Z-DAG: An interactive DAG protocol for real-time crypto payments with Nakamoto consensus security parameters

Jagdeep Sidhu, Msc.*, Eliot Scott*, Alexander Gabriel†
*Syscoin Core Developers, Blockchain Foundry Inc.
Email: jsidhu@blockchainfoundry.co, escott@blockchainfoundry.co
†Lincoln Centre for Autonomous Systems (L-CAS), University of Lincoln, United Kingdom
Email: agabriel@lincoln.ac.uk

***Abstract*—Z-DAG stands for Zero Confirmation Directed Acyclical Graph. A technology that allow for point-of-sale with high degree of statistical probability of settlement in real-time within an interactive protocol between a merchant and consumer application.**

1. Introduction

Z-DAG (Zero-Confirmation Directed Acyclic Graph) is an instant settlement protocol that is used as a complementary system to proof-of-work (PoW) in the confirmation of Syscoin [Sid] service transactions. In essence, a Z-DAG is simply a directed acyclic graph (DAG) where validating nodes verify the sequential ordering of transactions that are received in their memory pools. Z-DAG is used by the validating nodes across the network to ensure that there is absolute consensus on the ordering of transactions and no balances are overflowed (no double-spends).

1.0.1. Z-DAG. Z-DAG (Zero-Confirmation Directed Acyclic Graph) is an instant settlement protocol functioning across Syscoin Assets. An Asset ownership is proven through a private key that matches each unique address associated with the Asset. Z-DAG manages balances and state of transactions in a deterministic fashion. This helps protect against double spends where an Asset is transferred falsely by creating multiple transactions through multiple nodes within a short time.

1.0.2. DAG Settlement. What permits the instant settlement of Z-DAG transactions is confidence that there is a high probability of micro-transactions settling within a block than there was with the real-time transaction being received by each node by enforcing a interactive protocol whereby a minimum latency is respected amongst sender and receiver which permits high probability of miners seeing order of events as they happened in real-time. Participants in a transaction are able to agree upon settlements without block confirmation because the network is able to anticipate what transactions will be in the next block and how they will be ordered with confidence. During the creation of a block, the miner is tasked with constructing an ordered list of transactions out of their memory pool which they sort by time. Note that miners traditionally select the transactions with highest paid fees out of the memory pool and this rule does not change here, the additional ordering happens after the transaction set is selected based on highest fees. The time based sort has negligible processing requirements and thus is no detriment to a miner and is done automatically upon creation of a block. If a miner fails to do this they risk their block not being validated by peers because the balances may overflow if processing out of order. A strict validation happens by every peer to ensure that no cases are allowed where negative balances are accepted when processing a block just like how standard Bitcoin balance checks are done processing UTXOs. Successful miners write these lists into new blocks, which then go on to be either accepted or rejected by validating nodes across the network. The validating nodes use the transaction lists contained within these blocks to validate that no balances are overflowed, which they can cross-check to determine whether the network is in a deterministic state or not. If the network is not in a deterministic state, meaning that verifying nodes and miners have conflicting results, then the conflicting transactions and block will be rejected. If the validating nodes conclude that they are in absolute consensus, the transactions within the block are confirmed and added to state databases as normal.

At the beginning of a transaction a non-enforced 10 second delay is applied to subsequent transfers made by the same asset holder. This delay is intended to create a minimum latency, which eases the ordering of transactions by time. If a user sends two transactions, with the second being made 10 seconds after the first, then enough nodes across the network should be able to determine which transfer is the original. If transfers are made within this delay, a conflict state is set, and the receiving node will reject the transactions that do not adhere to the interactive protocol; see figure 2. Any discrepancies between the real-time state and PoW block will be resolved upon the confirmation of a block, as PoW always rolls back to the previous state and replays the correct order of events as seen by the miner. By using this probabilistic approach Z-DAG transactions are able to be settled in real-time with confidence and negligible risk of double-spending over the minimum latency time. The memory pool validation and network relay of transaction messages is highly optimized through parallel execution and

