Performance and Efficiency Tradeoffs in Blockchain Overlay Networks

Parikshit Hegde
hegde@utexas.edu
The University of Texas at Austin
Austin, Texas, USA

Gustavo de Veciana
deveciana@utexas.edu
The University of Texas at Austin
Austin, Texas, USA

ABSTRACT

Underlying blockchain's scalability and performance is a Peer-to-Peer (P2P) overlay network and protocols for relaying blocks and transactions among participating nodes. In this work, we model and perform a systematic analysis of blockchain communication protocols. We begin by introducing the performance metric of interest for blockchains, the sequence of *ordered-completion-times*, which captures the progress distributed nodes are making in jointly constructing a consistent blockchain. We then study the characteristics of block relaying protocols. In particular, we show that when nodes cannot perform cut-through relaying, there is no optimal causal block relaying protocol if block relaying times are deterministic. We propose a simple age-based block relaying protocol that is provably near-optimal in terms of minimizing ordered-completion-times when the P2P overlay network is a tree. This analysis is relevant to blockchain relay networks such as Bitcoin-Fibre, Bloxroute etc., where the core part of the overlay is a tree. Finally, using the insights derived for tree overlays, we explore the interplay between transaction and block relaying and prioritization (age, size, FCFS based) on both tree and fully connected overlays via simulation. We observe that policies that relay blocks and incorporate transactions into blocks based on 'age' outperform other natural policies. We also explore a fundamental tradeoff between *mining* and *computational efficiencies* and performance (in terms of ordered completion times).

CCS CONCEPTS

• **Networks → Peer-to-peer protocols**; **Network performance modeling**; **Peer-to-peer networks**.

KEYWORDS

blockchains, overlay networks, scheduling, performance

ACM Reference Format:
Parikshit Hegde and Gustavo de Veciana. 2022. Performance and Efficiency Tradeoffs in Blockchain Overlay Networks. In *The Twenty-third International Symposium on Theory, Algorithmic Foundations, and Protocol Design for Mobile Networks and Mobile Computing (MobiHoc '22), October 17–20, 2022, Seoul, Republic of Korea.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3492866.3549730

1 INTRODUCTION

A blockchain is operated by a set of nodes on a Peer to Peer (P2P) network. Transactions are submitted by clients and are distributed across the nodes for eventual inclusion in the blockchain. Following a selection rule, a node groups a subset of transactions it knows into a block and attempts to add the block to the blockchain by a process called *mining*. Apart from the set of transactions, the block also consists of a *block-header*, which includes the block's *proof-of-work*, indices of the contained transactions, and some protocol-dependent metadata about the block's predecessors. Once mined, the block needs to be distributed to all the participating nodes so that they can add it to their copies of the blockchain. In this work, one of our primary focus is on the design and analysis of policies to distribute transactions and blocks across overlay links in order to optimize the performance of the blockchain.

When a block is mined, it might contain transactions that have already been shared with a few other nodes in the network. In order to avoid transmitting redundant information during block distribution, Bitcoin employs a protocol called *compact block relay*[1]. Here, when a mined block is being sent to a peer node, only those transactions in the block that are not already available at the peer are sent. This is as opposed to sending all of the block's contents to the receiver irrespective of the transactions that the receiver already possesses. We will refer to this latter approach as the *naive block relay*. As we will see, the performance of scheduling protocols of a blockchain are rooted in an interplay between transaction and mined block dissemination, which is unique to blockchain systems.

Our aim is to take a systematic approach to the blockchain communication problem. Hence, we start by describing high level design objectives. The following categories are not necessarily independent of each other.

Performance: It is well known that the rate at which transactions can be securely confirmed by the blockchain is proportional to the rate at which blocks can distributed in the blockchain [7, 18]. This is because blockchains confirm transactions by reaching a consensus on an ordered sequence of blocks, where the blocks are ordered by the times at which they were mined. To this end, we note that knowing block i (indexed by mining time) is most useful to a node when it already has knowledge of all blocks mined prior to i. In this paper, we propose a performance measure called *ordered completion times* (OCT) for blocks, which is the time at which the given block and all blocks published prior to it are known to all the peers. We also propose and study scheduling protocols (or, policies) for blockchains that optimize OCT, thus optimizing the rate and throughput of transaction confirmation as well.

Physical Network and Overlay Network: The physical network layer of the blockchain consists of a set of compute nodes (peers) connected by a network of physical links. The overlay-network is a layer of abstraction over the physical layer, where peers are connected via logical links. The capacities and latencies of the logical links are thus dictated by the underlying physical network. Past works have observed that a completely randomized construction of the overlay network is a cause for slow block-propagation times [6, 14], and therefore partially "structured approaches" of network design have been proposed, such as Bitcoin-Fibre, Falcon and bloXroute [2, 3, 14]. In particular, these structured overlay networks consist of a core network that has a tree structure and has resources specifically provisioned for the blockchain (see Figure 1 for a small instance). In this paper, our analysis of protocols will focus on such structured overlay networks, in particular, tree networks, however this also provides key insights for the performance of protocols on more general overlay networks.

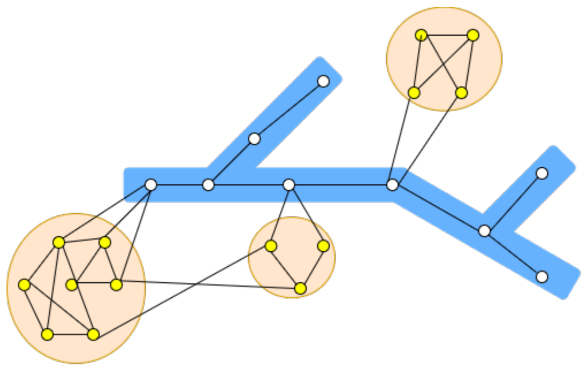


Figure 1: A small instance of a Blockchain Overlay Network, such as Bitcoin-Fibre, or Bloxroute. The core-network that consists of relay nodes with provisioned resources is shown in blue, and the nodes connecting to it are shown in yellow.

Efficiency: A couple of efficiency measures could be of interest. First, *mining efficiency*, which is the fraction of time that a node can be actively mining, impacts a miner's revenue. Second, in systems where nodes and clients have to perform certain set of operations per block, their *computational efficiency* could be affected by the number of transactions per block. These efficiency measures can be affected by the transaction arrival rate, network parameters and network policy, as we will demonstrate via simulation in Section 5.

1.1 Summary of Contributions:

A summary of the contributions of this paper is as follows:

- We begin by introducing a performance metric of interest, the sequence of *ordered-completion-times*, which captures the progress distributed nodes are making in jointly constructing a consistent blockchain. This metric captures both the rate at which transactions are confirmed in the blockchain, as well as the influence of block distribution on the security of the blockchain.
- We study the characteristics of block relaying/scheduling protocols. In particular, we show that when nodes cannot perform cut-through relaying, there is no optimal causal block relaying protocol if block relaying times are deterministic. We propose a simple age-based block relaying protocol that is provably near-optimal in terms of minimizing ordered-completion-times when the P2P overlay network is a tree. This analysis is relevant to core parts of blockchain relay networks where the overlay network is a tree.
- Driven by our theoretical results, we use simulation to explore the complex interplay between transaction and block relaying and prioritization (age, size, FCFS based). We observe that among the policies considered, the best performance is achieved by policies which prioritize block relaying based on their age (which is in line with our above analysis) and distribute unmined transactions either according to their size and/or their age.
- We explore the performance of blockchain systems in a natural regime where the block mining rate grows proportionally with the transaction arrival rate and show that it has a U-shape where it goes from being limited by mining delays to limited by network congestion. This tradeoff supports the existence of a natural sweet spot if one wishes to achieve both mining and computational efficiency, and performance. Moreover, in practice when network parameters are unknown, one might use sensitivity analysis and other optimization techniques to find this sweet spot.

