. First, for a validator to unbond (thereby recovering their collateral deposit and no longer earning fees to participate in the consensus), the deposit must be made untransferable for an amount of time known as the “unbonding period”, which may be on the order of weeks or months. Second, for a light client to be secure, the first time it connects to the network it must verify a recent block–hash against a trusted source, or preferably multiple sources. This condition is sometimes referred to as “weak subjectivity”. Finally, to remain secure, it must sync up with the latest validator set at least as frequently as the length of the unbonding period. This ensures that the light client knows about changes to the validator set before a validator has its capital unbonded and thus no longer at stake, which would allow it to deceive the client by carrying out a long range attack by creating new blocks beginning back at a height where it was bonded (assuming it has control of sufficiently many of the early private keys).

Note that overcoming the LRA in this way requires an overhaul of the original security model of proof-of-work. In PoW, it is assumed that a light client can sync to the current height from the trusted genesis block at any time simply by processing the proof-of-work in every block header. To overcome the LRA, however, we require that a light client come online with some regularity to track changes in the validator set, and that the first time they come online they must be particularly careful to authenticate what they hear from the network against trusted sources. Of course, this latter requirement is similar to that of Bitcoin, where the protocol and software must also be obtained from a trusted source.

The above method for preventing LRA is well suited for validators and full nodes of a Tendermint-powered blockchain because these nodes are meant to remain connected to the network. The method is also suitable for light clients that can be expected to sync with the network frequently. However, for light clients that are not expected to have frequent access to the internet or the blockchain network, yet another solution can be used to overcome the LRA. Non-validator token holders can post their tokens as collateral with a very long unbonding period (e.g. much longer than the unbonding period for validators) and serve light clients with a secondary method of attesting to the validity of current and past block–hashes. While these tokens do not count toward the security of the blockchain’s consensus, they nevertheless can provide strong guarantees for light clients. If historical block–hash querying were supported in Ethereum, anyone could bond their tokens in a specially designed smart contract and provide attestation services for pay, effectively creating a market for light-client LRA security.

Tendermint Light Clients

The need to sync all block headers is eliminated in Tendermint–PoS as the existence of an alternative chain (a fork) means $\geq 1/3$ of bonded stake can be slashed. Of course, since slashing requires that someone share evidence of a fork, light clients should store any block–hash commits that it sees. Additionally, light clients could periodically stay synced with changes to the validator set, in order to avoid long range attacks (but other solutions are possible).

In spirit similar to Ethereum, Tendermint enables applications to embed a global Merkle root hash in each block, allowing easily verifiable state queries for things like account balances, the value stored in a contract, or the existence of an unspent transaction output, depending on the nature of the application.

Preventing Long Range Attacks

Assuming a sufficiently resilient collection of broadcast networks and a static validator set, any fork in the blockchain can be detected and the deposits of the offending validators slashed. This innovation, first suggested by Vitalik Buterin in early 2014, solves the nothing-at-stake problem of other proof-of-stake cryptocurrencies. However, since validator sets must be able to change, over a long range of time the original validators may all become unbonded, and hence would be free to create a new chain from the genesis block, incurring no cost as they no longer have deposits locked up. This attack came to be known as the Long Range Attack (LRA), in contrast to a Short Range Attack, where validators who are currently bonded cause a fork and are hence punishable (assuming a fork-accountable BFT algorithm like Tendermint consensus). Long Range Attacks are often thought to be a critical blow to proof-of-stake.

Fortunately, the LRA can be mitigated as follows. First, for a validator to unbond (thereby recovering their collateral deposit and no longer earning fees to participate in the consensus), the deposit must be made untransferable for an amount of time known as the “unbonding period”, which may be on the order of weeks or months. Second, for a light client to be secure, the first time it connects to the network it must verify a recent block-hash against a trusted source, or preferably multiple sources. This condition is sometimes referred to as “weak subjectivity”. Finally, to remain secure, it must sync up with the latest validator set at least as frequently as the length of the unbonding period. This ensures that the light client knows about changes to the validator set before a validator has its capital unbonded and thus no longer at stake, which would allow it to deceive the client by carrying out a long range attack by creating new blocks beginning back at a height where it was bonded (assuming it has control of sufficiently many of the early private keys).