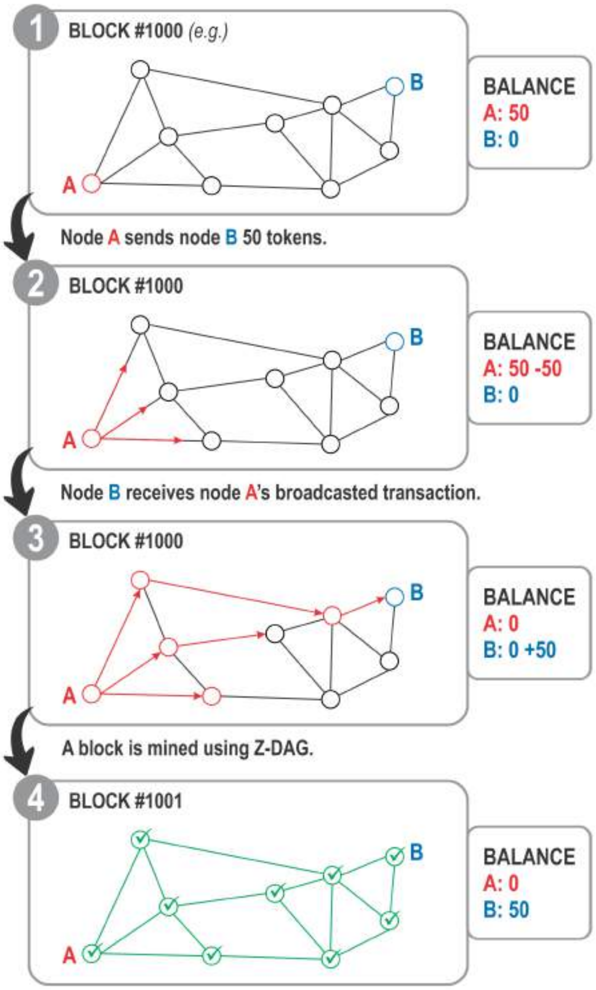

Figure 1: This represents real-time ordering of transactions as a linear ordering o = (v1,v2,. . .) of vertices in a DAG such that if u and v are connected by an edge e, u comes before v in o and e = (u → v). The balances are persisted in real-time and managed to detect double-spends. At Block 1001 the order is preserved by the miner and settled by PoW.

will allows for a probabilistic scenario whereby the chance of the miner not seeing transactions in the same order as the majority of the network after the minimum latency period is negligible.

1.0.3. PoW partition tolerance + DAG instant settlements. In addition to facilitating fast transactions, Z-DAGs also permit transactions to be settled upon a block without sacrificing any of the security benefits gained by using PoW as Syscoins primary method of consensus. The PoW is a fallback and partition tolerance of the DAG operation as every block a new DAG is constructed. If a receiver wishes to (if its not a micro-transaction, but for a larger amount) they may wish to simply wait for a minute/block for a block confirmation or 6 minutes/blocks for a more absolute confirmation. Typically, for token-based decentralized networks to improve transaction speeds some type of a comprise of functionality must be made. Systems that take this approach often choose to sacrifice decentralization and/or reward systems; examples can be made of the consensus protocols used by Ripple (XRP) [Dav] and Nano (NANO) [LeM]. Syscoin's solution retains decentralization, security, and rewards miners who sustain the network.

1.1. Technical outline

Syscoin's approach to settling transactions instantaneously shares similarities with an idea originally put forward by the originator of Bitcoin, [Naka] Satoshi Nakamoto. Satoshi argued in his/her response to the [Nakb] ”Bitcoin Snack Machine” problem that the threat of a double-spend is a race between a valid and malicious transaction to propagate the most nodes across a network first. By creating a delay between transactions, one transfer has an exponential advantage over another in spreading across the network first. Because of this, transactions are significantly easier to be ordered in sequence which enables Z-DAG to be constructed deterministically; this is an approach that other systems are structurally incapable of replicating unless they are decentralized systems with built-in rules for deterministic ordering coded into their consensus models. This is the essence of Z-DAG. See ”Related Works” section for further details on other works and how they fall short of solving the problem without centralization.

1.1.1. High level overview. The process of settling a Z-DAG transaction can be broken down as follows. The initiating node begins the process by broadcasting that it has sent some amount of a Syscoin Asset to another address. In Bitcoin and in crypto-token related projects in general all nodes who receive broadcasted transactions are tasked with generally verifying before relaying. However Syscoin optimized the verification procedure with a recourse model that allows for relaying before verifying; See Algorithm 1 in Appendix A. Syscoin uses an optimized multi-threaded parallel computing setup in the verification of signatures, see figure 3 and the ”Multithreading and signature verification” section. Once the transaction has reached the intended recipient, the sale is completed with confidence as the minimum latency period will have expired before the transaction was received. The process is which the receiver verifies the transaction is outlined in Algorithm 2 in Appendix A. The sender allocation is checked against double-spends or balance overflows by replaying the transaction through all in-transit allocation transactions in the memory pool. Miners order their queue of unconfirmed transactions by time and include these into a new block which they compete to mine onto the network. Once a miner succeeds in creating a block they broadcast it to the network. The networks validating nodes use the newly created blocks properties to determine the validity of the transactions it contains; see Algorithm 3 in Appendix A about the consensus code verification.

