Healthor: Protecting the Weak in Heterogeneous DLTs with Health-aware Flow Control

Jonas Theis[1,2], Luigi Vigneri[2], Lin Wang[1], and Animesh Trivedi[1]
[1]Vrije Universiteit Amsterdam and [2]IOTA Foundation

Abstract

Permissionless distributed ledger technologies (DLTs) utilize an underlying peer-to-peer network to disseminate transactions. These types of networks have been shown to be highly heterogeneous. However, current DLTs fail to consider this heterogeneity which can render low-end nodes to be unable to participate in consensus.

In this paper we introduce Healthor, a novel heterogeneity-aware flow-control mechanism that formalizes this heterogeneity in a notion of health of a node. In our early simulation results we show that Healthor enables high-end nodes to protect their weaker neighbors by buffering at their end while incurring only minimal overhead.

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

Keywords heterogeneity-aware flow-control, distributed ledger technologies, peer-to-peer overlay network

ACM Reference Format:
Jonas Theis[1,2], Luigi Vigneri[2], Lin Wang[1], and Animesh Trivedi[1]. 2020. Healthor: Protecting the Weak in Heterogeneous DLTs with Health-aware Flow Control. In *Workshop on Scalable and Resilient Infrastructures for Distributed Ledgers (SERIAL '20), December 7–11, 2020, Delft, Netherlands.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3429884.3430033

1 Introduction

First generation permissionless distributed ledger technologies (DLTs) such as Bitcoin-like blockchains reach consensus through expensive proof-of-work which leads to hashing power centralization in a few miners [7, 10]. This has consequences for decentralization, e.g., in terms of censorship resistance and device heterogeneity due to more efficient mining with specialized hardware. Alternative means to reach consensus in DLTs like proof-of-stake, Byzantine fault tolerance algorithms, and voting have since developed. Arguably, voting fosters a very heterogeneous and decentralized network because it allows each participant to take part in consensus without constraints on computing power or stake. More recent DLTs such as IOTA [8], Nano [1], HashGraph [4], etc., take a voting-based approach.

Every consensus mechanism has its own challenges. In the case of voting-based consensus the main challenges revolve around fair network usage and Sybil protection [1, 8]. Enabling fair network usage becomes a very challenging task due to the *heterogeneity* of nodes in a DLT and its underlying peer-to-peer (P2P) overlay network. Heterogeneity can be manifold: (1) bandwidth, latency, availability, and processing capabilities can vary between multiple orders of magnitude in P2P networks [9]; (2) protocols, geographical location, node freshness, and software versions can differ widely in DLTs [5]; (3) unsteady processing rates of a node due to competing tenant applications. It is a result of the very nature of permissionless P2P systems: anyone with a computing device can join. Such heterogeneity was a topic of active research with P2P content distribution systems such as Gnutella [2, 9]. However, DLTs have additional requirements. Every node needs to receive all ledger state updates (transactions) in a timely manner to be able to construct and verify the ledger state. If the network operates too fast without any flow control, only high-end nodes are able to keep up with the network activity. However, if every node should be able to maintain synced the network effectively can only operate at the speed of the slowest low-end node. It is important to find a middle ground between these two extremes so that every participant receives and processes all state updates in a timely manner, and is thus able to participate in consensus. In turn, the overall throughput should be high while maintaining a high degree of decentralization and heterogeneity.

To illustrate and quantify the impact of heterogeneity (when ignored) in DLTs, we run a simple experiment. In this experiment we consider a 3-node simplified network, where each node represents a transaction processing and voting node with a queue for pending transactions. There is a constant transaction influx at the rate of 1000 transactions per second (TPS) until second 60 and 0 TPS from 60-90 seconds in the network. We vary processing transaction rates of `node[2]`, and measure the queue size on each node to quantify the impact of heterogeneity on transaction processing. Figure 1 shows our results where there is a clear correlation between a processing rate lower than influx of transactions

