Byzantine Nodes in a Permissionless Blockchain Network

Philip Sigillito
University of Nebraska Omaha
College of Information Science & Technology
Omaha, USA
psigillito@unomaha.edu

***Abstract*—Blockchains and permissionless networks were introduced in 2008 with the release of the Bitcoin whitepaper. Permissionless networks are trustless, peer-to-peer, and decentralized networks where there is no authentication to join the network. Individual nodes are free to join and leave as they please. Permissionless networks maintain a distributed data structure called a blockchain which any node can contribute to. Permissionless networks face many challenges such as ensuring consensus of the authoritative blockchain, protecting from malicious nodes, incentivizing nodes to participate in the network, and choosing which node gets to add the next block to the chain.**

This paper investigates the impact of byzantine nodes participating in a permissionless network. We achieve this by running a simulated blockchain network and introducing faulty byzantine nodes to the network. Through our simulation, we find that byzantine nodes cause greater latency by making the network topology less connected. This results in more disagreement over which blockchain is authoritative and causes more blocks to be wastefully discarded. These effects cause the network to run more inefficiently and cause longer wait-times to confidently say a transaction has been successfully added to the blockchain.

Keywords—Permissionless Networks, Blockchain, Byzantine, Probabilistic Finality

I. Introduction

Bitcoin's introduction in 2008 was the first implementation of a blockchain network. For the first several years of its life, Bitcoin and blockchains remained a curiosity for excentrics on the Internet. However, over the years Bitcoin and competing cryptocurrencies have seen their value and popularity rise. In the last several years, cryptocurrencies have ballooned in value and are now household terms.

While the future of cryptocurrencies is uncertain, their popularity has encouraged an increased focus on blockchain and permissionless network technology. Various implementations of blockchains are now being applied to public and private applications including cryptocurrencies, supply chain management, inter-agency communication, auditing, contracts between parties, and distributed file storage.

Blockchains are append-only data structures. A good way to think of a blockchain is as a database where you can only write records to it. You cannot delete a record but you can nullify it by writing another record.[4] In a blockchain network, records are bundled together and appended to the database as a batch called a 'block'. As more blocks are added, they are linearly chained together, hence the term 'blockchain'.

There are variations to how a blockchain can be structured however, most blockchains still use the traditional layout of being chained together in a serial fashion such that the first block is chained to the second block which is chained to the third block and so on.

Blocks are chained together through hashing. Each block contains a header along with its bundle of transactions. The header contains a hash value partially based on the previous block. Since a block's hash is based on its parent, it is directly dependent on its parent and indirectly dependent on every block preceding its parent. As a result, tampering with one block invalidates its following blocks. For example, if a block is altered that will invalidate its stored hash and invalidate all of its childrens' hashes. This makes it easy for participants in the network to detect tampering of transactions within the blockchain and is why the network is described as 'trustless'.[4]

Each node maintains its own version of the blockchain which it considers to be the authoritative. There is no voting or coordination between the nodes to derive consensus in a traditional blockchain network. Nodes simply follow agreed upon rules for operating and validate proposed transactions and blocks through hashing encryption.

Many challenges arise for operating a permissionless blockchain network such as ensuring consensus on the authoritative blockchain, protecting from malicious nodes, incentivizing nodes to participate in the network, and choosing which node gets to add the next block to the chain. Nakamoto Consensus was introduced in the Bitcoin whitepaper,[1], to address these requirements:

"1) New transactions are broadcast to all nodes.

2) Each node collects new transactions into a block.

3) Each node works on finding a difficult proof-of-work for its block.

4) When a node finds a proof-of-work, it broadcasts the block to all nodes.

5) Nodes accept the block only if all transactions in it are valid and not already spent.

6) Nodes express their acceptance of the block by working on creating the next block in the chain, using the hash of the accepted block as the previous hash."[1]

Point three introduces the concept of proof-of-work (PoW) which is the mechanism for choosing which node gets to add the next block to the shared chain. Since there is no coordination within the network, there needs to be a way for determining who gets to write the next block to the chain. PoW is a 'logical puzzle' that the node must solve to gain the right to write the next block. Trying to solve the PoW puzzle is called 'mining'.

The PoW puzzle involves hashing the batch of transactions in the nodes proposed block along with data from the previous block and a proof value to generate a hash value. This generated hash value must meet an arbitrary requirement set by the network. The puzzle can only be solved through trial and error guesses by entering different proof values. When a node finds a working proof, they include that proof in their proposed block and share it with their neighbors to be distributed throughout the network. The neighbor nodes then validate the proof by confirming the hash value works and if it is valid and makes their version of the chain longer, they adopt it as the newest authoritative block and forward it to their neighbors and so on effectively flooding the network. If a node receives a block for a position in the chain that is shorter or equal to its stored blockchain version then it ignores the new block.