Note that overcoming the LRA in this way requires an overhaul of the original security model of proof-of-work. In PoW, it is assumed that a light client can sync to the current height from the trusted genesis block at any time simply by processing the proof-of-work in every block header. To overcome the LRA, however, we require that a light client come online with some regularity to track changes in the validator set, and that the first time they come online they must be particularly careful to authenticate what they hear from the network against trusted sources. Of course, this latter requirement is similar to that of Bitcoin, where the protocol and software must also be obtained from a trusted source.

The above method for preventing LRA is well suited for validators and full nodes of a Tendermint-powered blockchain because these nodes are meant to remain connected to the network. The method is also suitable for light clients that can be expected to sync with the network frequently. However, for light clients that are not expected to have frequent access to the internet or the blockchain network, yet another solution can be used to overcome the LRA. Non-validator token holders can post their tokens as collateral with a very long unbonding period (e.g. much longer than the unbonding period for validators) and serve light clients with a secondary method of attesting to the validity of current and past block-hashes. While these tokens do not count toward the security of the blockchain’s consensus, they nevertheless can provide strong guarantees for light clients. If historical block-hash querying were supported in Ethereum, anyone could bond their tokens in a specially designed smart contract and provide attestation services for pay, effectively creating a market for light-client LRA security.

ABCI Specification

ABCI consists of 3 primary message types that get delivered from the core to the application. The application replies with corresponding response messages.

The AppendTx message is the work horse of the application. Each transaction in the blockchain is delivered with this message. The application needs to validate each transactions received with the AppendTx message against the current state, application protocol, and the cryptographic credentials of the transaction. A validated transaction then needs to update the application state — by binding a value into a key values store, or by updating the UTXO database.

The CheckTx message is similar to AppendTx, but it's only for validating transactions. Tendermint BFT's mempool first checks the validity of a transaction with CheckTx, and only relays valid transactions to its peers. Applications may check an incrementing nonce in the transaction and return an error upon CheckTx if the nonce is old.

The Commit message is used to compute a cryptographic commitment to the current application state, to be placed into the next block header. This has some handy properties. Inconsistencies in updating that state will now appear as blockchain forks which catches a whole class of programming errors. This also simplifies the development of secure lightweight clients, as Merkle–hash proofs can be verified by checking against the block–hash, and the block–hash is signed by a quorum of validators (by voting power).

Additional ABCI messages allow the application to keep track of and change the validator set, and for the application to receive the block information, such as the height and the commit votes. ABCI requests/responses are simple Protobuf messages.

AppendTx

- Arguments:
- Data ([])byte): The request transaction bytes
- Returns:
- Code (uint32): Response code
- Data ([])byte): Result bytes, if any
- Log (string): Debug or error message
- Usage:

Append and run a transaction. If the transaction is valid, returns CodeType.OK

CheckTx

- Arguments:
- Data ([])byte): The request transaction bytes
- Returns:
- Code (uint32): Response code
- Data ([])byte): Result bytes, if any
- Log (string): Debug or error message
- Usage:

Validate a transaction. This message should not mutate the state. Transactions are first run through CheckTx before broadcast to peers in the mempool layer. You can make CheckTx semi-stateful and clear the state upon Commit or BeginBlock, to allow for dependent sequences of transactions in the same block.

Commit

- Returns:
- Data ([])byte): The Merkle root hash
- Log (string): Debug or error message
- Usage:

Return a Merkle root hash of the application state.

Query

- Arguments:
- Data ([])byte): The query request bytes
- Returns:
- Code (uint32): Response code
- Data ([])byte): The query response bytes

- Log (string): Debug or error message

Flush

- Usage:

Flush the response queue. Applications that implement types. Application need not implement this message — it's handled by the project.

Info

- Returns:
- Data ([]byte): The info bytes
- Usage:

Return information about the application state. Application specific.