SERIAL '20, December 7–11, 2020, Delft, Netherlands

ACM ISBN 978-1-4503-8208-3/20/12…$15.00
https://doi.org/10.1145/3429884.3430033

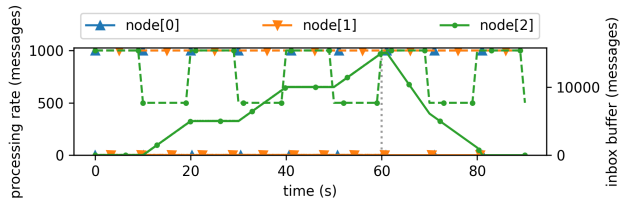

Figure 1. Correlation between growing queue size and varying processing rate is clearly visible when transaction influx is higher than processing rate. 1000 TPS ≤ 60s, 0 TPS 61-90s. A dashed line indicates a node's processing rate.

and growing queue length on a node. As the inbox queue builds up, it has detrimental effect on the performance. At time 10s there are 0 unprocessed transactions after 5 seconds of their issuing time whereas at time 60s more than 10000 transactions are not processed by `node[2]` after 5 seconds. This experiment shows that `node[2]` cannot keep up with the network activity, and therefore its local ledger has an inconsistent state. As a result, this node is not able to participate in the voting process, thus, increasing centralization in the DLT network. In summary, there is a clear need to address the heterogeneity challenges to maintain the egalitarian nature of a DLT network.

In this paper we present Healthor[1], a novel heterogeneity-aware, flow-control mechanism for DLT networks. Healthor captures heterogeneity by defining *health* of a node as function of the incoming queue length in relation to the network's target processing rate. Such queue-based indicators are also used in data centers for distributed congestion control mechanisms [6]. This health-awareness enables high-end nodes to *protect* weaker nodes from wasting unnecessary processing power, rapidly changing network load and bursts so that every node can stay in sync and participate in consensus.

We make following key contributions: (1) We make a case for heterogeneity-aware network management (flow-control) in DLT networks. (2) We design a novel health-based flow-control mechanism for DLT networks that leverages heterogeneity. (3) We present early simulation results showing the efficacy of the mechanism while incurring only minimal overhead.

2 Network model

The set of all nodes participating in the network is denoted as $\mathcal{M}$ where each node $m \in \mathcal{M}$ has a set of neighbors $\mathcal{N}_m \subset \mathcal{M}$. Neighbors are connected via bidirectional channels over which they exchange *messages*. A message can contain any data, e.g., monetary transactions. We use throughput and latency as main metrics to evaluate performance of nodes and network overall. Both are based on processing time, i.e., the time a node writes a message into its local ledger.

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

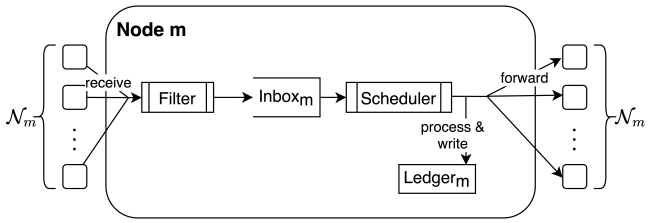

Figure 2. Model of a node m and its neighbors $\mathcal{N}_m$.

Figure 2 shows a node m's model and its neighbors $\mathcal{N}_m$. A node can perform different operations, namely issuing, processing, receiving and forwarding messages. The network's target processing rate ν defines the overall rate at which the network as a whole should operate. We assume that nodes efficiently exploit network resources using an underlying TCP-inspired congestion control mechanism [3]. In particular, depending on the congestion status of the network and a node's reputation, node m will be able to generate messages at rate λ_m.

On receipt of a message, a node filters duplicated messages out and pushes unseen messages to its inbox. The congestion control mechanism will schedule messages to be processed and forwarded from the inbox at fixed rate ν_m. Ideally, $\nu_m = \nu$ at any time, meaning that a node m is able to operate at the rate of the network. However, sometimes ν_m might be smaller than ν, which leads to accumulating messages at the inbox. This can happen because processing and writing messages to the local ledger is a costly operation due to cryptographic signature verification, and IO operations. Thus, we model ν_m to express heterogeneity between nodes and available processing capabilities with varying rates. Furthermore, we assume that filtering and forwarding messages have negligible cost. A message is forwarded to all neighbors except the one that sent the message.