An important property of PoW is that it limits access to write to the blockchain. A node cannot create and transmit lots of blocks overwhelming the network because it is limited to the rate at which it can find valid proofs. This avoids too many blocks from being proposed at the same time for the same location in the chain and reduces the ability of malicious nodes to manipulate the chain. [5] provides a good explanation as to why PoW puzzles thwart Sybil attacks which involve an attacker posing as multiple nodes in order to manipulate network consensus.

In PoW networks, the arbitrary requirements for a proof to be valid automatically adjusts to be easier or harder depending on the number of nodes actively trying to create the next block. The PoW difficulty adjusts in order to maintain a predefined average rate of new blocks being created. If a lot of mining nodes leave the network, the proof difficulty needs to be lowered so that the rate of new block creation stays the same. This ensures that throughput does not slow-down to an unacceptable rate. On the other hand, if lots of nodes start mining then the proof difficulty gets harder. This ensures the rate of new block creation stays approximately the same regardless of how many nodes are trying to create a new block. The interval is typically set to a time that allows the new block to propagate throughout the network before another block is discovered.[4]

There is still a potential for another block to be created in a part of the network where the first block has not propagated to yet. This results in two valid blocks in the network vying for the same location in the chain. Remember, both blocks cannot be added because their proof is based on the hash of their parent block; therefore, a proposed block is only valid at one position in the chain. Nakamoto Consensus also includes the longest-chain rule by which nodes accept the longest valid version of the chain as authoritative.[1]

If there are two valid blocks being propagated at the same time, nodes will adopt whichever block gets to them first. Suppose two blocks, A and B, are discovered at the same time on opposite ends of the network so that half the network has A as their latest block and half the network has B as their latest block. Nodes will then continue mining using either A or B as their parent hash input when trying to find a new proof. Eventually a node will discover a new proof. Suppose this new block, C, has A as its parent. Nodes with A as their latest block will confirm that C is valid and add it to their chain. Nodes with B as their latest block will see that A and C are valid and that if they discard block B and adopt A and C their chain will be longer so they do just that. This is a key development in permissionless networks because there is no voting between nodes; Nodes simply react to proposed changes following an internal set of rules.

As we will show in the test results, sometimes nodes in the network can disagree about several blocks in the chain. For example, one side of the network might think the valid chain is A, B, C, D, E while the other side thinks the valid chai is A, B, F, G, H. Eventually, one side of the network will discover a new block making their chain longer and propagate it throughout the network before a new block can be discovered off of the competing chain.

The possibility of competing chains existing in a blockchain network is why we say that they operate on probabilistic finality. [3] defines Probabilistic finality succinctly as "For any honest node, every new block is either discarded or accepted into its local blockchain. An accepted block may still be discarded but with an exponentially diminishing probability as the blockchain continues to grow"

There may be conflicting blocks but the blocks before the conflicting position in the chain will be the same throughout the network. For example, assume one side of a network thinks A, B, C, D, E is the valid chain and the other side of the network thinks A, B, C, F, G is the valid chain. Even though there is disagreement, all chains agree that the prefix of the chain A, B, C is in the chain. We say that A, B, and C are 'safely' in the chain.

Transactions in Bitcoin are considered 'safely' added to the blockchain if it has been followed by five more blocks.[4] This is because the probability of a competing chain that does not contain our transaction's block and has more than five blocks following it without reaching our node is extremely unlikely. In contrast, there is a reasonable chance that the latest block on your chain is competing with another proposed block

in the network for that position and your block will be discarded.

Finally, it should be noted that Nakamoto consensus includes the requirement that nodes be motivated to participate in the network. Running hashes to discover proofs, validating blocks, and forwarding transactions cost computing power and incur electrical and hardware costs. To offset this, nodes that solve the PoW puzzle and create the newest block add an extra transaction that rewards itself a set amount of currency as well as a transaction fee from each transaction in the created block.

II. Objectives

Research in permissionless blockchain networks has increased recently; although, very few research papers could be found where a blockchain network was simulated to gather results. There are two existing academic focused simulators named SimBlock and BlockSim but they appear to be abandoned projects. In order to understand permissionless networks better and gather actual results, this paper aims to contribute uniquely by simulating a network that follows Nakamoto consensus.

