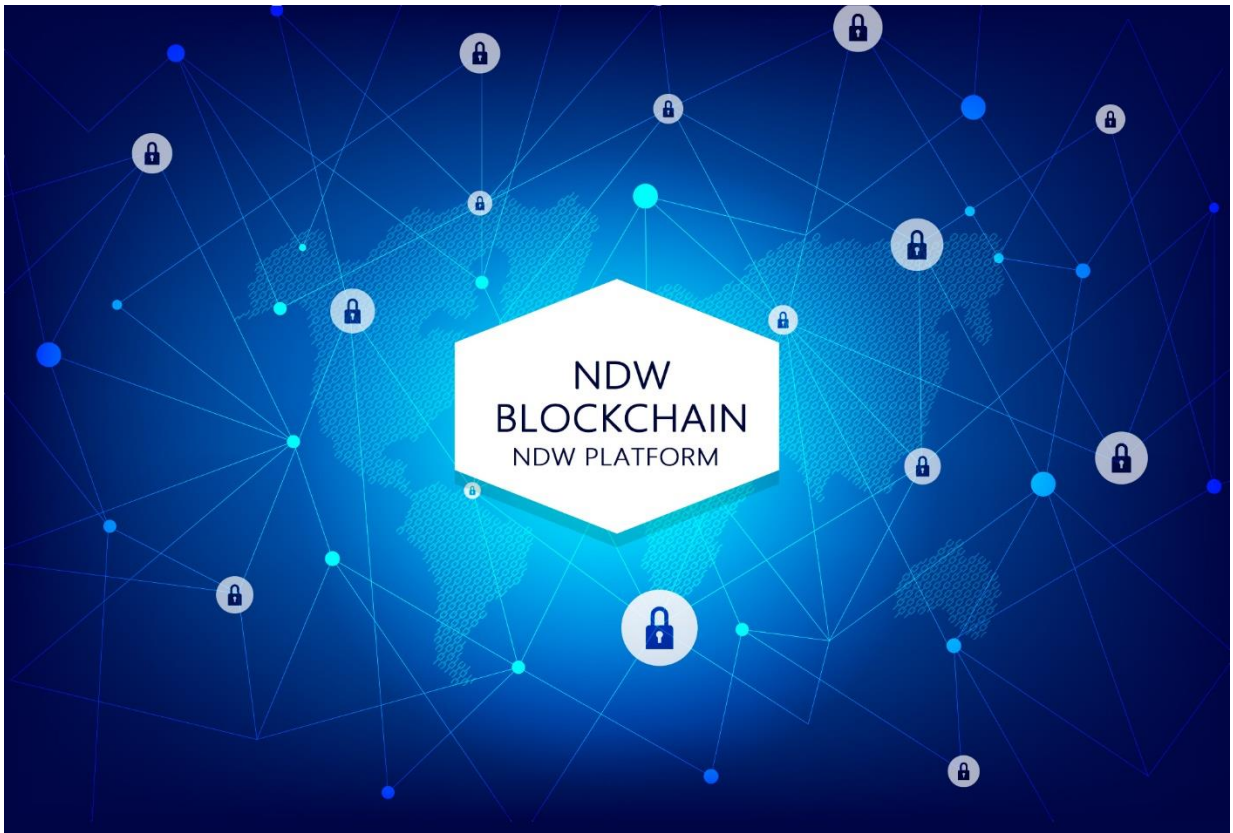


【WHITEPAPER】
NDW Network

NDW Development Management Department

2021.03

• Foreword

To many, “fintech” is a term simply associated with the trendy banking or payment services they use via their smartphone apps, or at the virtual counters of their banks. Internet banking and mobile payment applications are certainly important areas in the application of fintech, but they are far from the only ones. Other technologies, from artificial intelligence to big data analytics to virtual reality, are pushing out the possible frontiers of fintech every day. These technologies could bring a sea change to banking and payment services. The subject of this report – distributed ledger technology (DLT) – is just one key example of this beginning to happen.

DLT is perhaps better known as “blockchain”. It is essentially technology that supports networks of databases that enable participants to create, disseminate and store information in a secure and efficient manner. While database technologies are not new, what makes DLT special is that these networks of databases can operate smoothly and securely without necessarily being controlled and administered by a central party that is known and trusted by every participant.

The potential applications of DLT, as the fintech industry and many central banks and regulatory authorities soon found, are not limited to dealing in virtual currencies or commodities. The very fact that DLT allows information or records to be transferred and updated by network participants, and this to be done in a trustworthy, secure and efficient way, carries enormous potential. However, while the value proposition of DLT is gradually materialising, the use of DLT in financial services is also introducing new risks and giving rise to new legal and governance issues. These require in-depth study before its full potential can be realised. As a regulatory authority, we need to have a thorough understanding of the various governance, risk management and legal issues associated with DLT before its wider use begins in earnest.

• Blockchain Overview

Conceptual Description

Blockchain, or **distributed ledger** technology, is a database that is consensually shared, replicated, and synchronized.

To better understand the technical aspects of a blockchain, it is helpful to explain the concept through an example. When an individual deposits a sum of money into a banking institution, the individual trusts that the sum will be there until they decide to exchange it for goods or services. The individual trusts the bank will have an accurate record of the transaction, such as the amount, depositor, date, and time of the deposit. More broadly, society relies on central repositories, such as banks or governments, to collect, maintain, and protect the recorded actions of individuals or institutions.

Blockchain differs from centralized repositories in that it decentralizes the source of trust. An individual deposits funds into a digital wallet and the value is captured on the blockchain. If this individual purchases a digital song, the transaction is captured in the blockchain along with the change in fund level in the digital account. The bank is not required as a trusted third party. The trustworthy record is recorded in the blockchain shared by all the parties on the network.

Technical Description

The replication and storage of transactional data by each party, or node, on a blockchain network is known as a distributed ledger. Conflicts, or inaccuracies within the database, are automatically resolved with predefined ledger rules. The fundamental characteristics of the distributed ledger include:

> Operation with peer-to-peer networks,
> Decentralized transaction record keeping, Consensus or trust-based transactions, and Tamper resistance.

Blockchains, while similar to databases, are not used for general data storage, but rather hold information about transactions. Sometimes the blockchain will contain the transactions themselves or may include the proof a transaction is valid**.**

Blockchain Parts

Blockchains contain three core parts:

- **Block**: A list of recorded transactions over a period of time. Transactions can represent virtually any type of activity from registering a land deed to a single purchase. Any rules relating to the block itself are established when the network is first created. For example, the maximum number of transactions in a block or the size of each block can be limited.

- **Chain**: When the block reaches its maximum size of transactions, it is chained or linked to the preceding block through a **hash** as described in the section below. The hash value of one block is inserted into the next block. This makes a link between the new block and the previous block. Repeating a hash function on an unaltered block of data will always generate the same fixed-length value. If a block of data is altered, the resulting hash output will be different. A user can then see the hashes are different and will know the original block has been altered and may no longer be trustworthy.

