Project Boreas: Scaling Bitcoin through Off-Chain Verification and On-Chain Settlement

David Nettey

davidnettey@protonmail.com

Abstract

Bitcoin was created as a new type of currency that could improve on the short falls of fiat currencies. It is decentralized, permission-less, immune to hyperinflation, and open source. Despite all these advantages, it does have flaws and limitations. One of the most pressing limitations in Bitcoin is scaling. Bitcoin has a low transaction throughput, about 3 to 7 transactions per second [5]. When the network is congested, transaction fees can skyrocket. In order for Bitcoin to be used widely as a peer to peer cash system, it must solve its scaling issues.

To increase transaction speeds, we can make use of a second layer network. This network will act as a verification layer, allowing transactions to be viewed and spent much faster. The Avalanche consensus mechanism will be used to track and verify transactions off-chain. The Bitcoin network will act as the settlement layer. A user's wallet will periodically post a clearing transaction to the Bitcoin network containing verified but unsettled transactions. In order to save on fees, the clearing transaction will be comprised of multiple off-chain transactions. The second layer network essentially acts as a decentralized, trustless bank card issuer and payment processor for the Bitcoin network.

Background

Bitcoin

Satoshi Nakamoto's idea of a distributed ledger is the core of Bitcoin [3]. Each node on the bitcoin network has a copy of the ledger. This ledger is composed of blocks that are linked to one another, hence the name blockchain. Each block has

a set of transactions, a header, and a block hash. Only 1 MB of transactions can be added to a block. In the block's header's is metadata like the previous block's hash and time. The block's hash is the result of passing all the data in the block's header through a hash function called sha256.

Each person on the bitcoin network has a public and private key. The public key is used to "lock" bitcoin to an address. The private key is used to create signatures. These signatures are intrinsically linked to an address and transaction. A signature can be used to "unlock" the funds at that address. The signature also ensures that only the linked transaction can be performed.

Transactions are composed of inputs and outputs. Outputs are analogous to packages of bitcoin that belong to a someone. Each output is locked by the address using its owner(s)'s public key. Inputs point a set of existing, unspent outputs and uses them to create new outputs locked by new address. When a transaction is put on the blockchain, all outputs that the transaction's inputs point to are considered spent. The new outputs on that transactions are considered unspent. The sum of the spent outputs must be equal to the new, unspent outputs.

Blocks of transactions are confirmed by a mechanism called Proof-of-Work. Nodes called miners listen for new transactions and add them to a candidate block. When 1 MB of transactions have been added, the miners begin mining. There is a special field in the block's header that a miner can change called a nonce. When they change the nonce, the block's hash changes. Mining is a race between miners to change the nonce until the block hash meets certain conditions. The miner who meets the hash conditions first broadcasts the new block onto the network. Baked into the new block is the block reward, newly created bitcoin that goes to the miner. The miner also gets all fees in that block. If there are conflicting blocks, nodes create a fork in their copy of the blockchain. Each fork has one of the conflicting blocks. The nodes continue listening for new blocks. The longest fork is the one that will be trusted.

Proof-of-Work is a tried and true method of updating the blockchain. However, there are drawbacks. Mining uses large amounts of electricity. If this electricity is sourced from fossil fuels, it increases the amount of CO_2 in the atmosphere, contributing to climate change. Another drawback is the transaction throughput. The mining process was designed to take approximately 10 minutes. The difficulty

of mining automatically adjusts to ensure this timespan. This is already impractical for day to day transactions. If the network is congested, transactions can take far longer. This is due to the fee system. Transactions compete to be included into a candidate block. Transactions can offer high fees to the miner to be prioritized. The higher the fee, the quicker a transaction can be added to a block. This means transactions with low amounts must either offer disproportionately high fees or potentially wait for hours to be confirmed.

Avalanche

The Avalanche consensus mechanism is an alternative to Proof-of-Work and Proof-of-Stake created by a group called Team Rocket. Avalanche seeks to address the scaling issues in Proof-of-Work while achieving consensus in a leaderless fashion [5]. The fundamental building block of Avalanche is the Snowball algorithm [4].