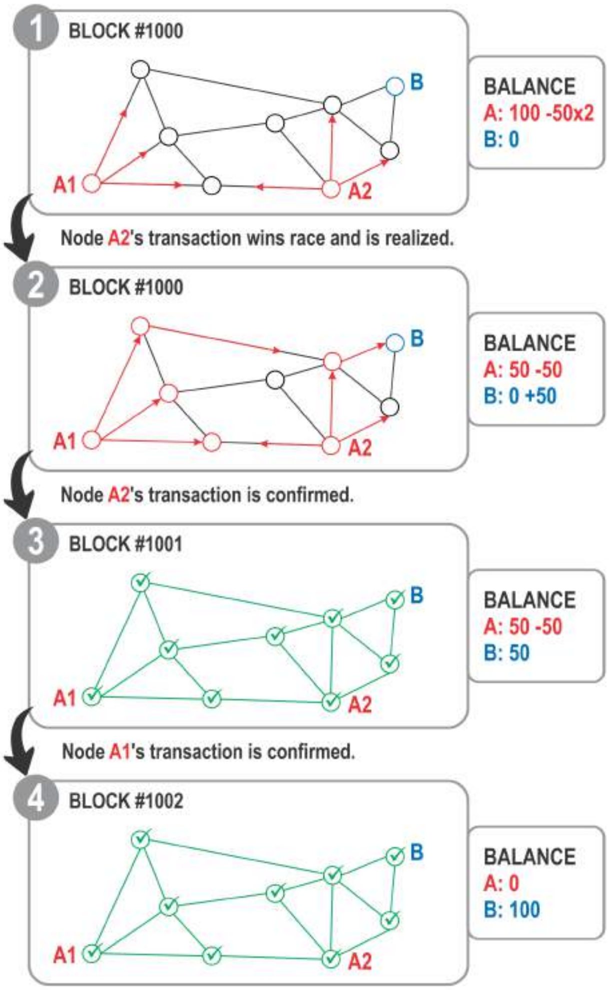

Figure 3: Node A broadcasts to the network that it is sending an amount an asset to node B. Once node B receives notice of the transaction, the balance is immediately realized in their wallet. Upon the mining of a block, the transaction is confirmed across the network. Upon block confirmation the state of the asset is reverted to that of the previous block and the ordered block is processed to arrive to a new balance. Notice that the balance remains the same as the process of verifying a block comes to the same conclusion to that of the real-time balance.

1.2. Interactive Game

The protocol consensus is enforced by the receiver of an asset transfer by calling the ”assetallocationsenderstatus” function which ensures that the minimum latency was respected for any transactions the sender made with assets, it also flags a major error state in case of a detected double-spend. If the sender was not following the protocol rules the receiver may decide to not honor a point-of-sale and either wait or reject their end of the deal. These rules are not enforced by network consensus and thus may be optimized later as network bandwidth and computing resources become cheaper and more efficient.

When a valid transfer is made, the miners who receive the broadcasted transaction include it in their queue of unconfirmed transactions to be mined into a block. Although the receiver of a transaction is able to realize their funds immediately upon receiving notice of the transaction, the transaction is not truly completed until a block has been mined; prior to the mining of a block, the transaction is simply settled with confidence that it will persist. Once a block has been mined, the transaction is confirmed, and the funds are finalized with confidence of PoW; see figure 4. In fact it is just the tale of two confidence models, the Z-DAG confidence model lies within the statistical probability that the PoW ordering will match with some certainty and the PoW confirmation allows for certainty that once confirmed over X blocks it is statistically unlikely that they will be rolled back. As a side note because of the confidence model that doesn’t persist over blocks it is over a time domain the fee market of such a model is also optimized to avoid cases where exponential rises in fees happen because blocks are full. Because in the Z-DAG confidence model you really only care that a double-spend has not been detected and that your transaction will eventually settle the fee one pays for a Z-DAG transaction may be a lot less than a standard one since one is not competing to get settled in the next block. It is also important to note that the main use-case for Z-DAG is for point-of-sale where a probabilistic settlement occurs and buyers are allowed to walk away from a sale with some type of fulfilled service without waiting for a block settlement with both parties confident that the transaction is accurate and complete.