- **Network**: The network is made up of nodes each containing a complete record of all transactions on a blockchain. No centralized "official" copy exists and no node is

"trusted" more than another. The data integrity is maintained by the blockchain being replicated on all of the nodes.

Think of a node as a cluster of servers running a blockchain. Node operators are incentivized to operate a node by receiving rewards for their efforts. For example, with cryptocurrencies, nodes compete to solve crypto-puzzles. The first node completing the puzzle has its solution verified by other nodes. Once the solution is verified, the node completing the puzzle adds the next block to the blockchain and is also rewarded with cryptocurrency for its effort. This process is called mining, with the resources involved called miners. Nodes are found across the globe and are challenging to operate. For example, the infrastructure of one cryptocurrency is supported by approximately 5000 nodes. Incentivized miners are required for cryptocurrency platforms, but are not necessarily part of other blockchain uses.

Behind the scenes, each blockchain has its own rules or algorithms governing how nodes validate transactions intended for entry into the blockchain. These rules are called a **consensus mechanism** and are established when the blockchain is created. By embedding a consensus mechanism, blockchains create a way for parties who do not know if they can trust each other to agree an entry should be added to the blockchain. This addresses the so-called **Byzantine Generals Problem**. Each blockchain has its own consensus mechanism depending on the type of transaction it is capturing. Some consensus mechanism are known as “proof of work”, “proof of space” or “proof of stake”. The mechanisms facilitate authenticity, or the immutability of transaction records.

To date, however, these blockchains have suffered from a number of drawbacks, including their gross energy inefficiency, poor or limited performance, and immature governance mechanisms. Proposals to scale Bitcoin’s transaction throughput, such as Segregated-Witness is vertical scaling solutions that remain limited by the capacity of a single physical machine, in order to ensure the property of complete auditability. The Lightning Network can help scale Bitcoin transaction volume by leaving some transactions off the ledger completely, and is well suited for micropayments and privacy-preserving payment rails, but may not be suitable for more generalized scaling needs.

An ideal solution is one that allows multiple parallel blockchains to interoperate while retaining their security properties. This has proven difficult, if not impossible, with proof-of-work. Merged mining, for instance, allows the work done to secure a parent chain to be reused on a child chain, but transactions must still be validated, in order, by each node, and a merge-mined blockchain is vulnerable to attack if a majority of the hashing power on the parent is not actively merge-mining the child. An academic review of alternative blockchain network architectures is provided for additional context, and we provide summaries of other proposals and their drawbacks in Related Work.

NDW
Nexus Digital Worth

A new database that integrates database and blockchain That is NDW

What is NDW

AI (artificial intelligence), BigData, IoT, quantum computers, and 5G.

Many advanced technologies, which have been called future technologies, are being researched and evolved every day to become reality.

NDW is a blockchain database for the next generation, which was originally designed and developed by NDW development team as an integrated management system for distributedcomputing, which is the infrastructure of these latest technologies.

As an infrastructure technology of a new era, we will protect important customer data,accelerate the evolution of solutions, and scale up our business.

That is the mission of NDW.

NDW has succeeded in achieving ultra-high speed by solving the three major existing bottlenecks in the blockchain.

In addition, the concept of high tamper resistance, zero downtime, and traceability, which could not be realized from the viewpoint of cost with the database, is acquired from the characteristics of the blockchain, and overwhelming usability that combines the advantages of the database and the blockchain is realized.

・Benefits of Database

Super fast approval
Significantly improved the approval speed, which was a bottleneck, and realized approval in 0.2 seconds per transaction.

High TPS by parallel processing
Eliminate the maximum number of processing cases per hour, which was a problem of blockchain, by parallel processing

Payment finality
Implemented finality processing, which was an issue for blockchain

・Benefits of Blockchain

Tamper resistance
High tamper resistance using blockchain technology

Zero downtime
Minimize downtime with distributed computing

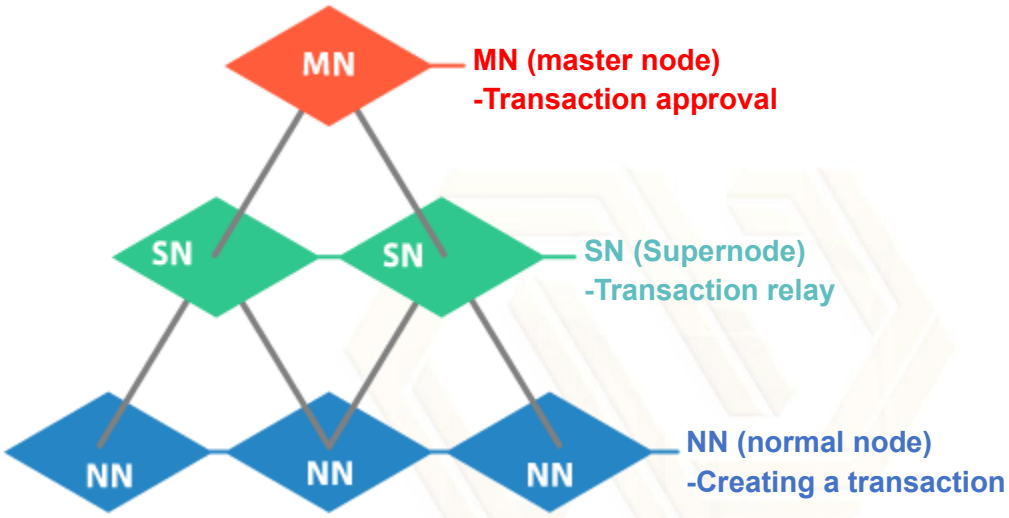
Cost reduction
Data server maintenance cost reduction
Reduction of learning cost by SQL conversion

Superiority of NDW

NDW		Compare items		Traditional blockchain
Approval of 0.2 seconds per transaction	○	Admit speed	×	**Approval time of 3 seconds to 10 minutes**
Approval with finality	○	Finality	×	**No finality**
Processing capacity of 40 million cases per second	○	TPS	×	**Processing capacity of 1 to 1000 cases per second**
Yes	○	Tamper resistance	○	**Yes**
Yes	○	Zero downtime	○	**Yes**
Low learning cost due to SQL compatibility	○	Cost reduction	×	**Requires acquisition of a dedicated language**
Versatile as a database	○	Versatility	×	**Limited usage due to specifications**
Low power consumption	○	power consumption	×	**High power consumption by mining**
Possible	○	Emergency control	×	**impossible**

