2019 IEEE International Conference on Blockchain (Blockchain)

FastChain: Scaling blockchain system with informed neighbor selection

Ke Wang, Hyong S. Kim
Carnegie Mellon University
kewang1@andrew.cmu.edu, kim@ece.cmu.edu

***Abstract*—The main advantage of the Bitcoin-style blockchain systems is immutability of data and decentralized authority. Despite its success, Bitcoin suffers from scalability issues. The potential of the blockchain technology cannot be fully realized unless it is made more scalable. We extend previous analysis on a fundamental trade-off between the throughput and forking rate of the blockchain system. We further propose FastChain to increase the throughput of the blockchain system by reducing the block propagation time. FastChain adopts bandwidth-informed neighbor selection. Miners disconnect from bandwidth-limited neighbors and favor the nodes with higher bandwidth. We build a Blockchain Dynamics Simulator (BDSim) to evaluate the performance of FastChain. FastChain improves the throughput by *20%~40%* in the normal operations.**

Keywords—Bitcoin, blockchain, scalability, throughput, forking rate, peer-to-peer network, neighbor selection

I. INTRODUCTION

Bitcoin is the most widely adopted cryptocurrency with more than *17* million bitcoins in circulation and *76* billion USD market capitalization as of March 2019. Bitcoin's underlying technology, the blockchain, is considered for applications in smart contract [1], health care [2], supply chain [3], review system [4] and data storage system [5].

The key properties of blockchain are immutability of data and decentralized authority based on peer-to-peer networks. A blockchain consists of continuously growing linked data blocks. In Bitcoin-style blockchains, a block is mined based on the proof of work. It requires computational or financial efforts to mine a block. These blocks are linked by a cryptographic hash of the previous blocks. Data in the blocks are impossible to modify without having to modify all following blocks. As long as the blockchain is growing, data in the blockchain is practically immutable. There is no central authority to validate the blockchain. Anyone can receive the data and mine to grow a blockchain without any centralized coordination. This peer-to-peer network of decentralized miners validates and maintains the integrity of data in the blockchain

A few parameters of a blockchain system are of interest. The *block rate* is defined as the number of blocks mined per second. The *block size* is defined as the number of bytes of a block. It takes time to transmit the block over the peer to peer network. The *block propagation time* is defined as the time needed for a miner to receive the new block. Since a block is received by different miners at different times, the distribution of the block propagation time is more commonly used. If a conflicting block is mined while another block is being propagated in the network, a blockchain fork occurs. Miners resolve blockchain forks by selecting the chain that contains the most blocks. Blocks not in the longest chain are discarded and the computing power put into them is wasted. The *forking rate* is defined as the number of discarded blocks per second. The *effective block rate* is the number of blocks added to the longest chain per second, i.e.,

$$block\ rate = forking\ rate + effective\ block\ rate$$

Despite its success, Bitcoin has scalability issues. The *throughput*, defined by the number of transactions per second, is limited by the maximum block size multiplied by effective block rate.

$$throughput = \frac{Transactions}{block} * \frac{effective\ blocks}{second}$$

Bitcoin currently has a maximum throughput of *7* transactions per second. We cannot simply increase the block size or mine blocks faster to improve the throughput. There is a three-way tradeoff between the block size, the block rate and the forking rate. Decker [6] shows that the block propagation time is linearly related to the block size. Increase in the block size increases the block propagation time, which in turn increases the forking rate. Increase in the block rate while maintaining the same block propagation time also increases the forking rate. High forking rate wastes the mining power and makes the blockchain system less secure.

There are a few proposed solutions to improve the scalability of the blockchain system. Bitcoin-NG [7] proposes to include two types of blocks in the blockchain. *Key blocks* are used for leader election. Miners compete to mine key blocks to become the leader. The leader then generates *micro blocks* that contain actual transactions. The throughput of the blockchain system greatly increases, as micro blocks are generated at a much higher rate. Payment networks [8, 9, 23, 24] propose off-chain transactions. Users open a payment channel by committing a single transaction to the blockchain. The users then perform multiple payments between each other without additional transactions on the blockchain. Transactions are posted to the public blockchain only when a payment channel is closed. The scalability increases as majority of transactions occur off the blockchain. Sidechain designs [11] propose to include multiple blockchains interfacing with the main blockchain. The throughput of the blockchain system could increase in certain cases. Some of the transactions could occur within the sidechains without impacting the main blockchain. However, sending funds between blockchains results in two transactions recorded in the main blockchain. This makes the scalability

978-1-7281-4693-5/19/$31.00 ©2019 IEEE
DOI 10.1109/Blockchain.2019.00058

issue even worse. Sidechains also require miner coordination between different blockchains.

In this paper, we approach the scalability of the blockchain system at its peer-to-peer network. We optimize blockchain system using better neighbor selection algorithm to reduce block propagation time. Existing solutions can also benefit from our work. For example, Bitcoin-NG may experience blockchain forks on every key block generation [7]. Reducing the block propagation time could decrease the key block forking rate. Falcon [13] and FIBRE [10] propose a specialized block relay network. The proposed auxiliary block relay network helps reduce the block propagation time. The neighbor selection in this paper is orthogonal to these works and targets at nodes in the general peer-to-peer network in the Bitcoin.