SetOption

- Arguments:
- Key (string): Key to set
- Value (string): Value to set for key
- Returns:
- Log (string): Debug or error message
- Usage:

Set application options. E.g. Key=“mode”, Value=“mempool” for a mempool connection, or Key=“mode”, Value=“consensus” for a consensus connection. Other options are application specific.

InitChain

- Arguments:
- Validators ([]Validator): Initial genesis–validators
- Usage:

Called once upon genesis

BeginBlock

- Arguments:
 - Height (uint64): The block height that is starting
 - Usage:
- Signals the beginning of a new block. Called prior to any AppendTxs.

EndBlock

- Arguments:
- Height (uint64): The block height that ended
- Returns:
- Validators ([]Validator): Changed validators with new voting powers (0 to remove)
- Usage:

Signals the end of a block. Called prior to each Commit after all transactions

CHAPTER 4

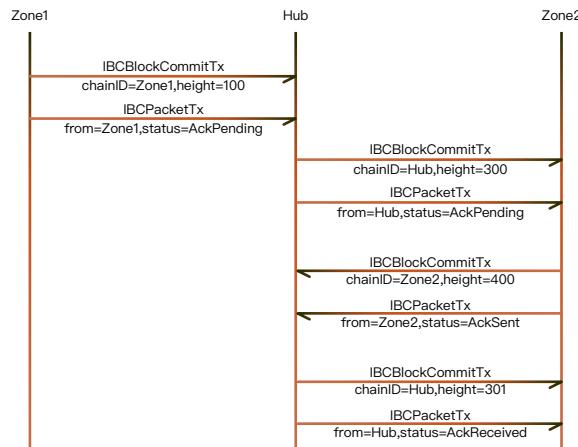
Underlying Technology

04

IBC Packet Delivery Acknowledgement

There are several reasons why a sender may want the acknowledgement of delivery of a packet by the receiving chain. For example, the sender may not know the status of the destination chain, if it is expected to be faulty. Or, the sender may want to impose a timeout on the packet (with the MaxHeight packet field), while any destination chain may suffer from a denial-of-service attack with a sudden spike in the number of incoming packets.

In these cases, the sender can require delivery acknowledgement by setting the initial packet status to AckPending. Then, it is the receiving chain's responsibility to confirm delivery by including an abbreviated IBCPacket in the app Merkle hash.



First, an IBCBlockCommit and IBCPacketTx are posted on “Hub” that proves the existence of an IBCPacket on “Zone1”. Say that IBCPacketTx has the following value:

- FromChainID: “Zone1”
- FromBlockHeight: 100 (say)
- Packet: an IBCPacket:
 - Header: an IBCPacketHeader:
 - SrcChainID: “Zone1”
 - DstChainID: “Zone2”
 - Number: 200 (say)
 - Status: AckPending
 - Type: “coin”
 - MaxHeight: 350 (say “Hub” is currently at height 300)
 - Payload: <The bytes of a “coin” payload>

Next, an IBCBlockCommit and IBCPacketTx are posted on “Zone2” that proves the existence of an IBCPacket on “Hub”. Say that IBCPacketTx has the following value:

- FromChainID: “Hub”
- FromBlockHeight: 300
- Packet: an IBCPacket:
 - Header: an IBCPacketHeader:

- SrcChainID: “Zone1”
- DstChainID: “Zone2”
- Number: 200
- Status: AckPending
- Type: “coin”
- MaxHeight: 350
- Payload: <The same bytes of a “coin” payload>