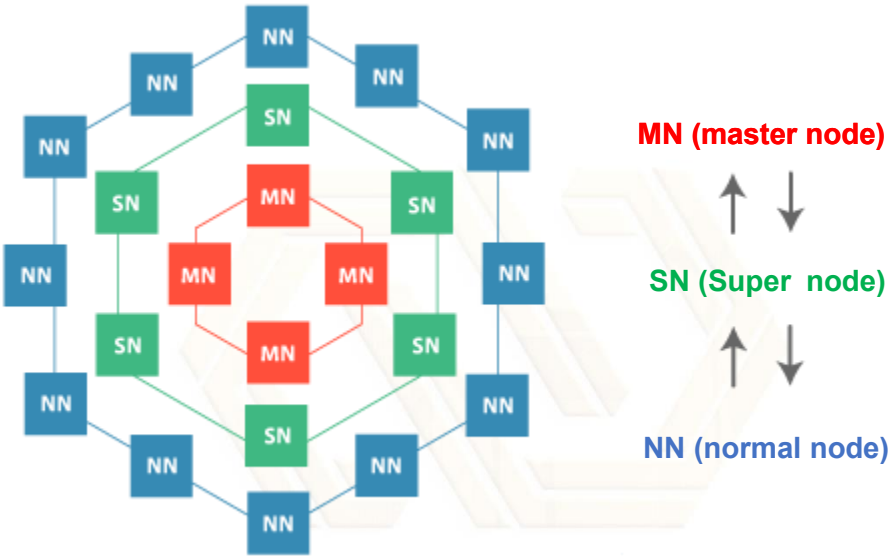
How NDW works

Introducing some of the systems that make up NDW
Node division of roles and optimization of processing by hierarchical structure



In a general blockchain, all nodes play the same role, so it is not possible to reduce inefficient processing such as duplication of the same work on different nodes. For this reason, conventional blockchains have not yet realized the implementation of speeds that can withstand business use.
In NDW, some of the centralized ideas used in the database are incorporated, and the nodes that make up the network are divided into three for each role.
We have built a hierarchical structure that optimizes transaction distribution, relaying, and approval processes by allocating each node to each layer.
As a result, by optimizing the transmission and processing of information and applying the machine power of each node to appropriate processing, we succeeded in dramatically improving the throughput, which was a problem in the conventional blockchain.
We have achieved a processing speed that can be used for business.

Streamlining information transmission through a circular network

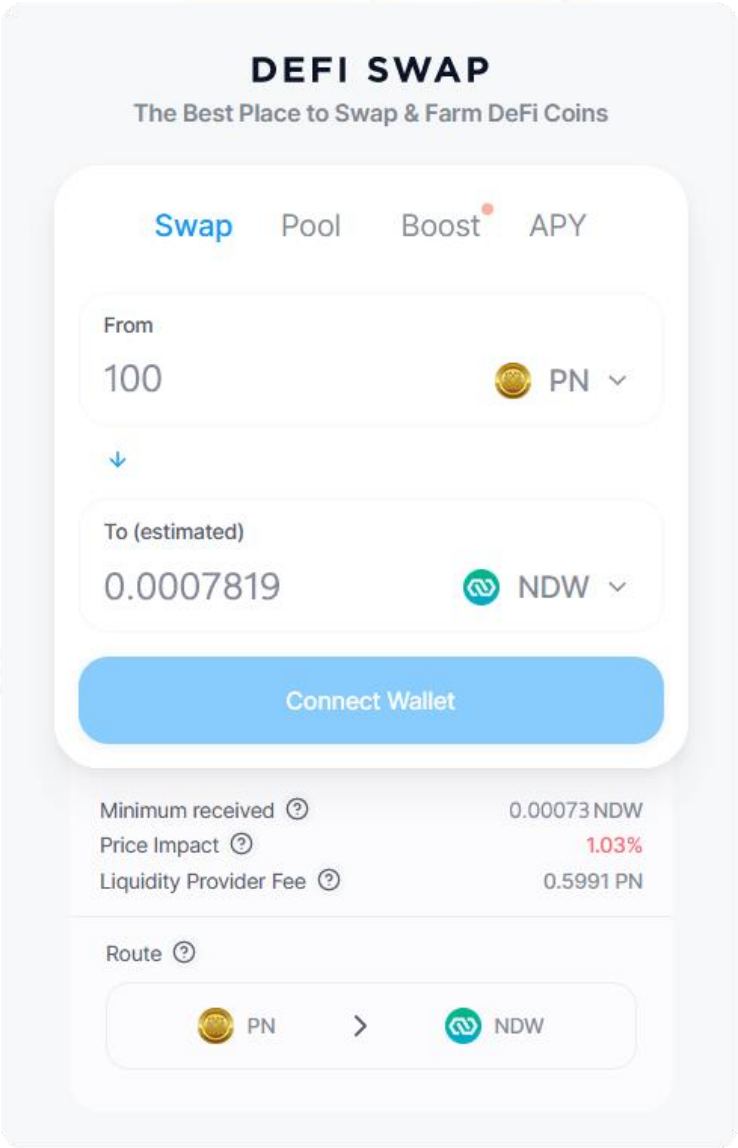


In the random network used in conventional blockchain systems, the larger the network scale, the longer it takes to transmit information, which hinders the improvement of processing speed.
The NDW network is built with a circular network structure centered on nodes that have the authority to approve data, minimizing the number of relays from the center to the end and optimizing data traffic.
Information in the network can be transmitted quickly and widely.
In addition, regardless of the size of the network, the arrangement from the center (MN) to the end (NN) of the ring remains constant and automatically expands or contracts, so it can be introduced regardless of the size of the system.