Adversaries may exceed the allowed generation rate λ_m and the processing rate ν_m. The analysis of such malicious behavior, however, is left as a future work.

3 Design

In this section, we present the main components of Healthor, our proposed heterogeneity-aware flow-control mechanism. At high-level, Healthor works as follows: Nodes periodically send their health, a notion of their current fitness, to their neighbors. A node adjusts its sending rates individually according to a neighbor's health and buffers messages for neighbors that are less healthy. Our key insight is that more capable nodes *store messages for weak neighbors*, thus shifting load away from low-end nodes (Section 4).

Healthor operates at the application-level and maintains connections to multiple neighbors in a group communication setup. It is inefficient to rely on existing mechanisms such as TCP-based backpressure because (1) TCP-based backpressure runs independently from the application-level resources, and would need modifications to the TCP implementation

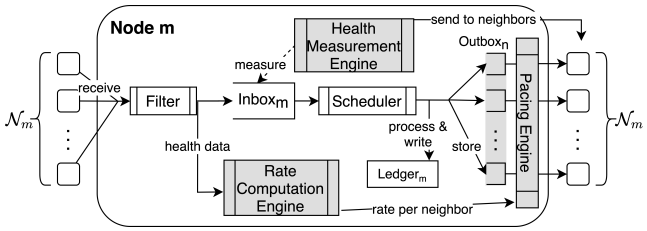

Figure 3. Model of a node m with our Healthor. New components compared to Figure 2 are highlighted in gray.

which is not feasible in a public DLT; and (2) TCP as a point-to-point protocol cannot efficiently handle group communication dynamics.

Figure 3 depicts an updated model of a node m with the new components highlighted. Healthor is a framework encompassing three separate engines, inspired by TIMELY [6], that interact together and form a heterogeneity-aware flow-control mechanism. Additionally, a node m has an $Outbox_n$ for every $n \in \mathcal{N}_m$ where references to messages for a neighbor n are stored before forwarding.

3.1 Health Measurement Engine

The *Health Measurement Engine* is at the heart of the mechanism. By introducing the notion of health $h_m \in [0, 1]$, a node m can express its fitness regarding current network traffic taking into account its specific computational resources available at this point in time. A node periodically calculates its health based on inbox occupation $len(Inbox_m) \stackrel{\text{def}}{=} l_m$ as shown in

$$h_m = \begin{cases} 1 & \text{if } l_m < \nu, \\ 1 - \frac{l_m}{\alpha\nu} & \text{if } l_m \geq \nu \wedge l_m \leq \alpha\nu \\ 0 & \text{if } l_m > \alpha\nu \end{cases} \quad (1)$$

where α is a tunable parameter, and sends the information to its neighbors.

Since all messages are delivered to all nodes, every node knows the current congestion level of the network. Therefore, inbox occupation gives a reasonable assessment of *how much in sync a node is*, i.e., whether it is able to receive and process state updates in a consistent and timely manner. A lower inbox occupation signals a node being able to process at the network's pace, being healthy. Conversely, a higher inbox occupation conveys that the node is struggling to keep up with network activity. Thus, it is unhealthy. Parameter α is tunable and allows to adjust for how quickly a node considers itself getting unhealthier.

3.2 Rate Computation Engine

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

$$r_n = \nu \cdot h_n. \quad (2)$$

It reduces the sending rate r_n linearly according to the neighbor's health h_n.

