Healthor: Heterogeneity-aware Flow Control in DLTs to Increase Performance and Decentralization

JONAS THEIS, Vrije Universiteit Amsterdam, and IOTA Foundation,
LUIGI VIGNERI, IOTA Foundation,
LIN WANG, Vrije Universiteit Amsterdam,
ANIMESH TRIVEDI, Vrije Universiteit Amsterdam,

Permissionless reputation-based distributed ledger technologies (DLTs) have been proposed to overcome blockchains' shortcomings in terms of performance and scalability, and to enable feeless messages to power the machine-to-machine economy. These DLTs allow machines with widely heterogeneous capabilities to actively participate in message generation and consensus. However, the open nature of such DLTs can lead to centralization of decision-making power, thus defeating the purpose of building a decentralized network.

In this paper, we introduce Healthor, a novel heterogeneity-aware flow-control mechanism for permissionless reputation-based DLTs. Healthor formalizes node heterogeneity by defining a health value as a function of its incoming message queue occupancy. We show that health signals can be used effectively by neighboring nodes to dynamically flow control messages while maintaining high decentralization. We perform extensive simulations, and show a 23% increase in throughput, a 76% decrease in latency and four times increased node participation in consensus compared to state of the art. To the best of our knowledge, Healthor is the first system to systematically explore the ramifications of heterogeneity on DLTs and proposes a dynamic, heterogeneity-aware flow control. Healthor's source code (https://github.com/jonastheis/healthor) and simulation result data set (https://zenodo.org/record/4573698) are both publicly available.

CCS Concepts: • **Networks → Peer-to-peer protocols**.

Additional Key Words and Phrases: application-layer protocols, network architecture, flow control, blockchain, distributed ledger technologies

1 INTRODUCTION

With its inception in 2008, Bitcoin has sparked a whole new world of distributed ledger technologies (DLTs) [48, 61]. DLTs are gaining popularity ever since, especially for trustless money transactions and code execution[1] enabling the recent trend towards decentralized finance with stablecoins, decentralized exchanges, and decentralized lending [33].

Conceptually, a distributed ledger is an immutable, replicated, and shared data structure that keeps track of ledger state entries, e.g., monetary transactions, in a distributed system without the need for a centralized authority but instead utilizes a distributed consensus mechanism [22, 61]. Ledger state updates are disseminated using a peer-to-peer (P2P) network between ledger participating nodes [24, 51, 59]. Theoretically, a permissionless DLT is open for anyone to join and keep track of the ledger and participate in consensus [59].

[1]Trustless code execution is usually referred to as *smart contracts* and enables decentralized applications to be built.

Authors' addresses: Jonas Theis, Vrije Universiteit Amsterdam, and IOTA Foundation,; Luigi Vigneri, IOTA Foundation,; Lin Wang, Vrije Universiteit Amsterdam,; Animesh Trivedi, Vrije Universiteit Amsterdam,

2769-6472/2022/8-ART $15.00
https://doi.org/10.1145/3555676

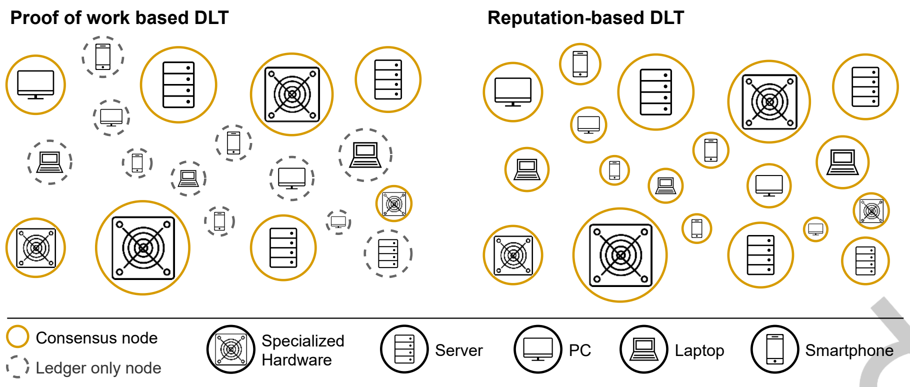


Fig. 1. Comparison of nodes participating in consensus and ledger only in proof of work based DLTs (left) and reputation-based DLTs (right). In proof of work based DLTs practically only high-end nodes and specialized hardware can participate in consensus whereas in reputation-based every node because participation in consensus is not dependent on processing capabilities. Connections between nodes are not pictured out of brevity. The size of nodes describes their processing capabilities.

However, in practice traditional DLTs present limitations in terms of not only decentralization but also scalability, performance, and energy efficiency [27, 28, 57]. Hence, a number of alternatives have been proposed in the last few years [25, 29, 67].

1.1 Heterogeneity in Permissionless DLTs

In this paper, we focus on *permissionless reputation-based* DLTs [7, 34, 50, 51, 54, 63], which achieve consensus through voting instead of expensive mining races. Reputation-based DLTs *can* enable a wide range of new application domains, such as machine-to-machine economy for the Internet of Things or public transparent supply chains [7, 50, 51], by overcoming Bitcoin's limitations: network throughput is not constrained at the protocol level, feeless messages are possible, low-power devices can participate in the consensus, etc. This class of DLTs adds design complexity to the original Bitcoin's blockchain, and faces a number of challenges which we describe and address throughout the paper.

One of the primary challenges in the Bitcoin network is the centralization of power. Miners typically use specialized hardware to compute a cryptographic puzzle faster than other nodes and add messages into the blockchain. This mechanism creates a costly filter formed by an elitist network (Figure 1 (left)) [57]. Conversely, the permissionless nature of the DLTs considered in this paper allows nodes with widely *heterogeneous* capabilities to participate in consensus and message generation (Figure 1 (right)). Such heterogeneity can be manifold:

- Bandwidth, latency, availability, and processing capabilities can vary between multiple orders of magnitude, as in traditional peer-to-peer (P2P) networks [55].
- Unsteady processing rates of a node due to competing tenant applications and performance variability, especially in the public cloud environment [23, 40, 65].
- Protocols, geographical locations, node freshness, and software versions can differ widely in DLTs [37].

1.2 Challenges

Heterogeneity is a key feature of permissionless reputation-based DLTs, but also introduces multiple challenges [7, 51]. First, *who can vote?* In order to start the voting procedure, a supermajority of nodes must have received the most recent messages necessary to construct and verify the ledger state. In these DLTs, a score (reputation) is assigned to each node to determine nodes' reserved throughput shares and weights used during voting. Many DLTs assign reputation by linking it to a constrained resource, such as stake (e.g., Proof-of-Stake (PoS) DLTs [26, 54] or IOTA's Mana [51]); more sophisticated technologies try to evaluate whether nodes are well-behaving and contributing to the security of the network [63]. A good reputation system should prevent Sybil attacks, where colluding nodes can gain disproportionate influence to manipulate the ledger state.

Second, unlike in Bitcoin, reputation-based DLTs require an explicit *distributed flow-control mechanism*. If powerful nodes issue new messages too fast without any flow control, then only high-end nodes will be able to keep up with the message processing and with the latest ledger updates. Hence, only few nodes, the ones with an updated ledger, can vote, thus increasing undesired DLT centralization.

Lastly, *maintaining maximum decentralization with high performance*. In DLTs, nodes are required to process all messages generated. Hence, to avoid any loss of synchronization or message drop, the network must operate at the speed of the slowest node, which can lead to low throughput. An additional challenge is given by the fact that the node's processing speed is not static as it varies based on the operating environment and performance fluctuations [23, 40, 65].

Though efforts have been put to tackle the issue of voting [26, 51, 54], limited attention has been paid to tackle decentralization and performance due to heterogeneity in DLTs. Such heterogeneity-related challenges are reminiscing of the early 2000s research in P2P content distribution systems [3, 13, 14, 39, 55], but the need for quality of service, and node reachability requirements sets modern DLTs apart from their P2P predecessors. We take inspiration from these works and recent networking research [47], and make a case for a *dynamic flow-control* protocol to react to the changing heterogeneity (i.e., computational capabilities) for maximizing throughput without sacrificing DLT decentralization.

1.3 Our Contributions

In this paper we present Healthor[2], a novel heterogeneity-aware, lightweight flow-control mechanism for permissionless reputation-based DLT networks. Healthor captures heterogeneity by defining a node's *health* as a function of its processing power and the current network activity. The health updates of neighboring nodes are then used to calculate the message forwarding rates, thus dynamically adjusting the flow control per node. This basic mechanism allows high-end nodes to buffer messages for unhealthy nodes, thus protecting weaker nodes from being overwhelmed with wasted processing and rapidly adapting network load and bursts. With such a flow-control design more nodes are able to keep up with the ledger updates and participate in DLT consensus, thus increasing decentralization and network performances. Unlike protocol-level solutions such as TCP, which operate independently of the applications, in this work we target an application level flow-control protocol.

Our key contributions in this work include:

- We identify challenges (high centralization, poor performance, security) due to lack of heterogeneity-aware flow control in permissionless reputation-based DLTs (Section 3).
- We propose Healthor, a lightweight distributed flow-control mechanism that leverages a node's health as a proxy of its heterogeneity and processing capabilities. We present its design choices, implementation, and trade-offs. In comparison to other DLTs, which use leader election or fixed computation, we are among the first to introduce networking concepts and optimizations to DLTs (Section 4). Healthor's source code is publicly available on GitHub (https://github.com/jonastheis/healthor).
- We evaluate Healthor in OMNeT++ simulations for up to 5,000 nodes. Our results demonstrate that Healthor increases the degree of decentralization by 78%, improves throughput by 23%, and 95 percentile message

[2]Union of the word *health* and the Germanic god *Thor* who is amongst other things associated with great strength and the protection of mankind.

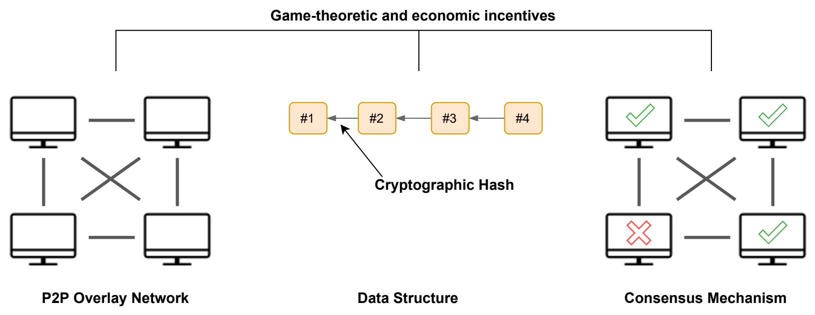


Fig. 2. Permissionless DLTs comprise three main components: a P2P overlay network (left), an immutable data structure (center), and a consensus mechanism (right) are deeply fused via game-theoretic and economic incentives.

latency by 76% while staying resilient against attacks (Section 5-6). The simulation result data set is publicly available on Zenodo (https://zenodo.org/record/4573698).

2 BACKGROUND: A SHORT INTRODUCTION TO DLTS

In 2008 the inconspicuous Bitcoin paper [48] written by the anonymous entity Satoshi Nakamoto sparked a revolution and ignited *distributed ledger technologies*. Though the underlying technologies were not novel [12, 18, 21, 32, 46], Bitcoin combined them in an ingenious way and created something that had been deemed impossible: consensus of replicated, shared, and synchronized data without a central entity in a permissionless and trustless setting where anyone can join and participate [48, 59]. Since then many flavors of DLTs have emerged, not only to enable monetary transactions but more so to enable trustless code execution and thus paving the way for many more use cases and a distributed, trustless Internet, enabling parties to interact without trusting anyone [22, 35]. DLTs broadly can be distinguished into permissioned, i.e., a central authority grants permission to participants, or permissionless, i.e., open access to anyone where participants do not know each other but cooperate through game-theoretical incentives [44]. The latter systems pose more challenges due to their open nature. In this paper, we focus on permissionless DLTs.

Generally, a DLT integrates three main components joining them together via its protocol and deeply ingrained game-theoretic and economic incentives as depicted in Figure 2. First, a P2P overlay network is utilized to disseminate state updates (Section 2.1). Second, every node keeps track of a shared, replicated, and immutable data structure (also called ledger) which is based on cryptographic primitives (Section 2.2). Lastly, nodes use a consensus mechanism to agree on a state in a distributed manner (Section 2.3).

2.1 P2P Overlay Network

Most DLTs build an unstructured decentralized P2P overlay network, either with manual peering, i.e., node owner are required to exchange connection details, or some form of automatic peer discovery and peer selection. Ledger state updates are disseminated in the form of messages (also called transactions) via epidemic broadcast so that every node eventually receives all ledger state updates [24, 51, 59].