Every node on the network chooses between a pair of binary choices. Lets call them A and B. A node has an initial preference for A. The node proceeds to query k random nodes on the network for their choice. If α nodes out of k have a preference for A, the node increments the number of consecutive successes. If B gets α out of k, then the node changes its preference to B and sets the consecutive successes to 1. If neither choice gets at least α votes, then the consecutive successes is set to 0. This repeats until the number of consecutive successes for a given preference reaches β; then the node finalizes its decision.

Avalanche uses a directed acyclic graph (DAG) in order to track transactions. A graph is a data structure that has packets of information, nodes, connected to each other through edges. If a graph is directed, that means an edge can only be traversed in one direction. Acyclic means that it is impossible to end where one started while traversing the graph. A graph node's ancestors are all the nodes that can be reached by following edges that lead away from the node. That graph node is a descendant of all those nodes along each path.

In Avalanche, each transaction in the graph is an output similar to those in Bitcoin. The consensus mechanism uses Snowball to ensure that conflicting transactions - such as a double spend - are not included in the DAG. Conflicting transactions belong to a conflict set. If there is only one transaction in the set, then it is likely to

be an honest transaction. When a node the network learns about a new transaction, it performs a modified version of the Snowball algorithm.

It queries k random nodes about whether they prefer the transaction. If α nodes or more prefer it, then the transaction gets a chit. A chit is a boolean stating whether the transaction got a quorum of votes or not. As the transaction gets descendants with chits, its confidence increases. Confidence is simply the sum of chits in all a transaction's descendants plus its own chit. Once a transaction's confidence reaches β, it is accepted. If the transaction belongs to a conflict set with two or more members, then the transaction that achieves β confidence is the only one that gets accepted.

Avalanche's byzantine fault tolerance increases as α increases. This means that the network can handle a higher proportion of nodes being malicious as α increases. Consensus is probabilistically guaranteed. There's an incredibly small chance that the algorithm would result in no consensus. Snowball is also very scalable. k and α can remain the same even as the number of nodes in the network increases. A network running Avalanche can theoretically process 3,400 transactions per second, with each transaction being verified in around 1.35 seconds [5].

Verification Layer

The verification layer is a network that allows transactions to be quickly verified and tracked. Nodes on the network have two jobs. The first job is using a modified Avalanche consensus mechanism to verify transactions. Nodes listen for new transactions from users' wallets. When a new transaction is added to the DAG ledger, Avalanche is used to quickly verify it.

The second job is securing funds that enter and leave the layer. When a user transfers funds from an external address, the funds are deposited into a multi-signature address. This address is generated by the user's wallet and the network working together. Two pairs of public-private key pairs are created. One is generated by the user's wallet. The other public-private key pair is generated and backed up by the network by leveraging Shamir's Secret Sharing algorithm. The wallet's public key and the network's public key are used to generate the multi-signature address.

Network Key-Pair Generation

First the user sends a request for a new address to a set of random nodes. Each node that gets the request forwards it to its peers. Then each node generates a random two dimensional, Cartesian coordinate and forwards that as well. Each node waits for T seconds. Assuming the network architecture approximates a tree, T is defined by this equation:

$$T = t * \log_p N$$

p is the number of peers per node. t is the average latency between nodes in seconds. N is the number of registered nodes on the network. After the wait, each node sorts the points it has received and its own point in ascending order and removes any duplicates. The top 2/3 of the list of points are interpolated into a polynomial. The result is checked through an instance of the Snowball algorithm. Once consensus is reached, the y-intercept hashed twice will be the network's new private key. The network will then generate a public key from the private key. Each node saves the public key and it's point if it was used during polynomial generation. If a node's point was not chosen to generate the polynomial, the node will select a random unselected point along the curve and save that.

The user's wallet will run its own instance of the Snowball algorithm to accept a public key from the network. The wallet will then create the address from its public key and network public key and propagate it through the network.

Signature Generation

When transferring funds to an external address or clearing verified transactions, both the network and user's wallet must generate the appropriate signatures for the transaction. This helps prevent the double spending of verified but unsettled funds.