Using our simulated network, this paper investigates how an implementation of Nakamoto Consensus performs when byzantine faults are introduced. To do this we will first run the simulation without any faulty nodes and then introduce faulty nodes to see their impact. For the sake of our experiments we can define a byzantine node as one that doesn't forward valid chains to neighbors, forwards invalid chains, and mines blocks that have invalid proofs (requiring its neighbors to detect the invalid proof).

Before getting into the simulation, we will first look at some of the relevant literature that has been published on the topic of blockchains. We will then go through how the simulator operates, the parameters of our tests and finally the results themselves.

III. Relevant Research

The amount of research related to permissionless networks has increased since they were introduced with the Bitcoin whitepaper in 2008. The Bitcoin whitepaper, [1], is perhaps the best starting point for describing the concepts behind permissionless networks although it does not provide research information regarding network performance.

Lots of academic research has been focused on replacements to proof of work (PoW). As more nodes join blockchain networks like Bitcoin, the networks' electrical energy consumption increases without throughput increasing since the hashing puzzle adjusts to become more difficult as more nodes join.

At the time of this writing (April 2021), Bitcoin's network has an annual carbon footprint equivalent to Sweden and consumes as much power as the Netherlands.[8]. A single Bitcoin transaction has the same carbon footprint as 1,169,819 VISA transactions and consumes enough electrical energy to power 38 average U.S. households for a day.[8]

With such staggering energy consumption, alternative proof methods have been adopted. Proof of Stake (PoS) is a common alternative that has been adopted by many second generation blockchains. PoS operates on the condition that mining nodes must make some unavailable and 'stake' its currency in order to try and solve a hashing puzzle. Typically, the more currency a node stakes, the easier its version of the puzzle will be. Additionally the puzzle is tied to a timer so that there is a new puzzle once per clock tick. This limits the amount of guesses a node can perform per puzzle. This method reduces the amount of hashing and makes nodes stake currency in the network, motivating them to want the network to operate properly.

There are many other alternative proofs such as proof of activity, proof of authority (PoA), proof of elapsed time (PoET), etc. [3] provides a thorough survey of these and other proof mechanisms for those interested.

Besides just reducing the amount of compute power consumed, some blockchains have harnessed the PoW mechanism to do meaningful work. Primecoin is a particularly clever implementation of this. Primecoin's proof of work involves discovering new prime numbers that are then made available to the public for scientific research.[9]

Research has also been conducted to see if different implementations of hashing impact the throughput and efficiency of blockchains. Faster hashing algorithms do increase the speed of determining if a transaction already exists in a node's list of proposed transactions although it does not increase the throughput of blocks mined.[2] The cause for this is because proposed transactions are typically stored in a hash map so faster hashing methods result in faster map lookup. Block creation does not increase with faster hashing because as we have already established the hashing puzzle difficulty will increase to keep the block throughput at a set interval.

Research has also been done to see if throughput can be increased. One approach is to move transactions off the main public chain and into small private chains so that the private nodes can transact with each other and then eventually submit one transaction to the slower public main chain when they are done interacting.[3]

Ethereum increases its transaction rate by decreasing its block interval to ten seconds per block. This is significantly less than the ten minute interval maintained by Bitcoin. Ethereum experiences many more competing chains than Bitcoin as a result. To cope with this, Ethereum uses the Greedy Heaviest Observed Subtree (GHOST) protocol. This protocol does not just consider which competing chain is longer but also considered the number of discarded blocks branching off of the chains.[10][6] It uses discarded blocks from the competing chains to better differentiate which is more authoritative resulting in less ties.

In the past couple years it appears that interest has also increased in private blockchains. These networks are permissioned and are inherently simpler, as outlined in [5], because participant identities are known, the number of participants can be known, late spawning nodes can be disallowed, and an upper bound on transmission delay or

timeout can be determined. Because of these reasons, private blockchains can operate using classical models for consensus.

Many research papers regarding private blockchains are case studies of applying blockchains and their ideas to a real-world business idea.

IV. Our simulation

A survey of existing simulation options was conducted before deciding to build a simulator. There are two academically focused simulators, SimBlock and BlockSim, although they appear to be abandoned projects. There are also testnets for popular blockchains such as Ethereum. These testnets do not cost money to operate and can be run privately. Testnets are intended for blockchain developers to test their applications on a network. Although they are robust, they are not easily customizable in terms of the fundamental ways they operate i.e. setting latency, toggling faultiness, toggling consensus mechanisms.