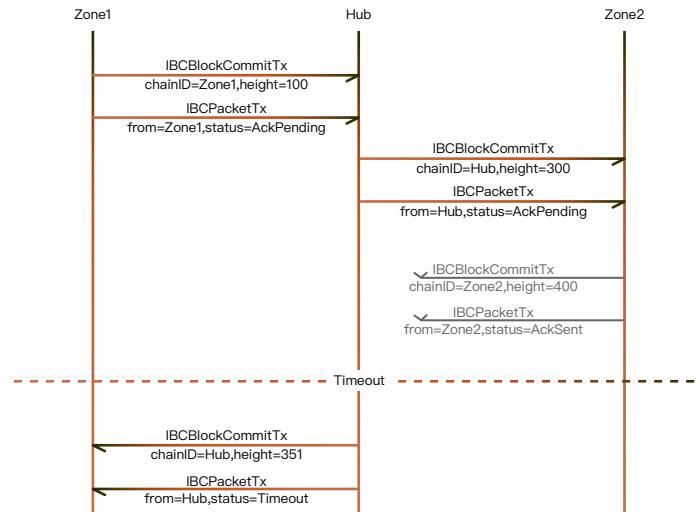
Next, “Zone2” must include in its app–hash an abbreviated packet that shows the new status of AckSent. An IBCBlockCommit and IBCPacketTx are posted back on “Hub” that proves the existence of an abbreviated IBCPacket on “Zone2”. Say that IBCPacketTx has the following value:

- FromChainID: “Zone2”
- FromBlockHeight: 400 (say)
- Packet: an IBCPacket:
- Header: an IBCPacketHeader:
- SrcChainID: “Zone1”
- DstChainID: “Zone2”
- Number: 200
- Status: AckSent
- Type: “coin”
- MaxHeight: 350
- PayloadHash: <The hash bytes of the same “coin” payload>

Finally, “Hub” must update the status of the packet from AckPending to AckReceived. Evidence of this new finalized status should go back to “Zone2”. Say that IBCPacketTx has the following value:

- FromChainID: “Hub”
- FromBlockHeight: 301
- Packet: an IBCPacket:
- Header: an IBCPacketHeader:
- SrcChainID: “Zone1”
- DstChainID: “Zone2”
- Number: 200
- Status: AckReceived
- Type: “coin”
- MaxHeight: 350
- PayloadHash: <The hash bytes of the same “coin” payload>

Meanwhile, “Zone1” may optimistically assume successful delivery of a “coin” packet unless evidence to the contrary is proven on “Hub”. In the example above, if “Hub” had not received an AckSent status from “Zone2” by block 350, it would have set the status automatically to Timeout. This evidence of a timeout can get posted back on “Zone1”, and any tokens can be returned.

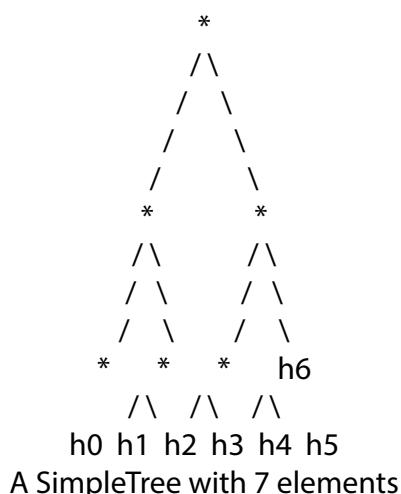


Merkle Tree & Proof Specification

There are two types of Merkle trees supported in the Tendermint/ Neutron ecosystem: The Simple Tree, and the IAVL+ Tree.

Simple Tree

The Simple Tree is a Merkle tree for a static list of elements. If the number of items is not a power of two, some leaves will be at different levels. Simple Tree tries to keep both sides of the tree the same height, but the left may be one greater. This Merkle tree is used to Merkle-ize the transactions of a block, and the top level elements of the application state root.



IAVL+ Tree

The purpose of the IAVL+ data structure is to provide persistent storage for key-value pairs in the application state such that a deterministic Merkle root hash can be computed efficiently. The tree is balanced using a variant of the AVL algorithm, and all operations are $O(\log(n))$.

In an AVL tree, the heights of the two child subtrees of any node differ by at most one. Whenever this condition is violated upon an update, the tree is rebalanced by creating $O(\log(n))$ new nodes that point to unmodified nodes of the old tree. In the original AVL algorithm, inner nodes can also hold key-value pairs. The AVL+ algorithm (note the plus) modifies the AVL algorithm to keep all values on leaf nodes, while only using branch-nodes to store keys. This simplifies the algorithm while keeping the merkle hash trail short.