1.2 Paper Organization

In Section 1.3, we review previous works that analyze blockchain network protocols, and works that use a simple network abstraction as a basis for blockchain analysis. In Section 2, we introduce our performance metric, OCT, and explain its significance. In Section 3, we describe our network model. In Section 4, we do a performance analysis in the case that the network is a tree, and show that although optimal policies are not possible with respect to minimizing OCT, a simple age-based policy is near optimal. Finally, in Section 5, we present simulation results on tree and fully connected networks and describe the tradeoff between performance and efficiency metrics.

1.3 Related Work

Most papers analyzing properties of blockchains use extremely simplified network models. [8, 9] use a synchronous model for the network where it is assumed that a broadcasted block is immediately received by all the nodes in the network. A popular model is the Δ-synchronous model[7, 10, 18] where the delay for all nodes to receive the broadcast of a mined block is at most Δ, where Δ is a deterministic parameter. A recent work [20] models Δ as a random variable which is i.i.d., across blocks. All these models fail to explore the underlying communication and queueing policies.

Two desirable properties for a Blockchain are *persistence* and *liveness* [9]. Persistence states that once a transaction is confirmed in a blockchain, it will stay confirmed forever with high probability. Liveness states that once a transaction is submitted to be included in the blockchain, it will be confirmed within a certain period of time with high probability. [7, 9, 10, 18, 20] all establish these two properties of the blockchain under the simple network models described above. For instance, in the Δ-synchronous network model, they conclude that the blockchain has persistence and liveness as long as the adversary controls fewer than a $1/(1+\Delta\lambda)$ fraction of the nodes, where λ is the rate at which blocks are mined. We believe that our proposed performance metric, ordered-completion-times, is the right metric to use when studying persistence and liveness properties of blockchains under a more comprehensive network model.

Works that consider more complicated network models are mostly restricted to the empirical domain. [6] performed an empirical study of the Bitcoin network and identified that network delay is a primary bottleneck for scaling blockchains. [5] perform a study of scalability issues in Bitcoin, in which they empirically observe the relationship between block size, network parameters and block propagation delay.

Some recent works model and analyze network dynamics of blockchains as well. [15, 16] study the queueing dynamics of transaction arrivals and block mining, however their model doesn't capture the queueing dynamics of transaction and block propagation. [11] study a model capturing general overlay networks. However, their work is restricted to studying the stability of the system, and not performance measures such as block propagation delay or ordered-completion-times.

Lastly, we note that there are several recent works that modify the underlying blockchain protocol to enable a high throughput of blocks and transactions [19, 21, 22]. The key idea in these works is to maintain a directed-acyclic-graph (DAG) of blocks rather than a chain. We stress that our analysis of network communication protocols is relevant to such DAG-based protocols as well.

Notation. $\mathbb{N}$ denotes the set of natural numbers. $(x_1, x_2, \dots)$ represents a sequence (ordered-set) of elements. $\{x_1, x_2 \dots\}$ represents an unordered set of elements. $[m, n]$ represents the set $\{m, m+1 \dots, n\}$, and $[k]$ represents the set $[1, k]$. For two sequences $\boldsymbol{x} = (x_i)_i, \boldsymbol{y} = (y_i)_i$, $\boldsymbol{x} \le \boldsymbol{y}$ if and only if $x_i \le y_i$ for every i.

2 PERFORMANCE IN BLOCKCHAIN NETWORKS

A blockchain is a directed acyclic graph with blocks as vertices such that each block has exactly one outgoing edge. Such a graph has a root-vertex, called the *genesis-block* in blockchains, such that all other blocks in the blockchain have a path to the genesis block. A chain in the blockchain is a path from one of the leaf blocks (a block with no incoming edges) to the genesis block. The *main chain* is the chain of longest length, with ties broken uniformly. Refer to Figure 2 for a small instance of a blockchain.

Blocks are added to the blockchain by network nodes through a process called *mining*. *Honest nodes* mine their blocks at the tip of their main chain, and publish their blocks soon after they are mined. *Adversarial nodes* could mine blocks at any point in the blockchain, and could wait an arbitrary amount of time before publishing their blocks. To simplify notation in the following sections, we will assume that blocks are indexed according to the order in which they are published (and not the order in which they are mined, which could be different due to adversarial behavior). When a node hears of a published block, it adds the block along with its outgoing edge to its copy of the blockchain.

Our proposed performance metric can be understood by the blockchain instance shown in Figure 2. Because of the sequential ordering of blocks in the blockchain, Block 3 can only be verified after both Blocks 1 and 2 are verified. At a transaction level, Transaction E can only be verified after transactions A, B, D, C and F, and the block-headers of blocks 1, 2 and 3 are also received and verified. Therefore, in terms of transaction confirmation, we note that receiving Block 3 is not useful to a peer until it has also received all the contents of Blocks 1 and 2. Hence, we define the ordered-completion-time for Block 3 as the earliest time at which all Blocks 1, 2 and 3 are received. Similarly, ordered-completion-time for Block 2 is the earliest time at which Blocks 1 and 2 are received.

Next, we focus on the features of a network performance metric required for a theoretical security analysis of blockchains. Most works [7–10, 18] that do a security analysis of the blockchain use a simplified network model, such as the Δ-synchronous model, which doesn't capture the dynamics of different blocks being distributed at different rates, and different network nodes receiving blocks at different rates. However, we observe that all these works intrinsically use the following general network metric. At any time t, they compute a previous time, $t - \Delta(t)$, such that all blocks that were published by time $t - \Delta(t)$ have been received by all network nodes by time t. For the simple Δ-synchronous network model, we would have $\Delta(t) = \Delta$, for all time t.

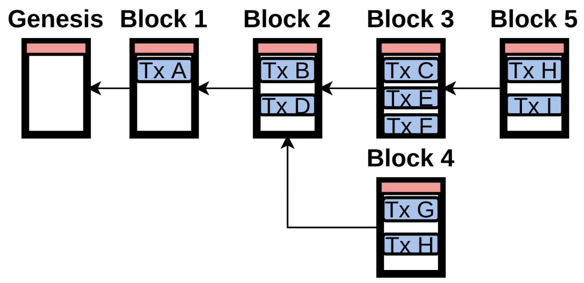


Figure 2: A small instance of a blockchain. The contents of the blocks are shown, where the red portion of the block is the block-header containing some metadata. The transactions indices are not "in-order" to indicate that the order in which transactions arrive to the system may be different from the order in which they are incorporated in to the blockchain.

In our work, we propose a network performance metric that captures both the practical purpose of transaction confirmation time, as well as the purpose of aiding a theoretical security analysis. To that end, let $\boldsymbol{A} = (\boldsymbol{a}_i)_{i=1}^{\infty}$ denote the *block arrival sequence*, where $\boldsymbol{a}_i = (n_i, t_i, m_i)$ is the mining node, time of publishing and block-size respectively of block i. Here, we note that although the blocks are numbered in sequence, this does not imply that they form a single chain in the blockchain. That is, they could be on different forks in the blockchain (as illustrated in Figure 2). However, our metric will be oblivious to the structure of forking in the blockchain, and will only depend on the block arrival sequence because, from a security point of view, it is still important to distribute blocks which are on short forks in the blockchain [7, 18].

Let $\boldsymbol{R} = (r_i)_{i=1}^{\infty}$ denote the *block receipt seqeunce*, where r_i is the time at which block i was received by all network nodes. Clearly, the sequence $\boldsymbol{R}$ is a function of the block arrival sequence $\boldsymbol{A}$, and a network communication policy, denoted π (network policies are discussed in detail in the following section). We define the ordered-completion-time for block i, denoted Θ_i, as the time by which all blocks up to index i have been received by all network nodes.