Our simulator is based on a tutorial provided by [7]. The tutorial provides a good starting point for creating a blockchain object, node instance, basic hashing, and polling neighbors for longer chains. In order for our simulation to comply with Nakamoto consensus and to be more robust we had to add additional functionality such as:

- Threading in order to mine and handle requests at the same time.
- Faulty node behavior when a node is set to faulty.
- Automatic forwarding when receiving a new transaction, mining a new block, or receiving a new chain.
- Functionality to manage the node's transaction pool and update it when a new transaction is received, the node receives a new block with transactions in the pool already added, and when the node has to abandon a block it is mining.
- Logging to record new transactions, when a node replaces its chain with a new valid chain, every block mined by a node, and when a node receives an invalid chain.

We also had to create scripts to interact with the network in order to run our experiment. These scripts included functionality to:

- Send a new unique transaction to a random node in the network to simulate new transactions being created by users.
- Send a request to mine periodically to keep the nodes mining.
- Send requests between the nodes to register each other as neighbors. This allows us to set the network topology by deciding which nodes are connected to each other.
- Send requests to start-up and shut-down the nodes.

Each node is implemented as a flask server containing a blockchain object. Flask is a micro framework commonly used for building small web servers. In our simulation, each node (flask server instance) is running on a different port on the same machine. Nodes then communicate with each other using http requests. Using http for API is useful because we can easily send requests manually or through scripts to customize each node's behaviour. For example, our simulation includes http requests to toggle a nodes faulty flag, show its current chain, show its neighbors, etc.

The hashing requirements for our PoW proofs are not difficult and cause minimal delay to the time it takes to mine a new block. Our PoW algorithm takes the parent block's proof and combines it with our current guess proof. We then pass the combined string into a SHA-256 hash and convert that into a hexadecimal string. If the hexadecimal string starts with '0000' then the proof is considered valid. To simplify simulating mining difficulty, a sleep timer runs for a random interval between 10 and 120 seconds on the mining thread. We still include the proof generation in order to validate blocks and chain the blocks together through hashing.

There were only a handful of challenges faced in implementing the simulation. The first was a limitation in hardware resources. Running a large network of dozens of nodes proved to quickly consume system memory causing lots of memory thrashing and slow response times. There were also issues with too many nodes trying to communicate with each other at the same time. In a large strongly connected network, the amount of traffic across the network caused servers to start refusing connections. Finally, poor implementation of the nodes caused unnecessary node inefficiency that had to be improved. For example, initial tests were conducted without the nodes threading its different tasks. This caused a node to be unresponsive while it was mining or validating a chain. To overcome this, threading was introduced for each blockchain task.

Again, source code is based on the tutorial provided by [7]. The full source code, including all of our code enhancements and scripts to interact with the simulation can be found at the link below.

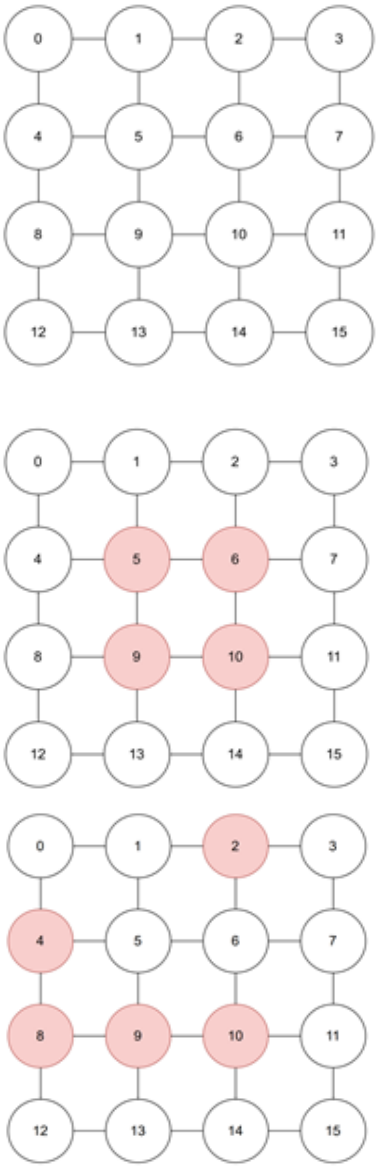
https://github.com/psigillito/block_chain_sim

V. Test Parameters

The simulator offers several customizable parameters that were set as follows:

- Latency of 0, 20, and 60 seconds between nodes for forwarding new blocks.
- Each simulation ran for five minutes.
- Each node was set to actively mining with an individual mining rate between 10 and 120 seconds.
- Unique transactions were sent to one random node in the network every 10 seconds.
- Nodes were set to proactively forward new longest chains instead of polling neighbors.