The AVL+ Tree is analogous to Ethereum's Patricia tries. There are tradeoffs. Keys do not need to be hashed prior to insertion in IAVL+ trees, so this provides faster ordered iteration in the key space which may benefit some applications. The logic is simpler to implement, requiring only two types of nodes — inner nodes and leaf nodes. The Merkle proof is on average shorter, being a balanced binary tree. On the other hand, the Merkle root of an IAVL+ tree depends on the order of updates.

We will support additional efficient Merkle trees, such as Ethereum's Patricia Trie when the binary variant becomes available.

Transaction Types

In the canonical implementation, transactions are streamed to the Neutron hub application via the ABCI interface. The Neutron Hub will accept a number of primary transaction types, including SendTx, BondTx, UnbondTx, ReportHackTx, SlashTx, BurnNTNTx, ProposalCreateTx, and ProposalVoteTx, which are fairly self-explanatory and will be documented in a future revision of this paper. Here we document the two primary transaction types for IBC: IBCBlockCommitTx and IBCPacketTx.

IBCBlockCommitTx

An IBCBlockCommitTx transaction is composed of:

- ChainID (string): The ID of the blockchain
- BlockHash ([]byte): The block-hash bytes, the Merkle root which includes the app-hash
- BlockPartsHeader (PartSetHeader): The block part-set header bytes, only needed to verify vote signatures
- BlockHeight (int): The height of the commit
- BlockRound (int): The round of the commit
- Commit ([]Vote): The >2/3 Tendermint Precommit votes that comprise a block commit
- ValidatorsHash ([]byte): A Merkle-tree root hash of the new validator set
- ValidatorsHashProof (SimpleProof): A SimpleTree Merkle-proof for proving the ValidatorsHash against the BlockHash
- AppHash ([]byte): A IAVLTree Merkle-tree root hash of the application state
- AppHashProof (SimpleProof): A SimpleTree Merkle-proof for proving the AppHash against the BlockHash

IBCPacketTx

An IBCPacket is composed of:

- Header (IBCPacketHeader): The packet header
- Payload ([]byte): The bytes of the packet payload. Optional
- PayloadHash ([]byte): The hash for the bytes of the packet. Optional

Either one of Payload or PayloadHash must be present. The hash of an IBCPacket is a simple Merkle root of the two items, Header and Payload. An IBCPacket without the full payload is called an abbreviated packet.

An IBCPacketHeader is composed of:

- SrcChainID (string): The source blockchain ID
- DstChainID (string): The destination blockchain ID
- Number (int): A unique number for all packets
- Status (enum): Can be one of AckPending, AckSent, AckReceived, NoAck, or Timeout
- Type (string): The types are application-dependent. Neutron reserves the “coin” packet type
- MaxHeight (int): If status is not NoAckWanted or AckReceived by this height, status becomes Timeout. Optional

An IBCPacketTx transaction is composed of:

- FromChainID (string): The ID of the blockchain which is providing this packet; not necessarily the source
- FromBlockHeight (int): The blockchain height in which the following packet is included (Merkle-ized) in the block-hash of the source chain
- Packet (IBCPacket): A packet of data, whose status may be one of AckPending, AckSent, AckReceived, NoAck, or Timeout
- PacketProof (IAVLProof): A IAVLTree Merkle-proof for proving the packet’s hash against the AppHash of the source chain at given height

The sequence for sending a packet from “Zone1” to “Zone2” through the “Hub” is depicted in {Figure X}. First, an IBCPacketTx proves to “Hub” that the packet is included in the app-state of “Zone1”. Then, another IBCPacketTx proves to “Zone2” that the packet is included in the app-state of “Hub”. During this procedure, the IBCPacket fields are identical: the SrcChainID is always “Zone1”, and the DstChainID is always “Zone2”.