To generate a signature for an address, the user's wallet creates a transaction for the bitcoin network, signs it, and sends it to the off-chain network. The nodes verify the proposed transaction through Snowball. Once the transaction is accepted, each node sends the secret share associated with the transaction's sending address to its peers. After sending its secret share, the node waits until it receives the requisite number of shares to reconstruct the private key. The node

interpolates the shares and generates the polynomial. The y-intercept of the polynomial hashed twice to produce the private key, then the node signs the transaction.

Transaction types

There are three types of transactions that can take place on the verification layer. The first type is simply the transfer of funds between addresses on the network. The funds are verified by the network then settled later on the Bitcoin network. These will be simply referred to as transfers.

The second type are deposits. Deposits happen when funds are sent from an external address address to an network address. The user first sends funds to the address. Upon detection of the new funds, the wallet sends a deposit transaction to the network. Nodes search the Bitcoin blockchain for the pertinent transaction to make their initial preference in the Snowball algorithm. The deposit is then verified normally through Avalanche.

External transfers are the last type. This is when funds from a network address are sent to an external address. Only funds that have been settled on the Bitcoin network can be used for external transfers.

Transaction Settlement

The off-chain network acts as a verification layer for Bitcoin transactions. Transactions can be verified in a timely fashion. After a certain number of transactions, unsettled transactions are sent to the Bitcoin network. When the transactions are included in a block, the transactions are considered settled and cleared.

In order to settle a group of transactions, first the user's wallet waits until a certain number of its transactions have been verified by the network. The wallet then sends a settlement request to the verification layer. The request contains the transactions to be cleared. The request is gossiped throughout the network.

Each node then finds the last settled ancestors of all the transactions that belong to the user. Then the nodes find the most recent descendants of each transaction in the request. A single transaction is constructed using the settled ancestors as inputs and the descendants or the transactions themselves as outputs. This

transaction is then verified through the Snowball algorithm. The nodes then reconstruct the appropriate private keys for the inputs and sign the transaction.

The user's wallet runs an instance of Snowball on random nodes on the network in order to get the agreed upon transactions and signatures. The wallet then signs the transaction and sends it to the Bitcoin network.

Each node on the network scans new blocks added to the Bitcoin blockchain for transactions associated with settlement requests. If one is found, the nodes mark each transaction in the paths from the inputs to the outputs as cleared.

Every transaction is associated with a fee. However the type of fee on the transaction differs depending on the source of the outputs. Fees on transactions that directly spend from settled outputs are calculated using segregated witness (segwit) fee rates for the Bitcoin network. The fee is equal to that single, 1440 block (~ 24 hour) confirmation time transaction with u multi-sig inputs, $n + 1$ outputs, and $2u$ signatures, divided by n. u is the number of settled outputs being spent from. n is the number of transaction verified per settlement transaction. These fees are reserved for the miners on the Bitcoin network when the settlement transaction is sent.

Fees on transactions that spend from verified, but unsettled outputs are calculated similarly. However, these fees go to the nodes on the verification layer. When a user's wallet creates a transfer using unsettled funds, the first node it selects to send the transaction to for verification gets the fee.

Fee Calculation

The size of a transaction is directly correlated to the size of the fee associated with the transaction. The transaction's size - or virtual size in the case of segwit - multiplied by the current rate for a given confirmation time yields the fee. A native transaction's fee can be expressed as:

$$f = rT_{size}$$

A segwit transaction's fee can be expressed as:

$$f_{segw} = r_v V_{size}$$

Where r is the native transaction fee rate for a given confirmation time in satoshis per byte; r_v is the native transaction fee rate for a given confirmation time in satoshis per virtual byte (vB); T_{size} is a transaction's size in bytes; V_{size} is a segwit transaction's virtual size.

Every transaction has overhead. The version field is 4 bytes. The input and output counts are both 1 byte as long as they are less than or equal to 252. The nLockTime field is also 4 bytes [7]. Each transaction also has inputs and outputs. Input and outputs can vary in size. A transaction can also have more than one or more inputs and outputs. The overall size of a native transaction can be expressed as:

$$T_{size} = n_{in}s_{in} + n_{out}s_{out} + 10$$

Where n_{in} is the number of inputs; n_{out} is the number of outputs; s_{in} is the size of the inputs in bytes; s_{out} is the size of the outputs in bytes.