Our three main tests were run with a 4x4 grid network topology. The topology was set by configuring each nodes' neighbors list. The first grid had no faults, the second grid had four faulty nodes, and the third grid had five faulty nodes. The topologies are depicted below.



VI. Test Results

A. *Set 1: 4x4 Grid with no Faulty Nodes*

To determine baseline functionality and confirm that our network is working properly, we first tested the 4x4 grid with no faulty nodes. We ran this network for five minutes and applied no additional latency.

After five minutes of the simulation running, the 16 nodes had complete agreement on the authoritative chain. Twelve blocks were mined in total and no blocks were discarded due to competing chains. The authoritative agreed upon chain consists of the genesis block plus the 12 mined blocks for a length of 13. The chain order and which node mined the block are outlined below.

4x4 No Latency No Faults		
Count	**Hash**	**Miner**
1	1	Genesis
2	ef86	3
3	90f1	2
4	711d	15
5	9b72	15
6	44f0	11
7	58b6	1
8	3e21	10
9	355f	1
10	8efc	15
11	5272	15
12	69a2	14
13	f161	2

The next test we ran operated with the same parameters except with the latency increased to 20 seconds. This means that a block mined by node 0 would take 120 seconds to be forwarded to node 15. This is significantly more delay than the latency experienced in the first experiment.

The second test resulted in 27 blocks being mined with the longest valid chain in the network being 9 blocks long. It is important to note that this experiment's network mined over twice as many blocks as the first experiment but yielded a shorter chain. This can be attributed to the added latency.

Latency delayed mining nodes getting notified of another block already existing for that position, allowing more time for them to try and generate a competing block. Any additional mining based on the parent block of that position is redundant therefore, it is more efficient for the entire network for that node to restart mining on the next block position.

Each node maintains its own copy of the blockchain. Outlined in the table below are the final blockchains maintained by each node in the network for the second experiment. Each column represents a node's blockchain. For example column 0 represents node 0. The first row in column 0 is the genesis block with a hash of 1. The following row, be30, is the second block in the chain. The last row in the column is the latest block known by the node. Matching chains are highlighted the same color to make similarities more apparent.

20 Second Latency 4x4 No Faults							
Nodes 0-7							
0	1	2	3	4	5	6	7
1	1	1	1	1	1	1	1
be30	be30	be30	be30	be30	be30	be30	be30
fc41	fc41	fc41	fc41	fc41	fc41	fc41	fc41
7e33	7e33	7e33	7e33	7e33	7e33	7e33	7e33
7e94	7e94	7e94	7e94	7e94	7e94	7e94	7e94
341c	341c	341c	341c	341c	341c	341c	341c
ef88	ef88	ef88	ef88	ef88	ef88	ef88	ef88
63e1	63e1	63e1	63e1	63e1	63e1	63e1	63e1
3c27	3c27		3c27	3c27	3c27	3c27	3c27
Nodes 8-15							
8	9	10	11	12	13	14	15
1	1	1	1	1	1	1	1
be30	be30	be30	be30	be30	be30	be30	be30
fc41	fc41	fc41	fc41	fc41	fc41	fc41	fc41
7e33	7e33	7e33	7e33	7e33	7e33	7e33	7e33
7e94	7e94	7e94	7e94	7e94	7e94	7e94	7e94
341c	341c	341c	341c	341c	341c	341c	341c
ef88	ef88	ef88	ef88	ef88	ef88	ef88	ef88
63e1	63e1	63e1	63e1	63e1	63e1	63e1	63e1
3c27	3c27	3c27	3c27	3c27	3c27	3c27	c59f

As the table shows, all the nodes agree on the first eight blocks of the blockchain. Following probabilistic finality, any transaction in one of the first eight blocks of the blockchain are considered 'safe'. No new introduced blocks will cause the first eight blocks to be discarded.

The network is now trying to come to an agreement on what the ninth block will be. Nodes 0,1, and 3 through 14 all have 3c27 as their ninth block. Node 2 has not been notified yet of the discovery of a ninth block. It will either mine a competing block or receive 3c27 before its mining completes. Node 15 mined c59f as its ninth block before being notified of the discovery of 3c27.

The transactions held in 3c27 and c59f are not considered 'safe' because we are uncertain which competing chain will win. There is a possibility that one of the nodes containing 3c27 will mine another block based on 3c27 making a new longest chain that is disseminated through the network, get adopted by node 15, and cause c59f to be discarded. It is also possible, although unlikely, that node 15 will mine another block based on c59f before any of the other nodes mine another block, giving it the longest chain. It would then get broadcast throughout the network causing 3c27 to be discarded.