When “Zone1” wants to send a packet to “Zone2” through “Hub”, the IBCPacket data are identical whether the packet is Merkle-ized on “Zone1”, the “Hub”, or “Zone2”. The only mutable field is Status for tracking delivery.

Chapter 5

Release Notes

05

Release Notes

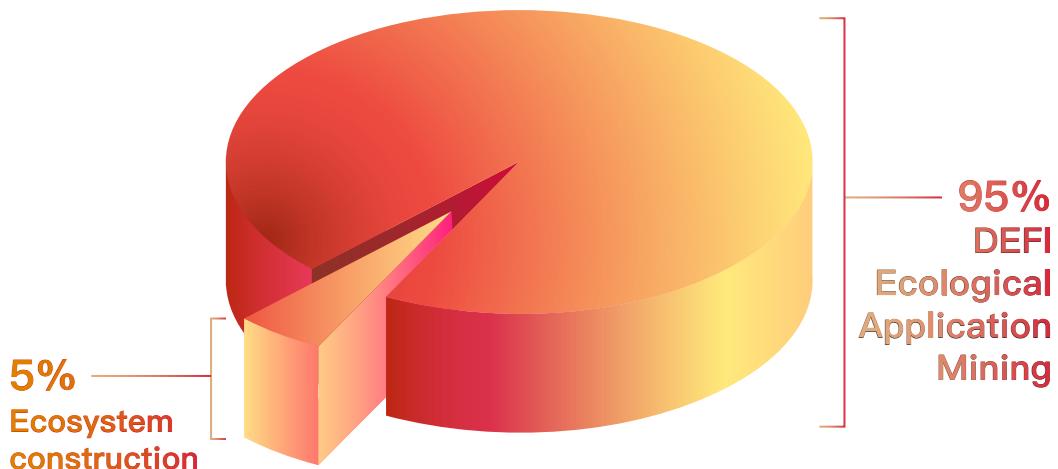
The current application of blockchain technologies are trying to issue tokens as a tool for economic incentives, promote the collaboration of various roles in the ecosystem. So, every blockchain project based on token incentives is trying to design a token-based economic system that can function well. Even an excellent project is hard to design a perfect economic model as in the digital of blockchain, the token economic model needs to be able to fully reflect the value of the real world, realize accurate model fitting based on the changes, but this vision is hard to achieve since there are too many variables to be considered, this makes the token economy of many projects are unstable, and ultimately failed. If the token economic system based on blockchain technology is regarded as a micro-perfect economic system, then the makers of system rules need to control the output channels, speed, quantity and etcetera to carry out macro-control. After the consideration of various factors, Neutron decided to adopt a constant deflation model to build a platform ecosystem, constant deflation can greatly simplify the interaction between the system platform and users, effectively realize resource isolation between usage rights and rights, allow the platform business unaffected by external economic fluctuations, ensure that the platform can operate continuously, stably, and healthily.

Issuing mechanism

Issuing main net	1. ERC20 2. BSC20 (Binance Smart Chain)
Issuance model	Constant deflation model
Total issuance	63000000

Issuance plan

The issuance of NTN token represents the opening of a decentralized multi-chain architecture, open the value interoperability of the global general industry, open up trust islands between platforms, realize a new situation of value sharing and win-win cooperation in a real sense. Once the NTN token is issued, it will never be issued again. In order to reflect its value and the true meaning of the deflation model, NTN is completely decentralized and open source, blockchain distributed accounting and issuance through planning and distribution in smart contracts, the plan is as below:



DEFI Ecological Application Mining: 60800000 (95%)

each miner participating in the pledge mining on the chain is an independent node, each node will form an on-chain LP in the corresponding node after completing the pledge and mining, and generate hashrate value. The computing power (T) will be the only value circulation and exchange for miners in the whole network mining.

Ecosystem construction: 3150000 (5%)

as the network ecological construction, the later ecology includes the founding team and the development team Neutron ecological chain construction, super node pledge, etc.