1.3. Multithreading and parallel signature verification

To improve the speed of newly broadcasted transactions crossing the network, Syscoin uses a unique approach to the verification of signatures and relaying of transactions. Nodes are tasked with first relaying any incoming transactions before verifying their signatures; see figure 5. Once a node has relayed a new transaction, it then adds that transaction to a thread-pool queue containing any other unchecked transactions. The thread-pool is based off of a Bounded MPMC Lock-free (no mutexes atomic RMW operations) queue with 1 CAS per en-queue/de-queue operation based off of the work by Dmitry Vyukov [Vyu]. By prioritizing the relay of transactions over signature verification, the intended recipient will receive notice of incoming funds exponentially faster, enabling their new balance to be realized sooner. Multithreading also greatly reduces the time that it takes for signatures to be verified; see Algorithm 1 of Appendix A on the mempool concurrent verification process.

1.3.1. Concurrent verification recourse policy.

As a preventive measure against bad actors spreading invalid transactions, Syscoin nodes are instructed to implement a simple

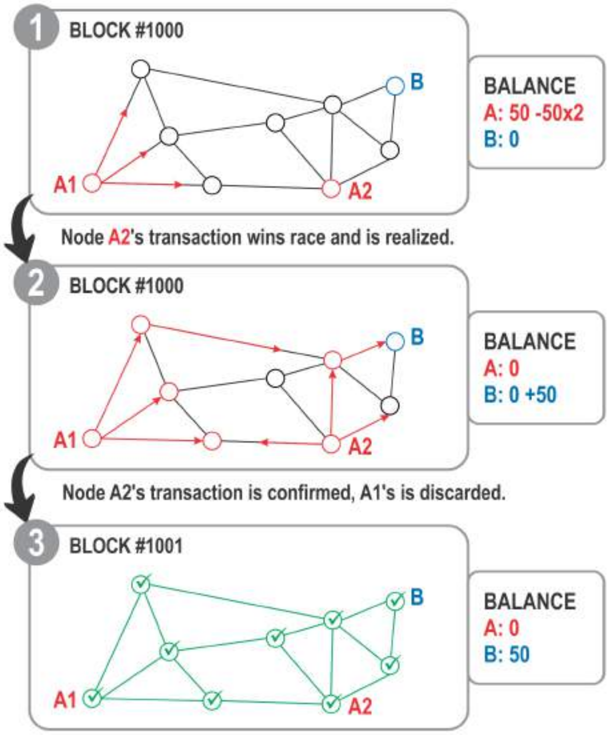

Figure 4: In the sequence diagram on the left, two nodes attempt to transact funds from the same address. Because the value of both transactions summed is within the sending addresses balance, the transactions are permitted to be confirmed over two blocks. In the sequence diagram on the right, two nodes attempt to transact funds from the same address, but one is rejected. After the first transaction is confirmed and the balance belonging to the sending address is reduced, the second transaction becomes illegal and is labelled as a double-spend and subsequently relayed to the rest of the network.

protocol upon encountering false signatures. When a node detects that a transaction has been wrongly signed, it sets a flag to return to single threaded mode for subsequent transactions requesting to enter the memory pool. The node then continues to parse through its remaining queue of unchecked transactions in single threaded mode for 60 seconds and discards any wrongly signed transactions it encounters in that time. Additionally, when a node is in single threaded mode it reverses its usual order of operations and parses through its queue of unchecked transactions before relaying them. By using this simple protocol valid transactions are able to propagate faster, while invalid transactions will be still caught and discarded quickly; see Algorithm 1 of Appendix A on the recourse policy of 60 seconds in context of the rest of the memory pool verification.

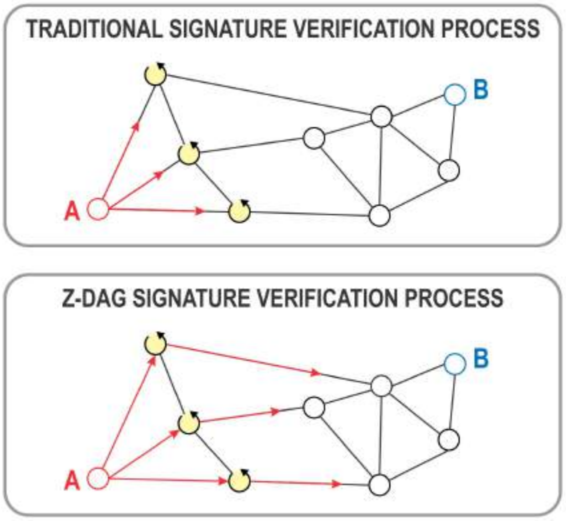
TRADITIONAL SIGNATURE VERIFICATION PROCESS
A
B
Z-DAG SIGNATURE VERIFICATION PROCESS
A
B

Figure 5: Traditional blockchain networks require each node to first check the signatures of incoming transactions before relaying them; this blocking technique bottlenecks broadcasting speed. The Z-DAG process as displayed above immediately relays transactions before checking signatures, resulting in significantly faster movement across the network

2. Network transmission and processing

Now that we have described how we solved the problem we can show the probabilities of detecting and processing double-spends in the context of Syscoin's masternode network. The intuition is that since we have decoupled verification from relaying we apply a statistical model to arrive to a conclusion that a 10-second verification window is sufficient for traditional commodity hardware to support speeds commensurate of VISA/Masternode networks operating at peak throughput.