The impact of latency is much more pronounced when it is increased to 60 seconds. Using 60 second latency, 49 blocks were mined with a max chain length of 7. Several competing chains emerged with several nodes containing blocks that had not been shared with neighbors. For example, node 7 has two blocks more than its direct neighbor node 3. These results indicate that the agreed upon blockchain is unstable. There is no clear consensus and 60 second latency is unsustainable.

60 Second Latency 4x4 No Faults							
Nodes 0-7							
0	1	2	3	4	5	6	7
1	1	1	1	1	1	1	1
ccf8	b6ed	b6ed	b6ed	ccf8	1875	1875	1875
b84a	ba6f	ba6f	ba6f	b84a	0b08	0b08	0b08
ff26	551b	551b	551b	ff26	c78e	c78e	c78e
a7f6	5426	5426	5426	a746	42d5	42d5	42d5
bfa2	27ec	27ec	27ec	bfa2	d394	d394	d394
				7490	b174	b174	b174
				c6fb			
Nodes 8-15							
8	9	10	11	12	13	14	15
1	1	1	1	1	1	1	1
2f44	4162	1875	2f44	2f44	4162	4162	2f44
e955	00af	0b08	e955	e955	00af	00af	e955
c47a	3d41	c78e	c47a	c47a	3d41	3d41	c47a
b748	b609	42d5	b748	b748	b609	b609	b748
b595	2bdd	d394	a226	ec352	2bdd	2bdd	a226
		b174				a8fa	

B. *Set 2: 4x4 Grid with Nodes 5, 6, 9, and 10 Faulty*

In this set of experiments, the central four nodes of the network (5, 6, 9, and 10) are faulty. These nodes mine incorrect blocks, forward invalid chains, and don't forward valid chains. Results from running this network with 0 latency were not included as they were not particularly interesting. There was agreement between the non-faulty nodes of the longest chain and only showed the efficiency of a network with four less productive nodes. For the second experiment with 20 second latency, The longest chain was 6 blocks with 29 blocks being mined in total. Of those 20 blocks, 8 were faulty and contained incorrect proofs. The final chains in the simulation are outlined below with the faulty nodes highlighted red.

20 Second Latency 4x4 5,6,9,10 Faulty							
Nodes 0-7							
0	1	2	3	4	5	6	7
1	1	1	1	1	1	1	1
7b89	7b89	7b89	7b89	7b89	7b89	7b89	7b89
c203	c203	c203	c203	c203	c203	c203	c203
4efd	4efd	4efd	4efd	4efd	4efd	4efd	4efd
bbe8	bbe8	bbe8	bbe8	bbe8	bbe8	bbe8	bbe8
bca2	bca2	bca2	bca2	bca2	bca2	bca2	bca2
		c50f	c50f	c50f	c50f		c50f
Nodes 8-15							
8	9	10	11	12	13	14	15
1	1	1	1	1	1	1	1
5b7a	7b89	7b89	5b7a	5b7a	5b7a	5b7a	5b7a
3bd6	c203	c203	3bd6	3bd6	3bd6	3bd6	3bd6
df0b	4efd	4efd	df0b	df0b	df0b	df0b	df0b
1b2f	bbe8	69e7	1b2f	1b2f	1b2f	1b2f	1b2f
06ce	bca2	c342	06ce	06ce	06ce	06ce	06ce
d316		7227	d316	d316	d316		

These results show more chain competition than the experiment with no faulty nodes and the same latency. It appears that the transmission and validation of bad blocks generated by the fault nodes is not the main cause of this inefficiency.

Due to the nature of the byzantine nodes not forwarding valid chains, they effectively stop all traffic from moving through them. This causes the number of hops between nodes on either side of the network to increase as valid traffic must go around the faulty nodes. This allows more time for competing blocks to be mined and for competing chains to emerge. I anticipate that given a long enough time, either the yellow or green chain would win out and become the authoritative chain. Future competing chains would emerge but the first several blocks in all chains would be the same, making those blocks 'safe'.

The emergence of competing chains are further exacerbated when we increase the latency to 60 seconds. Eight blocks are used in the longest chain but 42 blocks have been mined of which 5 are faulty blocks. With this high latency we also see the largest disparity in chain sizes to be 4.