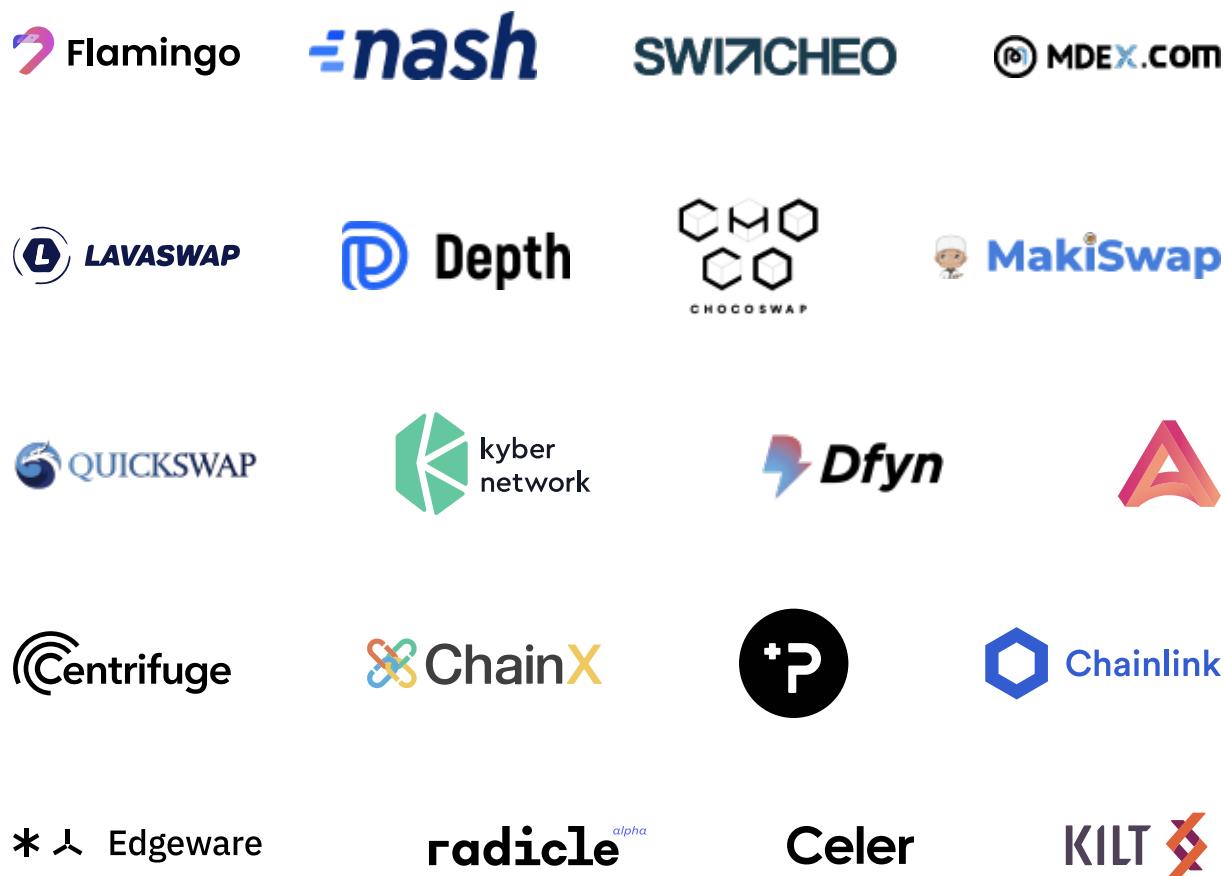
Chapter 6

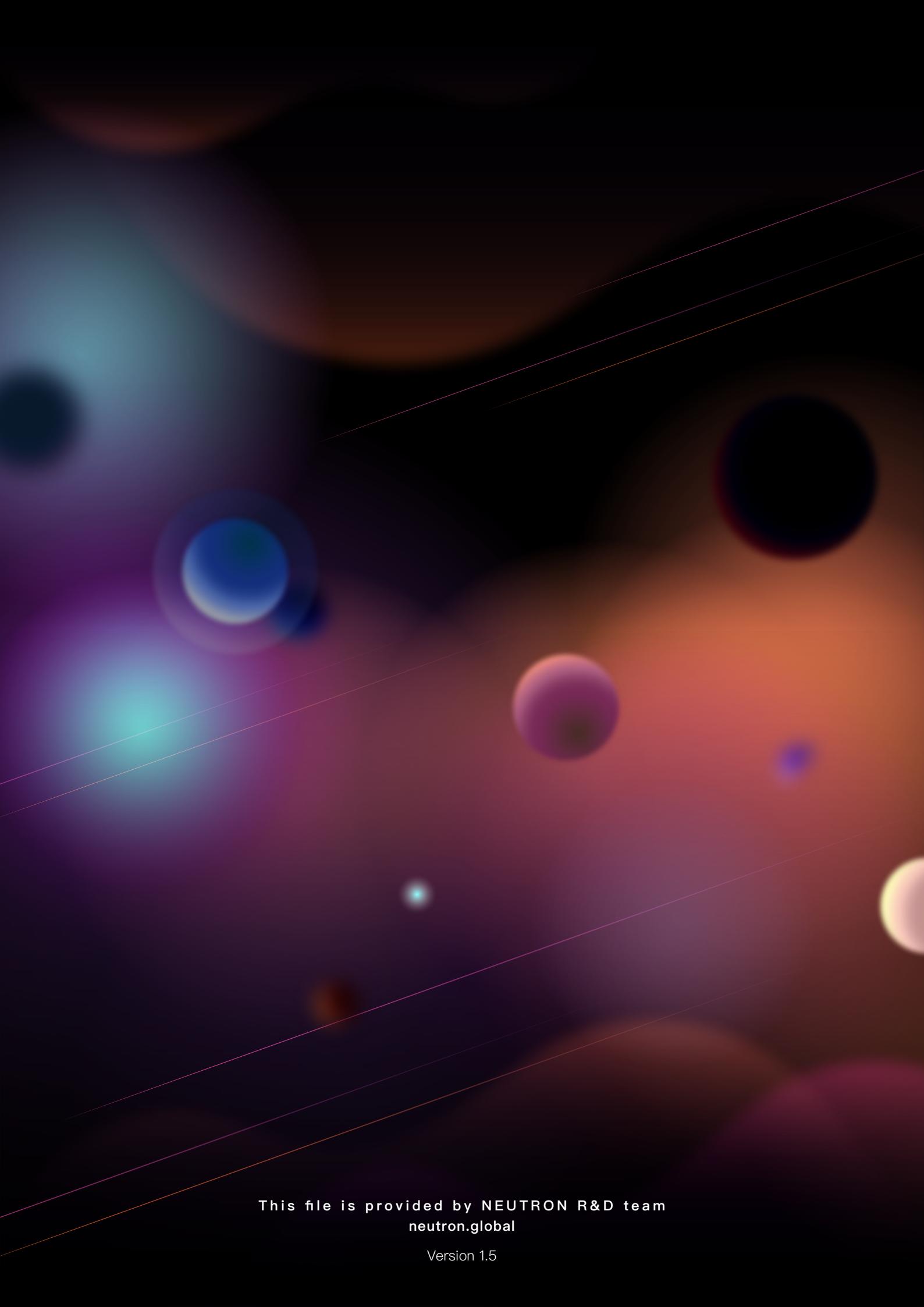
Neutron Partners

06

Neutron Partners

Neutron has formed strategic partnerships with the world's major laboratory giants through multiple dimensions, bring capital, advanced technology, and management experience to the platform, enhance the core competitiveness of enterprise technological progress and the ability to expand domestic and foreign markets, promote enterprise technological improvement and industry, achieve openness and win-win situation.



The background of the image is a dark, abstract space scene. It features several blurred, glowing circular shapes in various colors, including blue, purple, orange, and yellow. These circles of light are surrounded by thin, curved lines of the same color, creating a sense of motion and depth.

This file is provided by NEUTRON R&D team
neutron.global

Version 1.5