2.1. Transmission Path

The expected number of common peers between k masternodes depends on the total number of Masternodes m in the network and the number of peers p each masternode has. The expected number of common peers is then:

$$\mathbb{E}|cp_k| = pk - \sum_{m} \mathbb{E}[X_i] = pk - m\mathbb{E}[X_i] \tag{1}$$

where X_i is 1 when Masternode i is a common peer and 0 otherwise. Since Masternodes pick their peers randomly from the set of masternodes, this boils down to

$$\mathbb{E}|cp_k| = pk - m\left(1 - \left(1 - \frac{p}{m}\right)^k\right) \tag{2}$$

If we assume the total number of masternodes to be 1000, the expected number of common peers between two masternodes is 0.625. What is the probability of that happening?

For a pair of masternodes, the probability of having exactly n common peers is given by

$$p(|cp| = n) = \frac{\binom{M}{c}\binom{M-c}{p-c}\binom{M-p}{p-c}}{\binom{M}{p}^2}. \qquad (3)$$

For 1000 masternodes with 25 peers each, this gives us following probabilities:

$$\begin{aligned} p(|cp| = 1) &= 0.3462 & \quad p(|cp| > 0) &= 0.4731 \\ p(|cp| = 2) &= 0.1047 & \quad p(|cp| > 1) &= 0.0220 \\ p(|cp| = 3) &= 0.0193 & \quad p(|cp| > 2) &= 0.0027 \\ p(|cp| = 4) &= 0.0024 & \quad p(|cp| > 3) &= 0.0002 \end{aligned}$$

Let's play it safe and assume only 22 out of our 25 peers are unique. Then after two hops we can reach 484 masternodes and after 3 hops we can reach them all. For a higher total count of masternodes, these odds improve.

A usual path through the network then has 5 steps, 3 within the mesh of masternodes, and two at its borders.

2.2. Transmission Delay

We collected a data set of masternode to masternode and client to masternode 300 byte ICMP transmissions. This data set contains 2086 samples, each of which is the average transmission time of ten transmissions between two hosts. To model the transmission delay of a single hop, we fit a gamma distribution to this sample set.

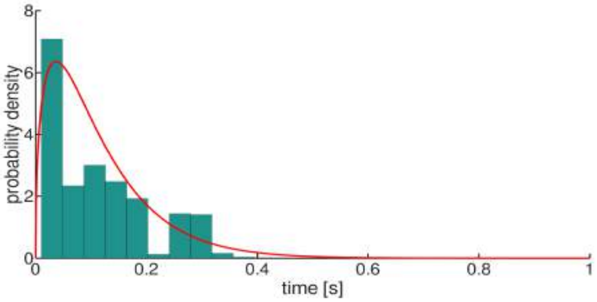

Figure 1. Matching the gamma distribution to our sample set

This gives us with the two defining parameters of a gamma distribution k and θ

$$k = 1.471680 \qquad \theta = 0.077714.$$

To get an estimation of the transmission time from end to end, we have to sum over 5 of our gamma distributions. This leaves us with a new set of defining parameters

$$k = 7.358398 \qquad \theta = 0.077714.$$

Now we can calculate the probability of the transmission arriving in a given time, and vice-versa how much time we have to wait to get the transmission with a desired probability.

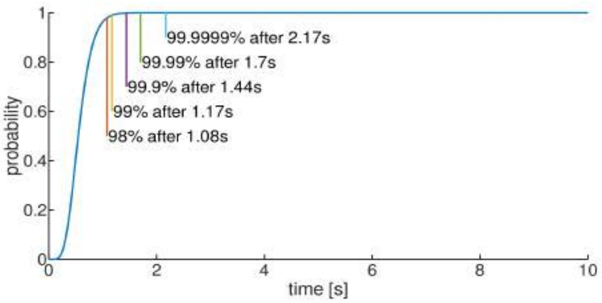

Figure 2. Probability of tx arrival after 5 hops over time

2.3. Network processing capacity

Since the receiver of a transaction is waiting 10 seconds before he accepts a point-of-sale transaction and up to 2.3 seconds of these are taken up by a possible double-spend transaction crossing the network, she has 7.7 seconds to spare. This time can be used for the verification of other signatures. Put another way, if a merchant has a backlog of 7.7s worth of signatures to verify, she can still detect the double-spend transaction in time.