60 Second Latency 4x4 5,6,9,10 Faulty							
Nodes 0-7							
0	1	2	3	4	5	6	7
1	1	1	1	1	1	1	1
81d1	81d1	81d1	81d1	81d1	81d1	81d1	81d1
16a0	16a0	16a0	16a0	16a0	16a0	16a0	16a0
528b	528b	528b	528b	528b	528b	528b	528b
9a68	9a68	9a68	9a68	9a68	9a68	9a68	9a68
b338	b338	b338	b338	b338	b338	b338	b338
0ad0	0ad0	0ad0	0ad0	0ad0	0ad0	0ad0	0ad0
a3a6	a3a6	a3a6	a3a6	a3a6	a3a6	a3a6	a3a6
		6726	2e39	2e39		6726	
		d7b1				d7b1	
		7890				7890	
Nodes 8-15							
8	9	10	11	12	13	14	15
1	1	1	1	1	1	1	1
81d1	5649	81d1	81d1	5649	5649	5649	5649
16a0	910c	16a0	16a0	910c	910c	910c	910c
528b	5ef2	528b	528b	5ef2	5ef2	5ef2	5ef2
9a68	4d7b	9a68	9a68	4d7b	4d7b	4d7b	4d7b
b338	fdd3	b338	b338	fdd3	fdd3	fdd3	fdd3
0ad0	b237	0ad0	0ad0	b237	b237	b237	b237
a3a6					e3f5	e3f5	

C. Set 3: 4x4 Grid with Nodes 2, 4, 8, 9 and 10 Faulty

For this testing set, we have nodes 2, 4, 8, 9, and 10 set faulty. The topology of non-faulty nodes is linear except for node 3. Similar to the previous set, we have not displayed the results for no latency because the experiment shows agreement between all non-faulty nodes. Depicted in the graph below are the final chains for a latency of 20 seconds. There is some discrepancy between the nodes as to the longest chain. As you can see in the results, Block 62ec was generated in the lower half of the network and is being transmitted upwards through the network.

20 Second Latency 4x4 2,4,8,9,10 Faulty							
Nodes 0-7							
0	1	2	3	4	5	6	7
1	1	1	1	1	1	1	1
5c59	5c59	5c59	5c59	5c59	5c59	5c59	5c59
70b2	70b2	70b2	70b2	70b2	70b2	70b2	70b2
8eb1	8eb1	8eb1	8eb1	8eb1	8eb1	8eb1	8eb1
2a5b	2a5b	2a5b	2a5b	2a5b	2a5b	2a5b	2a5b
c7fb	c7fb	f6d5	f6d5	f6d5	f6d5	f6d5	f6d5
9ce8	9ce8	96da	96da	96da	96da	96da	96da
d783	d783	008f	008f	008f	008f	008f	008f
			62ec			62ec	62ec
Nodes 8-15							
8	9	10	11	12	13	14	15
1	1	1	1	1	1	1	1
5c59	5c59	5c59	5c59	5c59	5c59	5c59	5c59
70b2	70b2	70b2	70b2	70b2	70b2	70b2	70b2
8eb1	8eb1	8eb1	8eb1	8eb1	8eb1	8eb1	8eb1
2a5b	2a5b	2a5b	2a5b	2a5b	2a5b	2a5b	2a5b
f6d5	f6d5	f6d5	f6d5	f6d5	f6d5	f6d5	f6d5
96da	96da	96da	96da	96da	96da	96da	96da
008f	008f	008f	008f	008f	008f	008f	008f
62ec	62ec	62ec	62ec	62ec	62ec	62ec	62ec

Increasing the latency to 60 seconds resulted in a longest chain of nine blocks out of 53 mined of which 16 were faulty. Of the blocks mined, only 9.4% are used in the longest chain. This is the lowest percentage of nodes used in the final chain in any of our experiments. A large contributor is that there are five nodes generating only faulty nodes. Additionally, the valid nodes are communicating almost completely serially. For example, if node 0 discovers a new block, it will have to hop through every valid node (except 3) to reach node 12. This allows lots of time for competing blocks to be mined and eventually discarded before the next block is received.

60 Second Latency 4x4 2,4,8,9,10 Faulty							
Nodes 0-7							
0	1	2	3	4	5	6	7
1	1	1	1	1	1	1	1
aaf3	aaf3	aaf3	aaf3	ea56	aaf3	aaf3	aaf3
ac62	ac62	ac62	ac62	70ba	ac62	ac62	ac62
5b81	5b81	5b81	5b81	a61b	5b81	5b81	5b81
443e	443e	6460	6460	d149	6460	6460	6460
							c4f5
Nodes 8-15							
8	9	10	11	12	13	14	15
1	1	1	1	1	1	1	1
aaf3	0052	adca	aaf3	acf4	acf4	1d7f	acb8
ac62	6cb5	83ec	ac62	b905	b905	5fb2	ccfb
5b81	0c2b	08eb	5b81	075d	075d	da57	e924
6460	e8a5	f199	6460	13b3	13b3	7a97	fd4b
	2b0c	5664			4b6d	cd66	b2b2
	769d						