We observe that Bitcoin nodes are provisioned with different network bandwidth [13]. The block propagation time is closely related to network bandwidth. Nodes with *500Mbps* bandwidth take an average of *1.55* seconds to receive a block, while nodes with *256Kbps* bandwidth takes *71.71* seconds on average [12]. Currently a miner chooses its neighbors randomly, agnostic to the bandwidth information [14]. Nodes with limited bandwidth easily become a bottleneck in the block propagation. We argue that miners should refreshing its connections by disconnecting the low bandwidth neighbors and randomly picking new neighbors. Using these observations, we propose FastChain, a simple and effective optimization algorithm to scale the blockchain system. FastChain prescribes miners to make informed neighbor selection rather than random selection. FastChain also take measures to avoid introducing eclipse attack targeting at this neighbor selection. FastChain reduces the block propagation time by *20%~40%* in most cases.

We design and build a Bitcoin blockchain simulator, BDSim (Blockchain Dynamics Simulator) to evaluate FastChain's performance. Existing Bitcoin simulators have various limitations and they are not suitable for our purpose. Bitcoin plug-in for Shadow [16] directly executes Bitcoin software inside the simulation framework. Executing the highly computational cryptographic operations limits the scalability of experiments [17]. Simbit [18] over-simplifies the peer-to-peer network of the blockchain system. Gervais's Bitcoin Simulator [19] is built on NS3 [20]. It simulates a static topology without mechanism for miners to refresh their neighbor connections. We thus build our own Bitcoin simulator. BDSim allows simulating the dynamics of Bitcoin's peer-to-peer network. BDSim contains various data probes. Statistics such as block propagation time and network-related metrics are collected. BDSim could be used to understand different block propagation mechanisms.

We make the following contributions in this paper.

- We extend previous analysis to formulate a linear relationship between the probability of forking and average block propagation time (Section III). This serves as the theoretical foundation of FastChain: reducing block propagation time reduces the forking rate, which in turn scales up the blockchain system.

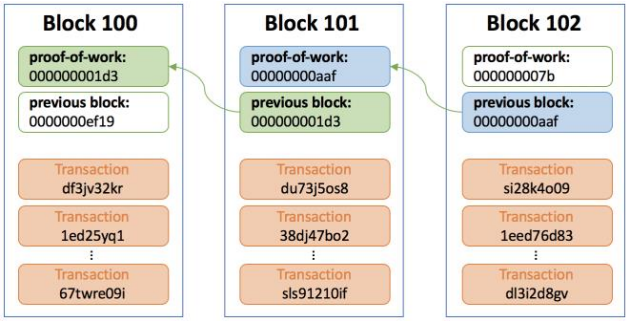

Figure 1. Bitcoin blockchain

- We propose FastChain (Section IV), an optimization algorithm for neighbor selection to reduce the block propagation time.
- We implement a discrete-event Bitcoin simulator, BDSim (Section V). BDSim is capable of simulating the dynamics of the blockchain peer-to-peer network at the scale of hundreds of thousands of miners. Using this simulator, we evaluate the performance of FastChain (Section VI). FastChain reduces block propagation time by *20%~40%*.

Although we develop and present FastChain in the context of Bitcoin blockchain, the underlying technique applies to other types of blockchain systems built on top of a peer-to-peer overlay network.

II. BACKGROUND

A. Bitcoin and its blockchain

Bitcoin is a decentralized crypto-currency [21]. It implements a blockchain protocol to record and serialize transactions among users. Transactions are recorded in individual blocks. Each block is built by choosing a nonce value such that the hash of the block meets a pre-set condition. The computational effort to find such nonce is known as the proof of work. Blocks are linked together as shown in Figure 1. Each block contains the hash of its previous block. Computing nodes compete with each other to be the first to build a block by finding a satisfactory nonce. These nodes are called miners. These miners are interconnected by a peer-to-peer network. We use miners and nodes interchangeably in the rest of the paper. Both newly built blocks and transactions are broadcast to miners through the peer-to-peer network. Miners then verify the newly built block independently and start working on the next block.

Blockchain can be viewed as a global append-only log. Each miner keeps a local copy of the log. Blockchain fork occurs due to the block propagation time. If a conflicting block is found while another block is propagating in the network, a blockchain fork occurs. Forked blocks may contain different sets of transactions and in different orders. As a result, miners can have different versions of the blockchain. Miners subsequently choose either of the branch to work on. Blockchain miners resolve forks by selecting the longest chain. When there are multiple branches with the same length, Bitcoin clients [14] select the branch it received first.

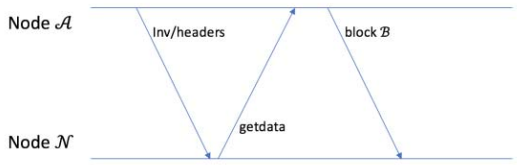

Figure 2. Advertisement based propagation

B. Bitcoin peer-to-peer network

Bitcoin nodes connect to neighbors over TCP/IP. Each node maintains a *tried* and *new* table to store public IPs. The *tried* table stores the IP addresses of neighbors to whom the node has established a connection. The *new* table stores the addresses of nodes to whom this node has not yet connected. The *new* table is initially populated with information from a DNS server when the node first joins the network. Nodes also exchange neighbor information via *addr* messages to update their *new* table periodically. The node randomly selects from the two tables when it needs to setup a new outgoing connection. In the default configuration [14], the Bitcoin node picks *8* nodes from the tables randomly to establish TCP connections. The node also accepts incoming connections. One node can have at most *125* connections in total.