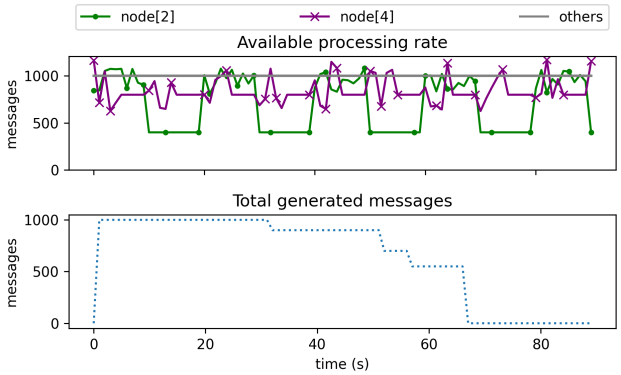

Figure 4. Available processing rates and generation rate.

3.3 Pacing Engine

The *Pacing Engine* gets 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 sending 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 Evaluation

We built a simulator using OMNeT++ to test our mechanism[2] and simulate a permissionless, voting-based DLT. We compare Healthor (Section 3) with the baseline scenario described in Section 2. The test network is made up of 10 nodes, each with 4 random neighbors. Channels have a random delay between 50ms and 150ms to simulate real network conditions. Figure 4 depicts available processing and message generation rates. The main point of interest here: `node[2]` and `node[4]` have a varying processing rate, different than that of the other nodes. This could be due to other tenant applications requiring too much CPU time or naturally because of heterogeneity between nodes. The total amount of generated messages at any point in time is less than or equal to $\nu = 1000$. According to our experiments, the parameter α seems to be at a sweet spot between growing inbox at an unhealthy node and outbox length at its neighbors at $\alpha = 10$, which we adopt for our experiments shown here. Every node's *Health Measurement Engine* computes its health h_m and sends it to its neighbors at an interval of 1 second. Since we do not consider malicious behavior and message priorities, a node's scheduler operates according to FIFO. We make use of unbounded buffers in our experiments to ease monitoring of system behavior.

Figure 5 shows the throughput over time. It is evident that all nodes with stable processing rates equal to ν can keep up with the influx of generated messages, without and with our

[2]The simulator and a framework to generate plots from Python can be found at https://github.com/jonastheis/healthor.

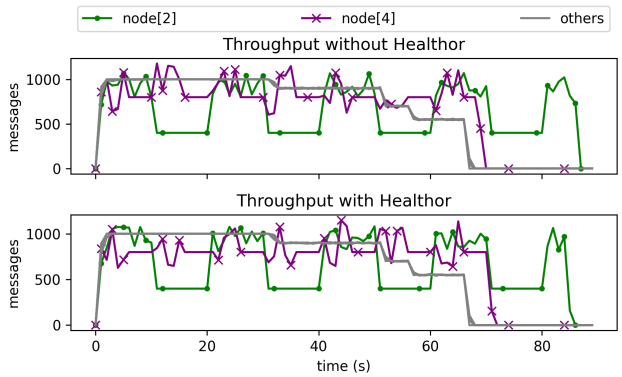

Figure 5. Throughput without and with Healthor.

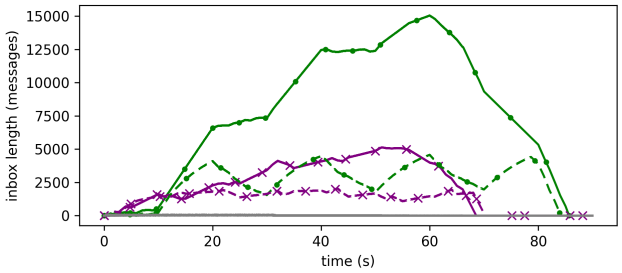

Figure 6. Inbox occupation over time without and with Healthor. Solid lines show occupation without Healthor; dashed lines with Healthor.

mechanism. Conversely, `node[2]` and `node[4]` can not always operate at ν and we can see a changing throughput. As a general trend it can be observed that the throughput in both cases is very similar. There seems to be a minimal penalty with our mechanism for unhealthy nodes which is due to the overhead of adding messages to an outbox and sending them via *Pacing Engine*. Latency (not depicted) paints a similar picture. Overall, with our mechanism latencies are minimally increased due to this overhead.