D. Additional Testing Set: Nine Strongly Connected Nodes

An additional set of tests were performed using nine strongly connected nodes. For this paper, strongly connected means each node is connected to every other node. Tables for the final chains have been omitted from this report. Results were similar to the previously described tests except faulty nodes and latency impact chain agreement significantly less. In a strongly connected network, each node is notified of a new block on the first broadcast. Higher latency does increase disagreement but higher latencies can be tolerated because you do not have to multiply the latency by the number of hops between nodes to determine the total latency. The most notable impact of introducing faulty nodes is that we removed a useful node from the mining pool.

VII. Conclusions

In reviewing byzantine behavior (as we defined it), it is clear that latency is the most important factor for the creation of competing chains. Latency results in more blocks being discarded and increases the time required for a block to be considered safely added to the chain. Byzantine nodes vary their impact depending on where they are located in the network and how they impact topology. Our tests imply that more connections and paths through the network reduce the ability of byzantine nodes to impact network performance.

Removing a node from the mining pool by making it byzantine is another obvious impact because it results in one less productive miner. Interestingly though, removing a node from the mining pool does not decrease the throughput of blocks mined. Most public blockchains, such as Ethereum and Bitcoin, adjust their hashing puzzle difficulty periodically to maintain a consistent rate of new blocks. If a mining node becomes faulty and stops mining valid blocks, the hashing puzzle difficulty will become easier. Since the byzantine node is no longer impacting throughput, it effectively becomes a listener in the network that periodically adds noise to the network through invalid blocks mined.

It should be noted that byzantine nodes in our experiments are different from malicious nodes which can work together to manipulate the production of new blocks.

VIII. Future Work

Our experiments were narrow in scope and there are lots of opportunities to expand simulation based experimenting. We only looked at the emergence of competing chains and how many mined blocks were discarded. We also only looked at a narrow definition of byzantine behavior, whereas in reality nodes could act faulty in numerous different ways.

Our metrics did not capture the cost of validating a bad chain received from a byzantine node. If this project were to be continued, I think that it would be useful to measure the compute or electrical cost of determining a received chain to be invalid.

Additionally, our simulation validated the hashes of blocks but did not validate the transactions of new blocks added to the chain. A node does not have to interact with any outside source to validate transactions since it has the entire history of transactions in its chain. It would be interesting to see how faulty transactions would impact the network's efficiency.

References

[1] Nakamoto, S. (n.d.). Bitcoin: A Peer-to-Peer Electronic Cash System. Retrieved from www.bitcoin.org

[2] F. Wang et al., "An Experimental Investigation Into the Hash Functions Used in Blockchains," in *IEEE Transactions on Engineering Management*, vol. 67, no. 4, pp. 1404-1424, Nov. 2020, doi: 10.1109/TEM.2019.2932202.

[3] Y. Xiao, N. Zhang, W. Lou and Y. T. Hou, "A Survey of Distributed Consensus Protocols for Blockchain Networks," in *IEEE Communications Surveys & Tutorials*, vol. 22, no. 2, pp. 1432-1465, Second Quarter 2020, doi: 10.1109/COMST.2020.2969706.

[4] Xu, X., Weber, I., & Staples, M. (2019). *Architecture for Blockchain Applications*. Cham: Springer International Publishing.

[5] R. Pass and E. Shi, "Rethinking Large-Scale Consensus," 2017 I*EEE 30th Computer Security Foundations Symposium (CSF)*, 2017, pp. 115-129, doi: 10.1109/CSF.2017.37.

[6] Yonatan Sompolinsky and Aviv Zohar. Secure high-rate transaction processing in bitcoin. In *Financial Cryptography*, 2015

[7] Learn blockchains by building one. (n.d.). Retrieved April 29, 2021, from https://hackernoon.com/learn-blockchains-by-building-one-117428612f46

[8] Bitcoin energy consumption index. (2021, April 10). Retrieved May 01, 2021, from https://digiconomist.net/bitcoin-energy-consumption

[9] Primecoin. (n.d.). Primecoin/primecoin. Retrieved May 01, 2021, from https://github.com/primecoin/primecoin/wiki/FAQ

[10] V. Buterin, "A next-generation smart contract and decentralized application platform", white paper, 2013. Retrieved May 01, 2021, from https://ethereum.org/en/whitepaper