Bitcoin adopts an advertisement based information request mechanism as shown in Figure 2. When node $\mathcal{A}$ learns about a new object (a transaction or a block), the miner first broadcasts an *inv* message or *headers* message to announce the existence of the new object. If neighbor $\mathcal{N}$ does not already have that object, it sends a *getdata* message to $\mathcal{A}$ to request the object. Node $\mathcal{A}$ then sends the actual object to $\mathcal{N}$. This advertisement based mechanism avoids unnecessary block transmissions. A block can be as large as *1MB* while an *inv* message is only a few bytes. The *inv* message only contains the type and hash of the object [22]. Node $\mathcal{N}$ requests a given object from the neighbor that sends the advertisement first [12]. Node $\mathcal{N}$ sets a timeout for *getdata* request. Currently the timeout value for block request is *10* minutes. This is to tolerate network congestion and slow nodes. Node $\mathcal{N}$ also selects three *high-bandwidth* (HB) mode neighbors. An HB neighbor sends a new block unsolicitedly through a compact block message.

III. Probability of Forking in Blockchain

We extend the analysis in [6] to formulate the linear relationship of the probability of forking and average block propagation time *D*. Denote P_{single} as the probability of any given nonce satisfying the proof-of-work requirement. P_{single} is configurable and represents the difficulty of mining. In Bitcoin, $P_{single} \approx 3.83 * 10^{-23}$ as of Mar 2019. Assume *n* miners in the blockchain system. For simplicity, we assume each miner possesses the same amount of computing power. Denote $g(t)$ as the rate at which a new block propagates to miners. $g(t)$ represents the percentage of miners that receive the new block at tth second after that block is mined. Denote $G(t)$ as the cumulative function of $g(t)$, i.e.,

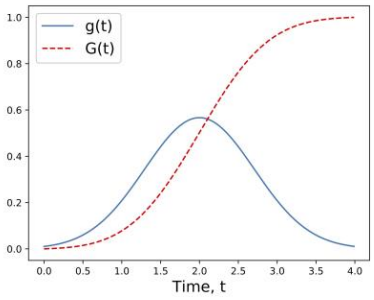

Figure 3. Block propagation distribution

$$G(t) = \int_0^t g(\alpha)d\alpha \quad (1)$$

An example of $G(t)$ and $g(t)$ is shown in Figure 3. Denote T_{max} as the maximal time for block propagation. All miners learn the existence of the new block in T_{max} seconds after it is mined. $G(t) = 1$ for $t \geq T_{max}$ and $g(t) = 0$ for $t \geq T_{max}$. Denote F as the number of conflicting blocks found while a given block is being propagated. Similar to [6], we consider $Pr(F = 0)$, the probability of no conflicting block found. Only nodes which have not received the new block may produce a conflicting block. Denote *s* as the number of hashes per second a miner can compute. The total hashes computed towards finding a conflicting block is $\int_0^{T_{max}} n * \big(1 - G(t)\big) * sdt$. No conflicting block is found when none of these hashes satisfies the proof-of-work requirement.

$$Pr(F = 0) = (1 - P_{single})^{\int_0^{T_{max}} n*(1-G(t))*sdt} \quad (2)$$

Equation 2 is first derived in [6] in a similar form with slightly different notations. We now further develop Equation 2 to relate the probability of forking with mean block propagation time. Using the approximation that $(1-\theta)^k \approx 1 - k\theta$ when θ is extremely small, Equation 2 can be approximated as

$$Pr(F = 0) \approx 1 - P_{single} * \int_0^{T_{max}} n * \big(1 - G(t)\big) * sdt \quad (3)$$

By the definition of $g(t)$, mean block propagation time *D* can be computed from $g(t)$ as follows.

$$D = \int_0^{T_{max}} t * g(t)dt$$

which is,

$$D = \int_0^{T_{max}} tdG(t)$$

leading to

$$D = T_{max} - \int_0^{T_{max}} G(t)dt \quad (4)$$

Substituting Equation 4 into Equation 3, we have the following.

$$Pr(F = 0) \approx 1 - P_{single} * ns * D \quad (5)$$

Correspondingly,

$$Pr(F \geq 1) \approx P_{single} * ns * D \quad (6)$$

The block mining process can be modelled as a Bernoulli process with each trial succeeding with probability P_{single}.

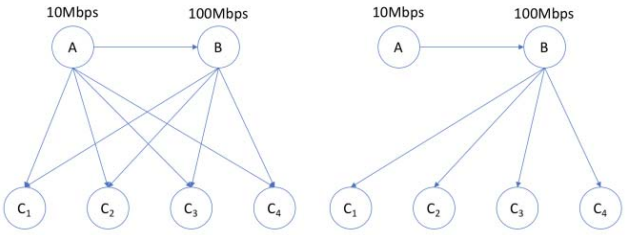

Fig 4.a Fig 4.b

Figure 4. Motivating example

Expected number of trials before the first successful trial is $1/P_{single}$. The average time to mine a block is $1/(ns * P_{single})$. Thus, we have raw block rate (all mined blocks including forked blocks) $B = ns * P_{single}$. Equation 6 can be re-written as

$$Pr(F \geq 1) = BD \quad (7)$$

Equation 7 shows that the probability of a blockchain fork is linearly related to the mean block propagation time D. Larger probability of forking $Pr(F \geq 1)$ indicates higher forking rate.

