# Gossip Layer Optimization for the IOTA Network

Gereon Mendler
*Dept. of Computer Science*
*University of Twente*
Enschede, The Netherlands
g.t.mendler@gmail.com

***Abstract*—With the goal of optimizing the gossip layer of the IOTA 2.0 network, we show that Push-Pull gossip offers a worthwhile tradeoff to the currently used flooding by substantially decreasing the network load. This comes at the cost of a predictable speed penalty, but the strong resiliency and reliability are retained. Further, we identify a strong relationship with the network topology, which defines the limits of gossip protocol potential but also embodies the mechanism by which its advantages can be fully exhausted and its weaknesses remedied. This optimization is motivated by the arbitrarily defined throughput limitation of the IOTA system, hence lower network overhead could facilitate correspondingly more transactions per second. For this purpose, we explored various network topologies within the bounds of the specifications and investigated the behavior and resiliency of the current system through event-based simulation of the P2P network with Peersim. Push-Pull gossip is identified as the most suitable protocol for this project, and good configurations emerge through a parameter sensitivity analysis enabled by the simulation. This configuration is then evaluated for resiliency and in conjunction with alternative topologies to identify recommendations with regard to the peering mechanism. By these means, we can propose a gossip protocol to replace flooding.**

***Index Terms*—IOTA, Distributed Ledger, Gossip, DAG, P2P Networks, Peering, Resilience, Simulation, Peersim**

## I. Introduction

Distributed Ledger Technologies (DLT) are logical constructs similar to append-only databases consisting of a set of transactional data that is geographically spread and replicated among multiple participating agents, forming a fully decentralized, coherent record of balances and transactions when combined. They can facilitate value transactions and contract enforcement between parties that lack trust in each other, without the need for a central administrator like a conventional bank. During the last few years, distributed ledgers have garnered much mainstream attention, partially due to the market value of the assets that they govern. These systems combine established concepts from previous research into peer-to-peer networks, distributed consensus, and cryptography and expand upon them.

Distributed Ledgers most commonly take the shape of Blockchains, which consist of a series of blocks that contain transactions. The state of the ledger can then be inferred by following the strictly ordered series of transactions, where available tokens come from yet unspent incoming transactions according to the UTXO model [23]. The blocks in the chain are cryptographically linked and thus the content of past blocks cannot be modified. The main challenge of these systems is determining and agreeing on the next block, as the chain can only be extended sequentially.

In practice, the network layer of a distributed ledger usually takes the form of a synchronized peer-to-peer communication network that facilitates the exchange of information between the parties that hold a replica of the ledger and constitute the nodes of this network. Proposed updates to the ledger in the form of transactions will be propagated to all nodes, who eventually form an agreement regarding the validity of the changes through a process called consensus. This structure requires that all participating nodes become aware of new updates, such that they have the same agreed-upon replica of the shared ledger. For this purpose, communication is usually accomplished using a propagation scheme such as flooding or gossiping.

IOTA is a distributed ledger based on a Directed Acyclic Graph (DAG) instead of a blockchain. New transactions are added to the graph individually in small blocks through attachment to other recent blocks acting as parent references. This way everyone can create transactions, wrap them in blocks to cryptographically proof immutability and placement in the DAG, and add them to the ledger themselves. Consequently, many different actors can update the shared ledger simultaneously without requiring a leader election as in blockchain networks. This way, DAG-based ledgers promise higher scalability and speed at the cost of the additional complexity of the consensus and Sybil protection measures. Currently, the research and development team behind IOTA is attempting to create the next major update to the protocol to replace the placeholder central coordinator that proved necessary to secure the network until the planned decentralized consensus was fully developed. This replacement is a novel distributed probabilistic consensus based on a reputation model using proof-of-stake, requiring that contributors stake their assets against the guarantee of their proper behavior [24]. This reputation, referred to as mana in the IOTA framework, determines the proportional, guaranteed share of system throughput available to each client. Both proof-of-stake and the DAG graph structure are relatively unexplored concepts in the confines of distributed ledgers, and their combination poses challenges that require intricate cooperation of the various protocol modules and layers to effectively deal with Sybil, Eclipse and DoS attacks [25] and ensure reliable consensus and fraud protection. Apart from addressing security concerns, the information dissemination protocol also influences the

operational seamlessness and usability of the network in terms of transaction throughput, network resilience, and finality time, which is the time required until a transaction becomes an immutable and thus reliable ledger entry.