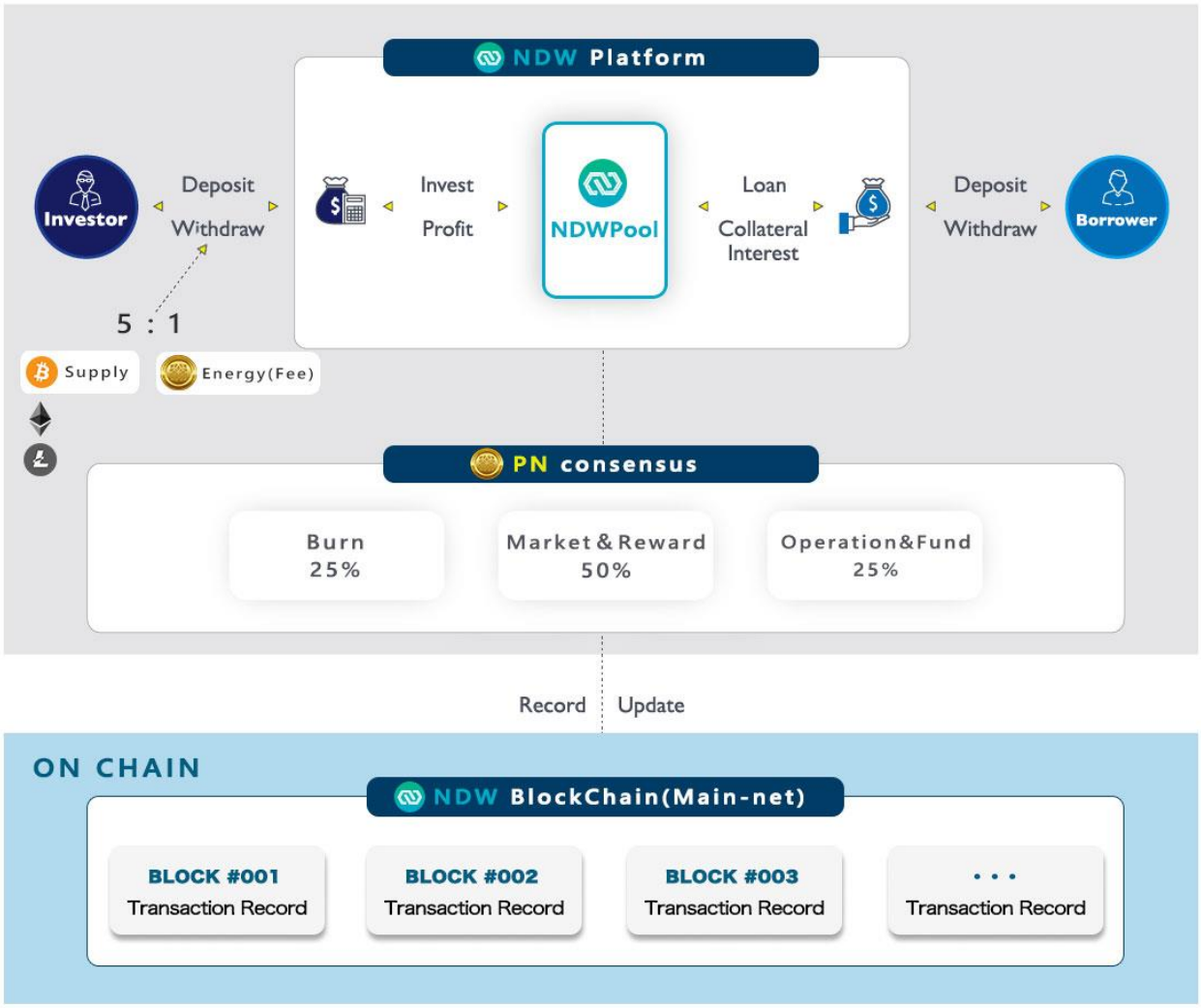
NDW Pool

The protocol is implemented as a set of “**smart contracts”** on top of the NDW blockchain. Smart contracts guarantee safety and do not require a middleman.

Each contract stores the optimised base curves using the corresponding parameters of each currency. This means that there is a mathematical function which determines the interest rate of each asset pool, with the interest rate changing based on the amount of borrowed funds and the total liquidity (i.e. utilisation) of the asset pool.



NDW Pool is a decentralised non-custodial liquidity protocol where users can participate as depositors or borrowers. Depositors provide liquidity to the market to earn a passive income, while borrowers are able to borrow in an over-collateralised (perpetually) or under-collateralised (one-block liquidity) fashion.



NDW Network
ClossChain

NDW network provides a uniform solution to cross-chain communication that meets the needs of both platform developers – no integration work is required from them, and application builders – one simple protocol and API to access global liquidity and communicate with the entire ecosystem.

NDW network consists of a decentralized network which bridges blockchain ecosystems that speak different languages and a protocol suite with APIs on top, making it easy for applications to perform cross-chain requests. The network connects existing stand-alone blockchains such as Bitcoin, Stellar, Terra, Algorand, and interoperability hubs such as solutions like Cosmos, Avalanche, Ethereum, and Polkadot. Our mission is to enable application developers to build such apps easier using a universal protocol and API without rolling out their proprietary cross-chain protocols underneath or rewriting applications as new bridges are developed. Towards this, we designed a protocol suite that includes Cross-Chain Gateway Protocol and Cross-Chain Transfer Protocol.

A core component of the network are the underlying decentralized protocols. Validators collectively maintain the NDW network and run the nodes that secure the NDW blockchain. They are elected through a delegation process by the users. Validators receive voting power pro-rata according to the stake delegated to them. The validators reach consensus on the state of multiple blockchains that the platform is connected to. The blockchain is responsible for maintaining and running the cross-chain routing and transfer protocols. Governance rules allow network participants to enact protocol decisions such as which blockchains to bridge and which assets to support.
NDW blockchain follows a Delegated Proof-of-Stake (DPoS) model similar to Cosmos Hub. Users elect validators who must bond their stake to participate in the consensus and maintain high-quality service. The DPoS model allows maintenance of large decentralized validator set and robust incentives to guarantee that the validators are responsible for maintaining bridges and shares of cryptographic threshold schemes. As part of consensus, validators run light-client software of other blockchains, allowing them to verify the state of other blockchains. The validators report these states to the NDW blockchain, and once enough of them report, the state of Bitcoin, Ethereum, and other chains is recorded on NDW.

Subsequently, the NDW base layer is aware of the state of external blockchains at any point in time, creating the “incoming bridges” from other blockchains. The validators collectively maintain threshold signature accounts on other blockchains (e.g., 80% of validators must approve and co-sign any transaction out of it), which allows them to lock and unlock assets and state across chains and to post state on other blockchains, the “outgoing bridges.” Altogether, one can view the NDW network as a decentralized crosschain read/write oracle.

The remainder of the document describes preliminaries and building blocks behind the network, some technical details of the network, cross-chain gateway protocol, and cross-chain transfer protocol.

Preliminaries

• Notation and Assumptions

Let V^r denote the set of NDW validators at round R. Each validator has a *weight*, a number in $(0,1]$ denoting the voting power of that particular validator. The weights of all validators add up to 1. A validator is *correct* if she runs a node that is consistent with the rules of the NDW protocol. To finalize blocks, or to sign cross-chain requests, NDW requires correct validators of total weight $> F$. We call the parameter $F \in [0.5,1]$ the *protocol threshold*.

NDW can be based on an *instant finality Delegated-Proof-of-Stake* blockchain. The validators run *Byzantine Fault Tolerant (BFT) consensus* at each round i to finalize the i_{th} block. Once the i_{th} block is finalized, new BFT consensus is run to finalize the $i + 1_{th}$ block, and so on. The validators are elected through stake delegation. A user with some stake may elect to run a validator node, or delegate their voting power (stake) to an existing validator, who then votes on their behalf. The validator set can be updated, validators join/leave the set, and users delegate/undelegate their voting power.