Equation 7 also shows that under the same probability of forking, P_{single} can be increased, i.e., mining made faster, if the block propagation time is reduced. Increasing P_{single} while keeping the same $Pr(F \geq 1)$ leads to higher throughput. This motivates FastChain to reduce the block propagation time to scale the blockchain system.

IV. FastChain: Informed Neighbor Selection

An example in Figure 4 shows the motivation behind FastChain. Node $\mathcal{A}$ has *10Mbps* upstream bandwidth. Node $\mathcal{B}$ has *100Mbps* upstream bandwidth. All links have *200ms* latency. For simplicity, we assume nodes broadcast a block directly without first sending *inv* messages. Figure 4.a corresponds to the random neighbor selection policy. Both $\mathcal{A}$ and $\mathcal{B}$ have *5* neighbors. Figure 4.b corresponds to bandwidth-informed neighbor selection policy. Node $\mathcal{A}$ is connected to node $\mathcal{B}$ only, while node $\mathcal{B}$ is connected to all other nodes. Suppose node $\mathcal{A}$ mines a block of size $10^6 Bytes$. In Figure 4a, bandwidth of node $\mathcal{A}$ is split among *5* connections. Each connection has *2Mbps* bandwidth. It takes *4.2s* for its neighbors to receive the new block. In Figure 4b, node $\mathcal{A}$ first transmits the block to node $\mathcal{B}$ in *1s*. Node $\mathcal{B}$ then transmits the block to the rest of the nodes in *0.52s*. The average block propagation time in the right topology is *1.416s*, almost *3* times less than the left topology.

Node $\mathcal{A}$ is the bottleneck for the block propagation. While node $\mathcal{A}$ is transmitting the block, other nodes with higher bandwidth are idle and not utilized. We propose the following policy to scale the blockchain system.

FastChain Policy: A node should avoid low bandwidth nodes when selecting neighbors. This neighbor selection policy avoids congesting bandwidth limited nodes.

A. FastChain implementation

FastChain consists of two phases: *Bandwidth Monitoring* and *Neighbor Update*.

Phase 1 Bandwidth Monitoring: Each node maintains a *recent-bandwidth* table. Whenever a node $\mathcal{A}$ receives a valid block from a neighbor $\mathcal{N}$, node $\mathcal{A}$ estimates the recent bandwidth of $\mathcal{N}$ using the following equation.

$$BD_{\mathcal{N}} = \frac{block\ size}{one\ hop\ block\ propagation\ time}$$

One-hop block propagation time represents the time taken for neighbor $\mathcal{N}$ to propagate the block to node $\mathcal{A}$. It is measured from the reception of *inv* message to the reception of the new block. $\mathcal{A}$ writes a new entry $BD_{\mathcal{N}}$ into the recent-bandwidth table. The entry expires after Δ blocks to ensure the bandwidth estimation reflects the recent state of the neighbor.

Phase 2 Neighbor Update: Each miner periodically refreshes its neighbor connections. A miner sorts its *recent-bandwidth* table every *T* blocks. It picks *K* neighbors with the least bandwidth. For each of the *K* neighbors, if its bandwidth is less than a threshold *B*, the miner disconnects from that neighbor and will not respond to the neighbor's *inv* message even if they are connected later again. The miner also picks up new neighbors randomly from its *new* or *tried* table to ensure that it has at least *M* outgoing connections. As the miner continuously disconnects from slow neighbors, its neighbor set will converge to a set of high-bandwidth neighbors.

Since FastChain adopts a bandwidth-dependent neighbor selection (as opposed to completely random neighbor selection), it is important not to introduce new vulnerabilities for eclipse attack. In an eclipse attack, the adversary induces the victim node to establish all connections to the attack nodes [15]. The adversary then perform attack such as not relaying the latest information to the victim [12, 25]. FastChain takes two measures for this purpose. First, a node in FastChain only disconnects a neighbor if its bandwidth is less than an absolute threshold *B*. Adversaries could not induce a node to evict a normal neighbor as long as that neighbor relays block in a timely manner. Second, when a node disconnects a slow neighbor, it will randomly select a new neighbor. Adversaries cannot guarantee the node will select an attack node as new neighbor.

Parameters T, K, B, M are configurable for the best performance. Smaller T enables a node to refresh its neighbor list more frequently. Nodes can adapt to network changes quickly. Higher *K* corresponds to a more aggressive policy. It may lead to thrashing of the network. As discussed above, the use of *B* is to ensure that there is no unnecessary disconnection when all neighbors send blocks reasonably fast. It helps the network converge and prevents eclipse attack. For example, *B* could be set such that as long as the neighbor relays the block within *10* seconds, there will be no disconnection. Selecting *M* outgoing connections makes sure no node or small group of nodes is completely disconnected from the main network. In Bitcoin *M* is set to *8*. We found in simulation that setting *T=5, K=2* and *M=8* could result in fast adaptation of network change. The network converges usually after *20-25* blocks are mined.