In Figure 6 inbox occupation is shown over time. For healthy nodes it is hovering around 0 without and with Healthor. For `node[2]` and `node[4]` it clearly can be observed that the occupation grows in times where there is less processing rate available than the network's target processing rate ν. Without Healthor the inbox of `node[2]` grows extremely large to up to 15000 messages because all its neighbors continue sending at rate ν. With our mechanism a different trend is visible: `node[2]`'s inbox only grows up to 4000 messages, also increases in times of less processing power but decreases in times of more processing power.

Recall that a growing inbox leads to a node becoming unhealthier as described in Section 3.1. Consecutively, its neighbors will slow down the sending rate accordingly. When the node has more processing rate available again, it starts draining its inbox; it becomes healthier and its neighbors speed up sending again. This balancing can be observed for both `node[2]` and `node[4]`. While the messages are not buffered at the unhealthy node they are buffered at its neighbors' outbox. These outboxes can grow up to 9500 messages. However, the message itself does not need to be stored but only pointers to already processed messages.

Ongoing work: From our experiments we can conclude that Healthor effectively shifts growing buffers from unhealthy nodes to their more capable neighbors while incurring only a minimal performance overhead. It is beneficial to build up queues at neighbors due to multiple reasons. First, there is less memory consumption on the low-end node that is already struggling to keep up to the network's pace. Second, in future versions of the mechanism this can be utilized to randomly drop messages at neighbors while still achieving high probability that the node receives all messages, and thus reducing the amount of messages to be received and filtered. Lastly, we have seen that the inbox of an unhealthy node can grow extremely large without our mechanism whereas this is mitigated with Healthor. If we consider bounded buffers, it becomes obvious that significant message drop would happen at an unhealthy node. We believe that in such scenarios a future version of Healthor can not only improve consistency but also latency and throughput. We are investigating Healthor with bounded buffers to verify our intuition.

5 Conclusion

Healthor is a novel heterogeneity-aware flow-control mechanism for DLT networks that captures heterogeneity by introducing health of a node. Therefore, more capable nodes can protect low-end nodes so that every node can continue to participate in consensus. In early simulation results we show that Healthor is capable of shifting memory load away from low-end nodes to their more capable neighbors without incurring significant overhead.

References

[1] Federico Matteo Bencic et al. 2018. Distributed Ledger Technology: Blockchain Compared to Directed Acyclic Graph. In *2018 IEEE 38th ICDCS*. 1569–1570.
[2] Yatin Chawathe et al. 2003. Making Gnutella-like P2P Systems Scalable. In *SIGCOMM '03*. New York, NY, USA, 407–418.
[3] Andrew Cullen et al. 2020. On Congestion Control for Distributed Ledgers in Adversarial IoT Networks. *ArXiv* abs/2005.07778 (2020).
[4] Nabil El Ioini et al. 2018. A Review of Distributed Ledger Technologies. In *On the Move to Meaningful Internet Systems. OTM 2018 Conferences*. Vol. 11230. Springer International Publishing, 277–288.
[5] Seoung Kyun Kim et al. 2018. Measuring Ethereum Network Peers. In *Proceedings of the ACM IMC 2018*. 91–104.
[6] Radhika Mittal et al. 2015. TIMELY: RTT-Based Congestion Control for the Datacenter. In *Proceedings of the 2015 ACM SIGCOMM*. 537–550.
[7] Satoshi Nakamoto. 2009. Bitcoin: A Peer-to-Peer Electronic Cash System. (2009).
[8] Serguei Popov et al. 2020. The Coordicide: Realizing IOTA's vision of a permissionless and scalable distributed ledger technology.
[9] Stefan Saroiu et al. 2001. Measurement Study of Peer-to-Peer File Sharing Systems. In *Multimedia Computing and Networking 2002*.
[10] Florian Tschorsch et al. 2016. Bitcoin and Beyond: A Technical Survey on Decentralized Digital Currencies. *IEEE Communications Surveys & Tutorials* 18, 3 (2016), 2084–2123.