Different blockchains work under different network assumptions. *Synchronous communication* means that there is a fixed upper bound Δ on the time messages take to be delivered, where Δ is known and can be built into the protocol. *Asynchronous communication* means that messages may take arbitrarily long to be delivered, and it is known that BFT protocols cannot be built for asynchronous networks even in the presence of just one malicious validator. A realistic compromise between synchrony and asynchrony is the assumption of *partially synchronous communication*. The network may be completely asynchronous until some unknown global stabilization time (GST), but after GST communication becomes synchronous with a known upper bound Δ .

Typical blockchains work under the assumption of $> F$ correct validators. For synchronous networks $F = 1/2$ is typically set, but for the weaker assumption of a partially synchronous network $F = 2/3$. Bitcoin, its forks, and the current Proof-of-Work version of Ethereum only work assuming synchrony. Others like Algorand and Cosmos only require partial synchrony. When connecting chains through NDW, the connection works assuming the strongest network assumptions out of these chains, which is synchrony in the case of connecting Bitcoin and Cosmos, for instance. The NDW blockchain itself works in a partially synchronous setting and thus requires $F = 2/3$, but it is possible to improve the threshold requirement by assuming that other existing blockchains are secure and leveraging their security.

• Cryptographic Preliminaries

Digital Signatures. A *digital signature scheme* is a tuple of algorithms (*Keygen,Sign,V erify*). *Keygen* outputs a pair of keys (*PK,SK*). Only the owner of *SK* can sign messages, but anyone can verify the signatures given the public key *PK*. Most blockchain systems today use one of the standard signature schemes such as ECDSA, Ed25519, or a few of their variants.

Threshold Signatures. A *threshold signature scheme* enables a group of *n* parties to split a secret key for a signature scheme in such a way that any subset of *t* + 1 or more parties can collaborate to produce a signature, but no subset of *t* or fewer parties can produce a signature or even learn any information about the secret key. The signatures produced by the threshold protocols for ECDSA and EdDSA look identical to the signatures produced by the stand-alone algorithms.

A threshold signature scheme replaces the *Keygen* and *Sign* algorithms for an ordinary signature scheme with distributed *n*-party protocols *T.Keygen*, *T.Sign*. These protocols typically require both a public broadcast channel and private pairwise channels among the parties, and they typically involve several rounds of communication. After successful completion of *T.Keygen* each user holds a share s_i of a secret key SK and the corresponding public key PK. The *T.Sign* protocol allows these parties to produce a signature for a given message that is valid under public key PK. This signature can be verified by anyone using the *Verify* algorithm of the original signature scheme.

• Properties of Threshold Signatures

There are several properties a threshold scheme might have that are especially desirable for decentralized networks:

Security against a dishonest majority

Some threshold schemes have the restriction that they are secure only when a majority of the *n* parties are honest. Thus, the threshold parameter *t* must be smaller than *n/*2 . This restriction is typically accompanied by the fact that 2*t* + 1 honest parties are needed to sign, even though only *t*+1 corrupted parties can collude to recover the secret key. Schemes that do not suffer from this restriction are said to be *secure against a dishonest majority*.

Cross-chain platforms must maximize the safety of their networks and be able to tolerate as many corrupted parties as possible. Thus, schemes that can tolerate dishonest majority are necessary.

Pre-signatures, non-interactive online signing

In an effort to reduce the burden of communication upon the parties to sign a message, several recent protocols have identified a significant portion of the work for a signature that can be done “offline”, before the message to sign is known. The output of this offline phase is called a *pre-*

signature. The production of pre-signatures is viewed as a separate protocol *T.Presign* distinct from *T.Keygen* and *T.Sign*. The outputs of the pre-signature protocol must be kept private by the parties until they use them at the signing phase. Later, when the message to sign becomes known, only a small amount of additional “online” work remains to be done in *T.Sign* in order to complete the signature.

> The online *T.Sign* phase does not require any communication among the parties. Each party simply does a local computation on the message and pre-signature and then announces her share s_i of the signature. (Once public, these signature shares $s_1,...,s_{t+1}$ are easily combined by anyone to reveal the actual signature s.) This property is called *non-interactive online signing*.

Robustness

Threshold schemes guarantee only that a subset of malicious parties cannot sign messages or learn the secret key. This guarantee does not, however, preclude the possibility that bad actors can block everyone else from producing keys or signatures. In some schemes, malicious behaviour by even a single party can cause *T.Keygen* or *T.Sign* to abort with no useful output. The only recourse is to restart the protocol, possibly with different parties.

Instead, for decentralized networks, we want *T.Keygen* and *T.Sign* to succeed if at least $t + 1$ of the parties are honest, even if some malicious parties send malformed messages or drop messages in the protocols. This property is called *robustness*.

Fault attribution

The ability to identify bad actors in *T.Keygen* or *T.Sign* is called *fault attribution*. Without fault attribution it is difficult to reliably exclude or punish bad actors, in which case the costs imposed by bad actors must be borne by everyone. This property is also important for decentralized networks where malicious behavior should be identifiable and economically disincentivized via slashing rules.

Security in concurrent settings

The signature scheme needs to be secure in a concurrent setting, where multiple instances of the keygen and signing algorithms can be involved in parallel. (Drijvers et al. for instance, showed an attack against Schnorr multisignature schemes in these settings). There are versions of both ECDSA and Schnorr schemes that satisfy these properties

NDW Network

• Designing an Open Cross-Chain Network

The bridges that NDW network maintains are backed up by threshold accounts such that (almost) all validators must collectively authorize any cross-chain request. Designing a network where anyone can participate to secure these bridges requires meeting the following technical requirements:

- *Open membership.* Any user should be able to become a validator (following the rules of the network).
- *Updates to membership.* When a validator leaves the system honestly, their key needs to be revoked appropriately.
- *Incentives and slashing.* Malicious validators should be identifiable and their actions must be identified and addressed by the protocol.
- *Consensus.* Threshold schemes on their own are defined as stand-alone protocols. To propagate messages between nodes we need both broadcast and point-to-point private channels. Moreover, validators need to agree on the latest state of each invocation of threshold schemes since they often have multiple round of interactions.
- *Key-management.* Just as ordinary validators in any PoS system must carefully guard their keys, so too must NDW validators guard their threshold shares. Keys need to be rotated, split between online and offline parts, etc.

NDW starts with Delegated Proof-of-Stake model, where the community elects a set of validators to run the consensus. Note that standard threshold schemes treat every player identically and have no notion of “weight” in the consensus. Hence, the network must adapt them to take validators’ weight into account. A simple approach is to assign multiple threshold shares to larger validators. Outlined below are three basic functions that validators collectively perform.

- *Threshold Key Generation.* Existing threshold key generation algorithms for standard blockchain signature schemes (ECDSA, Ed25519) are interactive protocols between multiple participants. A special transaction on the NDW network instructs the validators to commence execution of this stateful protocol. Each validator runs a threshold daemon process that is responsible for the secure keeping of the secret state. For each phase of the protocol:

  1. A validator keeps the state of the protocol in its local memory.
  2. It calls the secret daemon to generate the messages as per the protocol description for other validators.
  3. It propagates the messages either via the broadcast or via the private channels to other validators.