Although the effective dissemination of messages through the network is a core capability of any DLT network, no efforts have yet been made to integrate a customized gossip-based protocol into the IOTA protocol stack. Instead, the current implementation follows a naive flooding approach that propagates duplicate messages without considering the network topology, node conditions, or other relevant contexts. Gossip protocols can improve upon this by mimicking the way diseases or rumors spread through a population using probabilistic means. Such protocols have proven themselves to be significantly more efficient than the naive solutions for all-to-all multicast transmissions, but need to be customized to the specific requirements of the system at hand. Furthermore, the ability to rapidly spread information through the network constitutes a risk in case of incorrect or malicious data. The IOTA network is mainly characterized by well-defined data allowing strong validation and verification, and traffic that can originate from every node, but to different degrees.

With this research, we aim to aid in the optimization of the gossip layer for the IOTA 2.0 protocol stack, with particular emphasis on the propagation speed of new blocks and the network load caused by them. Since the IOTA system has an arbitrarily defined throughput limitation, lower network overhead could facilitate correspondingly more transactions per second. The investigated problem can be broadly separated into three consecutive questions:

1) To what extent do IOTA system components and their parameters impact message propagation?
2) How can messages be best propagated through the network using gossiping?
3) Does gossiping achieve increased propagation efficiency at the cost of resilience against attacks and failures?

Gossip protocols have proven themselves to be extremely efficient in distributing information, but require customization to fit the system they are supposed to serve. Previous work shows numerous ways to improve specific performance metrics and characteristics of gossip in the DLT space. To this end, the relevant components of the system, and the degree of their dependence need to be explored using simulation. A gossip protocol can then be designed to adhere to these requirements and tuned to its environment. A direct comparison with flooding serves to explore possible shortcomings under varying conditions and unusual scenarios, which serve to judge the resiliency of the protocol proposal. Finally, we aim to make recommendations on the important characteristics of a gossip protocol in the context of IOTA and make a specific proposal fitting those needs.

Section II serves to summarize previous work about general and DLT-specific gossip protocols, followed by an explanation of the methodology in section III. The simulation setup and IOTA system components and their implementations are detailed in section IV. Then, variants of network topology are compared in section V after which the simulation results with flooding are discussed in section VI, providing the insights needed to construct the gossip protocol presented in section VII. Results about parameter sensitivity, varying loads, and resilience are shown in section VIII before conclusions are drawn in section IX.

## II. Background

### A. Basic Gossip

Gossip protocols used to encompass a very wide domain, but are now more commonly described as being run in a regular, periodic, relatively lazy, symmetric, and decentralized manner [12]. In essence, they are convergently consistent, meaning that they propagate any new information to all nodes that will be affected by the information within time logarithmic in the size of the system. There are two prevailing purposes of gossip protocols, aggregation, and information dissemination. Here we are only interested in the latter one, which can be achieved broadly by either periodically pulling updates from peers (anti-entropy) or pushing information as new events occur (rumor-mongering). A node that received new information is seen as infected and further infects a certain number of random peers defined as the fan-out parameter [11]. This is repeated for each data object for some rounds, which is followed by the death of that peer.

Generally, gossip protocols are simple, proactively robust to transient network disruptions, and efficiently disseminate information with bounded cost [11], albeit perhaps less quickly than local flooding. However, due to the probabilistic nature of gossip there is a major trade-off between efficiency in terms of duplicate messages and the coverage, stated as the portion of nodes that eventually receive the message. The authors of [11] identify 4 main issues for gossip: membership maintenance, network awareness, buffer management, and message filtering. For IOTA, membership maintenance is managed by the peer discovery process and is assumed to provide a near-complete view of the network. Buffer management is a non-issue due to the bounded message sizes and scheduling rate, and the everyone-to-everyone multicast foundation of DLT systems answers the message filtering question. Therefore the question remains of how to effectively make connections between peers.