2.2 Data Structure

The ledger state in a DLT is derived from an immutable data structure, that can be compared with an append-only log. Every node in the network has a copy of the ledger state and can thus verify the validity of new updates

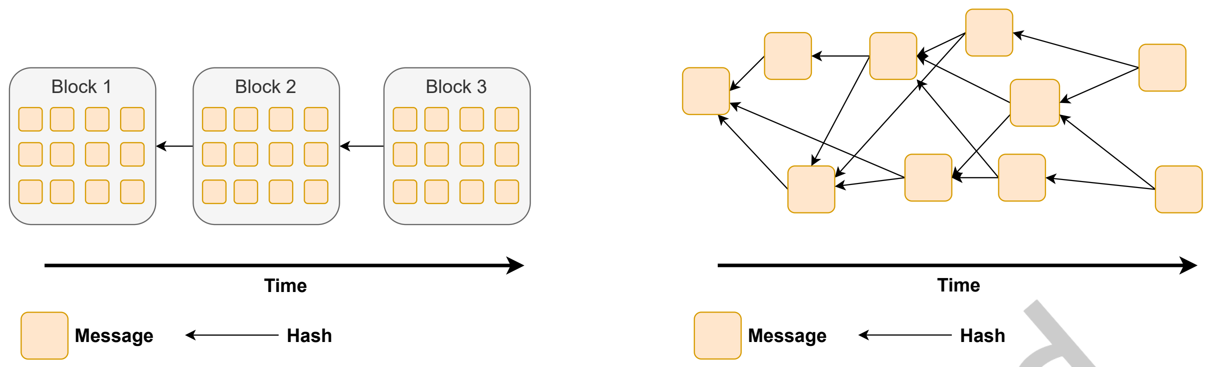


Fig. 3. Data structures in DLTs. A blockchain groups messages into blocks and links blocks together (left). The Tangle consists of messages that reference 2 other messages (right), thus forming a DAG.

locally. Newly joining nodes can bootstrap by simply downloading the ledger from their neighbors, and then make sure that the ledger state is valid locally [7, 22, 50, 59, 61].

Blockchain. A blockchain is a linked list of blocks where each new block contains a cryptographic hash of its predecessor's content as shown in Figure 3 (left). A block mainly consists of a number of state updates, e.g., in the case of Bitcoin monetary transactions. This essentially creates an immutable chain of blocks where any change to a block invalidates all future blocks as well. The longer the chain, the harder it is to change any content because all future blocks would be invalidated, hence it is tamperproof. A blockchain is totally ordered: blocks are issued, e.g., with consensus on the longest chain, at regular intervals and state updates within a block are deterministic [22, 59, 61].

Directed Acyclic Graph (DAG). A directed acyclic graph (DAG) is a graph without directed cycles, i.e., it grows in one direction. IOTA's Tangle [50] is a DAG where messages are linked together via their cryptographic hashes instead of being grouped into blocks. Figure 3 (right) shows this data structure. Similar to a blockchain, linking messages together via their cryptographic hashes makes the data structure immutable and tamperproof. The Tangle is not totally ordered as messages can be attached simultaneously by multiple users which promises better scalability compared to blockchains. However, there is added complexity for nodes to verify the ledger state and come to consensus. For example, before a message can be verified it needs to be solid, i.e., its entire history needs to be known to the node. In case a node is missing a message it can ask its neighbors via *solidification request* [22, 50].

2.3 Consensus Mechanism

The consensus mechanism is at the core of every permissionless DLT. It is a set of rules that combines the P2P overlay network, data structure as well as some form of leader election, e.g. to select a block producer, and (virtual) voting with game-theoretical incentives and bakes the results into the immutable ledger. In this way, the data structure does not only serve as an immutable ledger database but also as a verifiable instrument of consensus. It enables a Byzantine Fault Tolerant P2P network of anonymous nodes that are free to join and leave at will [10, 64].

Name	**Date**	**Size**	**Hosting**	**Top Providers**
Bitcoin (Bitnodes [66])	21/11/2020	11,122	4,195 (38%)	Tor Network: 2,827 (25%)† Hetzner: 1,049 (9%) Amazon: 803 (7%) OVH: 490 (4%) DigitalOcean: 455 (4%) Google: 360 (3%)
Bitcoin (Mariem et al. [43])	07/05/2019	9,476	6,159 (65%)	Hetzner: 1,042 (11%) Amazon: 805 (8.5%) DigitalOcean: 616 (6.5%) OVH: 550 (5.8%) Comcast: 351 (3.7%)
Ethereum (ethernodes.org [9])	21/11/2020	9,517	5,855 (62%)	Amazon: 1,778 (19%) Alibaba: 1,106 (12%) Hetzner: 541 (6%) Google: 385 (4%) DigitalOcean: 326 (3%)
Ethereum (Kim et al. [37])	08/05/2018	8,309	3,722 (44.8%)∗	Amazon Alibaba DigitalOcean OVH Hetzner Google
IOTA (thetangle.org [58])	21/11/2020	302	202 (69%)	Contabo: 66 (22%) Hetzner: 63 (21%) Netcup: 37 (12%) Amazon: 10 (3%) DigitalOcean: 6 (2%)

Table 1. DLT network size (publicly reachable) and distribution of nodes running on cloud hosting providers.
∗ Only top 8 ASes, no exact values published in [37].
† Not included in *Hosting* due to unknown service provider.

3 HETEROGENEITY IN DLTS: CHALLENGES AND OPPORTUNITIES

With the basic working model explained in Section 2 in mind, we now discuss what is the extent of the heterogeneity in contemporary DLT networks, what happens if heterogeneity is ignored, and what opportunities present if we can leverage it.

3.1 The Nature of DLT Heterogeneity

To get an estimation of the level of heterogeneity and its impact, we analyzed recently published literature on DLT deployments and decentralization [27, 37, 43]. Our analysis shows that (Table 1) a significant amount of nodes in DLT networks run on only few big cloud hosting providers. For example, 62% of publicly reachable Ethereum nodes are running on cloud hosting providers. In the IOTA network this is even more extreme with 69% of publicly reachable nodes running in the cloud.

Though, on the surface cloud hosting seems to offer a more homogeneous environment, but it is not the case in practice. First, cloud providers offer a bewildering array of choices in terms of configurations, capabilities, and cost of systems resources like virtual machines (VMs), which has lead to a series of work in workload optimizations for heterogeneous cloud resources [20, 38, 42, 52]. Such heterogeneous choices imply that there is no single ideal VM that DLTs can choose to deploy. Moreover, even with the choice of a VM there is significant performance fluctuation over time [23, 40, 60, 65]. For example, we took cloud performance traces from [23], which has collected the performance of CPU-intensive benchmarks (comparable with cryptographic signature verification widely used in DLTs [59]) over a period of 30 days from three cloud providers (AWS, Azure, and Dimension Data), and normalized the performance to the mean value in the performance. We plot the performance variation

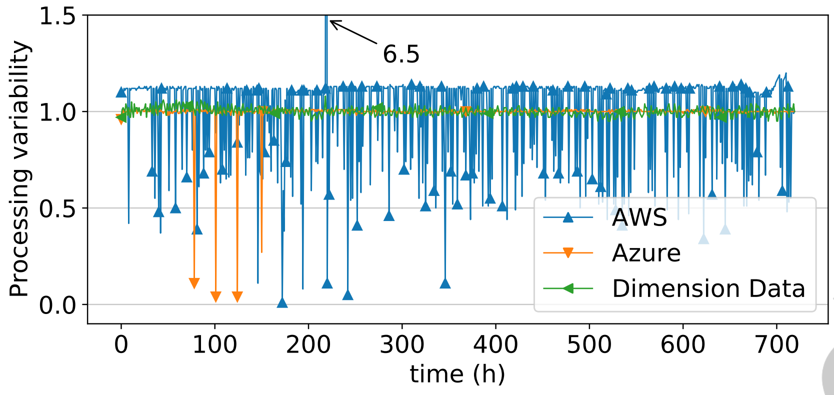


Fig. 4. Processing variability in cloud providers. Data from [23] measured hourly over a period of 30 days. Normalized (mean=1) to show relative performance variability on a single cloud provider.

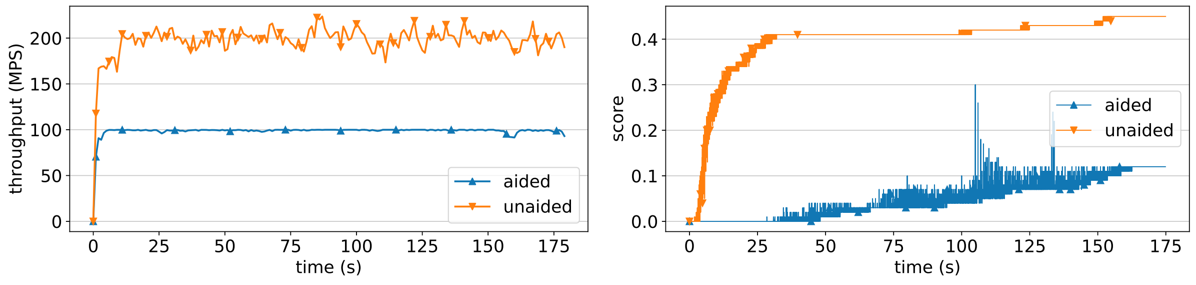
time series (on the y-axis) with respect to time (on the x-axis) in Figure 4. The key observation from the figure is that there exists more than an order of magnitude performance variability in hosted cloud providers. It is this variability that leads to processing heterogeneity even for cloud-hosted DLTs.

3.2 What If We Disregard Heterogeneity in DLT Processing?

Disregarding heterogeneity leads to centralization of voting power. To quantify the impact of heterogeneity on centralization, we run an experiment with 100 nodes in OMNeT++. In the experiment, the incoming rate of new messages is set to 200 messages per second (MPS). The 100 nodes are modeled with a mean processing rate between 90-350 MPS with their message processing rate modeled (Section 5.1 for details) after the traces from Figure 4. During the experiment, we measure a metric called *centralization value*, which is calculated as a ratio of nodes which are left behind and can not vote (due to their inability to process high rates of new ledger update messages) and the total number of nodes (Section 5.2 for details). Hence, the lower the score, the higher the participation in voting, thus, higher DLT decentralization (desired). We further investigate two network scenarios: aided and unaided. In the aided setup, a DLT can enforce a fixed throughput rate (i.e., flow control) which is calculated keeping the slowest node(s) in mind (e.g., with a minimum DLT joining requirement), thus ensuring a certain level of decentralization at the expense of resource utilization. The current IOTA 2.0 solution proposes this [51]. In the unaided setup there is no network-level support for heterogeneity.

Figure 5 shows our results for aided and unaided cases. First, we look at the throughput (the y-axis) with time (on the x-axis) of both cases as shown in 5a. As expected, the aided case leads to a stable throughput of 100 MPS while underutilizing the remaining processing capability of the network. In comparison, in the unaided case the throughput increases until around 200 MPS (the orange line), thus proving that the network has spare, underutilized capacity. However, when we analyze the centralization values of aided and unaided configurations we observe an opposite picture (Figure 5b). As the unaided configuration delivers higher performance, it leaves a large chunk of slow nodes out of sync (up to 40% by the time 30 seconds), i.e., the incoming rate of new messages is greater than the processing capabilities of a node. These out-of-sync nodes can not participate in voting, thus, allowing only faster nodes to have fully control of the consensus procedure. In contrast, the fixed-rate aided DLT only leaves less than 10% nodes out of sync, though at the cost of poor, underutilized DLT network.

However, both previously shown scenarios are not optimal as they either statically trade throughput for decentralization or decentralization for throughput. A desirable solution should deliver high throughput at



(a) Mean throughput (higher is better)

(b) Centralization value (lower is better)

Fig. 5. Comparison of aided and unaided heterogeneity in voting-based DLTs in a heterogeneous network with 100 nodes. (a) Mean throughput (processed messages) of all nodes. Higher is better. (b) Centralization value (Section 5.2). Lower score indicates higher degree of decentralization (desired).

all times while guaranteeing a high degree of decentralization, and this is exactly what we propose with our Healthor.

4 DESIGN OF HEALTHOR

Healthor is a distributed flow-control protocol to improve the decentralization and performance in a heterogeneous DLT network. Before we introduce the details of the Healthor protocol, we first briefly present the network model and our assumptions in the following section.

4.1 Network Model and Assumptions

We denote the set of all nodes participating in the network as $\mathcal{M}$, where each node $m \in \mathcal{M}$ has a set of inbound and outbound neighbors denoted by $\mathcal{N}_m \subset \mathcal{M}$. Figure 6 shows a node model and its neighbors. A node and each of its neighbors are connected via bidirectional channels over which they exchange *messages*. A message contains data, e.g., monetary transactions, and a hashed reference to a previous message as its parent, building an immutable directed acyclic graph.

Network membership management and bootstrapping. Permissionless network membership implicitly emerges through a multi-tiered bootstrapping process. First, the P2P overlay network needs to be established so that nodes are able to communicate. We assume this to happen via reciprocal manual peering, meaning that node owners exchange address information about their nodes offline and configure their nodes to connect to each other. An alternative to manual peering is a Kademlia-like [45] peering mechanism where initially trusted, so-called entry nodes are contacted to provide a list of known nodes of the network. In that way nodes can discover themselves automatically. Second, once the P2P overlay network is established, nodes need to agree on the initial ledger state. Generally, this is done via a trusted *genesis* which contains all the necessary information for a DLT to bootstrap. The *genesis* is decided upon by social consensus by the developers, community members and participants of a DLT before it is started. After this step, nodes can then start exchanging messages. It is important to note that the above items refer to any permissionless DLT. Conversely, Healthor does *not* need to be bootstrapped specifically: due to the local nature of operations such as health propagation and flow-control decision-making, only depending on a node's direct neighbors, and the periodical, adaptive mechanism, a special bootstrap procedure is not necessary.