We ran tests of our new parallel signature verification mechanism to create a sample set which we can use to model the signature verification time. We use libsecp256k1 for ECDSA signature validation of transaction inputs entering the mempool [Wui]. libsecp256k1 uses efficiently-computable endomorphism to split the P multiplicand into 2 half-sized ones to give us a 30 percent speed boost. Bitcoin has this disabled but is due to enable it in a future release. We have also enabled all hardware optimization's. We stuck with conventional hardware for typical throughput's of average off-the-shelf computing hardware found on cloud providers which Masternodes are running on today (quad-core CPUs with hyperthreading for 7 concurrent threads, 1 reserved for the executing thread). The numbers are representative of what should be possible today with low-cost hardware and not forward-looking which will be much more favourable as the speeds and infrastructure surrounding the networks grow organically.

	1 core	4 cores, hyperthreading
μ	55.170	13.030
σ^2	4.0890	2.2690

Using this model, we can find out how many signatures the merchant can verify in the 7.7s that are left of the 10 second wait time.

293.3 thousand signatures verified in the ten second window means the network has a maximum theoretical throughput of 29.33 thousand transactions per second with a 4-core receiver node. Each asset transaction can hold up

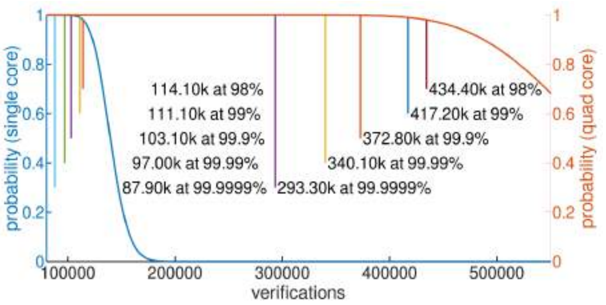

Figure 3. Probability of verifying signatures in 7.7s

to 250 receivers meaning effectively a maximum throughput of roughly 733 thousand transactions per second assuming a virtual transaction between a sender and receiver tuple. If the merchant runs a faster machine, these numbers increase significantly.

3. Related works

Syscoin's method of consensus is fundamentally centered around Satoshi's proof-of-work algorithm. The Z-DAG protocol is an integral secondary layer of consensus that serves the purpose of permitting transactions to be settled before they are confirmed; Z-DAG cannot exclusively ensure consensus alone. By simultaneously using a DAG and PoW, Syscoins approach to solving the fast-transaction problem is both efficient and unique. Many other decentralized services make use of DAG based network structures entirely, and even more continue to use PoW or PoS. Projects such as IOTA [Pop] and NANO have created their own DAG based protocols which largely forego the proven security of PoW but in return gain the ability to process transactions in rapid speeds. Many older services, namely Bitcoin and similar projects forked from it, continue to use PoW for the sake of security, but their users are frequently subjected to slow and restrictive transaction times.

3.1. Related works comparison

As previously mentioned, IOTA and NANO are two exemplary DAG based services that have created their own unique methods of ensuring a decentralized consensus amongst their networks node. It can be argued that NANO's approach was inspired by Satoshi's proof-of-work as it is used in ensuring the networks consensus. However, NANO's consensus is fundamentally attained through the use of their DAG structure which they refer to as the "block-lattice". Unlike Syscoin, nodes on the Nano network maintain their own individual blockchains. Transactions are ordered by their timestamps and are used to form directed acyclic graphs. During a transaction, both the sender and receiver produce their own blocks which reflect their respective participation in a transaction, which they then broadcast to the network. Both the sender and receiver also verify their own transactions using PoW which eliminates the need for designated miners on the network. To prevent foul play, NANO employs PoS and a voting system to be used if any transactions do not meet the networks required criteria.

IOTA's approach to ensuring their networks consensus does-away with traditional models almost entirely. IOTA depends on a network shared main-DAG which they refer to as the "Tangle". Within the tangle, transactions exist as vertices and form long chains. When a transaction attempts to join the tangle to be confirmed across the network it must approve two others prior to it. Thus, by participating in an IOTA transaction, the user is helping to speed up the network. Consequently, and similarly to NANO, each node can be viewed as their own miner. Harsh criticisms have been made against IOTAs system, however, as it is not entirely decentralized at present time. Right now, IOTA is structurally vulnerable in its current maturity to what is called a "34 percent attack". A "34 percent attack" is much like the "51 percent attack" that threatens PoW systems, but with a significantly greater vulnerability risk. While the tangle grows overtime as transactions join the DAG, it's resilience against such an attack increase. As a temporary remedy until the tangle reaches a certain maturity point, IOTA has instated a centralized coordinator node (COO) to prevent 34 percent attacks specifically.