4. Each validator executes state transition functions to update its state, proceed to the next phase of the protocol, and repeat the above steps.

At the end of the protocol, a threshold public key is generated on the NDW chain, and it can be displayed back to the user (e.g., for deposits) or to the application that generated the initial request.

- *Threshold Signing*. Signing requests on the NDW network are processed similarly to the key-generation requests. These are invoked, for instance, when a user wants to withdraw an asset from one of the chains. These are interactive protocols, and state transition between the rounds is triggered as a function of the messages propagated via the NDW blockchain view and every validator's local memory.

- *Handling Validator Membership Changes.* The validator set needs to be rotated periodically to allow for new stakeholders to join the set. Upon a validator set update, we need to update the threshold key to be shared across the new set. Thus if we allowed anyone to join at any time, we would have to update the threshold key very frequently. To prevent this, we rotate validators every T blocks. Within intervals of T rounds, the set V^R and the threshold key are fixed. At every round that is an integral multiple of the parameter T, we update the validator set as follows:

  1. At any round R, the NDW state keeps track of the current validator set V^R. $V^{R+1} = V^R$ unless $R + 1$ is a multiple of T.
  2. During rounds $((i-1)T, iT]$, users post bonding/unbonding messages.
  3. At the end of round iT, these messages are applied to V^{iT-1} to get V^{iT}.

- *Threshold Key Generation and Signing in the Presence of Rotating Validators.* NDW blockchain may issue a request for a new key or a threshold signature at round R. The signing process takes longer than one round, and we don't want to slow down consensus, so we request that the signature is produced before round $R + 10$ starts. In particular, validators start round $R + 10$ only after seeing a certificate for round $R+9$ and a signature for each keygen/signature request issued at round R. The outcome of all round R requests must be included in block $R+11$. In other words, a round R block proposal that does not contain the outcomes from a round $R - 11$ is considered invalid, and validators don't vote on it. To ensure that all threshold messages are signed before a validator set update, NDW does not issue any threshold requests during a round equal to $-1, -2, ..., -9$ modulo T.

Network Security

The security of blockchain systems relies on various cryptographic and game theoretic protocols, as well as the decentralization of the network. For instance, in proof-of-stake blockchains, without the proper incentives validators may collude and rewrite the history, stealing other users' funds in the process. In proof-of-work networks, without sufficient decentralization, it is quite easy to create long forks and double spend, as the multiple attacks on Bitcoin Gold and Ethereum Classic have proven.

Most of the research on blockchain security has focused on sovereign chains. But once chains interoperate, new attack vectors have to be considered. For instance, assume that Ethereum talks to a small blockchain X through a direct bridge controlled by two smart contracts, one on Ethereum and one on X. Besides the engineering challenges we summarized in Section, one must decide what happens when the trust assumptions of X are violated. In this case, if ETH has moved to X, the validators of X may collude to forge a history of X where they hold all the ETH, post the forged consensus proofs on Ethereum and steal the ETH. The situation is even worse when X is connected with multiple other chains through direct bridges, where if X forks the effects propagate through every bridge. Setting up recovery governance guidelines for each pairwise bridge is an overwhelming task for any individual project.

NDW network addresses the security concerns using the following mechanisms:

- *Maximum Safety.* NDW sets the safety threshold to 90%, meaning that almost all validators will need to collude to withdraw any funds that are locked by its network or forge state proofs[1]. In practice, it has been observed that PoS validators have very high up-time (close to 100%), assuming they are properly incentivized. Hence, NDW network will produce blocks even despite this high threshold. However, in the rare case that something goes wrong and the network stalls, the network needs robust fall-back mechanisms to reboot the system described next.

- *Maximum Decentralization.* Since the network uses threshold signature schemes, the number of validators can be as large as possible. The network is not bounded by the number of validators we can support, transaction limits or fees that would arise from using, for instance, multi-signatures on different chains where the complexity (and fees) increase linearly with the number of validators.[2]

- *Robust Fall-back Mechanisms.* The first question that must be addressed in a network with high safety thresholds as above is what happens when the network itself stalls. Suppose NDW network itself stalls. Can we have a fall-back mechanism that would allow users to recover their funds? To address any potential stall of the NDW network itself, each threshold bridge account on a blockchain X that the NDW validators collectively control has an "emergency unlock key". This key can be shared across thousands of parties and may even be a custom key for blockchain X that is shared across the community of that chain. Hence, if NDW network stalls, this key will act as a fall-back and enable recovery of the assets (see below for more details).

- *Maximum Decentralization of Fall-Back Mechanisms.* This fall-back mechanism includes a secondary *recovery set* of users, in which just anyone can participate without any cost. These users do not need to be online, run nodes, or coordinate with each other. They are only "called on duty" if NDW network stalls and cannot recover. The network's security is enhanced by a very high threshold on the primary validator set and a maximally decentralized secondary recovery set.
- *Shared Governance.* A common protocol governs the NDW network. Collectively, the users can vote on which chain should be supported through its network. The network will also allocate a pool of funds that can be used to reimburse users in case of unexpected emergencies, controlled via the governance protocols as well.

Various security mechanisms are discussed below.

Fall-Back Mechanisms. When NDW stalls due to the high threshold, an "emergency unlock key" takes control of the network. There are multiple ways to instantiate this unlock key, and certain chains/applications may opt to utilize a different variation for the "recovery set" or opt-out completely:[3]

- *Option a.* Share the key across foundations of blockchain projects and reputable people in the community.
- *Option b.* Share the across parties elected through the delegated PoS mechanism.
- *Option c.* For accounts managing assets and information for chain/application X, share a custom key across the stakeholders/validators of X. Assuming X has governance mechanisms in place, the same governance mechanisms can be applied to determine a course of action if NDW stalls.

Now, given the recovery users' identities and their public keys, a simple protocol generates shares of the recovery key that no-one knows. Moreover, the users of recovery set do not need to be online until called to recover via the governance mechanisms. Following the standard distributed key-generation protocols, each NDW validator shares a random value. The recovery secret key is generated by summing up these values. Instead of doing the summations in the clear, all shares are encrypted under the public keys of the recovery users and then added up homomorphically (this assumes additively homomorphic encryption and an additional layer of zero-knowledge, both of which are easily obtainable). The result of this protocol is a recovery public key *RPK* and potentially thousands of encryptions (under the public keys of the recovery users) of the shares of the corresponding secret key $Enc_i(s_i)$ that are distributed to their owners (e.g., posted on chain). NDW bridge contracts include an option to recover funds using *RPK* under certain conditions. Finally, it is also possible to update this recovery key and even change the set of users holding its shares without requiring any work from the participating shareholders.