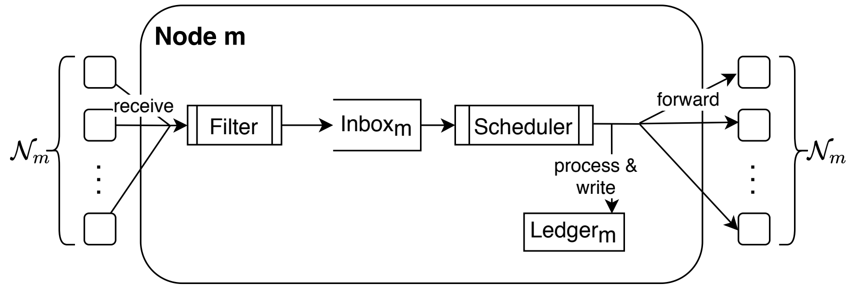


Fig. 6. Model of a node m and its neighbors $\mathcal{N}_m$.

Node operations. A node can perform various operations, namely issuing, receiving, processing, and forwarding messages. We assume that a node can issue new messages at a recommended rate which is enforced by the network congestion mechanisms [17]. In this work we focus only on the end-host flow-control mechanism. On receipt of a message, the node filters out duplicates, thus preventing replay attacks, and pushes unseen messages to its inbox, a buffer with limited size. Based on the message's cryptographic signature, the node can identify if it has processed all parent messages for a new message. In case of missing parent messages, it requests the missing messages from its neighbors via an implementation-specific synchronization mechanism like pull-action in Gossip-based networks [56]. For any other message for which the node has all parent messages (i.e, entire history), the message is scheduled for processing which includes cryptographic signature verification, and then writing ledger updates to persistent storage. After processing, the message is forwarded to the neighbors.

Network modes. We distinguish between two different operating modes of a network, *aided and unaided heterogeneity*. In the aided case, we assume that an overall processing rate v_{net} (messages per second (MPS)) of the network is defined. This is the message rate at which the network as a whole should operate (we will discuss more about it in Section 5.4). Let v_m be the variable message processing rate of a node m. Hence, node m would be able to process and forward messages at rate $\min(v_{net}, v_m)$. Ideally, $v_m = v_{net}$ at any time, meaning that a node m is able to operate at the overall network rate. However, operating conditions may lead to performance fluctuation, thus leading to accumulating messages at the inbox. In case the inbox runs full a tail drop policy is used, i.e., new messages that would cause the buffer to overflow are dropped. In the unaided case, a node m is free to forward messages at its rate v_m. The current Healthor protocol design is for the aided case, but in Section 8 we discuss how we can relax knowing about v_{net} and what implications it has.

4.2 Workings of Healthor

The basic idea behind the protocol is to rate-limit message forwarding in a DLT network based on neighbor's message processing capacity, termed as its *health*. Intuitively, the notion of health captures the dynamic heterogeneity of the DLT network, that might be changing over time. Figure 7 presents an example showing the intuition behind the Healthor's design at a high level. The figure shows a DLT network with 3 nodes (A, B, and C). Node A has new messages to forward to nodes B and C. Before A calculates the forwarding rate, it receives the health updates (0.5 for B, 1.0 for C), and then calculates the rate based on the updates. In this example, node C gets all the messages, whereas B only gets half, while the other half is buffered by node A on behalf of node B. This basic mechanism is the key insight in our flow-control protocol where more capable nodes can buffer messages to accommodate performance fluctuations in weaker nodes.

Healthor operates at the application layer and maintains connections to neighbors in a group communication setup. It is inefficient to rely on existing mechanisms such as TCP-based backpressure because (1) TCP-based

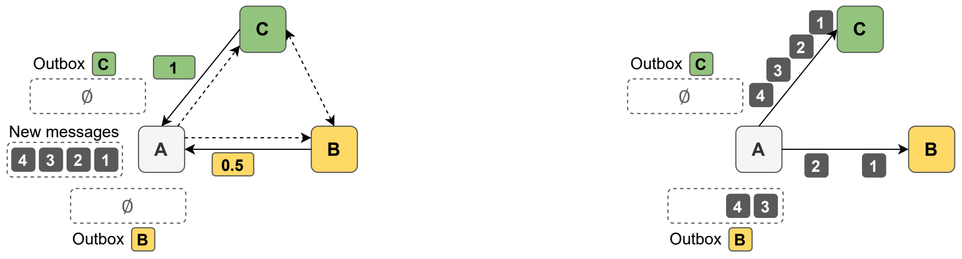


Fig. 7. High-level design of Healthor. Nodes periodically send their health to their neighbors (left). A adjusts its rate according to a neighbor's health and buffers messages in an outbox per neighbor (right).

backpressure runs independently of the application level, and would need modifications to the TCP implementation which is not feasible in public DLTs; and (2) TCP is a point-to-point protocol which cannot efficiently handle group communication dynamics.

Figure 8 depicts an updated model of a node m with the new components from Healthor highlighted. Healthor is a framework encompassing four separate engines, inspired by TIMELY [47], that uses local queue building and health updates as a signal. Additionally, a node m has an $Outbox_n$ for every neighbor $n \in \mathcal{N}_m$ where references to messages for n are stored before forwarding. In the following section we introduce these new engines and associated design decisions.

4.3 Health Measurement Engine

The *Health Measurement Engine* is at the heart of the mechanism. By introducing the notion of health $h_m \in [0, \infty)$, a node m can express its fitness regarding processing the maximum number of messages as defined by the expected network's processing rate v_{net} (in the aided case). A node periodically calculates its health based on its inbox occupation $l_m \overset{\text{def}}{=} len(Inbox_m)$ as following

$$h_m = \begin{cases} \frac{v_m}{v_{net}} & \text{if } l_m < v_{net} \text{(per second)}, \\ 1 - \frac{l_m}{l_{capacity}} & \text{otherwise.} \end{cases} \quad (1)$$

where $l_{capacity}$ is the maximum size of the inbox, configured on startup of node m. In (1) we can compare l_m (number of messages) with v_{net} (messages per second) because health measurements are done every second, thus, making v_{net} the expected number of messages. However, if a different interval is used the calculation needs to be adjusted according to the number of messages in this duration.

Choosing a signal. It is not trivial to find a reliable signal for rate control in a distributed, permissionless DLT setting. Relying on special hardware support such as Explicit Congestion Notification or changes to the operating system stack are infeasible in a public DLT. Due to the ever-changing nature of available computing resource an initial announcement of capabilities is also not practical. Therefore, the solution must come from the end-host's application level and not from the protocol-level like TCP.

Recall that in a DLT all messages are delivered to all nodes eventually. If every node keeps track of received messages in relation to v_{net}, every node knows the current health level of the network. Therefore, the inbox occupation gives a reasonable assessment of *how much a node is in sync*, i.e., whether it is able to receive and process state updates in a consistent and timely manner. A low inbox occupation signals a node being able to process at the network's pace, being healthy. Conversely, a high inbox occupation conveys that it is struggling to keep up with network activities. Thus, the node is unhealthy.

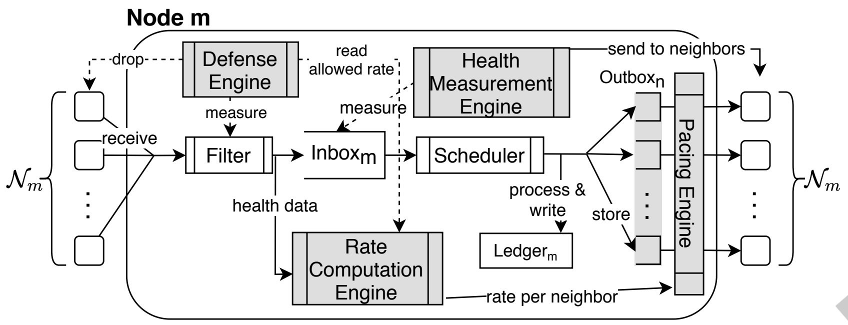


Fig. 8. Model of a node m with Healthor. New components compared to Figure 6 are highlighted in gray.

Different means to compute health. As *health* is such a central component of the mechanism, we experimented with several approaches before concluding on the one described in Eq. (1). Firstly, only using the second branch of Eq. (1) leads to $h_m \in [0, 1]$ and a node m always being slightly unhealthy as soon as there is any message in the inbox. Our initial simulation results showed consistently poorer performance than the ideal aided case (i.e., $v_m = v_{net}$) as the ideal case represents the achievable theoretical maximum. By introducing the condition $l_m < v_{net}$ a node can still be seen as healthy if its inbox is occupied with a few messages as long as they remain less than the maximum messages expected at the network rate, the performance was equal to the aided case. Lastly, allowing a health rate greater than 1 enables nodes to temporarily go faster than v_{net}, which we adopted as our final way to compute h_m.

Health messages. A health message is very lightweight, containing simply the health of the node m as a double precision floating point number. It therefore is only 64 bits of data. Additionally, in a real system, like any message, it should contain a node signature to verify a valid origin of a message. If a health message is lost for any reason, a neighbor n simply continues forwarding at its last known rate for a node m until it receives a new health message.

4.4 Rate Computation Engine

On receipt of a neighbor n's health data the node's *Rate Computation Engine* calculates the message forwarding rate for this neighbor n as shown in

$$r_n = v_{net} \cdot h_n. \tag{2}$$

The node computes the forwarding rate r_n linearly according to the neighbor's health h_n. Therefore, it can even go faster than the target network rate v_{net} if its neighbors can process messages at the forwarding rate of r_n. Nevertheless, a node can only forward as much as there is network activity (new messages are issued), and it is able to process itself.

4.5 Outboxes

As in Figure 8 illustrated, a node m has an *Outbox$_n$* for every neighbor $n \in \mathcal{N}_m$. When a message is processed, a reference to it is added to every outbox. The original message is stored in the node's local ledger. The *Pacing Engine* takes care of forwarding messages to neighbors at their respective rates.

Drop policy. Similar to the inbox, an outbox is a buffer of limited size, defined on node startup. In case an outbox runs full, a tail drop policy is used (Figure 9 top), i.e., new messages that would cause the buffer to overflow are dropped while message requests are prioritized as shown. We also experimented with random (Figure 9 bottom) and head drop (Figure 9 center) policies. However, these turned out to be unfavorable due to the fact that

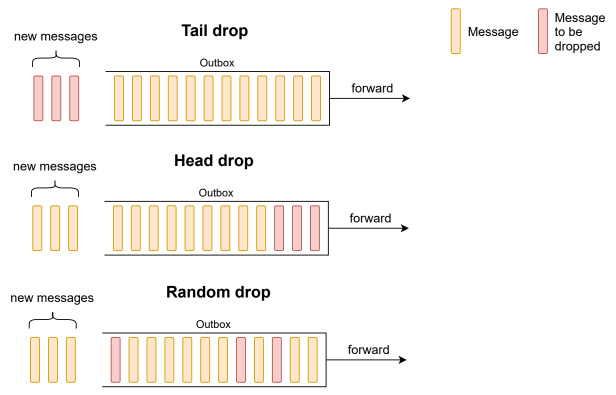


Fig. 9. Drop policies of outboxes.

messages are generally forwarded in order. If messages from the beginning of the buffer are dropped, the receiver needs this already dropped message to process later messages. It, therefore, needs to request these dropped messages adding even more overhead.

4.6 Pacing Engine

The *Pacing Engine* fetches messages from an $Outbox_n$ for a neighbor n and forwards them at the rate calculated by the *Rate Computation Engine*. It essentially controls each flow of messages to every neighbor to achieve the given forwarding rate r_n. A possible implementation in a real-world system could make use of one thread per $Outbox_n$ that pulls messages from the outbox and forwards them while inserting delays to match r_n.

4.7 Defense Engine

It is essential to protect nodes against exploitation and adversarial behavior in a permissionless DLT setting. Therefore, the *Defense Engine* locally monitors a node's neighbors behavior and initiates appropriate actions if a protocol violation is suspected. Fundamentally, it provides incentives, hardens the flow-control mechanism, and makes nodes resilient against attacks. There are two main attack vectors on Healthor, namely exceeding the allowed forwarding rate and manipulation of health updates.

Exceeding the allowed forwarding rate. The basic idea is that, in the aided case, the expected network rate v_{net} is known to all the nodes in the network. Hence, a node's neighbor n calculates the forwarding rate as defined in Eq. (2) and its pacing engine forwards out messages at this rate (i.e., the allowed rate). However, a neighbor may diverge from this rate because of being unhealthy itself, connection issues, or adversarial behavior. In any case, an extremely large divergence of the allowed rate cannot be tolerated. Therefore, every node m keeps track of the rates of every neighbor n by simply counting the received messages per neighbor. The *Defense Engine* of a node m periodically creates a moving average of the allowed rate within the time window tw as well as a moving average within the same window for every neighbor's receive rate. If a neighbor exceeds the allowed rate β-times in a row, the neighbor is dropped. Likewise, if a neighbor falls below the rate β-times.

Manipulation of health updates. A node m sends the same health update to all its neighbors. Therefore, health updates can be absent or manipulated by an adversary. If no health updates are received, a node m simply uses the known previous health of a neighbor n. At startup every node considers all its neighbors to be healthy, i.e., $h_n = 1, \forall n \in \mathcal{N}_m$. Generally, an adversary can only influence its own view on the network traffic by manipulating its health. For example, if she lies and sends different health messages to distinct neighbors, each neighbor will send at different rates according to the protocol. However, this does not have any influence on the neighbors.