V. BDSim: Blockchain Dynamics Simulator

BDSim is a discrete event simulator for blockchain systems. It simulates the major aspects of Bitcoin protocol, including the peer to peer network, block mining and block relay. Each node

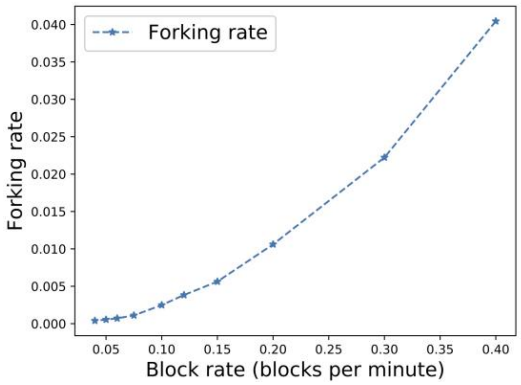

Figure 6.a. Impact of block rate

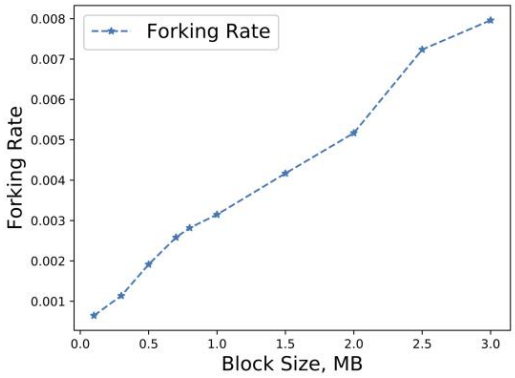

Figure 6.b. Impact of block size

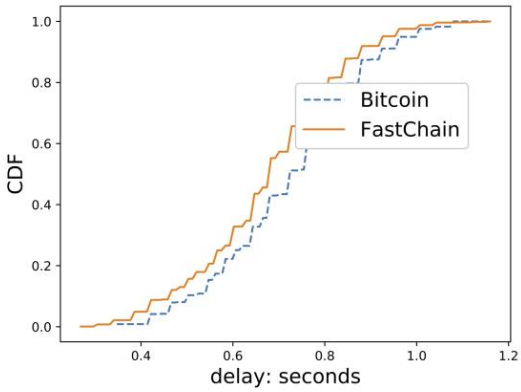

Fig 7.a. Constant

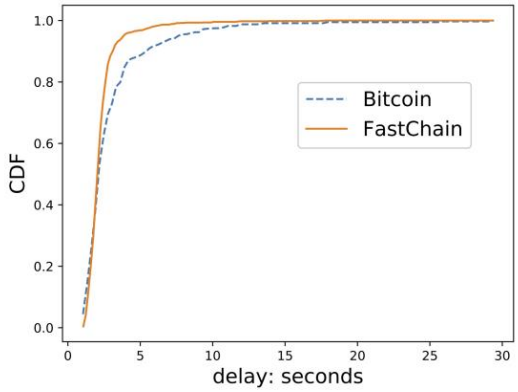

Fig 7.b. Uniform

in the blockchain is abstracted as a data object. The data object keeps all states of the node, denoted as *node state*. The node state includes the node's local view of the blockchain, list of its neighbors, its *tried* table, *new* table and the *recent-bandwidth* table. The node state also includes statistics, such as the latency for each block received by the node. We define the *system state* as the union of all node states. BDSim follows a standard discrete event simulator. The system state is updated by *events* at discrete time points. BDSim contains the following types of events.

- Type 1 - *New block mined event*. This indicates a miner completes mining a new block. The miner updates its copy of the blockchain and schedules events for advertising the block to its neighbors.
- Type 2 - *Message received event*. A message sent from a source node is received by a destination node. The destination node takes corresponding actions depending on the message type. The message type includes *addr*, *inv*, *getdata*, *block*.
- Type 3 – *Refreshing neighbor event*. A node refreshes its neighbor connections. The node decides to disconnect certain neighbors according to the neighbor selection policy. Two policies are implemented: random neighbor selection and bandwidth informed neighbor selection. The node then establishes new connections as needed.

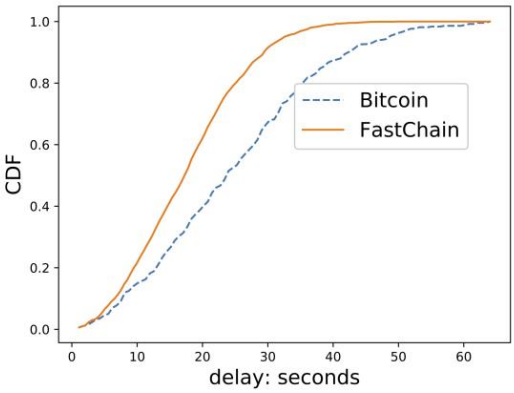

Fig 7.c. Heavy-tail

Each event is associated with a scheduled execution time. Events are stored in a priority queue ordered by the execution time. Simulation progresses as the next earliest event is executed