Segregated witness was introduced to Bitcoin in Bitcoin Improvement Proposal (BIP) 141 [7]. Each field in a segwit transaction has a weight associated with it depending on whether it is witness data. Witness data is the overhead associated with segwit transactions: the segwit marker, segwit flag, and the witness. The non-witness data are the fields that segwit and native transactions have in common. A segwit transaction's weight can be expressed as:

$$W = 4T_{size} + n_{in}w + 2$$

Where w is the size of the witness for a given input in bytes. When calculating fees for a segwit transaction, the weight must be divided by 4 to get the virtual size.

$$V_{size} = \frac{W}{4} = T_{size} + \frac{n_{in}w + 2}{4}$$

Each input has a transaction id (32 bytes), output selection (4 bytes), a sequence (4 bytes) [7]. A transaction also has the script size and the unlocking script itself. The script size is 1 byte as long as its less than or equal to 252. s_{in} can be expressed as:

$$s_{in} = u + 41$$

Where u is the unlocking script's size.

Each output has a value (8 bytes), a script size (1 bytes as long as its less than or equal to 252), and the locking script itself [7]. s_{out} can be expressed as:

$$s_{out} = l + 9$$

Where l is the locking script's size.

Comparison of fees

We will now compare the costs of sending a single transaction on-chain and the cost of sending a single transaction off-chain using the verification layer network.

The most common transaction type on the Bitcoin network are Pay-To-Pubkey-Hash (P2PKH) transactions [2]. We will use this type as the typical on-chain transaction. We will assume one input to draw funds from and two outputs. One output locks a part of the funds from the input into the recipients address. We will assume the locking script of the recipient's address is also P2PKH. The other output locks the remaining funds from the input into the sender's address, acting as the sender's "change".

The average size of a P2PKH locking script is 25 B [1]. Therefore $s_{out} = 25 + 9 = 34$. The average size of the unlocking script is 106.5 [1]. Therefore, $s_{in} = 106.5 + 41 = 147.5 \approx 148$. Therefore $T_{size} \approx 148 + (2 * 34) + 10 \approx 226$.

The average transaction fee will be about $226r$ satoshis.

Applying segwit to a P2PKH transaction would result in a Pay-To-Witness-Pubkey-Hash (P2WPKH) transaction. The locking script for a P2WPKH transaction is 22 B on average. The unlocking script is moved to the witness, so it is 0 B. The witness' size on average is 107.5 B, 108 B rounded up [1].

Therefore

$$s_{out} = 22 + 9 = 31$$

$$s_{in} = 41$$

$$T_{size} = 41 + (2 * 31) + 10 = 83$$

$$V_{size} = 83 + \frac{108 + 2}{4} = 110.5 \approx 111$$

The average transaction fee for a P2WPKH transaction would be $111r_v$ satoshis. If $r = r_v$, then applying segregated witness to the transaction reduced fees by about 49%.

The fee of a transaction off-chain will be equal to the fee of the clearing transaction divided by n; where n is the number of off-chain transactions included in the clearing transaction. We propose using segregated witness for all transactions in order to reduce fees.

$$F = \frac{f_{clear}}{n} = \frac{V_{size} r_v}{n}$$

We must determine the size of the clearing transaction. We will use Pay-To-Witness-Script-Hash (P2WSH) 2-of-2 multisig as our locking script and witness. The locking script is on average 34 B. The unlocking script is moved to the witness, so it is 0 B. The witness is on average 219 B [1].