On the other hand, an adversary could try to inflate outboxes of neighbors and to slow them down by pretending to be unhealthy. Inflation of outboxes, however, is not possible due to their limited size and drop policy. Nonetheless, a neighbor should not waste resources and therefore the *Defense Engine* implements a similar strategy to the forwarding rate. A node m keeps track of the health of every neighbor n by simply recording the outbox occupation. Recall that a node buffers messages for unhealthy neighbors. The unhealthier a neighbor n the higher the occupation will be. If a neighbor n is unhealthy for too long, $Outbox_n$ will run full and eventually the neighbor is dropped after β measurements.

5 EVALUATION

We evaluate Healthor at two different scales. First, we explore *node properties* such as throughput, latency and individual load at a small scale. These microbenchmarks are conducted simulating a network of 10 nodes such that tunable parameters of Healthor can be separately investigated. Second, we examine global *network properties*. In macrobenchmarks we shift focus on the overall performance of the network and study decentralization, throughput, and tail latencies. Along with this evaluation we discuss the following questions:

- *Does Healthor take load away from unhealthy nodes and allow nodes to stay (longer) in sync?* Our findings in Section 5.3 suggest that Healthor reduces load on low-end nodes and allows nodes to stay in sync when they could not with aided heterogeneity.
- *What is the influence of processing heterogeneity on decentralization and throughput?* We observe in Section 5.3 that some nodes that fulfill the network requirements nominally, i.e., v_m is greater than v_{net}, fall out of sync because of heterogeneity, thus increasing centralization.
- *Can Healthor improve decentralization, throughput and/or latency?* Our results in Section 5.4 demonstrate that, indeed, Healthor can improve all three properties.
- *Does Healthor provide the above improvements while staying resilient against attacks?* We show in Section 6 that nodes can detect protocol violations and protect themselves against attacks reliably.

5.1 Setup

We built a discrete-event simulator using OMNeT++ to simulate the permissionless, voting-based DLT modeled in Section 4.1 and test our mechanism. In our experiments we employ small-world networks of various sizes $|\mathcal{M}|$ where each node has between 2-4 random neighbors to model the properties real-world networks as such random pairing is done in applications that adopt a Kademlia-like [45] peering mechanism such as Ethereum [24], IPFS [8], BitTorrent [16], and Storj [62]. The distance between two randomly chosen nodes is in the order of $\log |\mathcal{M}|$ [45]. If a node gets out-of-sync, it goes offline and its neighbors repeer with other random nodes that have less than 4 neighbors.

Why simulation? Design and development of a DLT needs a careful analysis of nature of decentralization, performance, and node heterogeneity. Among a variety of potential consensus mechanisms available [36], voting-based, public-permissionless is now gaining traction, but large-scale deployments (>1,000 nodes) are not yet available [58, 68]. In this work, we use system simulation to evaluate Healthor. The use of simulation allows us to explore the bounds of our design systematically, and perform in-detail sensitivity analysis on specific configuration parameters. Nonetheless, we are aware of the limitation of such simulation studies. A practical deployment has additional challenges such as bootstrapping overheads, monitoring data collection, and connectivity issues, which we have not addressed in this study. As a next step, we will perform validation of our results on a small-scale deployment to increase confidence in our results.

Parameters. We adopt a Poisson process as the network processing rate $\nu_{net} = 100$ MPS for our experiments with aided heterogeneity and Healthor. Theoretically, ν_{net} is not defined for unaided. However, for simplicity we assume $\nu_{net} = 200$ MPS for our experiments in this case. A random subset of nodes issues messages following a Poisson distribution with parameter λ_m, so that $\sum_m \lambda_m = \nu_{net}$. Channels are assumed to have a delay between 50ms and 150ms uniformly at random to simulate real network conditions [27, 37].

The variable processing rate ν_m of a node m is modeled according to the cloud performance traces shown in Figure 4. As a conservative approximation of real-world DLT P2P networks [9, 27, 37, 43, 66], we adopt a distribution of 50% constant, 25% AWS, 15% Azure, and 10% Dimension Data. Every ν_m is randomly scaled between $0.9\nu_{net}$ and $3.5\nu_{net}$ (in the case of unaided, the original ν_{net} value with which is compared is used). For example, if a node m is of type *AWS* with $1.5\nu_{net}$ its *mean* processing capabilities are $\bar{\nu}_m = 150$ MPS but vary over time as pictured in Figure 4 where the x-axis is randomly shifted.

Every node's *Health Measurement Engine* computes its health h_m and sends it to its neighbors every 1 second. Since we do not consider message priorities, a node's scheduler operates according to FIFO. A node's *Defense Engine* calculates allowed rates at an interval of 1 second and creates moving averages over a time window $tw = 3$ seconds. A neighbor is dropped after violating the protocol $\beta = 5$ times in a row.

5.2 Metrics

Centralization score: quantifying (de)centralization. In P2P networks, decentralization is the property of not relying on any centralized component. DLTs work, by definition, in a decentralized way. However, while in theory no centralized components are present, to prevent Sybil attacks nodes have different influence on consensus. Hence, if a node assumes too much power (e.g., hashing power in Bitcoin [59], concentration on major cloud hosting providers [27, 37, 43]), we can conclude it has exceeding control of the network. Decentralization is a fundamental property of DLTs, and with the *centralization score* we introduce an easy way to compare the degree of decentralization in a voting-based DLT with the ratio of nodes being able to participate in voting.

The *centralization value* is the number of nodes that are not able to process all messages within a defined time window d normalized by the total number of nodes and is defined as

$$c_{value}(t) = \frac{\sum_{m \in \mathcal{M}} unsync_m(t)}{|\mathcal{M}|}, \tag{3}$$

where

$$unsync_m(t) = \begin{cases} 1 & \text{if } m \text{ processed all messages in } [t, t+d], \\ 0 & \text{otherwise.} \end{cases}$$

As such, a lower value is better because more nodes in the network are able to participate in consensus. In our evaluation we adopt a time window $d = 5$ seconds, i.e., a node is considered not being able to participate in consensus if at least one message has been received by the node later than 5 seconds from the time the message has been issued, also taking into account network delays. Considering that voting usually takes place in rounds it is reasonable to assume that a node being a bit behind is still able to participate. In case of the IOTA 2.0 protocol voting rounds are initiated every 10 seconds [51]. Recent PoS DLTs like Polkadot [63] adopt block times of around 5 seconds. Also in these systems a node needs to be able to receive and process a transaction within this bounded time window in order to be able to produce or validate a block, thus taking part in consensus.

To show the evolution of centralization we plot the centralization value over time, which in simulation can be easily determined and in real world scenarios can be inferred by the voting participation of nodes. Accordingly, the *centralization score* is the mean of the centralization value over a given duration D and expresses the degree

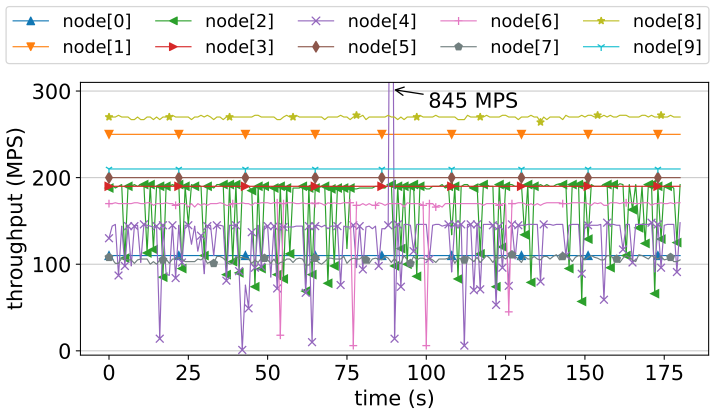


Fig. 10. Processing rates v_m in a network with 10 nodes. A node's mean processing rate $v_m \geq v_{net}$.

of decentralization in a single number. It is defined as

$$c_{score}(D) = \frac{\sum_{t=0}^{D} c_{value}(t)}{D}. \tag{4}$$

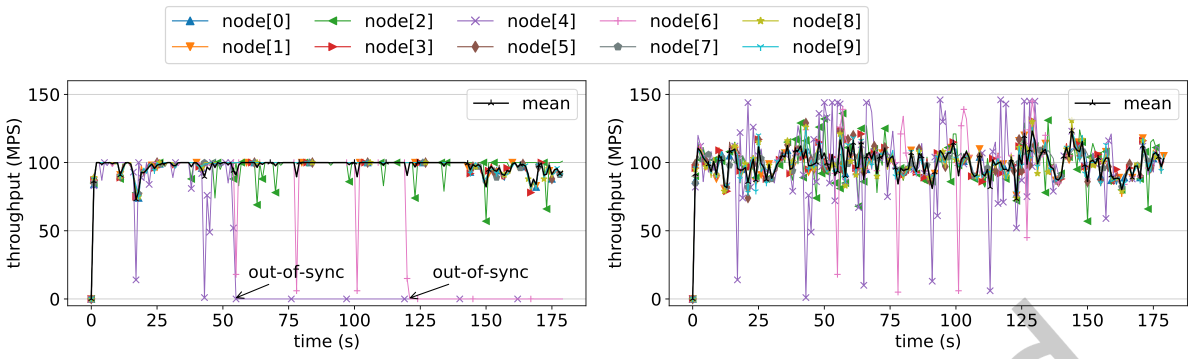
Throughput: measuring network performance. We measure the processing time of messages on a node and aggregate it by the granularity of one second. This constitutes the throughput of a particular node at a given time. To evaluate the network's mean throughput we sum up the throughput of all nodes divided by the count of nodes. This calculation excludes out-of-sync nodes, i.e., only participating nodes contribute to the network's throughput.

Latency: assessing delays. Delays are important in every network where small delays correspond to quick response times. In voting-based DLTs a small latency is crucial for nodes to be able to participate in consensus. We measure latency of any given node m as the duration between message issuance time and processing time on node m. When assessing latency as global network property, we consider the 95 percentile latency of all in-sync nodes.

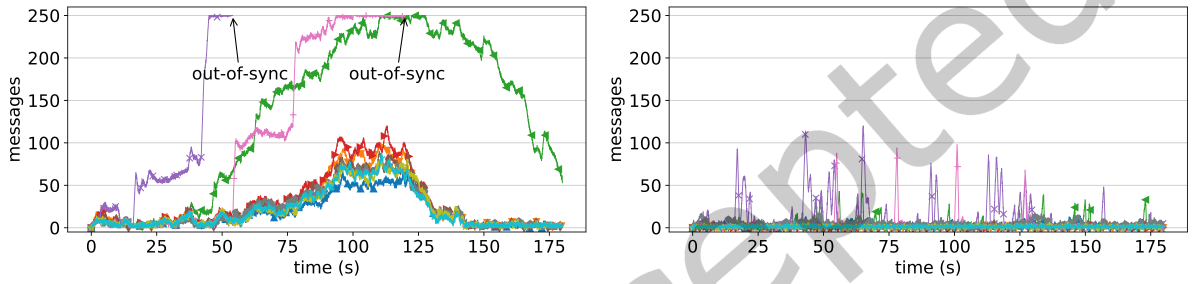
5.3 Microbenchmarks

We compare aided and Healthor in a network with 10 nodes. Figure 10 depicts that each node's mean processing rate v_m is larger than or equal to v_{net}. It can be seen that node[2] ($\bar{v}_2$ = 170 MPS), node[4] ($\bar{v}_4$ = 130 MPS), and node[6] ($\bar{v}_6$ = 170 MPS) sometimes fall below the network processing rate v_{net} = 100 MPS. Therefore, these nodes are of special interest. Due to the limiting nature of aided the actual processing rates used are $\min(v_{net}, v_m)$ and higher rates can not be leveraged.

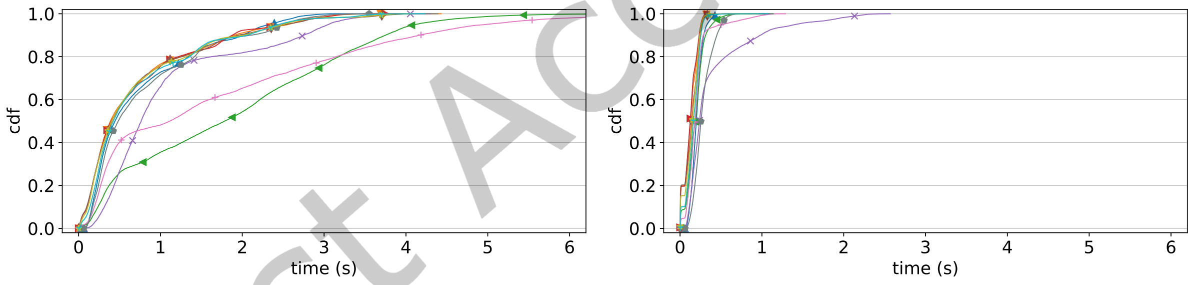
Comparison of aided and Healthor. Figure 11a details the throughput of aided (top) and Healthor (bottom). The x-axis shows the throughput in messages per second, and the y-axis the time in seconds. Each node's throughput can be directly related to its available processing rate as depicted in Figure 10. For aided, the throughput is capped at a maximum rate v_{net} ($mean$ = 97.51 MPS), while Healthor temporarily allows higher throughput ($mean$ = 99.86 MPS) which fluctuates around—instead of being limited by—the value v_{net}. In aided, node[4]'s throughput drops to 0 around second 55. Similarly, node[6]'s throughput falls to 0 at 120 seconds. This indicates that both nodes are out-of-sync, i.e., their inboxes are filled up with too many messages without their known history, so that no newly received message can be scheduled. node[4] and node[6] go offline from this point in time and cannot participate in consensus anymore until a heavy re-sync operation is performed. In



(a) Throughput.