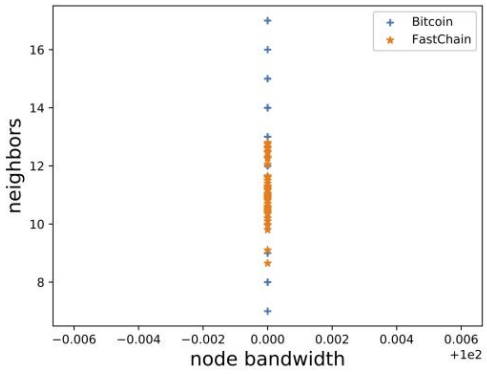

Fig 8.a. Constant

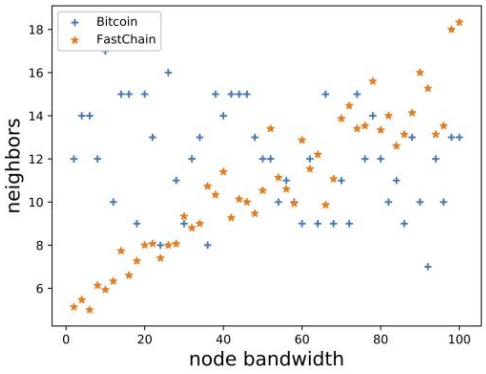

Fig 8.b. Uniform

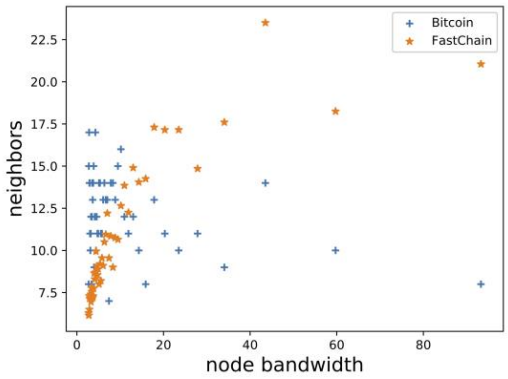

Fig 8.c. Heavy-tail

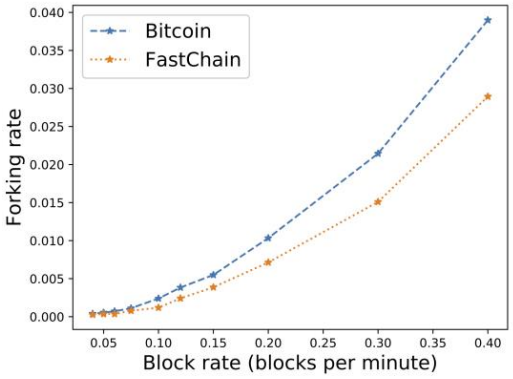

Figure 9.a. Forking rate

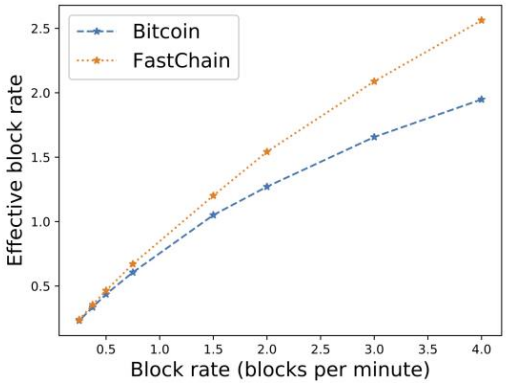

Figure 9.b. Effective block rate

from the queue and the system state is updated accordingly. The simulation stops when there is no more event in the queue.

Simulation of Cryptographic Operation: We do not simulate the computationally intensive cryptographic operation to find the proof-of-work. Instead, we model the required time to mine a block to be an exponentially distributed random variable. The mean of the exponential random variable is configured to represent the average block mining time.

Simulation of Network Layer: We do not simulate the entire TCP/IP stack. We focus on the block propagation time and simplify the propagation process as follows. The latency of relaying a message depends on two factors: the link latency and the link bandwidth.

$$latency = T_{link} + \frac{data\ size}{link\ bandwidth} \quad (8)$$

Link latency is a static property associated with the link. It is set when the link is initiated and does not change with time. We use logical links between two nodes. Link bandwidth is the bandwidth allocated to this link to transmit the message. Link bandwidth depends on the bandwidth of the source and destination nodes and may change over time. For instance, if the source node $\mathcal{A}$ has *100Mbps* upstream bandwidth and is transmitting a block to *10* neighbors concurrently, then the link bandwidth for each transmission is *10Mbps*. In the middle of the transmission, if $\mathcal{A}$ decides to start to transmit a message to the *11th* neighbor, the receiving time of the block for previous *10* other neighbors would be delayed, as the bandwidth to transmit this block decreases. BDSim is capable to capture this kind of change of network condition. Admittedly, this is only a simplified model and it does not capture the full feature of the underlying physical network. We leave that to our future work. Still, we validate BDSim against production measurements in [6]. We found that under the same network configuration, the average and median block propagation time is within the limit of production Bitcoin system.