Definition 2.1 (Ordered-Completion-Time). *Θ_i is the earliest time at which all the networks nodes have received blocks $1, \ldots, i$, for some arrival sequence $\boldsymbol{A} = (\boldsymbol{a}_{i'})_{i'=1}^{\infty}$ and network policy π,:*

$$\Theta_i(\pi, \boldsymbol{A}) = \max_{i' \in [1,i]} r_{i'}(\pi, \boldsymbol{A}).$$

The sequence of Θ_i's is represented as $\Theta(\pi, \boldsymbol{A}) = (\Theta_i(\pi, \boldsymbol{A}))_{i=1}^{\infty}$.

We are interested in designing network policies that minimize Θ_i, for each block i. To do so, we first describe the communication model.

3 NETWORK MODEL

We consider a blockchain network with N peers (or, nodes) inter-connected via an overlay graph $G = (N, E)$, where E denotes a set of directed edges, which we also refer to as *links*. Links are bidirectional, meaning that if $(u, v) \in E$, then $(v, u) \in E$. We will use the terms graph and network interchangeably. Without loss of generality, the nodes are labelled from 1 to N.

For a node u in G, let $P(u)$ denote the set of neighbours of u, i.e., $P(u) = \{v : (v, u) \in E\}$. Each edge $(u, v) \in E$ in the overlay network represents an overlay link (u, v) transmitting data from node u to node v. The graph G is said to be a tree if there is a single path between any two nodes u, v with a unique sequence of links.

Developing a model and analysis for a general overlay network G is very difficult. In fact, a far simpler problem of characterizing the stability region of communication protocols on general overlay-networks G is still an open problem [11]. Therefore, we will focus our analysis on the case where G is a tree. Moreover, this assumption is relevant for structured networks such as in Bitcoin-Fibre, Falcon[2, 3] etc., where the core network is a tree.

Assumption 1. *The capacity of a link (u, v) is denoted as $c(u, v)$. A link has capacity c means that it takes m/c seconds to serve a file/packet of size m bits. Moreover, the capacity of each link is assumed to be decoupled from the traffic in the rest of the network.*

The above assumption provides a natural but simplified model for overlay link capacities in an overlay network built on top of possibly shared physical resources. Block/transaction relaying over such links is mediated via TCP sessions whose long term throughput is known to depend on round-trip-times (geography) and packet loss (underlay network congestion), see e.g.. [17]. Thus, roughly we assume that the TCP throughput among nodes is stable and representative of the achievable rate in transferring data amongst the overlay nodes involved. We are also further assuming that the capacities of overlay links are decoupled, even if they share physical links, and/or if the links share the same network interface, i.e., they are not limited by a node's protocol processing power at the network interface. In the core network infrastructure of structured networks such as Bitcoin-Fibre and Falcon, resources are provisioned, and therefore, the blockchain VPN would enjoy a degree of isolation from other traffic on the network, making Assumption 1 reasonable for such networks.

In the next two paragraphs, we introduce notation to track the distribution of blocks and transactions in the graph. A directed subgraph of the graph G, denoted by its set of links T, is a directed graph such that if a directed edge (v_1, v_2) is in T, then (v_1, v_2) is in G. Links in T do not have to be bidirectional. T is called an *arborescence* on G if it has an associated node called the *root*, denoted $\mathrm{root}(T)$, such that there is a unique directed path from $\mathrm{root}(T)$ to every other node v in G. Notice that if a block i was mined by node u, then there exists an arborescence T with root u such that the block i was transmitted on every link $(v_1, v_2) \in T$. Let $\mathcal{T}_G$ be the collection of all arborescences in G. Since the graph G is assumed to be a tree, there is a single arborescence associated with each root node u. In this case, we denote by $T(u)$ the arborescence with root u.

Define $\mathrm{path}_T(v)$ as the unique path in the arborescence T from its root node, $\mathrm{root}(T)$, to node v.

Definition 3.1 (Diameter). *The diameter of the weighted graph G, denoted* $\mathrm{diam}(G)$ *is defined as,*

$$\mathrm{diam}(G) = \max_{T \in \mathcal{T}_G} \max_{v \neq \mathrm{root}(T)} \sum_{(v_1, v_2) \in \mathrm{path}_T(v)} \frac{1}{c(v_1, v_2)}. \tag{1}$$

Assuming there is no congestion (queuing) in the overlay network, the diameter $\mathrm{diam}(G)$ is an upper bound on the time it takes for a block of unit size mined at any node in the network to be distributed to all other nodes in the graph.

We further introduce an alternate weight for a tree graph G called the critical-synchronized-delay. This is best defined in a recursive manner.

Definition 3.2 (Critical-Synchronization-Delay). *Let (u, v) be a link on the graph G. Then, define its synchronization delay as,*

$$\mathrm{sd}(u, v) = \begin{cases} 0 & , \text{ if } P(u) = \{v\}, \\ \max_{w \in P(u) \setminus \{v\}} \mathrm{sd}(w, u) & \\ \quad + \max_{x \in P(u) \setminus \{v\}} \frac{1}{c(x,u)} & , \text{ otherwise.} \end{cases} \tag{2}$$

Then the critical-synchronization-delay of the graph, denoted $\mathrm{csd}(G)$, *is defined as:*

$$\mathrm{csd}(G) = \max_{(u,v) \in E} \mathrm{sd}(u, v). \tag{3}$$

For an intuitive description of the critical-synchronization-delay, consult Figure 3. For a given node, its children are the nodes who have links into the node in the given tree (for instance, x_1's children are y_1 and y_2, and x_1 is their parent). Consider the following computational scenario. All the leaf nodes are said to be ready at time 0. Every other node needs to begin and finish a computation, following which it is said to be ready. A node can begin its computation when all its children nodes are ready. And, the time that a node takes to finish the computation is equal to the maximum of the inverse of the capacities of links from its children to it. Then, the synchronized-delay of link (u, v) is the time at which node u is ready in the tree shown in Figure 3. Following these rules, notice that nodes x_1 and x_2 are ready at times 1 and 10 respectively. Therefore, node u can begin its computation at time equal to 10, and takes $\max\{10,1\}$ time to finish computation. So, node u is ready at time 20: $\mathrm{sd}(u,v) = 20$.

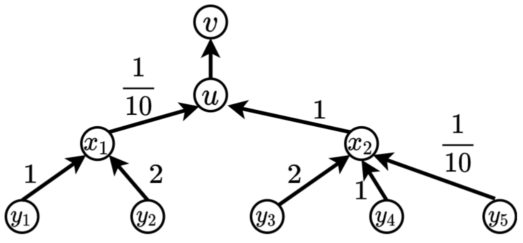


Figure 3: A subset of a graph G depicting links that lead up to link (u,v). The link capacities are shown next to the corresponding links.

In section 4, we show that $\mathrm{diam}(G)$ and $\mathrm{csd}(G)$ appear as additive gaps of our proposed algorithm from the optimal in the *non-fluid* regime, where nodes have to receive the whole block before they can begin relaying it to their neighbours.

As noted in the introduction, a blockchain communication protocol could be running in two modes of operation: naive, or compact block relay. Our analysis is restricted to naive block relay in which all the block's contents are relayed on links even if the receiver already possesses some of the block's transactions. Therefore, when analyzing naive block relay we can ignore transaction-distribution and solely focus on block distribution. To this end, we only introduce notation for block-distribution in this section. We will introduce notation for transaction distribution in Section 5 when we discuss compact block relay.