If chain X that is connected to NDW breaks, there are a couple options:

- Impose limits on the USD value of assets that can be moved in/out of X on any single day. Thus a malicious chain X can only steal a small fraction of all assets that are bridged to it

before NDW validators detect this, and the governance mechanisms from the following bullets kick in.

- The NDW governance module can be used to vote on what happens in those situations. For instance, if there is a benign bug and the community restarts X, NDW governance can determine to restart the connection from where it left off.
- If ETH had moved to X, a custom Ethereum recovery key can determine what happens to the ETH assets.

Cross-Chain Gateway Protocol (CGP)

In this section, we explain the cross-chain gateway protocol and routing mechanisms on two core examples common between many applications' needs:

State synchronization. Post information about the state of a source blockchain *S* into the state of a destination blockchain *D*.

(For example, post a Bitcoin block header to the Ethereum blockchain.)

Asset transfer. Transfer a digital asset from *S* to *D* and back again.

(For example, transfer bitcoins from the Bitcoin blockchain to the Ethereum blockchain, and then back to the Bitcoin blockchain.)

For simplicity we assume that chain *D* has at least minimal support for smart contracts but *S* can be any blockchain whatsoever.

• Accounts on other chains

To bridge different chains, threshold accounts are created on each chain that control the flow of value and information across them. For chain *Chain*, denote the account by $Chain_{NDW}$.

Bitcoin account. For Bitcoin and other non-smart contract chains NDW validators create a threshold ECDSA key as per section. This key controls the ECDSA account on Bitcoin, and is the destination address where users send deposits. Personalized threshold keys may be created per user request. The key may be updated periodically, and the latest key and personalized keys can be found by querying an NDW node.

Threshold bridge account on chains with smart contracts. Denote the chain by SC. the validators create a threshold ECDSA or ED25519 key as per section, depending on which key type the chain supports. We denote this key by PK_{NDW}, when there is no ambiguity as to which chain we are referring to. This key controls a smart contract account on SC, denoted by SC_{NDW}, and any application on SC can query SC_{NDW} to learn the PK address of that key. This way, any SC application can recognize messages signed by SK_{NDW}. The protocol also needs to account for rotating values of PK_{NDW}. This happens as follows:

1. Initialize SC_{NDW} on SC. It stores PK_{NDW} as part of its state, which is initialized as its genesis value on NDW. SC_{NDW} also includes rules for updating the PK.
2. To update PK_{NDW}, a transaction of the format $(update, PK_{new})$ must be submitted with a signature from the current SK_{NDW}. Then the contract sets $PK_{NDW} = PK_{new}$.

3. Every time the validators update the threshold key for SC from PK^i to PK^{i+1}, NDW requests that validators use SK^i to sign $(update, PK^{i+1})$. Subsequently this signature is posted to SC_{NDW} which updates PK_{NDW}.

• State synchronization

Let q_S denote an arbitrary question about the state of chain S. Examples of such questions include:

- *"At what block round, if any, did a transaction tx appear?"*
- *"What is the value of a certain data field?"*
- *"What is the Merkle root hash of the entire state of S at block round 314159?"*

Let a_S denote the correct answer to q_S and suppose an end-user or application demands that a_S be posted to chain D. NDW network meets this demand as follows:

1. The user posts a request q_S on one of the bridge accounts (which are subsequently picked up by the the validators) or directly to the NDW blockchain.
2. As part of NDW consensus, each validator must run node software for chains S, D. NDW validators query the API of their chain S node software for the answer a_S and report the answer to the NDW chain.
3. Once > F weighted validators report the same answer at round R, NDW asks validators to sign a_S.
4. Using threshold cryptography the validators sign a_S. The signature is included in block R + 11.
5. Anyone can take the signed value a_S from block R + 11 and post it to D.
6. The request has been serviced. Any application on D may now take the signed value a_S, query D_{NDW} for the latest PK_{NDW}, and verify that the signature of a_S corresponds to PK_{NDW}. The validators also post a_S to the bridge account on chain D, which applications can retrieve.

• Cross-Chain Asset Transfer

The network enables cross-chain transfers of digital assets by extending the state synchronization workflow of Section.

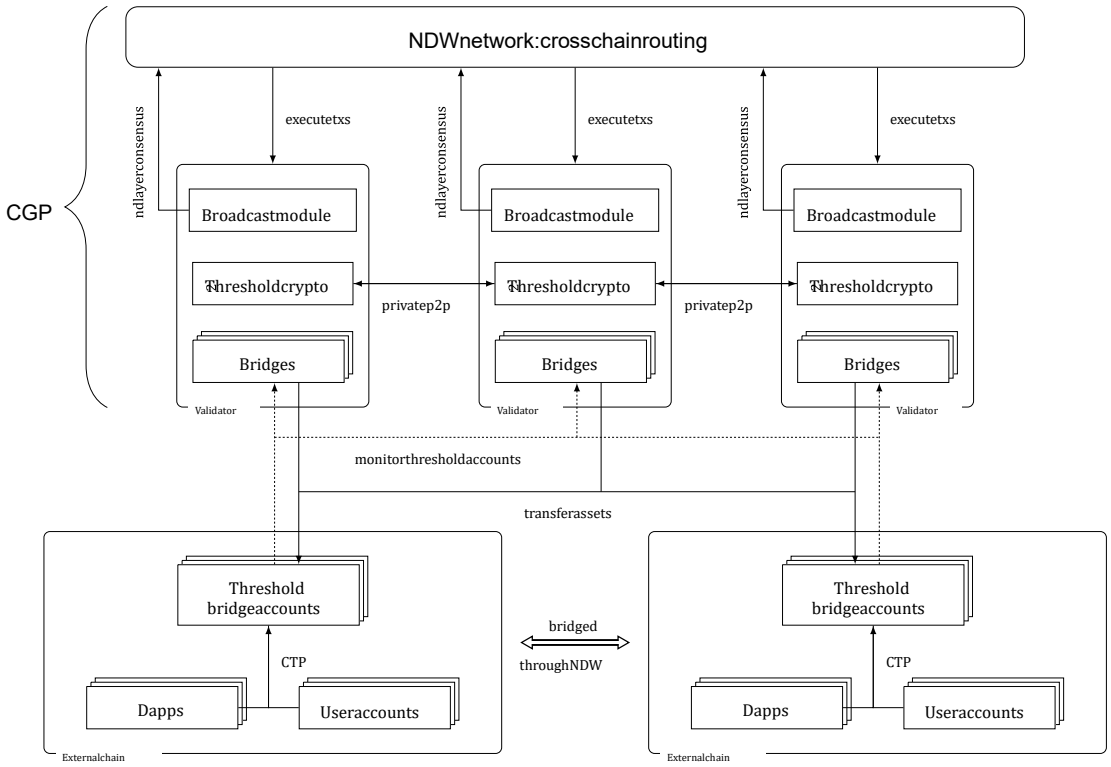
A sufficient supply of pegged-S tokens is printed and controlled by D_{NDW} upon its initialization. Suppose a user demands to exchange x amount of tokens on source chain S for x amount of pegged-S tokens on destination chain D, to be deposited at a D-address w_D of the user's choice. We present the fully general workflow, which supports arbitrary source chains S—even chains such as Bitcoin that do not support smart contracts:

1. The user (or an application acting on the user's behalf) posts a transfer request (x, w_D) to the threshold bridge account which is subsequently routed to the NDW network.
2. NDW validators use threshold cryptography to collectively create a fresh deposit address d_S for S. They post d_S to the NDW blockchain.
3. The user (or an application acting on the user's behalf) learns d_S by monitoring the NDW blockchain. The user sends x amount of S-tokens to address d_S via an ordinary S-transaction tx_S using her favourite software for chain S.