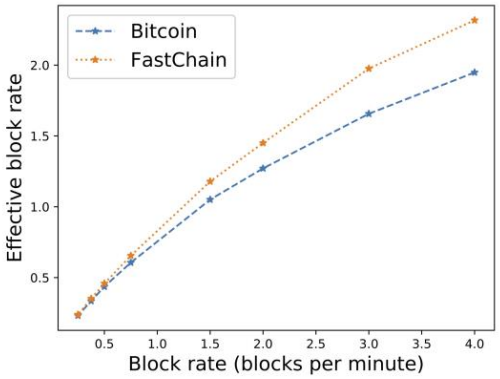

Fig 10.a. linear proportional

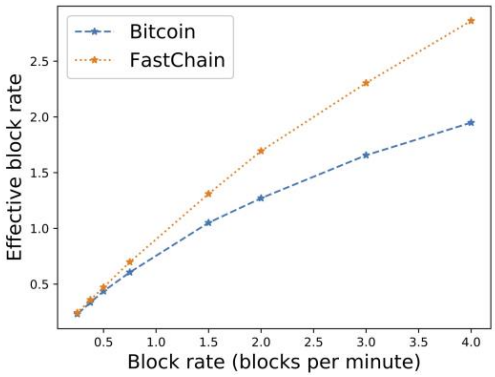

Fig 10.b. reverse proportional

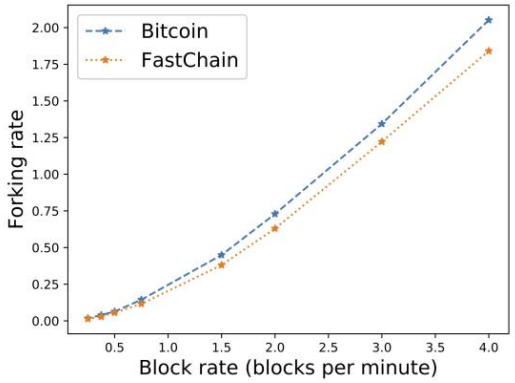

Fig 11.a. high link latency, forking rate

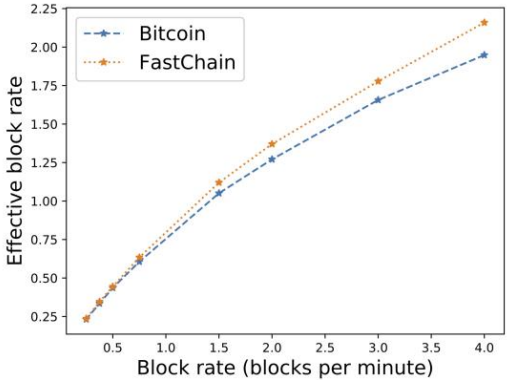

Figure 11.b. high link latency

VI. Evaluation

A. *Forking rate*

We first demonstrate that forking rate increases as either block rate or block size increases. Recall that forking rate is defined as the number of discarded blocks per second. In Figure 6.a, we fix the block size at *1MB* and vary the mining difficulty to vary the block rate. In Figure 6.b, we fix the block rate to one block every *10* minutes and vary the block size. Both figures confirm our analytical result in Section III. Increasing block size increases the block propagation time. Increasing block rate decreases the block mining interval. As shown in the result, this leads to higher probability for miners to mine conflicting blocks while newly mined blocks are being broadcast to miners.

B. *Impact of node bandwith*

FastChain reduces the average block propagation time in Bitcoin. The advantage of FastChain is significant when the nodes have highly varying bandwidths. We evaluate *3* situations. 1) The node bandwidth is the same for all nodes. 2) The node bandwidth follows a uniform distribution. 3) The node bandwidth follows a heavy tail distribution. Among the three situations, the heavy tail distribution is the closest to Bitcoin as reported in [13]. Figure 7 shows the Cumulative Distribution Function (CDF) of block propagation time for both Bitcoin protocol and FastChain. The difference of average block propagation time between FastChain and Bitcoin is *only 6%* for the constant case. However, the reduction of the block propagation times is *19.8%* for the uniform distribution, and *28.0%* for the heavy tail distribution. For the constant bandwidth case, there is almost no improvement for FastChain. FastChain directs higher bandwidth nodes to be connected to more neighbors. This neighbor selection policy does not have much impact when all the nodes have the same bandwidth. The only gain (*6%*) comes from the fact that FastChain results in a load-balanced topology, where random neighbor selection could congest a few nodes.

In Figure 8, each dot represents a node. The *x* coordinate represents the bandwidth of the node. The *y* coordinate represents the number of neighbors of the node. For the constant bandwidth case, all nodes have the same bandwidth. A few nodes in Bitcoin have distinctively more neighbors than others due to the random neighbor selection. These neighbors could get congested when transmitting and receiving blocks. Nodes in FastChain result in a more balanced network where each node has similar number of connections. More nodes get congested when the nodes have highly varying bandwidth, as in the case of uniform and heavy tail distribution. Nodes with less bandwidth are easier to get congested due to the random selection policy.

In FastChain, the number of neighbors for a node is linearly proportional to the node bandwidth as shown in Figure 8. Nodes with lower bandwidth have less neighbors on average.

Due to the lower block propagation time, FastChain achieves less forking rate and higher throughput of transactions. As shown in Figure 9.a, we configure the block size to be *1 MB* and vary the block rate. The forking rate for both FastChain and Bitcoin increases as blocks are generated faster. However, FastChain lowers the block propagation time and reduces forked blocks as shown in Figure 9.a. Correspondingly the effective block rate of FastChain is *30%* higher than that of Bitcoin at the block rate of *4* blocks per minute, as shown in Figure 9.b.

C. Impact of mining power distribution

FastChain is still effective when miners possess different mining power. In previous sections, we assume all the nodes have the same computational power to mine a block. However, different Bitcoin miners have different computational power [13]. Two cases are shown in Figure 10. The first case has the mining power that is linearly proportional to the node bandwidth. Nodes with higher bandwidth have more power to mine a block. The second case has the mining power that is inversely proportional to the node bandwidth. FastChain improves the effective block rate of two cases as much as *25%* and *40%* respectively. The higher improvement in Figure 10.b is caused by the fact that bandwidth-limited nodes are now more likely to generate blocks. Congesting these nodes in the random selection policy would result in more forked blocks.

D. Impact of link latencies

FastChain is more effective when the link latency is shorter. Experiments in Figure 7-10 assume the average link latency of *100ms*. We now increase the average link latency to *500ms*. Figure 11 shows that the effective block rate of FastChain is only *15%* higher than that of Bitcoin. The block propagation time depends on both the link latency and the node bandwidth. When the link latency is higher, improvement is not significant. The advantage of FastChain increases as link latency decreases.