Recall that the mining process is represented by the block arrival sequence $\boldsymbol{A} = (\boldsymbol{a}_i)_{i=1}^{\infty}$, where $\boldsymbol{a}_i = (n_i, t_i, m_i)$ is the i-th block arrival. n_i, t_i and m_i represent the node that mined the block, arrival-time and the block-size respectively for the i-th block. For analysis purposes, it will be useful to truncate $\boldsymbol{A}$ to the first k blocks, which we will denote as, $\boldsymbol{A}|_k = (\boldsymbol{a}_i)_{i=1}^{k}$.

For the link (u,v), which is the link transmitting blocks from peer u to peer v, there exists a corresponding *link-policy* $\pi(u,v)$ that decides which of the available blocks at peer u to transmit to v. The link-policy is *local* which means that it is a causal function of the state of the peers u and v, and not of the state of any other peers in the system. The link-policy can work in a *preemptive-resume* manner, which means that it can interrupt the service of a block in order to serve another block. It can later resume the service of the former block starting from where it was interrupted. The ensemble of all link-policies in the network is represented as: $\pi = \{\pi(u,v) : (u,v) \in E\}$. We will refer to π simply as the *policy*.

We consider two modes for operation: the fluid-regime and the non-fluid-regime. In the *fluid regime*, a node can start forwarding (fluid) bits of a block before it has received the whole block from its neighbours. That is, the nodes can employ cut-through relaying to speed up communication [12]. In the *non-fluid regime*, a node can only start forwarding a block after it has received the entire block. In the context of blockchain, the latter is more realistic since a block is typically "verified" before it is forwarded.

A link-policy is said to be *work-conserving* if it is serving a block whenever there is a block available to serve(in the fluid regime, when there is a bit of a block to serve).

Let $b_i^u(\pi, \boldsymbol{A})$ denote the time at which peer u receives the last bit of the i^{th} block (denoted as block i) in the arrival sequence $\boldsymbol{A}$ under policy π. If u is the node that mined block i (that is, $n_i = u$), then $b_i^u(\pi, \boldsymbol{A}) = t_i$, and the peer forwards the block to all its neighbours. If u is not the miner of the block, then it received it from one of its neighbours, and it needs to forward it to each of its other neighbours in the tree. Then, the block receipt, r_i time of block i (first defined in Section 2), can be expressed as,

$$r_i(\pi, \boldsymbol{A}) = \max_{u \in [1,N]} b_i^u(\pi, \boldsymbol{A})$$

4 PERFORMANCE ANALYSIS

In this section, our goal is to characterize the performance of certain protocols π on the blockchain system under naive block relay, assuming that the overlay network G is a tree.

We separate our analysis into two main cases: the non-fluid regime and the fluid regime. We first discuss the non-fluid regime, and show later that the results for the fluid regime can be obtained as a corollary to the non-fluid regime results.

4.1 Non-Fluid Regime

We first show that, in the non-fluid regime, there exists no causal and decentralized policy π that is optimal in terms of minimizing $\Theta_i(\pi, \boldsymbol{A})$ for every block i, and every arrival sequence $\boldsymbol{A}$. However, later in this section, we propose a simple policy called π_{np}^* that achieves near-optimality guarantees with respect to minimizing ordered-completion-times metric under any arrival sequence $\boldsymbol{A}$. In fact, the near-optimality guarantee holds even when comparing to non-causal and centralized policies.

There are two factors that could lead to there being no optimal policies. First, since we require policies to be causal, uncertainty around future arrivals might lead to there being no optimal (causal) policies. In the interest of space, we skip showing this formally.

Second, since our proposed performance metric $\Theta(\cdot,\cdot)$ is a vector, the performance of two policies π_1 and π_2 may not always be comparable. That is, for some arrival sequence $\boldsymbol{A}$, it maybe that $\Theta(\pi_1, \boldsymbol{A}) \nleq \Theta(\pi_2, \boldsymbol{A})$ and $\Theta(\pi_2, \boldsymbol{A}) \nleq \Theta(\pi_1, \boldsymbol{A})$. Moreover, for a given arrival sequence $\boldsymbol{A}$, there might not exist a (possibly non-causal) policy π^* such that $\Theta(\pi^*, \boldsymbol{A}) \leq \Theta(\pi, \boldsymbol{A})$, for any other policy π. This is illustrated formally in Lemma 4.1. The crux of the proof is in the following situation. Suppose that a link was midway through serving a block when it received an older block (a lower-indexed block) from another node. Here, it faces the dilemma of whether to preempt the current service and serve the older block, or to complete the current service before serving the older block. In the former case, it might take a longer time for the next link in the path to begin service, since it cannot start service until it has received a full block. In the latter case, an older block is having to wait for a newer block to complete service, which could hurt the metric $\Theta(\pi, \boldsymbol{A})$, since the metric prioritizes the completion of older blocks. This phenomenon is illustrated in detail in the proof of the following Lemma, which can be found in Appendix A

Lemma 4.1. *Assuming G is a tree, there exists no causal policy π^* such that for every arrival sequence $A = (\boldsymbol{a}_i)_{i=1}^{\infty}$ and any other policy π,*

$$\Theta(\pi^*, \boldsymbol{A}) \leq \Theta(\pi, \boldsymbol{A}).$$

This is true even if all block sizes are equal, i.e., $m_i = m$, and all link capacities are equal, i.e., $c(u, v) = c$.

As a consequence of Lemma 4.1, we turn to constructing near-optimal policies. The ordered-completion-times metric $\Theta(\pi, \boldsymbol{A})$ suggests that blocks should be distributed in the order of their age. Therefore, it seems that a good policy π should prioritize serving the oldest available blocks at every link. However, from Lemma 4.1, it is unclear whether a policy should preempt the service of a newer block in order to serve an older block. Also, if there are two blocks one of which is much smaller in size than the other, it is unclear if one should disregard the ages of the block and serve the smaller sized block first.

In this work, we propose a very simple policy called π^*_{np} which non-preemptively serves the oldest block at each link. Despite the many complicated scenarios that could arise as described above, we prove that this simple policy achieves near-optimality guarantees on tree overlay-networks on any arrival sequence $\boldsymbol{A}$. The policy is defined formally below.

Definition 4.2. *The policy $\pi^*_{np} = \left\{\pi^*_{np}(u, v) : (u, v) \in E\right\}$ is such that for every link (u, v) and at every time t,*

- *if $\pi^*_{np}(u, v)$ was serving a certain block i at t^- and its service was not completed, then it continues serving i at time t,*
- *otherwise, if there are available blocks, it starts serving the oldest available block at that link (that is the block with the smallest t_i).*

The *np* in π^*_{np} stands for non-preemptive since the policy doesn't preempt the service of a block to serve another. Observe that it is also a work-conserving policy. Also, it is a decentralized policy since at each link the policy only depends on the state of the nodes connected to that link and not on the global state of the system. Below, we characterize the performance of π^*_{np}.

The near-optimality gap for π^*_{np} can be split into two cases. In the first case, when all the block sizes in the arrival sequence are equal, we show that $\Theta_i(\pi^*_{np}, \boldsymbol{A})$ is larger by additive-gap at most $m_{\max} \operatorname{diam}(G)$ compared to $\Theta_i(\pi, \boldsymbol{A})$, for any arrival sequence $\boldsymbol{A}$, policy π and block index i. In the second case, when the block sizes are allowed to be different, we show that the gap is at most $m_{\max}(\operatorname{diam}(G) + \operatorname{csd}(G))$.

Note that the near-optimality of π^*_{np} holds even when the policies π we compare to are non-causal and know the global state of the system (centralized). The following theorem states it formally.