   (Due to the threshold property of d_S, tokens cannot be spent from d_S unless a threshold number of the validators coordinate to do so.)
4. tx_S is posted on NDW. The validators query the API of their chain S node software for existence of tx_S and, if the response is "true", report the answer to the NDW chain.
5. Once > F weighted validators report "true" for tx_S at round R, NDW asks validators to sign a transaction a_D that sends x amount of pegged-S tokens from D_{NDW} to w_D.
6. Using threshold cryptography the validators sign a_D. The signature is included in block R + 11.
7. Anyone can take the signed value a_D from block R + 11 and post it to D.
8. The request has been serviced, once a_D is posted on D the transfer is processed.

Now suppose a user demands to redeem x^0 amount of wrapped-S tokens from chain D back to chain S, to be deposited at a S-address w_S of the user's choice. The workflow is as follows:

1. The user initiates a transfer request (x^0, w_S) by depositing x^0 amount of wrapped-S tokens into c_D via an ordinary D-transaction using her favourite software for chain D.
2. (x^0, w_S) is posted on NDW. The validators query the API of their chain D node software for existence of (x^0, w_S) and, if the response is "true", report the answer to the NDW chain.


NDWnetwork:crosschainrouting
CGP
ndlayerconsensus
executetxs
Broadcastmodule
Thresholdcrypto
privatep2p
Bridges
Validator
monitorthresholdaccounts
transferassets
Threshold
bridgeaccounts
CTP
Dapps
Useraccounts
Externalchain
bridged
throughNDW

3. Once > F weighted validators report ”true” for (x^0,w_S) at round R, NDW asks validators to sign a transaction a_S that sends x^0 amount of S tokens from S_{NDW} to w_S.
4. Using threshold cryptography the validators sign a_S. The signature is included in block R + 11.
5. Anyone can take the signed value a_S from block R + 11 and post it to S.
6. The request has been serviced, once a_S is posted on S the transfer is processed.

Additional requests supported by the CGP routing layer include locking, unlocking or transferring assets across chains.

Achieving Atomic Cross-Chain Transaction Flow. Depending on the cross-chain request type, NDW tries to ensure that the corresponding transactions are executed on multiple chains or none. Towards this, every request can be in one of the following states in NDW blockchain: *(initialized, pending, completed, timed out)*. If a *timeout* at the pending stage is triggered, the request returns an error code. Some timeout events also begin a *refund* event: for instance, if an asset from one chain needs to be transferred into an asset on another chain, if the receiving chain did not process the transaction, the asset is refunded back to the original user.

Cross-Chain Transfer Protocol (CTP)

CTP is an application-level protocol that makes it easy for applications to leverage cross-chain features. We explain the integration by focusing on asset transfer features (e.g., used in DeFi). These applications typically consist of three main components: front-end GUI, smart contracts on one chain, and an intermediary node that posts transactions between the front-end and the smart contracts. The front-ends interact with the user's wallets to accept deposits, process withdrawals, etc. Applications can leverage cross-chain features by calling CTP queries analogous to HTTP/HTTPS GET/POST methods. These queries are subsequently picked up by CGP layer for execution and results are returned back to the users.

- *CTP Queries*. Application developers can host their applications on any chain and integrate their smart contracts with threshold bridge accounts to execute CTP queries.
- *Threshold bridge accounts*. Suppose an application developer builds their contracts on chain A. Then, they would reference threshold bridge contracts to obtain cross-chain support. This contract allows applications to:
  - Register a blockchain it would like to communicate with.
  - Register assets on that blockchain that it would like to leverage.
  - Perform operations over the assets such as accept deposits, process withdrawals, and other functions (similar to, say, ERC-20 contract calls).

Suppose a prominent DeFi application, ⓐDeFi, that natively resides on chain A registers with a threshold bridge account. The NDW validators collectively manage the contract itself on the corresponding chain. Suppose a user wants to submit a deposit into a trading pair between assets X and Y that reside across the two chains, respectively. Then, when a user submits such a request, it is routed via the threshold bridge account to the NDW network for processing. Form there, the following steps are performed:

1. NDW network understands that this application registered for the cross-chain support across the assets. It generates the deposits key leveraging threshold cryptography and consensus for the user on the corresponding chains A and B.
2. The associated public keys are returned to the application and displayed to the user who can use their favorite wallets to submit deposits. The corresponding secret key is shared across all NDW validators.
3. When the deposits are confirmed, NDW updates its cross-chain directory to record that the user on the corresponding chains has deposited these assets.
4. The NDW validators execute multi-party protocols to generate a threshold signature that allows updating the threshold bridge account on chain A where the application resides.
5. The CTP query is then returned to the DeFi application smart contracts, which can update its state, update its yield formulas, exchange rates, or execute other application state-related conditions.

Throughout this process, the NDW network, on a high-level, acts as a decentralized cross-chain read/write oracle, CGP is the routing layer in between chains, and CTP is the application protocol.

Additional Cross-Chain Requests. CTP supports more general cross-chain between applications across blockchains such as:

- Perform Public Key Name Services (PKNS). This is a universal directory for mapping public keys to phone numbers/twitter handles (a few projects, such as Celo, provide these features within their platforms).
- Cross-chain application triggers. An application on chain A can update its state if some another application on chain B satisfies a search criteria (interest rate < X).
- Smart contract composability. Smart contract on chain A can update its state based on state of contracts on chain B, or trigger an action to update a smart contract on chain B.

On a high-level, these requests can be processed since collectively, the protocols CTP, CGP, and NDW network can pass and write arbitrary verifiable state information across blockchains

Summary

Over the next years, significant applications and assets will be built on top of multiple blockchain ecosystems. NDW network can be used to plug-in these blockchains into a uniform cross-chain communication layer. This layer provides routing and application-level protocols that meet both platform builders and application developers' demands. Application developers can build on the best platforms for their needs and leverage a simple protocol and API to access global cross-chain liquidity, users, and communicate with other chains.