(b) Inbox occupation.



(c) Latencies.

Fig. 11. Experiment results in a network with 10 nodes with aided heterogeneity (left) and Healthor (right).

practice this could mean manual intervention and restart with a trusted, previously downloaded ledger state snapshot.

Figure 11c shows the CDFs of the latencies for aided and Healthor, respectively. In aided, node[2] and node[6] have by far the largest latency, exceeding 5 seconds at the tail. Recalling the definition of centralization score in Section 5.2, this indicates that these nodes are too far behind to participate in consensus for some messages. Indeed, as shown in Figure 11a, node[4] and node[6] run out-of-sync. However, node[4]'s latency does not exceed 5 seconds which implies that it got out-of-sync quickly. With Healthor, the network latency is significantly lower by 73% compared to aided. Especially node[2]'s and node[6]'s 95 percentile latencies stand out with an improvement of 91% and 89%, respectively.

Inbox	c_{score}		**MI** `node[4]`		**MI** `node[6]`	
	aided	**Healthor**	**aided**	**Healthor**	**aided**	**Healthor**
100	0.14	0	100	100	100	90
250	0.11	0	250	120	250	98
500	0.08	0	500	120	400	115
1000	0.08	0	920	120	400	115
2000	0.08	0	920	120	400	110

Table 2. Different inbox sizes with aided and Healthor. MI=maximum measured inbox occupation.

A closer look at inbox sizes. Figure 11b shows the number of messages stored at each node's inbox over time, i.e., inbox occupation. Every node can store up to 250 messages. Generally, we observe that buffers have higher occupation in aided, which leads to higher latencies and indicates higher memory load. Messages reside longer in the inbox before they can be scheduled because of the enforced processing rate limit. `node[4]`'s and `node[6]`'s inboxes run full around second 40 and 90, respectively. Once this happens for too long the node is nonrecoverable out-of-sync and, simultaneously, its throughput drops to 0, hence it goes offline. Temporarily, also `node[2]`'s inbox runs full at $\sim$ 120 seconds, but it does not go out-of-sync. Instead, it is able to recover via pull action and its inbox occupation drops towards the end of the simulation. Inbox occupation with Healthor follows a different pattern: none of the nodes' inbox runs full, thus all nodes stay in sync. Furthermore, the inbox fills and empties in a zigzag pattern reacting to health changes. Recall that a node gets unhealthier when its inbox grows, and its neighbors forward at slower rate until the node gets healthier again.

Table 2 shows the centralization score and the maximum inbox occupation of `node[4]` and `node[6]` for various inbox sizes. The centralization score captures when a node is not able to participate in consensus. Therefore, it is a reasonable tool to assess the effect of various inbox sizes. On the one hand, a too small inbox can be easily fulfilled. Hence, new messages are dropped with high probability up to the point where the node falls out-of-sync. On the other hand, a too big inbox size might consume too much memory while only increasing delays. Either way a node is not able to vote.

We observe that `node[4]` and `node[6]` get out-of-sync in the aided case when the inbox size is 100, because both nodes' inboxes run full. With Healthor, the centralization score is 0 which signals that no nodes got out-of-sync, even though `node[4]`'s inbox reached 100. In this case, `node[4]` could recover via a pull action and with the help of its neighbor's buffering while it is unhealthy. Larger inbox sizes seem to improve the centralization score slightly for aided but the fundamental problem of `node[4]` and `node[6]` getting out-of-sync remains. With Healthor this problem is already alleviated with an inbox size of 100, where every node can stay in sync. However, `node[4]`'s inbox is at its maximum capacity, and the node needed to recover via pull actions. Inbox sizes larger than 250 only let nodes look healthier, but cannot improve the centralization score (which is already 0). Therefore, the maximum inbox occupation measured is slightly higher. Henceforth, we adopt an inbox size of 250 for all of our experiments as a reasonable tradeoff between memory consumption and storing capacity.

A closer look at outbox sizes with Healthor. Recall that a node m has an $Outbox_n$ for every $n \in \mathcal{N}_m$. It is therefore important to establish a fitting outbox size, so that enough messages can be stored, but no unnecessary overhead is created. Table 3 shows the centralization score, maximum inbox occupation of `node[4]` and `node[6]` as well as maximum outbox occupation of both nodes' neighbors. Larger outboxes indicate less pressure on an unhealthy node up to an outbox size of 250, as is evident by the higher inbox occupation of `node[4]` with an outbox size of 100. However, this trend can only be observed until an outbox size of 250, where the outbox occupation at `node[4]`'s neighbors and its inbox occupation stabilize. An outbox size of 250 seems to offer a good

Outbox	c_{score}	**MI node[4]**	**MON node[4]**	**MI node[6]**	**MON node[6]**
100	0	160	100	98	100
250	0	120	200	98	100
500	0	120	200	98	100
1000	0	120	200	98	100
2000	0	120	200	98	100

Table 3. Outbox sizes with Healthor. MI=maximum measured inbox occupation, MON=maximum outbox occupation at neighbor node.

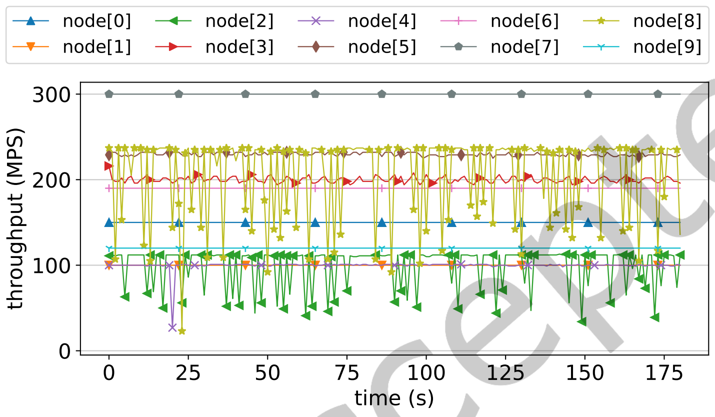


Fig. 12. Processing rates v_m in a network with 10 nodes (additional scenario). A node's mean processing rate $v_m \geq v_{net}$.

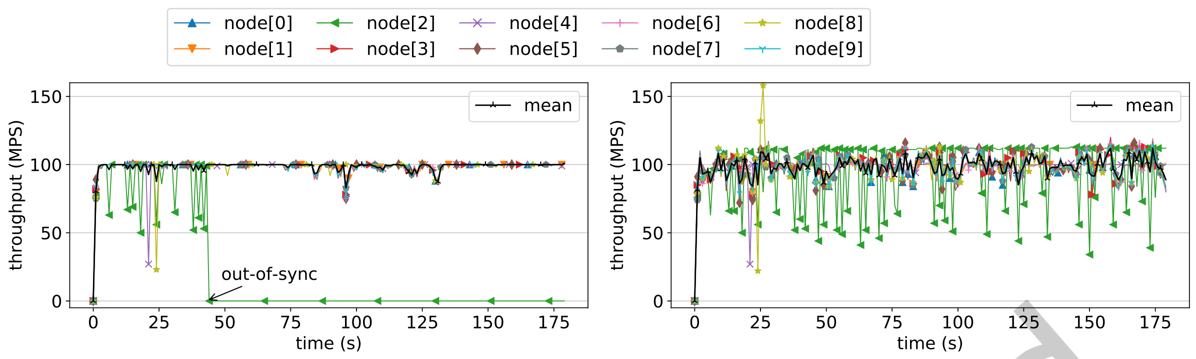
tradeoff between decreasing load on an unhealthy node and creating overhead on its neighbors. We therefore adopt an outbox size of 250 for our experiments.

Additional Scenario The previous analysis was done with the same network and node heterogeneity setup. To gain a better intuition of Healthor's behavior for different networks, we now take a look at the same set of figures (Figure 13) for throughput, inbox occupation, and latencies for a different 10 node network setup. Figure 12 shows the nodes' processing rates v_m, where each node's mean processing rate v_m is larger than or equal to v_{net}. The network's parameter are as described in Section 5.1. We observe that the mean of v_m for node[1], node[2], and node[4] are equal to the network processing rate v_{net} = 100 MPS. Hence, these nodes are of special interest.

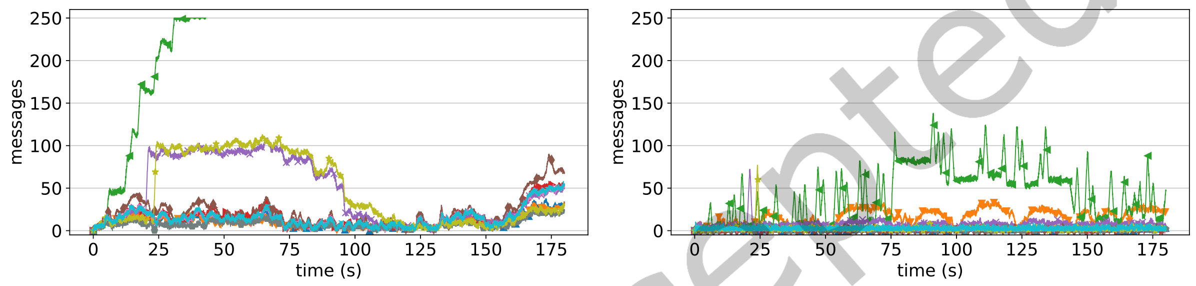
Figure 13a details the throughput of aided (left) and Healthor (right). We show, for the former, the throughput is capped at a maximum rate v_{net}, while the latter, temporarily allows higher throughput, which fluctuates around — instead of being limited by — the value v_{net}. In aided, node[2]'s throughput drops to 0 around second 45 which means that the node is out of sync and offline from this point in time and cannot participate in consensus anymore.

In Figure 13b the inbox occupation is displayed. At the same time, we observed node[2]'s throughput dropping to 0, i.e., getting out of sync, we can see its inbox becoming full for the aided case (left). Comparing to Healthor (right), node[2]'s inbox does not grow larger than 145 and follows the zigzag pattern reacting to health changes, as we have seen for the first scenario already. Overall, inboxes do not grow as full with Healthor as in aided.

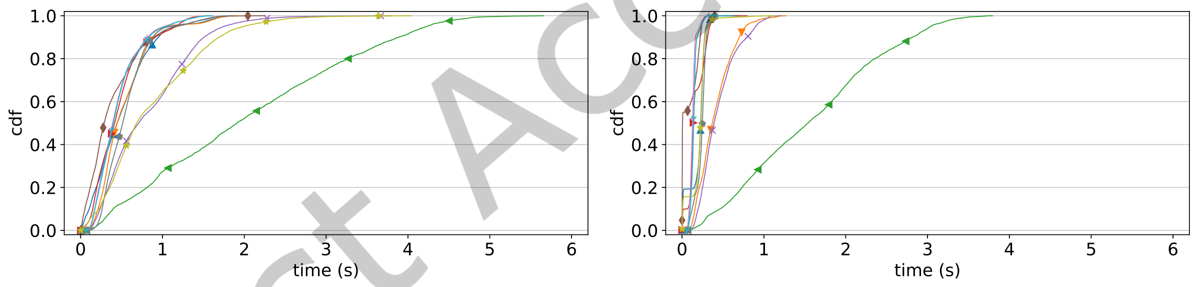
The CDFs of the latencies for aided (left) and Healthor (right) are shown in Figure 13c. In aided, node[2]'s latency exceeds the 5s mark and indicates, once again, that this node is getting out of sync. With Healthor the



(a) Throughput.



(b) Inbox occupation.

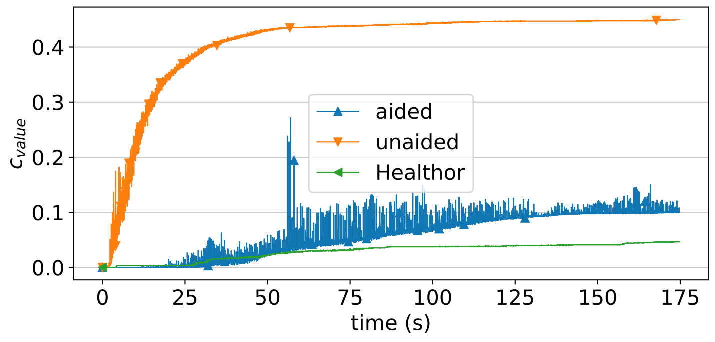


(c) Latencies.

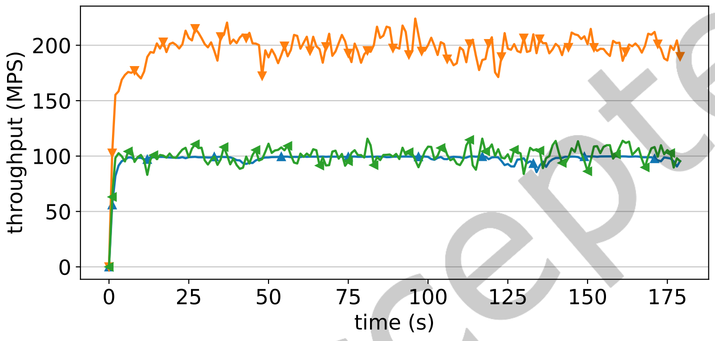
Fig. 13. Experiment results in a network with 10 nodes (additional scenario) with aided heterogeneity (left) and Healthor (right).

latency for node[2] is lower and this is not the case. For all other nodes especially tail latencies are large in aided whereas this is not the case with Healthor.