VII. Conclusion

We analyze a fundamental trade-off between the throughput and the forking rate of the blockchain system. We observe that reducing the block propagation time improves the scalability of the blockchain without increasing the forking rate. We propose FastChain to scale the effective block rate of blockchain systems. FastChain reduces the block propagation time through its informed neighbor selection policy. Miners periodically refresh their connections and disconnect from bandwidth-limited neighbors. We evaluate the performance of FastChain through extensive simulations. FastChain increases effective block rate by up to *40%*. Network topologies with shorter average link latency can benefit significantly from FastChain. FastChain is effective when nodes have different mining powers. When the mining power of a node is inversely proportional to the node bandwidth, the advantage of FastChain is the largest.

References

[1] https://www.ethereum.org

[2] Mettler, M., 2016, September. Blockchain technology in healthcare: The revolution starts here. In e-Health Networking, Applications and Services (Healthcom), 2016 IEEE 18th International Conference on (pp. 1-3). IEEE.

[3] Su, S., Wang, K. and Kim, H.S., 2018, July. SmartSupply: Smart contract based validation for supply chain blockchain. In *The 2018 IEEE International Conference on Blockchain* (Blockchain-2018), (2018), ISBN: 978-1-5386-7975-3; 988 - 993.

[4] Wang, K., Zhang Z., and Kim, H.S., 2018, July. ReviewChain: Smart contract based review system with multi-blockchain gateway. In *The 2018 IEEE International Conference on Blockchain* (Blockchain-2018), (2018), ISBN: 978-1-5386-7975-3; 1521 - 1526.

[5] Liang, X., Shetty, S., Tosh, D., Kamhoua, C., Kwiat, K. and Njilla, L., 2017, May. Provchain: A blockchain-based data provenance architecture in cloud environment with enhanced privacy and availability. In Proceedings of the 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (pp. 468-477). IEEE Press.

[6] Decker, C. and Wattenhofer, R., 2013, September. Information propagation in the bitcoin network. In 2013 IEEE Thirteenth International Conference on Peer-to-Peer Computing (P2P), (pp. 1-10). IEEE.

[7] Eyal, I., Gencer, A.E., Sirer, E.G. and Van Renesse, R., 2016, March. Bitcoin-NG: A Scalable Blockchain Protocol. In NSDI (pp. 45-59).

[8] Poon, J. and Dryja, T., 2016. The bitcoin lightning network: Scalable off-chain instant payments.

[9] Decker, C. and Wattenhofer, R., 2015, August. A fast and scalable payment network with bitcoin duplex micropayment channels. In Symposium on Self-Stabilizing Systems (pp. 3-18). Springer, Cham.

[10] M. Corallo. FIBRE: Fast internet Bitcoin relay engine. https://github.com/bitcoinfibre/bitcoinfibre, retrieved Apr. 2017..

[11] Back, A., Corallo, M., Dashjr, L., Friedenbach, M., Maxwell, G., Miller, A., Poelstra, A., Tim on, J., Wuille, P.: Enabling blockchain innovations with pegged sidechains. https://www.blockstream.com/sidechains.pdf.

[12] Gervais, A., Ritzdorf, H., Karame, G.O. and Capkun, S., 2015, October. Tampering with the delivery of blocks and transactions in bitcoin. In Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security (pp. 692-705). ACM.

[13] Gencer, A.E., Basu, S., Eyal, I., van Renesse, R. and Sirer, E.G., 2018. Decentralization in bitcoin and ethereum networks. arXiv preprint arXiv:1801.03998.

[14] https://github.com/bitcoin/bitcoin

[15] Heilman, E., Kendler, A., Zohar, A. and Goldberg, S., 2015, August. Eclipse Attacks on Bitcoin's Peer-to-Peer Network. In USENIX Security Symposium (pp. 129-144).

[16] Miller, A. and Jansen, R., 2015. Shadow-Bitcoin: Scalable Simulation via Direct Execution of Multi-threaded Applications. IACR Cryptology ePrint Archive, 2015, p.469.

[17] T. Neudecker, P. Andelfinger, and H. Hartenstein, "A simulation model for analysis of attacks on the bitcoin peer-to-peer network," in Proc. IFIP/IEEE Int. Symp. Integr. Netw. Manage. (IM'15), May 2015, pp. 1327–1332.

[18] https://github.com/ebfull/simbit

[19] Gervais, A., Karame, G.O., Wüst, K., Glykantzis, V., Ritzdorf, H. and Capkun, S., 2016, October. On the security and performance of proof of work blockchains. In Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security (pp. 3-16). ACM.

[20] https://www.nsnam.org

[21] Nakamoto, S., 2008. Bitcoin: A peer-to-peer electronic cash system.

[22] https://en.bitcoin.it/wiki/Protocol_documentation

[23] Green, M. and Miers, I., 2017, October. Bolt: Anonymous payment channels for decentralized currencies. In Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security (pp. 473-489). ACM.

[24] Prihodko, P., Zhigulin, S., Sahno, M., Ostrovskiy, A. and Osuntokun, O., 2016. Flare: An approach to routing in lightning network. White Paper.

[25] Apostolaki, M., Zohar, A. and Vanbever, L., 2017, May. Hijacking bitcoin: Routing attacks on cryptocurrencies. In Security and Privacy (SP), 2017 IEEE Symposium on (pp. 375-392). IEEE.