Decentralized services that rely purely on PoW consensus reap the benefit of its rigorously tested and proven security but are bottle-necked by slow transaction times. For transactions to be confirmed they must be included in a block which are often mined far apart; in Bitcoin's case, once every 10 minutes. This mandatory time constraint makes exchanges unfeasible for many real-world applications of cryptocurrency. Typically, the ordering of transactions within a block is arbitrarily decided by the miner. The only exception to this is that transactions have to appear after any transactions upon which they depend. This randomness makes it impossible for a block to be confidently predicted; hence why Syscoin orders transactions by their topological ordering before validating blocks. This allows the real-time view of the transactions to mirror the ordering within the block and this requires protocol support which is lacking from Bitcoin and other PoW implementations. Thus technologies relying on such PoW chains to do zero-confirmation settlements cannot fundamentally have the same security properties than those that have protocol support at the core layer.

Bitcoin-XT also implements a zero-confirmation settlement scheme by relaying double-spending transactions and having merchants detect the double-spends in real-time, however it is susceptible to a delay attack as described by [Ger] which can be effectively averted in Syscoin by having merchant's query randomly selected masternodes for their responses to assetallocationsenderstatus.

Implementations that are not backed by PoW suffer from partition tolerance issues. Inherently the trade off has been made by DAG implementations based on performance over consistency. A PoW chain will likely have greater consistency over time and not end up having the issue of coordinating DAG structures such as IOTA has. Implementing a PoW + DAG in Syscoin meant using the PoW as a reference for each DAG and thus no coordinate action is required to maintain consistency in how the DAG is formed or validated.

Syscoin's protocol for Z-DAG poses no restrictions on how receivers validate transactions, only that assteallocationsenderstatus is a reference implementation ensuring minimum latency between adjacent sender transactions is met as well as ensuring that a double-spend has not been actively detected for that sender. Users are free to implement their own or even adjust the latency between acceptable real-time transactions from 10 seconds to something less depending on how network bandwidths increase to allow for faster and faster propagation of transactions. In fact perhaps an optimization can be done to dynamically adjust the latency to optimally follow the rate at which the propagation is happening on average. For example when the mempool is relatively empty it would be relatively quick to propagate a transaction across the network compared to if the mempool was filling up at a rate where the verification mechanism was the bottleneck. We leave this as an exercise for future work.

3.2. Centralized services using Bitcoin

Another approach to solving what has been previously referred to as the 'fast-transaction' problem are application level wrap-around solutions. Entities such as BitPay offer blockchain billing services for businesses that are built on top of cryptocurrencies; e.g. BitPay is built on top of Bitcoin. BitPay's model uses an invoice and crypto-to-fiat conversion technique. BitPay invoices clients for a Bitcoin payment which it then receives and converts back into fiat to return to the business. Additionally, BitPay offers business and clients the ability to choose a level of risk tolerance that theyre willing to assume with their transactions. This risk-mitigation system scales its levels of risk in accordance to how many blocks have been mined on the Bitcoin network since the transactions initiation. If a transaction requires a minimal number of blocks to be mined before a payment is confirmed it is at a greater-risk of fraud, and vice-versa if a larger number of blocks are required.

4. Future Work

The work here can be improved to increase the statistical probability of validation in the minimum latency period through a variety of techniques. One method is to check a random quorum of bonded validators for verification.

4.1. SPV Z-DAG

A simple payment verification method for Z-DAG is possible by asking a random quorum at the receivers discretion for details regarding Z-DAG transactions in question. By relying on a random quorum you can offload the probability of invalid detection of double-spends from the host node on a receiver to a set of nodes and likeliness of gamesmanship of the random quorum selected. As can be seen in the random quorum selection for the Instant Send protocol [Duf], it is safe without a reasonable doubt to use such a verification mechanism for point-of-sale applications especially if the randomness is seeded by the receiver and not by a deterministic seed which may be easier to take advantage of.

This type of verification may be especially useful for Point-Of-Sale terminals that are not running full nodes but may reference end-points to a randomized quorum of bonded validator full nodes to do the verification for them for each transaction.

5. Conclusion

As can be seen by comparing Syscoin to related works, the Z-DAG protocol is an elegant and efficient solution to solving the aforementioned 'fast-transaction' problem. It offers users the ability to instantly settle transfers of Syscoin services with negligible risk of fraud, and without the involvement of any third party. Z-DAG transactions do not require that merchants or customers take on any additional steps to make fast-transactions on the Syscoin network. By using Z-DAG, customers transactions can be settled quickly and with the absolute security proven by Bitcoin.