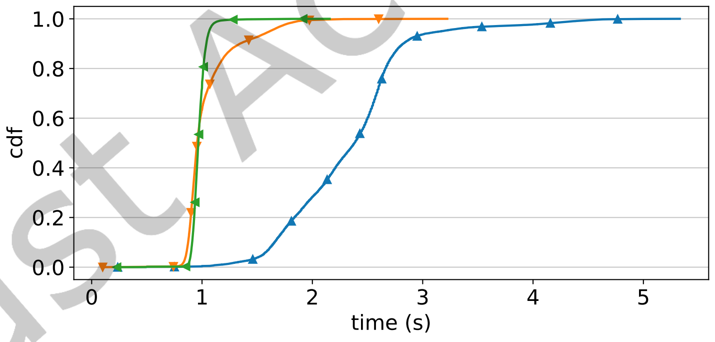
Overheads. In our evaluation we configure Healthor to exchange health messages every 1 second. Naturally, this adds message overheads compared to aided. The theoretical maximum overhead of a node for a simulation of length T time can be calculated as $o_{max}(m) = |\mathcal{N}_m|T$. Assuming that a node has $|\mathcal{N}_m| = 4$ neighbors during a simulation of 180 seconds results in $o_{max} = 720$ health messages sent and received. In a flooding-based P2P network with $v_{net} = 100$ MPS and the same configuration the maximum number of sent and received messages is 72,000. It follows that, Healthor incurs a maximum overhead of 1% in the deployed configuration. However, we are aware that the overhead is a function of message rate and can be high with low throughput.



(a) Centralization value.



(b) Throughput.



(c) 95 percentile latency.

Fig. 14. Experiment results in a network with 2,000 nodes.

Discussion. In two different scenarios, we have shown that Healthor enables lower latencies and reduces load on low-end nodes compared to aided mainly by allowing temporarily higher throughput. It therefore unlocks enormous potential: *nodes can make use of resources when they are available, irrespective of network activity.* However, the mechanism does not come without overheads as our calculations show.

$\|\mathcal{M}\|$	aided			unaided			Healthor		
	c_{score}	L	T	c_{score}	L	T	c_{score}	L	T
100	0.04	1.93	98.24	0.40	0.83	198.38	0.02	0.73	101.10
500	0.09	4.24	97.15	0.38	1.17	196.20	0.02	0.95	100.81
1000	0.07	4.47	97.82	0.38	1.45	195.20	0.03	0.99	100.12
2000	0.06	3.17	97.79	0.41	1.61	196.98	0.03	1.06	100.82
5000	0.07	4.26	97.00	0.40	1.73	197.21	0.03	1.19	100.60

Table 4. Decentralization vs. throughput experiment results. L=95 percentile latency, T=mean throughput.

5.4 Macrobenchmarks

In the previous section we investigated Healthor closely and established sensible default parameters for inbox and outbox sizes. Following, we compare aided, unaided, and Healthor at a larger scale. We present overall network-level results for heterogeneous networks with up to 5,000 nodes in line with the description provided in Section 5.1.

Decentralization vs. throughput. Figure 14a shows the centralization value for aided, unaided, and Healthor in a network with 2,000 nodes. Clearly, unaided is the least decentralized whereas aided is significantly more decentralized and Healthor even more so. Figure 14b shows the throughput. As expected, aided is limited at $\nu_{net} = 100$, Healthor permits temporarily higher throughput around ν_{net}, and unaided is only limited by demand (here at $\nu_{net} = 200$). In Figure 14c, the CDF of 95th percentile delay of in-sync nodes details how delays are much higher in aided than in unaided ($\sim 2x$ higher) and Healthor ($\sim 3x$ higher). Messages in aided stay much longer in a buffer before being able to be processed and forwarded due to the limited rate. It is interesting to see that while Healthor slightly increases throughput and makes the network significantly more decentralized compared to aided, it still provides comparable latencies to unaided.

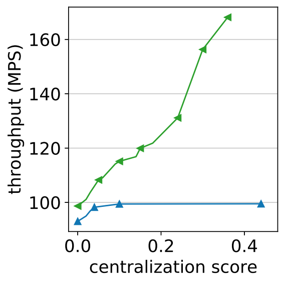
A similar trend can be observed in Table 4. In various network settings, unaided allows the highest throughput and has low latency, but it is also the least decentralized. Aided guarantees a fair decentralization but has poor latency and a maximum fixed throughput. Healthor offers the best out of both worlds: almost complete decentralization (78% improvement compared to aided), a slightly better throughput than aided and its latency is on par with unaided.

Sensitivity analysis. Looking closer at the behavior of aided and Healthor with the same degree of decentralization can reveal in which settings our mechanism improves throughput and latency. This is shown in Figure 15, where throughput (Figure 15a) and latency (Figure 15b) are plotted on the y-axis and centralization score on the x-axis. As expected, the throughput of aided converges towards ν_{net} and cannot exceed it with increasing centralization scores. We basically observe the system breaking down, because there are more messages issued than the fixed throughput ν_{net}. With Healthor it looks decidedly different: the throughput correlates almost linearly with the centralization score, hence is not limited at a fixed rate (desired). The network remains functional, however, it is in a similar elitist mode as unaided, where only high-end nodes can continue to participate. In Figure 15b we observe a correlation between rising latency and increasing centralization score in aided (undesired). Conversely, with Healthor we see a flat latency, hovering around 1 second, regardless of increasing centralization scores.

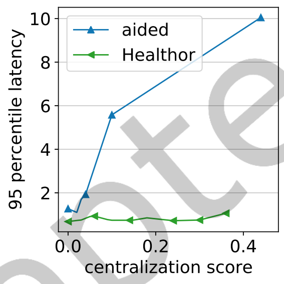
Table 5 shows by how much Healthor improves latency and throughput compared to aided with the same degree of decentralization for networks of various sizes. Again, we can observe much higher latencies with aided and a significant improvement of up to 76% with Healthor. Similarly, Healthor improves the throughout by up to 23% compared to aided in a network with 500 nodes.

$\|\mathcal{M}\|$	c_{score}	Latency			Throughput		
		aided	**Healthor**	**Dec. (↓)**	**aided**	**Healthor**	**Inc. (↑)**
100	0.04	1.93	0.76	61%	98.24	105.45	7%
500	0.09	4.24	1.06	75%	97.15	119.51	23%
1000	0.07	4.47	1.08	76%	97.82	115.10	18%
2000	0.06	3.17	1.12	65%	97.79	108.96	11%

Table 5. Comparing throughput and latency in aided and Healthor with same degree of decentralization.



(a) Correlation throughput and c_{score} (higher is better).



(b) Correlation latency and c_{score} (lower is better).

Fig. 15. Sensitivity analysis of unaided compared to Healthor in a network with 100 nodes.

Discussion. Aided shows one side of the extreme and limits throughput to increase decentralization. In light of processing heterogeneity, however, this is not very effective as our experiments have shown. The problem is that nodes may experience temporary slowdowns below the network processing rate v_{net}. As long as the throughput is close to v_{net}, these nodes lag behind further and further as they experience slowdowns creating higher latency. On the other hand, the limit on throughput prevents these nodes from falling out-of-sync completely and therefore provides decentralization.

With unaided nodes can simply operate at their own speed at all times. This clearly offers lower latency, but it creates an elite-network where only the strongest high-end nodes can participate, thus reduces decentralization. Since throughput is only limited by demand, it is practically only limited by the processing capabilities of the fastest participant(s) at the cost of excluding slower devices and becoming more centralized.

Healthor provides the best out of both worlds by embracing heterogeneity. It allows nodes to temporarily go faster than v_{net} and hence offers low latencies comparable to unaided, higher throughput than aided and most importantly improves decentralization even more than aided.

6 ATTACK ANALYSIS

Permissionless P2P systems are required to face challenges posed by adversarial behavior. In the case of DLTs this is exacerbated through financial motives. Therefore, protocols in permissionless DLTs should be tamperproof and resilient against exploitation. Healthor employs several inherent defense mechanisms as described in Section 4.7. We evaluate how the *Defense Engine* operates in targeted attacks.

Attack model. In the presented attacks we assume an omniscient adversary. Hence, the adversary is aware of every node's health, can intercept both messages and health messages. She is able to tamper with the aspects of

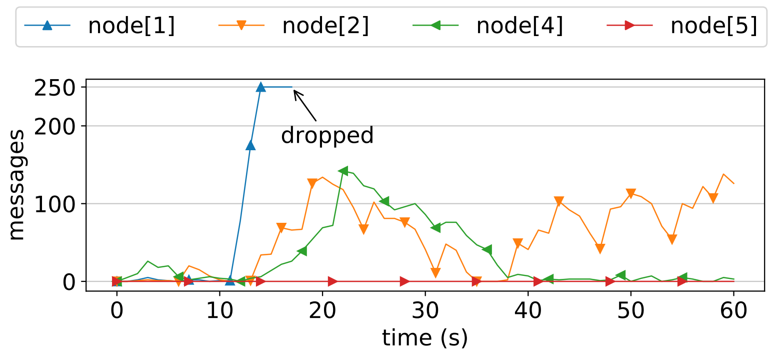


Fig. 16. Node[3]'s outbox occupation. Node[1] attacks node[3] by pretending to be unhealthy.

Healthor's protocol. Attacks on the network topology, such as eclipse attacks, and the underlying ledger are out of scope of this paper. We use the same 10 nodes network and parameters of Section 5.3 for the attacks shown in this section.

6.1 Manipulation of health updates

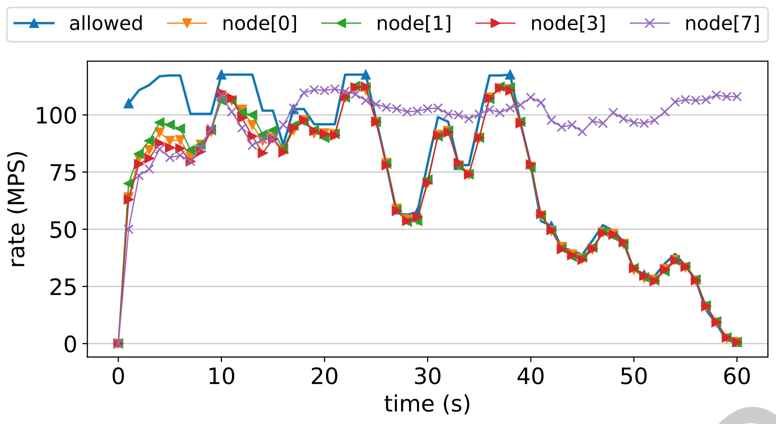
Not sending health updates. Recall that a node assumes a neighbor n to be healthy $h_n = 1$ when connecting to each other. Further, it uses the last known health status if a neighbor does not report any health. Therefore, an adversary cannot cause any harm by withholding health updates.

Lying about health. Since health is computed locally and then sent to the neighbors a node can lie about its health and even send different health status to distinct neighbors. From the viewpoint of a neighbor n of such an adversarial node m_a, node m_a behaves normally as long as no other protocol violations, such as being unhealthy for too long, exceeding the allowed sending rate, or sending too little, are detected. In essence, an adversary has only very limited capabilities which only affect itself.

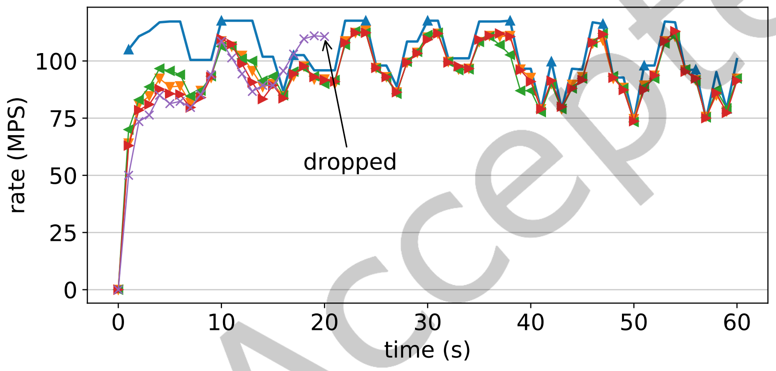
Pretending to be unhealthy. An adversary could try to inflate outboxes on its neighbors by pretending to be unhealthy. Healthor intrinsically averts this by employing limited lengths for buffers. However, an adversary could make its neighbors waste resources up to this limited amount which is not favorable. The *Defense Engine* implements a mechanism to detect whether a node is unhealthy for too long indirectly via its outbox occupation. If the outbox of a neighbor n is full for too long, it is dropped. Figure 16 shows this scenario where `node[1]` attacks `node[3]`. $Outbox_{node[1]}$ grows rapidly up to the maximum capacity of 250. Eventually, `node[3]` drops `node[1]` because the outbox has been full for too long. It can now repeer with another node.