Therefore

$$s_{out} = 34 + 9 = 43$$

$$s_{in} = 41$$

$$T_{size} = 41n_{in} + 43(n + 1) + 10 = 41n_{in} + 43n + 53$$

$$V_{size} = 41n_{in} + 43n + 53 + \frac{219n_{in} + 2}{4} \approx 96n_{in} + 43n + 54$$

$$F \approx \frac{96n_{in} + 43n + 54}{n} r_v$$

The percent savings or percent premium in fees for off-chain transactions to on-chain transactions can be expressed as:

$$P = \frac{F}{f_v} \approx \frac{96n_{in} + 43n + 54}{111n}$$

The following figure shows values of P for a given number of inputs and transactions:

	Inputs									
Txs	1	2	3	4	5	6	7	8	9	10
1	1.738739	2.603604	3.468468	4.333333	5.198198	6.063063	6.927928	7.792793	8.657658	9.522523
2	1.063063	1.495495	1.927928	2.36036	2.792793	3.225225	3.657658	4.09009	4.522523	4.954955
3	0.837838	1.126126	1.414414	1.702703	1.990991	2.279279	2.567568	2.855856	3.144144	3.432432
4	0.725225	0.941441	1.157658	1.373874	1.59009	1.806306	2.022523	2.238739	2.454955	2.671171
5	0.657658	0.830631	1.003604	1.176577	1.34955	1.522523	1.695495	1.868468	2.041441	2.214414
6	0.612613	0.756757	0.900901	1.045045	1.189189	1.333333	1.477477	1.621622	1.765766	1.90991
7	0.580438	0.70399	0.827542	0.951094	1.074646	1.198198	1.32175	1.445302	1.568855	1.692407
8	0.556306	0.664414	0.772523	0.880631	0.988739	1.096847	1.204955	1.313063	1.421171	1.529279
9	0.537538	0.633634	0.72973	0.825826	0.921922	1.018018	1.114114	1.21021	1.306306	1.402402
10	0.522523	0.609009	0.695495	0.781982	0.868468	0.954955	1.041441	1.127928	1.214414	1.300901
11	0.510238	0.588862	0.667486	0.74611	0.824734	0.903358	0.981982	1.060606	1.13923	1.217854
12	0.5	0.572072	0.644144	0.716216	0.788288	0.86036	0.932432	1.004505	1.076577	1.148649
13	0.491337	0.557866	0.624394	0.690922	0.75745	0.823978	0.890506	0.957034	1.023562	1.09009
14	0.483912	0.545689	0.607465	0.669241	0.731017	0.792793	0.854569	0.916345	0.978121	1.039897
15	0.477477	0.535135	0.592793	0.65045	0.708108	0.765766	0.823423	0.881081	0.938739	0.996396
16	0.471847	0.525901	0.579955	0.634009	0.688063	0.742117	0.796171	0.850225	0.904279	0.958333
17	0.466879	0.517753	0.568627	0.619502	0.670376	0.721251	0.772125	0.822999	0.873874	0.924748
18	0.462462	0.510511	0.558559	0.606607	0.654655	0.702703	0.750751	0.798799	0.846847	0.894895
19	0.458511	0.50403	0.54955	0.595069	0.640588	0.686107	0.731626	0.777146	0.822665	0.868184
20	0.454955	0.498198	0.541441	0.584685	0.627928	0.671171	0.714414	0.757658	0.800901	0.844144

For the off-chain fees to be less than the on-chain fees, P must be less than 1:

$$n > \frac{96n_{in} + 54}{68} \approx 1.4n_{in} + 0.79$$

$$1.4n_{in} + 0.79 \leq 3n_{in}, \forall n_{in} \geq 1$$

$$\frac{n}{n_{in}} \geq 3$$

In order to save on fees consistently, the clearing transaction must settle 3 or more off-chain transactions for every input.

For a savings of greater than 50%, P must be less than 0.5:

$$n > \frac{96n_{in} + 54}{12.5} \approx 7.7n_{in} + 4.32$$

$$7.7n_{in} + 4.32 \leq 13n_{in}, \forall n_{in} \geq 1$$

$$\frac{n}{n_{in}} \geq 13$$

In order to save more than 50% in fees per transaction consistently, the clearing transaction must settle 13 or more off-chain transactions for every input.

Vulnerabilities

As with any network, the verification layer network will have certain vulnerabilities. This section addresses some of these vulnerabilities and possible solutions for these vulnerabilities.