Theorem 4.3. *Assume that G is a tree. Let $\boldsymbol{A} = (\boldsymbol{a}_i)_{i=1}^{\infty}$ be an arrival sequence with the maximum block size being $m_{\max}$.*

If all the block sizes in $\boldsymbol{A}$ are equal to $m_{\max}$, then for all $k \in \mathbb{N}$ and all (possibly non-causal and centralized) policies π,

$$\Theta_k(\pi^*_{np}, \boldsymbol{A}) \leq \Theta_k(\pi, \boldsymbol{A}) + \operatorname{diam}(G_S) m_{\max}. \quad (4)$$

If the block sizes are different, then for all $k \in \mathbb{N}$ and all (possibly non-causal and centralized) policies π,

$$\Theta_k(\pi^*_{np}, \boldsymbol{A}) \leq \Theta_k(\pi, \boldsymbol{A}) + \Big(\operatorname{csd}(G) + \operatorname{diam}(G)\Big) m_{\max}. \quad (5)$$

The proof of the Theorem can be found in Appendix B.

4.2 Fluid Regime

In the fluid-regime, links can start serving bits of a block before they have received the whole block. We will consider the bits to be infinitesimally small, thus mimicking a fluid. Denote by π_{fl} the policy that, at every link and at every time, serves a bit from the oldest available block (which is probably only partially received) at that node. The following theorem says that π_{fl} is optimal in terms of the ordered-completion-time for any arrival sequence.

Theorem 4.4. *Assuming that G is a tree, for any arrival sequence $\boldsymbol{A}$,*

$$\Theta(\pi_{fl}, \boldsymbol{A}) \leq \Theta(\pi, \boldsymbol{A}),$$

for any other policy π.

Proof Sketch. The fluid-regime can be considered to be equivalent to the non-fluid regime where $m_{\max} \to 0$. Therefore, substituting $m_{\max} = 0$ in Theorem 4.3 implies this result. □

So far, we have analyzed the performance of blockchain systems on tree-overlays while only considering the distribution of blocks (instead of both transactions and blocks). We concluded that although no optimal policies exist, a simple policy that relays the oldest block non-preemptively achieves near-optimal performance. In the following section, we carry this insight into a more realistic scenario where we consider both tree and complete-overlays, as well as the distribution of both transactions and blocks.

5 SIMULATION

In this section, we further explore factors that impact blockchain system performance-efficiency via simulation. We start by briefly describing our simulation model. For simplicity we consider two classes of overlay network G: linear and fully connected networks. Each node has transactions arriving as a Poisson process of rate λ_t. Upon arrival, transactions are disseminated through neighboring nodes and become candidate transactions for inclusion in mined blocks. Transaction sizes are uniformly distributed between 0.5 and 1.5 units. A node stores all its known transactions that have not yet been included in the blockchain in its *transaction-mempool*, or simply mempool. When a node becomes aware that a certain transaction in its mempool has been mined into a block (possibly at a different node) it removes it from its mempool.

When a node has received all the completed blocks in the blockchain, it groups at most $m_{\max}$ transactions from its mempool according to a mining strategy (elaborated later) and tries to mine them into a block. The block mining process at each node occurs at rate λ_b. A block consists of the block-header, and the list of transactions included in the block. Usually, the block header is much smaller compared to the rest of the block. Therefore, for simplicity, we assume that the block-header size is 0.

If a node successfully mines a block, say block i, it immediately announces the corresponding block header to all the other nodes in the network. Since a blockchain is a sequence of blocks containing a consistent order of transactions, nodes need to receive and verify the contents of block i before they can append their blocks on top of block i. Therefore, the other nodes temporarily pause mining until they receive the complete block (i.e., all its associated transactions) and verify it. Once the block is received and verified, a node can resume mining. In practice, nodes may start mining blocks with 0 transactions in them while they wait to receive other completed blocks, instead of pausing their mining. However, since the size of such blocks will be negligible (as they only consist of block headers), we shall simply ignore such blocks.

When a node u receives a transaction tx (either as a new arrival or forwarded by another node), it announces the availability of tx to all its other neighbours through an *inventory message*. Those neighbours that have not yet received tx request it from node u by sending a *getdata message*. Upon receiving a getdata message from a neighbour v, node u adds the transaction to the corresponding link (u, v)'s queue. If node v receives tx from a different neighbour before node u is able to send it, then node v notifies it, and u removes the transaction from link (u, v)'s queue. In our simulation, we have set the sizes of inventory messages and getdata messages to 0. We shall also have Assumption 1 in place, that is each link (u, v) in G serves data at a rate $c(u, v)$.

We shall simulate systems which are stable, meaning that the mempool and the link-queue sizes should not diverge to infinity. For the mempools to be stable, transactions should be incorporated into blocks at a rate *at least* as high as the rate of transaction arrivals. That is, we need $m_{\max}\lambda_b > \lambda_t$. Since a node's mining process could be paused part of the time (while it waits to receive previously mined blocks) and all the blocks may not be full, in our simulations we set $m_{\max}\lambda_b \geq 3\lambda_t$.

Now, we comment on the stability of the links. On tree networks, we note that there is exactly one set of paths for a transaction originating at a particular node to take. Therefore, denoting $H(u, v)$ as the number of nodes whose transaction need to be served on link (u, v),

$$H(u,v) = \Big|\{x : x \in [N] \text{ and } (u,v) \in T(x)\}\Big|. \tag{6}$$

And, if we assume that the rate of transaction arrivals at all the nodes is equal to λ_t, then, for a tree-network, all the links are stable if for all links (u, v), $c(u, v) > \lambda_t H(u, v)$.

Stability of link-queues in fully connected networks is non-trivial [11]. However, here, we use a heuristic to estimate an upper bound as follows. Each arriving transaction needs to be communicated a total of $N - 1$ times in order to be distributed to all the nodes. Since the transaction arrival rate at each node is λ_t, the total communication rate demanded by the transactions per node is $\lambda_t(N - 1)$. We estimate the system capacity to be the minimum over the nodes of the sum of link capacities out of each node. An upper bound on the transaction arrival rate is then calculated as $(N - 1)\lambda_t <$ system capacity.

Different communication protocols are possible based on the order in which transactions in the link queues are served. Since our performance metric, ordered-completion-times, naturally prioritizes the distribution of blocks, transactions in the link-queue that have already been mined into a block are marked as mined-transactions. In all the protocols we consider (except FCFS), mined-transactions are served before unmined transactions. We consider the following communication protocols (which are all non-preemptive) and mining strategies,

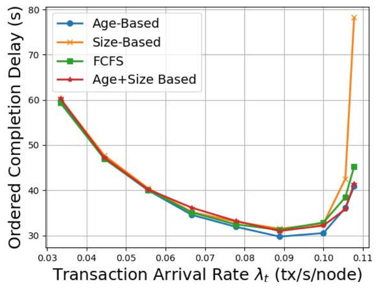
(1) Age − Based: The age of a transaction is the time since the transaction arrived to the system. If there are any mined-transactions in the link-queue, serve the oldest of those transactions. Otherwise, serve the oldest unmined transaction. During mining, sort the transactions in the mempool in descending order of their age, and group the first $m_{\max}$ transactions into a candidate block (if there are fewer than $m_{\max}$ transactions in the mempool, group all of them into the block).
(2) FCFS: Serve transactions on a First Come First Serve (FCFS) basis at all link queues (without regard to whether a transaction is mined or unmined). Incorporate transactions into blocks on a FCFS basis as well.
(3) Size − Based: Serve the smallest-sized transaction among the mined transactions first. Otherwise, serve the smallest-sized transaction amongst the unmined transactions. During mining, sort the transactions in the mempool in ascending order of their sizes and group the first $m_{\max}$ transactions into a candidate block.
(4) Age + Size − Based: If there are mined transactions to be sent, serve the smallest transaction from the oldest available block. Otherwise, send the smallest unmined transaction. Mining strategy is the same as for the Size − Based policy.