6.2 Manipulation of forwarding rate

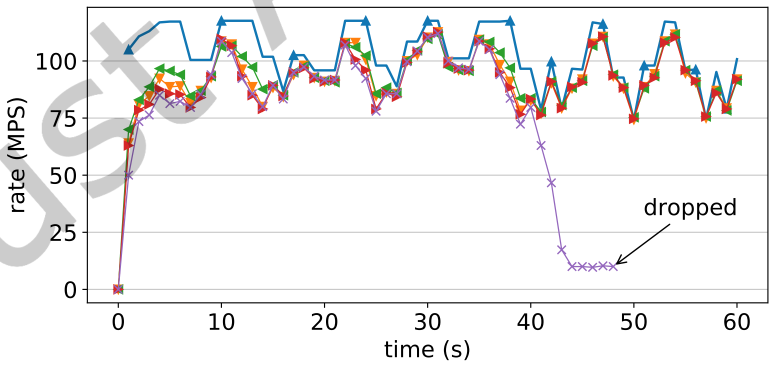
Forwarding more than allowed rate to neighbor. An adversary can choose to diverge from the allocated forwarding rate by sending more and trying to overload a victim. This is displayed in Figure 17a where `node[7]` attacks `node[2]` without *Defense Engine*. The figure shows the forwarding rates of `node[2]`'s neighbors as recorded on receipt by `node[2]`. The blue line on top indicates the maximum allowed forwarding rate for any neighbor. It is clear that `node[7]` exceeds this rate starting around second 15. As `node[7]` continues to send at high rate, `node[2]` gets unhealthier. Thus, the maximum allowed forwarding rate drops and other neighbors of `node[2]` reduce their rate accordingly. As a result, we can see that without the *Defense Engine* an attacker can overload a node by forwarding above the allowed rate.



(a) Disabled *Defense Engine*. Node[7] exceeds allowed forwarding rate.



(b) Enabled *Defense Engine*. Exceeding of forwarding rate is quickly detected.



(c) Enabled *Defense Engine*. Slowing down substantially is quickly detected.

Fig. 17. Receiving rates of node[2]'s neighbors, measured locally. Node[7] attacks node[2].

Figure 17b shows the same attack but with the *Defense Engine* enabled. `Node[2]` quickly detects `node[7]` divergence and drops the connection to it around second 20. This results in `node[2]` staying healthy and being able to process messages by other nodes until the end of the simulation.

Forwarding too little to neighbor. A node might receive none or very little messages from a specific neighbor. This could be due to a neighbor very unhealthy or a targeted attack to slow down and exclude the node from participating. Exactly this attack is pictured in Figure 17c. At second 40 `node[7]` slows down its forwarding rate to `node[2]` to 10 MPS which is shortly after detected by `node[2]`. The connection is dropped and `node[2]` can peer with another node.

7 RELATED WORK

Flow control in the permissionless DLT setting builds on many topics from networking research such as P2P systems, network security, and distributed flow control. However, to the best of our knowledge Healthor is the first system to systematically combine a health-based, heterogeneity-aware distributed flow-control protocol in permissionless DLTs.

Congestion and flow control in traditional networks and data centers is a deeply studied topic [1, 4, 11, 31, 47]. Generally, mostly end-to-end communication settings rather than group communication is considered. However, with Healthor we focus on application-level message flow control in a group communication setting. Congestion control protocols employ specific signals such as delay measurements or packet loss to detect congestion. Some of which require switch support [1], NIC support [47], or are software-based [4, 11, 31]. In Healthor we use explicit health messages from neighbors as a flow-control signal without needing any hardware assistance. Its architecture converting health to per-neighbor forwarding *rates* is inspired by TIMELY [47] (in contrast to window-based TCP variants [1, 31]).

P2P systems and specifically content distribution systems such as Gnutella, KaZaA and BitTorrent were studied widely in the early 2000s [3, 14, 39]. Similar to modern DLTs, these systems are highly heterogeneous [55] and faced challenges with scalability and performance [3]. In [13, 41] Gia, a scaling approach for the Gnutella network, is proposed that leverages heterogeneity. It dynamically changes the network topology and puts high capacity nodes within short reach of most nodes. Additionally, an active flow-control mechanism based on available capacity is used to avoid overloaded hot-spots. Gia is similar to Healthor in spirit as both mechanisms leverage heterogeneity to increase utilization. However, the way Gia is designed is unsuitable for permissionless DLTs as topology changes can lead to cliques construction of powerful nodes, thus defeating the aim of decentralization in DLTs. Also, in Gnutella it is not essential for nodes to retrieve every search query whereas in DLTs state updates need to be propagated to every node eventually.

More recent research on DLTs [5, 6] focuses more on the aspects of consensus [30, 59], decentralization [27, 37, 43], and scalability [15], the networking aspect of DLTs has received only little attention [19]. As such, to the best of our knowledge there is no flow-control mechanism for DLTs similar to Healthor.

Scalability research in DLTs is mainly concerned with scaling consensus instead of the network layer [68]. There are some proposals to improve transaction and block dissemination such as Kadcast [53] and Erlay [49]. However, heterogeneity is often not considered at all and merely considered a by product caused by performance variability of cloud providers [23, 40, 65]. With Healthor we provide a solution to this emerging problem. As an alternative to permissionless public DLTs, there are also permissioned DLT designs [2] that can potentially give more control over participants and their network activities.

8 CONCLUSION AND ON-GOING WORK

In this paper we have presented Healthor, one of the first distributed flow-control mechanisms to leverage node heterogeneity to dynamically improve performance and decentralization in a permissionless DLT network.

Healthor achieves this by rate controlling message forwarding rates based on a node's health. Health is defined as a function of a node's message inbox occupancy. With extensive simulation of Healthor on DLT networks of up to 5,000 nodes, we have shown that this simple health-based signal can increase the degree of decentralization by 78%, improves throughput by 23%, and decreases 95 percentile message latency by 4×.

As next steps we are specifically investigating how to design a self-stabilizing mechanism for Healthor (without a network target rate) to converge to an optimal equilibrium based on the current network load and health of network participants. Furthermore, like early P2P designs, right incentives plays an important role for protocol adherence in any open network. Currently, Healthor does not provide any hardened incentives for nodes to follow the protocol except risking being dropped by a neighbor. One promising design is to incorporate Healthor's neighbor-assisting behavior into a DLT's reputation system incentives. Another interesting research direction is the detailed analysis of a colluding, malicious majority and its effects on the Healthor protocol.

Healthor's source code (https://github.com/jonastheis/healthor) and simulation result data set (https://zenodo.org/record/4573698) are both publicly available.

REFERENCES

[1] Mohammad Alizadeh, Albert Greenberg, David A. Maltz, Jitendra Padhye, Parveen Patel, Balaji Prabhakar, Sudipta Sengupta, and Murari Sridharan. 2010. Data Center TCP (DCTCP). In *Proceedings of the ACM SIGCOMM 2010 Conference (SIGCOMM '10)*. Association for Computing Machinery, New Delhi, India, 63–74. https://doi.org/10.1145/1851182.1851192

[2] Elli Androulaki, Artem Barger, Vita Bortnikov, Christian Cachin, Konstantinos Christidis, Angelo De Caro, David Enyeart, Christopher Ferris, Gennady Laventman, Yacov Manevich, Srinivasan Muralidharan, Chet Murthy, Binh Nguyen, Manish Sethi, Gari Singh, Keith Smith, Alessandro Sorniotti, Chrysoula Stathakopoulou, Marko Vukolić, Sharon Weed Cocco, and Jason Yellick. 2018. Hyperledger Fabric: A Distributed Operating System for Permissioned Blockchains. In *Proceedings of the Thirteenth EuroSys Conference (EuroSys '18)*. Association for Computing Machinery, New York, NY, USA, Article 30. https://doi.org/10.1145/3190508.3190538

[3] Stephanos Androutsellis-Theotokis and Diomidis Spinellis. 2004. A Survey of Peer-to-Peer Content Distribution Technologies. *Comput. Surveys* 36, 4 (Dec. 2004), 335–371. https://doi.org/10.1145/1041680.1041681

[4] Venkat Arun and Hari Balakrishnan. 2018. Copa: Practical Delay-Based Congestion Control for the Internet. In *Proceedings of the Applied Networking Research Workshop*. ACM, Montreal QC Canada, 19–19. https://doi.org/10.1145/3232755.3232783

[5] Badr Bellaj, Aafaf Ouaddah, Emmanuel Bertin, Noel Crespi, and Abdellatif Mezrioui. 2022. SOK: a comprehensive survey on distributed ledger technologies. In *ICBC 2022 : IEEE International Conference on Blockchain and Cryptocurrency*. IEEE, Shanghai, China, 1–16. https://hal.archives-ouvertes.fr/hal-03609651

[6] Marianna Belotti, Nikola Bozic, Guy Pujolle, and Stefano Secci. 2019. A Vademecum on Blockchain Technologies: When, Which, and How. *IEEE Communications Surveys & Tutorials* 21, 4 (2019), 3796–3838. https://doi.org/10.1109/COMST.2019.2928178

[7] Federico Matteo Bencic and Ivana Podnar Zarko. 2018. Distributed Ledger Technology: Blockchain Compared to Directed Acyclic Graph. In *2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS)*. IEEE, Vienna, 1569–1570. https://doi.org/10.1109/ICDCS.2018.00171

[8] Juan Benet. 2014. IPFS - Content Addressed, Versioned, P2P File System. *arXiv:1407.3561 [cs]* (July 2014). arXiv:1407.3561 [cs] http://arxiv.org/abs/1407.3561

[9] bitfly gmbh. 2020. Clients - Ethernodes.Org - The Ethereum Network & Node Explorer. https://ethernodes.org/

[10] Christian Cachin and Marko Vukolić. 2017. Blockchain Consensus Protocols in the Wild. *arXiv:1707.01873 [cs]* (July 2017). arXiv:1707.01873 [cs] http://arxiv.org/abs/1707.01873

[11] Neal Cardwell, Yuchung Cheng, C. Stephen Gunn, Soheil Hassas Yeganeh, and Van Jacobson. 2017. BBR: Congestion-Based Congestion Control. *Commun. ACM* 60, 2 (Jan. 2017), 58–66. https://doi.org/10.1145/3009824

[12] David Chaum, Amos Fiat, and Moni Naor. 1990. Untraceable Electronic Cash. In *Advances in Cryptology — CRYPTO' 88 (Lecture Notes in Computer Science)*, Shafi Goldwasser (Ed.). Springer, New York, NY, 319–327. https://doi.org/10.1007/0-387-34799-2_25

[13] Yatin Chawathe, Sylvia Ratnasamy, Lee Breslau, Nick Lanham, and Scott Shenker. 2003. Making Gnutella-like P2P Systems Scalable. In *Proceedings of the 2003 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM '03)*. Association for Computing Machinery, New York, NY, USA, 407–418. https://doi.org/10.1145/863955.864000

[14] Bram Cohen. 2003. Incentives Build Robustness in BitTorrent. In *Workshop on Economics of Peer-to-Peer Systems*, Vol. 6. Berkeley, CA, USA, 68–72.

[15] 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: (A Position Paper). In *Financial Cryptography and Data Security*, Jeremy Clark, Sarah Meiklejohn, Peter Y.A. Ryan, Dan Wallach, Michael Brenner, and Kurt Rohloff

(Eds.). Vol. 9604. Springer Berlin Heidelberg, Berlin, Heidelberg, 106–125. https://doi.org/10.1007/978-3-662-53357-4_8

[16] Scott A Crosby and Dan S Wallach. 2007. An Analysis of BitTorrent's Two Kademlia-Based DHTs. (2007), 29. https://hdl.handle.net/1911/96357

[17] Andrew Cullen, Pietro Ferraro, William Sanders, Luigi Vigneri, and Robert Shorten. 2020. On Congestion Control for Distributed Ledgers in Adversarial IoT Networks. *arXiv:2005.07778 [cs]* (May 2020). arXiv:2005.07778 [cs] http://arxiv.org/abs/2005.07778

[18] W. Diffie and M. Hellman. 1976. New Directions in Cryptography. *IEEE Transactions on Information Theory* 22, 6 (Nov. 1976), 644–654. https://doi.org/10.1109/TIT.1976.1055638

[19] Maya Dotan, Yvonne-Anne Pignolet, Stefan Schmid, Saar Tochner, and Aviv Zohar. 2022. Survey on Blockchain Networking: Context, State-of-the-Art, Challenges. *Comput. Surveys* 54, 5 (June 2022), 1–34. https://doi.org/10.1145/3453161

[20] Ronald P. Doyle, Jeffrey S. Chase, Omer M. Asad, Wei Jin, and Amin M. Vahdat. 2003. Model-Based Resource Provisioning in a Web Service Utility. In *Proceedings of the 4th Conference on USENIX Symposium on Internet Technologies and Systems - Volume 4 (USITS'03)*. USENIX Association, USA, 5.

[21] Cynthia Dwork and Moni Naor. 1993. Pricing via Processing or Combatting Junk Mail. In *Advances in Cryptology — CRYPTO' 92 (Lecture Notes in Computer Science)*, Ernest F. Brickell (Ed.). Springer, Berlin, Heidelberg, 139–147. https://doi.org/10.1007/3-540-48071-4_10

[22] Nabil El Ioini and Claus Pahl. 2018. A Review of Distributed Ledger Technologies. In *On the Move to Meaningful Internet Systems. OTM 2018 Conferences*, Hervé Panetto, Christophe Debruyne, Henderik A. Proper, Claudio Agostino Ardagna, Dumitru Roman, and Robert Meersman (Eds.). Vol. 11230. Springer International Publishing, Cham, 277–288. https://doi.org/10.1007/978-3-030-02671-4_16

[23] Jamie Ericson, Masoud Mohammadian, and Fabiana Santana. 2017. Analysis of Performance Variability in Public Cloud Computing. In *2017 IEEE International Conference on Information Reuse and Integration (IRI)*. IEEE, San Diego, CA, 308–314. https://doi.org/10.1109/IRI.2017.47

[24] Ethereum. 2020. Ethereum/Devp2p. https://github.com/ethereum/devp2p

[25] Ittay Eyal, Adem Efe Gencer, Emin Gün Sirer, and Robbert van Renesse. 2016. Bitcoin-Ng: A Scalable Blockchain Protocol. In *USENIX NSDI*. 45–59.