Sybil Attack

A Sybil attack is when a malicious entity creates multiple identities in order to get disproportionate influence over a network [6]. Bitcoin has a natural deterrent to Sybil attacks through Proof-Of-Work. In order for a malicious actor to get control over the network, they must control more than 50% of the network hash rate. It doesn't matter how many nodes a malicious actor has if they do not have 51% or more of the hash rate. As an actor gains more hash rate, it becomes more expensive to maintain. This is due to the fact that mining takes large amounts of electricity and computing resources.

Unlike Proof-of-Work, the Avalanche consensus mechanism does not have an inherit deterrent to Sybil attacks. A way to remedy this is to borrow from Proof-of-Stake. Each node on the network must lock a certain amount of bitcoin for each node they posses. If the node doesn't have the requisite amount of staked funds, they simply can't participate in Avalanche. Ideally, the amount stake would be low enough for many different participants to have a node, but high enough to deter Sybil attacks. We propose that the stake for each node be 0.1 to 0.5 BTC.

Network Spoofing

To steal funds from a user, an attacker could impersonate the network. The user creates a request to deposit funds to the false network. The attacker then

generates their own public-private key pair and shares the public key. The user then mistakenly locks their funds in the multi-sig address that requires permission from the attacker to unlock. From there, the attacker could take the funds hostage to extort the user. The attacker can simply destroy their private key so that the user loses their funds. To reduce the risk of network spoofing, wallets can use the Tor network or a VPN to obscure their traffic. Wallets can also have a list of trusted nodes from which to download a list of peers.

Unclearable Funds

In order to clear funds, the user who controls the original settled outputs must provide their signature. If a user loses their private key and device, transactions linked to that user's settled outputs cannot be cleared.

To mitigate this risk, wallets can encourage the user to securely save seed words to recover a private key on another device. Funds should be cleared as soon as possible. Wallets can keep track of a user's transaction history and settled outputs. When a user reaches their clearing threshold, the clearing process takes place immediately. The maximum clearing threshold can be limited so that users funds are cleared sooner. Wallets can also suggest sending transactions larger than a certain threshold directly to the bitcoin network rather than through the verification layer.

Conclusion

Large bank card companies such as Visa and Mastercard provide customers with a quick way to spend money in their bank account without having to wait for a transfer. These companies verify that a customer has enough funds in their bank account to perform a transaction. The companies notify the banks of the transaction. The bank later clears transactions in large batches. The scaling solution presented can be likened to a decentralized, trustless version of these companies. The Bitcoin network acts as a decentralized, trustless bank.

The utility and efficiency of the second layer encourages users to join the network. As more users use this network for transactions, space in the memory pool of Bitcoin nodes is freed up. This can lead to a reduction in transaction times and fees. The reduction further reduces fees and clearance times on the

secondary layer. The use of a second layer network like this can effectively scale Bitcoin for every day applications.

References

[1] Bistarelli, Stefano, et al. “An Analysis of Non-Standard Transactions.” Frontiers in Blockchain, vol. 2, 2019, doi:10.3389/fbloc.2019.00007.

[2] Hofmann, Johannes. “On Bitcoin Script and Witness Sizes.” On Bitcoin Script and Witness Sizes | by Dr. Johannes Hofmann | Coinmonks | Medium, 14 June 2020, medium.com/coinmonks/on-bitcoin-transaction-sizes-97e31bc9d816.

[3] Nakamoto, Satoshi. “Bitcoin: A Peer-to-Peer Electronic Cash System.” Bitcoin - Open Source P2P Money, 2008, bitcoin.org/bitcoin.pdf.

[4] Popovic, Jovica, and Jay Kurahashi-Sofue. “Developer Documentation.” Avalanche, 2021, docs.avax.network/.

[5] Rocket, Team, et al. “Scalable and Probabilistic Leaderless BFT Consensus through Metastability.” Ava Labs: Build the Internet of Finance, Cornell University, 24 Aug. 2020, www.avalabs.org/.

[6] Trifa, Zied, and Maher Khemakhem. “Sybil Nodes as a Mitigation Strategy Against Sybil Attack.” Procedia Computer Science, vol. 32, 2014, pp. 1135–1140., doi:10.1016/j.procs.2014.05.544.

[7] Walker, Greg. “How Does Bitcoin Work?” How Does Bitcoin Work?, 2015, learnmeabitcoin.com/.