The Age − Based and Age + Size − Based policies are motivated by our analysis of the π^*_{np} policy in Section 4 which suggests that serving mined transactions based on the age of their corresponding blocks is near optimal. The Size − Based policy prioritizes the smallest mined transactions on a link, and thus takes advantage that smaller transactions would propagate faster through the network.

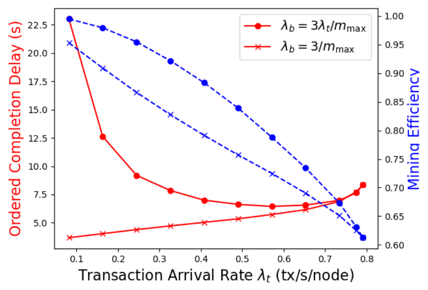
When it comes to unmined transactions, apart from ensuring a quick distribution of transactions in the network, it might be beneficial to prioritize transactions that have already been distributed in the rest of the network. This is so that when such transactions do get mined into a block, the rest of the nodes do not have to pause for long to receive and verify them. Since we need communication policies to be decentralized, we use the following heuristics in designing the policies. First, the Age − Based policy also prioritizes the dissemination of unmined transactions by their age, with the idea that older transactions are more likely to be already distributed in the rest of the network. Second, the Size − Based and Age + Size − Based policies prioritize the smallest unmined transactions on a link since they propagate faster through links. We use the simple FCFS policy as a natural and simple baseline.

Additionally under all the above policies, we ensure that the mining strategy of incorporating transactions into blocks is consistent with the communication policy used to relay transactions. As one might expect, we found that if the two policies were not consistent, it led to bad performance (in the interest of space, we skip showing these results).

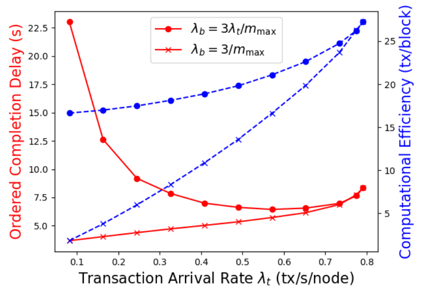
In our simulations performance comparisons for various policies are based on the average *ordered-completion-delay* of transactions. The ordered-completion-delay for a transaction is the difference of its arrival time and the ordered completion time of the corresponding block that it is eventually mined into. The average ordered



(a) Plot depicting ordered-completion-delay for transactions versus transaction arrival rate for different policies. Linear Network with N=10 nodes. All link capacities are equal to 1. $\lambda_b = 3\lambda_t/m_{\max}$**, and** $m_{\max} = 50$**.**



(b) Plot depicting tradeoff between ordered-completion-delay (solid lines) and mining efficiency (dashed lines). Fully connected network with N = 10 nodes. All link capacities sampled i.i.d., Unif[0.5,1.5]. The policy used is the Age − Based **policy.**



(c) Plot depicting tradeoff between ordered-completion-delay (solid lines) and computational efficiency (dashed lines). Fully connected network with N = 10 nodes. All link capacities sampled i.i.d., Unif[0.5,1.5]. The policy used is the Age − Based **policy.**

Figure 4

completion delay for a linear network with $N = 10$ nodes and all the link capacities being equal to 1 is shown in Figure 4a. We observe that for low values of transaction arrival rate, all policies have a similar performance. This is because, in this region, transactions do not build up in the link-queues, thus all the policies end up behaving similarly. However, for high transaction arrival rates, we observe that the Age − Based and Age + Size − Based policies outperform the others in terms of ordered completion delays. This is in-line with our intuition from Section 4 since these policies prioritize mined transactions according to the age of the corresponding blocks. In terms of scaling, when λ_t reaches 97% of system capacity, FCFS has 10% more delay than Age − Based policy. Whereas, on a linear network with N=20 nodes, it had 17% more delay. We speculate that the gap in performance could increase for larger networks.

In the left-axis of Figure 4b, we show the performance of Age − Based policy on a fully connected network with N=10 nodes, for different mining rates. First, when the mining rate is held fixed ($\lambda_b = 3/m_{\max}$) across the different transaction arrival rate λ_t, we notice that the ordered-completion-delays low for small λ_t (where there is no network congestion), and the delay grows with increasing λ_t because of increasing network congestion.

Second, Figure 4b also depicts the performance when we set the mining rate to be proportional ($\lambda_b = 3\lambda_t/m_{\max}$) to the transaction arrival rate λ_t. Here, we notice that the ordered-completion-delay exhibits a U-curve shape w.r.t., varying λ_t. The increase in delay for high λ_t is due to network congestion as explained above. Whereas, for small λ_t, the increase in delay is due to increase in inter-arrival time of blocks. Therefore, under this scaling of mining rates, we note that there is a sweet-spot for transaction arrival rate as the system goes from being driven block mining delays to network congestion.

Purely in terms of ordered-completion-delay, it seems that holding the mining rate fixed is a better strategy as compared to varying it linearly with transaction arrival rate. However, we note that there is a tradeoff between performance (ordered-completion-delay) and the efficiency of the system. We propose the following two measures to quantify the efficiency.

- Mining Efficiency: This is measured in terms of the fraction of time that a node's mining is paused while it waits to receive mined blocks from other nodes. The system has high mining efficiency if the fraction of a nodes' paused time is low. A low mining efficiency implies that node's mining resources are either held dormant or are wasting energy for a fraction of the time, both of which are bad for the miner.
- Computational Efficiency: This is measured in terms of the average number of transactions per mined block. A lower computational efficiency could mean that nodes and clients need to perform more computation per transaction. For instance, in certain blockchain consensus protocols such as Ouroboros Proof of Stake[13], all the nodes in the system need to perform a joint computation to decide which node gets to mine the block. Similarly, in recently popular protocols such as stateless blockchains [4], the computation required by the clients is proportional to the block mining rate. In either examples, lower number of transactions per block means more computation by nodes and clients per transaction.

In Figure 4b we show the tradeoff between performance and mining efficiency. Note that although higher mining rate (such as when we have fixed λ_b with respect to λ_t) leads to better performance, the more frequent mining of blocks means that transactions have lesser time to be distributed to all the nodes before they are mined. Therefore, we observe that nodes are paused for longer (i.e., they have lower mining efficiency) when the λ_b is higher.

Similarly, the tradeoff between performance and computational efficiency is shown in Figure 4c. Here, we note that with a higher mining rate (such as when λ_b is held fixed), the more frequent mining of blocks means that the number of transactions per blocks (computational efficiency) goes down quickly as the transaction arrival rate goes down.

In conclusion, we note that there is an inherent tradeoff between performance and efficiency in blockchain systems. This means that, based on the relative importance of performance and efficiency, there exists a sweet spot in terms of choosing the transaction arrival rate and block mining rate. In systems where transaction arrival rates is dictated by external forces, the sweet spot can be expressed in terms of the link capacities that the nodes in the system need to invest in.

6 CONCLUSION

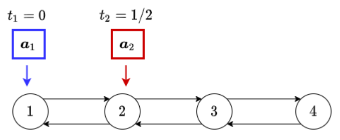
We have explored the interplay among transaction and block relaying policies and the mining strategies used to incorporate transactions into blocks and their impact on blockchain performance in terms of order-completion-delays as well as tradeoffs in efficiency. This is certainly a complex system dynamics to analyze but our theoretical results on tree overlays and simulation results on tree and fully-connected overlays provide support for the claim that consistent age-based strategies provide robust performance over a variety of networks and loads for such systems.