[26] Matthias Fitzi, Peter Gazi, Aggelos Kiayias, and Alexander Russell. 2020. Proof-of-Stake Blockchain Protocols with near-Optimal Throughput. *IACR Cryptol. ePrint Arch.* 2020 (2020), 37.

[27] Adem Efe Gencer, Soumya Basu, Ittay Eyal, Robbert van Renesse, and Emin Gün Sirer. 2018. Decentralization in Bitcoin and Ethereum Networks. In *Financial Cryptography and Data Security*, Sarah Meiklejohn and Kazue Sako (Eds.). Vol. 10957. Springer Berlin Heidelberg, Berlin, Heidelberg, 439–457. https://doi.org/10.1007/978-3-662-58387-6_24

[28] Arthur Gervais, Ghassan O. Karame, Karl Wüst, Vasileios Glykantzis, Hubert Ritzdorf, and Srdjan Capkun. 2016. On the Security and Performance of Proof of Work Blockchains. In *ACM CCS*. 3–16. https://doi.org/10.1145/2976749.2978341

[29] Yossi Gilad, Rotem Hemo, Silvio Micali, Georgios Vlachos, and Nickolai Zeldovich. 2017. Algorand: Scaling Byzantine Agreements for Cryptocurrencies. In *ACM SOSP*. 51–68. https://doi.org/10.1145/3132747.3132757

[30] Huaqun Guo and Xingjie Yu. 2022. A Survey on Blockchain Technology and its security. *Blockchain: Research and Applications* (Feb. 2022), 100067. https://doi.org/10.1016/j.bcra.2022.100067

[31] Sangtae Ha, Injong Rhee, and Lisong Xu. 2008. CUBIC: A New TCP-Friendly High-Speed TCP Variant. *ACM SIGOPS Operating Systems Review* 42, 5 (July 2008), 64–74. https://doi.org/10.1145/1400097.1400105

[32] Stuart Haber and W. Scott Stornetta. 1991. How to Time-Stamp a Digital Document. In *Advances in Cryptology-CRYPTO' 90 (Lecture Notes in Computer Science)*, Alfred J. Menezes and Scott A. Vanstone (Eds.). Springer, Berlin, Heidelberg, 437–455. https://doi.org/10.1007/3-540-38424-3_32

[33] Vikas Hassija, Gaurang Bansal, Vinay Chamola, Neeraj Kumar, and Mohsen Guizani. 2020. Secure Lending: Blockchain and Prospect Theory-Based Decentralized Credit Scoring Model. *IEEE Transactions on Network Science and Engineering* 7, 4 (2020), 2566–2575. https://doi.org/10.1109/TNSE.2020.2982488

[34] C. Huang, Z. Wang, H. Chen, Q. Hu, Q. Zhang, W. Wang, and X. Guan. 2020. RepChain: A Reputation Based Secure, Fast and High Incentive Blockchain System via Sharding. *IEEE Internet of Things Journal* (2020), 1–1. https://doi.org/10.1109/JIOT.2020.3028449

[35] ITU-T Focus Group on Application of Distributed Ledger Technology (FG DLT). [n.d.]. Distributed Ledger Technology Use Cases. ([n.d.]). https://www.itu.int/en/ITU-T/focusgroups/dlt/Documents/d21.pdf

[36] Niclas Kannengießer, Sebastian Lins, Tobias Dehling, and Ali Sunyaev. 2020. Trade-Offs between Distributed Ledger Technology Characteristics. *Comput. Surveys* 53, 2 (July 2020), 1–37. https://doi.org/10.1145/3379463

[37] Seoung Kyun Kim, Zane Ma, Siddharth Murali, Joshua Mason, Andrew Miller, and Michael Bailey. 2018. Measuring Ethereum Network Peers. In *Proceedings of the Internet Measurement Conference 2018*. ACM, Boston MA USA, 91–104. https://doi.org/10.1145/3278532.3278542

[38] Ana Klimovic, Heiner Litz, and Christos Kozyrakis. 2018. Selecta: Heterogeneous Cloud Storage Configuration for Data Analytics. In *Proceedings of the 2018 USENIX Conference on Usenix Annual Technical Conference (USENIX ATC '18)*. USENIX Association, USA, 759–773.

[39] N. Leibowitz, M. Ripeanu, and A. Wierzbicki. 2003. Deconstructing the Kazaa Network. In *Proceedings the Third IEEE Workshop on Internet Applications. WIAPP 2003*. IEEE Comput. Soc, San Jose, CA, USA, 112–120. https://doi.org/10.1109/WIAPP.2003.1210295

[40] Ang Li, Xiaowei Yang, Srikanth Kandula, and Ming Zhang. 2010. CloudCmp: Comparing Public Cloud Providers. In *Proceedings of the 10th Annual Conference on Internet Measurement - IMC '10*. ACM Press, Melbourne, Australia, 1. https://doi.org/10.1145/1879141.1879143

[41] Qin Lv, Sylvia Ratnasamy, and Scott Shenker. 2002. Can Heterogeneity Make Gnutella Scalable? In *Peer-to-Peer Systems*, Gerhard Goos, Juris Hartmanis, Jan van Leeuwen, Peter Druschel, Frans Kaashoek, and Antony Rowstron (Eds.). Vol. 2429. Springer Berlin Heidelberg, Berlin, Heidelberg, 94–103. https://doi.org/10.1007/3-540-45748-8_9
[42] Ashraf Mahgoub, Alexander Michaelson Medoff, Rakesh Kumar, Subrata Mitra, Ana Klimovic, Somali Chaterji, and Saurabh Bagchi. 2020. OPTIMUSCLOUD: Heterogeneous Configuration Optimization for Distributed Databases in the Cloud. In *2020 USENIX Annual Technical Conference (USENIX ATC 20)*. USENIX Association, 189–203. https://www.usenix.org/conference/atc20/presentation/mahgoub
[43] Sami Ben Mariem, Pedro Casas, Matteo Romiti, Benoit Donnet, Rainer Stutz, and Bernhard Haslhofer. 2020. All That Glitters Is Not Bitcoin – Unveiling the Centralized Nature of the BTC (IP) Network. In *NOMS 2020 - 2020 IEEE/IFIP Network Operations and Management Symposium*. IEEE, Budapest, Hungary, 1–9. https://doi.org/10.1109/NOMS47738.2020.9110354
[44] Juri Mattila. 2016. The Blockchain Phenomenon. *Berkeley Roundtable of the International Economy* (2016), 16.
[45] Petar Maymounkov and David Mazières. 2002. Kademlia: A Peer-to-Peer Information System Based on the XOR Metric. In *Peer-to-Peer Systems (Lecture Notes in Computer Science)*, Peter Druschel, Frans Kaashoek, and Antony Rowstron (Eds.). Springer, Berlin, Heidelberg, 53–65. https://doi.org/10.1007/3-540-45748-8_5
[46] Ralph C. Merkle. 1978. Secure Communications over Insecure Channels. *Commun. ACM* 21, 4 (April 1978), 294–299. https://doi.org/10.1145/359460.359473
[47] Radhika Mittal, David Zats, Vinh The Lam, Nandita Dukkipati, Emily Blem, Hassan Wassel, Monia Ghobadi, Amin Vahdat, Yaogong Wang, and David Wetherall. 2015. TIMELY: RTT-Based Congestion Control for the Datacenter. In *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication - SIGCOMM '15*. ACM Press, London, United Kingdom, 537–550. https://doi.org/10.1145/2785956.2787510
[48] Satoshi Nakamoto. 2009. Bitcoin: A Peer-to-Peer Electronic Cash System. (2009), 9.
[49] Gleb Naumenko, Gregory Maxwell, Pieter Wuille, Alexandra Fedorova, and Ivan Beschastnikh. 2019. Erlay: Efficient Transaction Relay for Bitcoin. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security (CCS '19)*. Association for Computing Machinery, New York, NY, USA, 817–831. https://doi.org/10.1145/3319535.3354237
[50] S. Popov. 2015. The Tangle. https://assets.ctfassets.net/r1dr6vzfxhev/2t4uxvsIqk0EUau6g2sw0g/45eae33637ca92f85dd9f4a3a218e1ec/iota1_4_3.pdf
[51] Serguei Popov, Hans Moog, Darcy Camargo, Angelo Capossele, Vassil Dimitrov, Alon Gal, Andrew Greve, Bartosz Kusmierz, Sebastian Mueller, Andreas Penzkofer, et al. 2020. The Coordicide. (2020). https://files.iota.org/papers/20200120_Coordicide_WP.pdf
[52] Chenhao Qu, Rodrigo N. Calheiros, and Rajkumar Buyya. 2016. A Reliable and Cost-Efficient Auto-Scaling System for Web Applications Using Heterogeneous Spot Instances. *Journal of Network and Computer Applications* 65, C (April 2016), 167–180. https://doi.org/10.1016/j.jnca.2016.03.001
[53] Elias Rohrer and Florian Tschorsch. 2019. Kadcast: A Structured Approach to Broadcast in Blockchain Networks. In *Proceedings of the 1st ACM Conference on Advances in Financial Technologies*. ACM, Zurich Switzerland, 199–213. https://doi.org/10.1145/3318041.3355469
[54] Fahad Saleh. 2020. *Blockchain Without Waste: Proof-of-Stake*. SSRN Scholarly Paper ID 3183935. Social Science Research Network, Rochester, NY. https://doi.org/10.2139/ssrn.3183935
[55] Stefan Saroiu, P Krishna Gummadi, and Steven D Gribble. 2001. Measurement Study of Peer-to-Peer File Sharing Systems. In *Multimedia Computing and Networking 2002*, Vol. 4673. 156–170.
[56] Devavrat Shah. 2007. Gossip Algorithms. *Foundations and Trends® in Networking* 3, 1 (2007), 1–125. https://doi.org/10.1561/1300000014
[57] Michael Bedford Taylor. 2017. The Evolution of Bitcoin Hardware. *Computer* 50, 9 (2017), 58–66. https://doi.org/10.1109/MC.2017.3571056
[58] thetangle.org. 2020. Public IOTA Nodes. https://thetangle.org
[59] Florian Tschorsch and Bjorn Scheuermann. 23. Bitcoin and Beyond: A Technical Survey on Decentralized Digital Currencies. *IEEE Communications Surveys & Tutorials* 18, 3 (23), 2084–2123. https://doi.org/10.1109/COMST.2016.2535718
[60] Alexandru Uta, Alexandru Custura, Dmitry Duplyakin, Ivo Jimenez, Jan Rellermeyer, Carlos Maltzahn, Robert Ricci, and Alexandru Iosup. 2020. Is Big Data Performance Reproducible in Modern Cloud Networks?. In *17th USENIX Symposium on Networked Systems Design and Implementation (NSDI 20)*. USENIX Association, Santa Clara, CA, 513–527. https://www.usenix.org/conference/nsdi20/presentation/uta
[61] Roger Wattenhofer. 2017. *Distributed Ledger Technology: The Science of the Blockchain* (second ed.). CreateSpace Independent Publishing Platform, North Charleston, SC, USA.
[62] Shawn Wilkinson, Tome Boshevski, Josh Brandoff, and Vitalik Buterin. 2014. *Storj A Peer-to-Peer Cloud Storage Network*. https://storj.io/storj.pdf
[63] Gavin Wood. 2016. Polkadot: Vision for a Heterogeneous Multi-Chain Framework. (2016). https://pdfs.semanticscholar.org/12ea/67d037ddd0a5c7beb40fbd4a70bc90631644.pdf
[64] Y. Xiao, N. Zhang, W. Lou, and Y. T. Hou. Secondquarter 2020. A Survey of Distributed Consensus Protocols for Blockchain Networks. *IEEE Communications Surveys Tutorials* 22, 2 (Secondquarter 2020), 1432–1465. https://doi.org/10.1109/COMST.2020.2969706
[65] Yunjing Xu, Zachary Musgrave, Brian Noble, and Michael Bailey. 2013. Bobtail: Avoiding Long Tails in the Cloud. In *10th USENIX Symposium on Networked Systems Design and Implementation (NSDI 13)*. USENIX Association, Lombard, IL, 329–341. https://www.usenix.org/conference/nsdi13/technical-sessions/presentation/xu_yunjing

[66] Addy Yeow. 2020. Global Bitcoin Nodes Distribution. https://bitnodes.io/
[67] Mahdi Zamani, Mahnush Movahedi, and Mariana Raykova. 2018. RapidChain: Scaling Blockchain via Full Sharding. In *ACM CCS*. 931–948. https://doi.org/10.1145/3243734.3243853
[68] Qiheng Zhou, Huawei Huang, Zibin Zheng, and Jing Bian. 2020. Solutions to Scalability of Blockchain: A Survey. *IEEE Access* 8 (2020), 16440–16455. https://doi.org/10.1109/ACCESS.2020.2967218