Acknowledgments

We would like to thank Satoshi Nakamoto and the Bitcoin Core developers for their continued excellence in software engineering, which has made it possible for others to develop innovative products on top of their accomplishments.

References

[Dav] Arthur Britto David Schwartz Noah Youngs. *The Ripple Protocol Consensus Algorithm*. URL: https://ripple.com/files/ripple_consensus_whitepaper.pdf.

[Duf] Evan Duffield. *Dashpay*. URL: https://github.com/dashpay/dash/wiki/Whitepaper.

[Ger] Arthur Gervais. *Tampering with the Delivery of Blocks and Transactions in Bitcoin*. URL: https://scalingbitcoin.org/papers/bitcoin-block-transaction-delivery.pdf.

[LeM] Colin LeMahieu. *Nano: A Feeless Distributed Cryptocurrency Network*. URL: https://nano.org/en/whitepaper.

[Naka] Satoshi Nakamoto. *Bitcoin: A peer-to-Peer Electronic Cash*. URL: https://bitcoin.org/bitcoin.pdf.

[Nakb] Satoshi Nakamoto. *Bitcoin snack machine (fast transaction problem)*. URL: https://bitcointalk.org/index.php?topic=423.20.

[Pop] Serguei Popov. *The Tangle*. URL: https://assets.ctfassets.net/r1dr6vzfxhev/2t4uxvsIqk0EUau6g2sw0g/45eae33637ca92f85dd9f4a3a218e1ec/iota1_4_3.pdf.

[Sid] Jagdeep Sidhu. *Syscoin 3.0: A Peer-to-Peer Electronic Cash System Built For Business Applications*. URL: https://syscoin.org/Syscoin_3.0_Whitepaper__Condensed.pdf.

[Vyu] Dmitry Vyukov. *Bounded MPMC queue*. URL: http://www.1024cores.net/home/lock-free-algorithms/queues/bounded-mpmc-queue.

[Wui] Pieter Wuille. *Optimized C library for EC operations on curve secp256k1*. URL: https://github.com/bitcoin-core/secp256k1.

6. Appendix A: Z-DAG protocol pseudocode

Algorithm 1: Mempool parallel verification and Z-DAG consensus in real-time

```
Result: Asset allocation balances will transfer in real-time as soon as the message is received across network
        participants
Call assetallocationsend RPC;
Sign and send TX to network;
for each transaction received from peer do
    Accept to memory pool;
    if $now() - nLastMultithreadMempoolFailure < 60$ seconds then
        set bMultiThreaded = false;
    else
    Preliminary input checks same as Bitcoin;
    if bMultiThreaded == false then
        for all inputs do
            Do signature verification;
            if If signature verification fails then
                return false;
            else
        Add transaction to memory pool;
        Relay transaction to network peers;
    else
        Add transaction to memory pool;
        Relay transaction to network peers;
        Add transaction to thread pool queue for signature verification;
        Thread pool concurrently checks for signature validity;
        if signature verification fails then
            set nLastMultithreadMempoolFailure = now();
            return;
        else
        Zero confirmation Syscoin consensus updates;
        if update fails then
            set nLastMultithreadMempoolFailure = now();
            return;
        else
```

Algorithm 2: Asset Allocation Sender Status RPC

```
Result: Check that a transaction received is indeed valid according to Z-DAG interactive protocol rules
set status = OK;
if sender was found in assetAllocationConflicts (double-spend detection buffer) then
    set status = MAJORCONFLICT;
else
    Order all in-transit asset allocations from sender in order by ascending time;
    set mapBalances[sender] = sender balance from last PoW block (last known good state);
    for all in-transit asset allocations do
        set txRef = in-transit asset allocation fetched from mempool;
        if txRef is invalid then
            continue;
        else
        if time received of 2 adjacent transactions $<=$ minimum latency (10 seconds) then
            set status = MINORCONFLICT;
        else
        for all allocations sent in this transaction do
            set senderBalance = senderBalance $-$ sending amount;
            set mapBalances[sender] = mapBalances[sender] $-$ sending amount;
            if senderBalance $<= 0$ then
                set status = MINORCONFLICT;
            else
return status;
```

Algorithm 3: Block Construction

```
Result: A Block is constructed out of transactions related to Syscoin and/or Syscoin assets
Craft block from transactions queued in the memory pool, ordered by highest fee first;
Order all in-transit asset allocations from sender in order by ascending time;
Test validity of Syscoin asset transactions in block;
if If block is invalid because transactions cause balance overflows then
    remove invalid transactions from block and call Block Construction again;
else
Test validity of standard Syscoin block transactions;
Solve block PoW and relay block to the network;
```