Appendices

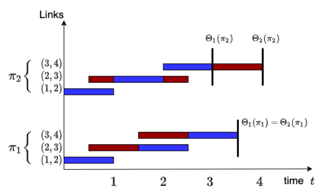
Appendix A PROOF OF LEMMA 4.1

Proof. The proof is by counter-example. Consider a linear network with $N = 4$ peers. Let the capacity at all the links be $c = 1$, and all block sizes be $m = 1$. We will only consider the first 2 block arrivals (future arrivals can be assumed to be arriving much later with respect to these two blocks). The idea is to construct an arrival sequence such that any policy that optimizes Θ_1 is non-optimal in terms of Θ_2, and vice versa. Let the two block arrivals be: $\boldsymbol{a}_1 = (1, 0, 1)$ and $\boldsymbol{a}_2 = (2, 1/2, 1)$. Refer Fig. 5(a) for a pictorial depiction of the network.

For the links $(2, 1)$ and $(1, 2)$, the link-policies are simple. $\pi(2, 1)$ needs to serve block 2 from time $1/2$ to $3/2$, and $\pi(1, 2)$ needs to serve block 1 from time 0 to 1.



(a) Linear network with N=4 and 2 block arrivals.



(b) Temporal depiction of block service at different links under policies π_1 (bottom) and π_2 (top). Block 1's service is shown in blue, and block 2's in red. Other links are not shown since they are inconsequential for this example.

Figure 5: Pictorial Illustration for Lemma 4.1.

For the link $(2, 3)$, we need to consider two link-policies, $\pi_1(2, 3)$ and $\pi_2(2, 3)$. Let $\pi_1(2, 3)$ serve block 2 from time $1/2$ to $3/2$, and serve block 1 from time $3/2$ to $5/2$. Let $\pi_2(2, 3)$ serve block 2 from $1/2$ to 1 and then from 2 to $3/2$, and let it serve block 1 from time 1 to 2. And, let $\pi_3(3, 4)$ serve the block that arrives to the link $(3, 4)$ first.

Let π_1 be the policy containing the link-policy $\pi_1(2, 3)$, and let π_2 be the policy containing the link-policy $\pi_2(2, 3)$. For any other policy π, it can be shown that either $\Theta(\pi_1, \boldsymbol{A}) \leq \Theta(\pi, \boldsymbol{A})$, or $\Theta(\pi_2, \boldsymbol{A}) \leq \Theta(\pi, \boldsymbol{A})$. Refer to Fig. 5(b) for a pictorial representation of the two policies.

The ordered completion times for the two policies can be computed to be: $\Theta(\pi_1, \boldsymbol{A}) = (7/2, 7/2)$, and $\Theta(\pi_2, \boldsymbol{A}) = (3, 4)$. Since these two sequences are not comparable, the proof by counter-example is complete. □

Appendix B PROOF OF THEOREM 4.3

Proof. In the interest of space, we will only prove (4) here. The proof of (5) is similar. Denote by $z_k^{(u,v)}(t, \pi)$ the number of the first k blocks that have been transferred on link (u, v) by policy π on arrival sequence $\boldsymbol{A}$ (for brevity, we skip including $\boldsymbol{A}$ as a parameter). Similarly, define $z_k^u(t)$ as the number of blocks among the first k blocks that are mined by node u by time t. Define, $\text{diam}(u, v) \triangleq \sum_{(v_1,v_2)\in \text{path}_{T(u)}(v)} \frac{1}{c(v_1,v_2)}$. Then, diam$(G)$, defined in Definition 3.1, can be expressed for trees as, $\text{diam}(G) = \max_{u,v} \text{diam}(u, v)$.

We prove the lemma statement by proving the following stronger statement:

$$z_k^{(u,v)}(t, \pi_{np}^*) \geq z_k^{(u,v)}(t - m_{\max} \,\text{diam}(u, v), \pi) \quad ,\forall t. \qquad (7)$$

Before proving (7), let's show how it implies (4). Since G is a tree, the number of the first k blocks of $\boldsymbol{A}$ that need to be served on link (u, v) is equal under any policy, call it $k^{(u,v)}$. Under policy π, let $t^{(u,v)}$ be the earliest time when all the corresponding $k^{(u,v)}$ blocks are transferred, i.e., $z_k^{(u,v)}(t^{(u,v)}, \pi) = k^{(u,v)}$. Then, (7) implies that,

$$z_k^{(u,v)}(t^{(u,v)} + m_{\max} \,\text{diam}(G), \pi_{np}^*) \geq k^{(u,v)}.$$

That is, π_{np}^* will complete serving the bits on link (u, v) by time at most $t^{(u,v)} + m_{\max} \,\text{diam}(u, v)$. $\Theta_k(\pi, \boldsymbol{A})$ is the maximum of $t^{(u,v)}$'s taken over all links $(u, v) \in E$. Therefore, $\Theta_k(\pi_{np}^*, \boldsymbol{A}) \leq \Theta_k(\pi, \boldsymbol{A}) + m_{\max} \,\text{diam}(G)$.

Now, we prove (7). Recall $H(u, v)$ from (6). The proof will be by induction on $H(\cdot, \cdot)$. As the base case, consider links (u, v) with $H(u, v) = 0$. In this case, the only blocks that are served on (u, v) are those mined by node u. Since π_{np}^* is a work conserving policy that serves oldest blocks first, we have $z_k^{(u,v)}(t, \pi_{np}^*) \geq z_k^{(u,v)}(t, \pi)$.

As the induction hypothesis, assume that (7) is true for all links (x, y) such that $H(x, y) \leq n$, for some integer n. If there are no links (u, v) with $H(u, v) = n + 1$, then the induction hypothesis is vacuously true for all links (x, y) with $H(x, y) \leq n + 1$ as well. Consider that there is a link (u, v) such that $H(u, v) = n+1$. Let $P(u) \setminus \{v\} = \{x_1, \ldots, x_l\}$ be all the other neighbours of u, i.e., $(x_j, u) \in E$ for all $j \in [l]$. Notice that for all these neighbouring links (x_j, u), we have $H(x_j, u) \leq n$. Therefore, from the induction hypothesis, all these neighbouring links satisfy (7). As a contradiction, assume that (7) fails at link (u,v). Let t be the earliest time that it fails:

$$z_k^{(u,v)}(t, \pi^*_{np}) < z_k^{(u,v)}(t - m_{\max}\operatorname{diam}(u,v), \pi). \tag{8}$$

Link (u,v) takes $m_{\max}/c(u,v)$ time to deliver a block. Since, t is the first time that the induction hypothesis fails, we have,

$$\begin{aligned} & z_k^{(u,v)}\left(t - \frac{m_{\max}}{c(u,v)}, \pi^*_{np}\right) \\ & \quad = z_k^{(u,v)}\left(t - \frac{m_{\max}}{c(u,v)} - m_{\max}\operatorname{diam}(u,v), \pi\right). \end{aligned} \tag{9}$$

To see why the above equation is true, consider the two alternate cases. If, it was "<" instead of "=", then t is not the earliest time that (7) fails. Instead, if it was ">", then for (8) to be true, π would have to serve more than 1 block on link (u,v) in a time interval of length $m_{\max}/c(u,v)$, which is impossible.

In other words, (8) and (9) imply that policy π delivered a block from the first k blocks on link (u,v) in the time interval $(t - m_{\max}/c(u,v) - m_{\max}\operatorname{diam}(u,v), t - m_{\max}\operatorname{diam}(u,v)]$. This implies that at time $t - m_{\max}/c(u,v) - m_{\max}\operatorname{diam}(u,v)$, node u has at least an extra block from the first k blocks than node v under policy π, which it sends over in that interval. That is,

$$\begin{aligned} & z_k^u\left(t - \frac{m_{\max}}{c(u,v)} - m_{\max}\operatorname{diam}(u,v)\right) \\ & \quad + \sum_{x\in P(u)\setminus\{v\}} z_k^{(x,u)}\left(t - \frac{m_{\max}}{c(u,v)} - m_{\max}\operatorname{diam}(u,v), \pi\right) \\ & \quad > z_k^{(u,v)}\left(t - \frac{m_{\max}}{c(u,v)} - m_{\max}\operatorname{diam}(u,v), \pi\right). \end{aligned} \tag{10}$$

On the other hand, equations (8) and (9) imply that π^*_{np} did not deliver any of the first k blocks on link (u,v) in time interval $(t - m_{\max}/c(u,v), t]$.

By the induction hypothesis at all links $(x,u) \in P(u) \setminus \{v\}$,

$$\begin{aligned} & z_k^u\left(t - \frac{2m_{\max}}{c(u,v)}\right) \\ & \quad + \sum_{x\in P(u)\setminus\{v\}} z_k^{(x,u)}\left(t - \frac{2m_{\max}}{c(u,v)}, \pi^*_{np}\right) \\ & \geq z_k^u\left(t - \frac{2m_{\max}}{c(u,v)}\right) \\ & \quad + \sum_{x\in P(u)\setminus\{v\}} z_k^{(x,u)}\left(t - m_{\max}\left(\frac{2}{c(u,v)} + \operatorname{diam}(x,u)\right), \pi\right) \\ & \overset{(a)}{\geq} z_k^u\left(t - m_{\max}\left(\operatorname{diam}(u,v) + \frac{1}{c(u,v)}\right)\right) \\ & \quad + \sum_{x\in P(u)\setminus\{v\}} z_k^{(x,u)}\left(t - m_{\max}\left(\operatorname{diam}(u,v) + \frac{1}{c(u,v)}\right), \pi\right) \\ & \overset{(b)}{>} z_k^{(u,v)}\left(t - m_{\max}\left(\frac{1}{c(u,v)} + \operatorname{diam}(u,v)\right), \pi\right), \\ & \overset{(c)}{=} z_k^{(u,v)}\left(t - \frac{m_{\max}}{c(u,v)}, \pi^*_{np}\right). \end{aligned} \tag{11}$$

(a) follows as z_k^u and $z_k^{(\cdot,u)}$ are non-decreasing with time and since $\operatorname{diam}(u,v) = 1/c(u,v) + \max_{x\in P(u)\setminus\{v\}}\operatorname{diam}(x,u)$. (b) follows from (10), and (c) follows from (9).

(11) implies that, under policy π^*_{np}, node u already has more blocks from the first k blocks to send to node v at time $t - 2m_{\max}/c(u,v)$, than has been sent by link (u,v) by time $t - m_{\max}/c(u,v)$. However, it might have to wait non-preemptively for a block with index greater than k to complete service on link (u,v). That block would take at most $m_{\max}/c(u,v)$ time to complete service. Therefore, the bits from the first k blocks can begin service by at most time $t - m_{\max}/c(u,v)$, and will complete service by at most time t. This is a contradiction to (8) and (9). Therefore, the induction hypothesis is true for link (u,v). □

REFERENCES

[1] [n. d.]. https://bitcoincore.org/en/2016/06/07/compact-blocks-faq/. Accessed: 2021-07-01.

[2] [n. d.]. https://bitcoinfibre.org/. Accessed: 2021-03-07.

[3] [n. d.]. https://www.falcon-net.org/. Accessed: 2021-03-19.

[4] Dan Boneh, Benedikt Bünz, and Ben Fisch. 2019. Batching techniques for accumulators with applications to iops and stateless blockchains. In *Annual International Cryptology Conference*. Springer, 561–586.

[5] Kyle Croman, Christian Decker, Ittay Eyal, Adem Efe Gencer, Ari Juels, Ahmed Kosba, Andrew Miller, Prateek Saxena, Elaine Shi, Emin Gün Sirer, Dawn Song, and Roger Wattenhofer. 2016. On Scaling Decentralized Blockchains. In *Financial Cryptography and Data Security*, Jeremy Clark, Sarah Meiklejohn, Peter Y.A. Ryan, Dan Wallach, Michael Brenner, and Kurt Rohloff (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 106–125.

[6] C. Decker and R. Wattenhofer. 2013. Information propagation in the Bitcoin network. In *IEEE P2P 2013 Proceedings*. 1–10. https://doi.org/10.1109/P2P.2013.6688704

[7] A. Dembo, S. Kannan, Ertem Nusret Tas, D. Tse, Pramod Viswanath, Xuechao Wang, and O. Zeitouni. 2020. Everything is a Race and Nakamoto Always Wins. *IACR Cryptol. ePrint Arch.* 2020 (2020), 601.

[8] Ittay Eyal and Emin Gün Sirer. 2014. Majority is not enough: Bitcoin mining is vulnerable. In *International conference on financial cryptography and data security*. Springer, 436–454.

[9] Juan Garay, Aggelos Kiayias, and Nikos Leonardos. 2015. The bitcoin backbone protocol: Analysis and applications. In *Annual international conference on the theory and applications of cryptographic techniques*. Springer, 281–310.

[10] Peter Gaži, Aggelos Kiayias, and Alexander Russell. 2020. Tight consistency bounds for bitcoin. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*. 819–838.

[11] Aditya Gopalan, Abishek Sankararaman, Anwar Walid, and Sriram Vishwanath. 2020. Stability and scalability of blockchain systems. *Proceedings of the ACM on Measurement and Analysis of Computing Systems* 4, 2 (2020), 1–35.

[12] Parviz Kermani and Leonard Kleinrock. 1979. Virtual cut-through: A new computer communication switching technique. *Computer Networks (1976)* 3, 4 (1979), 267–286. https://doi.org/10.1016/0376-5075(79)90032-1

[13] Aggelos Kiayias, Alexander Russell, Bernardo David, and Roman Oliynykov. 2017. Ouroboros: A provably secure proof-of-stake blockchain protocol. In *Annual International Cryptology Conference*. Springer, 357–388.

[14] Uri Klarman, Soumya Basu, Aleksandar Kuzmanovic, and Emin Gün Sirer. 2018. bloxroute: A scalable trustless blockchain distribution network whitepaper. *IEEE Internet of Things Journal* (2018).

[15] Quan-Lin Li, Jing-Yu Ma, and Yan-Xia Chang. 2018. Blockchain queue theory. In *International Conference on Computational Social Networks*. Springer, 25–40.

[16] Quan-Lin Li, Jing-Yu Ma, Yan-Xia Chang, Fan-Qi Ma, and Hai-Bo Yu. 2019. Markov processes in blockchain systems. *Computational Social Networks* 6, 1 (2019), 1–28.

[17] Jitendra Padhye, Victor Firoiu, Don Towsley, and Jim Kurose. 1998. Modeling TCP throughput: A simple model and its empirical validation. In *Proceedings of the ACM SIGCOMM'98 conference on Applications, technologies, architectures, and protocols for computer communication*. 303–314.

[18] Rafael Pass, Lior Seeman, and Abhi Shelat. 2017. Analysis of the blockchain protocol in asynchronous networks. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 643–673.

[19] Serguei Popov. 2018. The tangle. *White paper* 1, 3 (2018).

[20] Suryanarayana Sankagiri, Shreyas Gandlur, and Bruce Hajek. 2021. The Longest-Chain Protocol Under Random Delays. *arXiv preprint arXiv:2102.00973* (2021).

[21] Yonatan Sompolinsky, Yoad Lewenberg, and Aviv Zohar. 2016. SPECTRE: a fast and scalable cryptocurrency protocol. *IACR Cryptol. ePrint Arch.* 2016, 1159 (2016).

[22] Yonatan Sompolinsky and Aviv Zohar. 2018. Phantom. *IACR Cryptology ePrint Archive, Report 2018/104* (2018).