For DLTs, absolutely every node needs to receive all messages eventually, but the highly interconnected structure of the DAG also offers the opportunity for a dynamic repair mechanism in the form of Solidification, a blocking process that ensures local knowledge of all relevant previous blocks, either by waiting or by explicitly requesting them from peers. Since each message refers to some older messages, references to missing messages can be discovered, and the full objects requested from peers. However, this mechanism does have limits in the extreme sense, as only those messages are discovered to be unknown whose successors are passed to the node. Therefore, Solidification cannot ensure that blocks without successors (DAG tips) are received, and there may be a possibility for entire branches to remain unknown for a while.

A classical reliable multicast algorithm is a direct competitor and likely outperforms gossip drastically as the redundancy is much lower [12]. However such an algorithm may be much harder to design and configure, and it might be less resistant to malicious behavior. The inherent simplicity and randomness of gossip protocols is ultimately its strength, as there is very little attack surface. They are therefore uniquely suitable for a network underlying a DLT, as only a very limited amount of trust is required between nodes. Additionally, although there is usually a high degree of churn and topology changes, it may be dangerous to trust anything not directly observed.

### B. Bitcoin

With its creation in 2009, Bitcoin was the first blockchain project and is still the project with the highest market cap today. More than 10,000 nodes cooperate to maintain the perhaps simplest blockchain application, which hasn't changed much since its inception over a decade ago. Nodes in the network are only aware of directly connected peers, of which there may be up to 125, and keep a message queue for each of them together with individual Poisson timers. When a timer expires, a list of digests of the buffer entries is offered to that neighbor in an INVentory message, which can then be selectively requested using GETDATA responses. Thus, while any node may receive multiple INV offers, each complete data object is only communicated once.

### C. Ethereum

Ethereum is one of the largest blockchain projects and a well-established network by now. With roughly 10,000 active nodes, it is similar in size to the intended IOTA network, however, the communication patterns differ substantially due to the inherent differences between blockchains and DAG-based ledgers. Kiffer et al [13] observed that the Ethereum network suffers from a high churn rate, as 70% of nodes they observed are active for at most one day. Further, during each hour 20% of neighbors drop their connections after 10 seconds or less.

For Ethereum, any node can be used as an entry point into the network, and the Node Discovery Protocol v5 [8] can then be used to find other participants at very low cost and for any purpose. Communication between nodes takes place using the TCP-based RLPx Transport Protocol and facilitates synchronization of the shared chain, block propagation, and the exchange of requested transactions. When a new block announcement is received from a peer, it will be immediately forwarded to the random small fraction of peers (square root of the total amount). Then, the block is thoroughly validated by executing all transactions contained inside it. If it is considered valid, the hash of the block will be passed to all peers who weren't contacted before and didn't already announce the same block. This approach aims to distribute blocks as fast as possible while reducing unnecessary duplicates. In Ethereum the distribution speed of new blocks is vitally important due to the inherent characteristics of a blockchain, where new additions to the ledger are appended by miners to the most recent block, therefore miners want to hear of new blocks as fast as possible. Since the IOTA network has no such incentive structure and new transactions are added regardless of parallel work by other nodes, the requirements on timing and reliability are less strict.

### D. Hyperledger Fabric

Hyperledger Fabric is a more recent permissioned blockchain project that was designed for high throughput applications, and uses gossip-based broadcast for many purposes, including the dissemination of new blocks to all peers (within an organization) [10]. Under the assumption that all peers are known to each other, the gossip protocol operates in three independent parts: Firstly, there is a fast-paced infect-and-die push model [11], where freshly received blocks are propagated once only to three randomly determined neighbors. Whereas this is done every 10 milliseconds, the pull component that is intended to fill the gaps is executed much less frequently. Every 4 seconds, a block contacts 3 random peers and requests the digests of recent blocks, from which the missing blocks are retrieved in a second step. The third part of the protocol facilitates nodes that (re)join the network to catch up with the ledger in a recovery process, which is not relevant to this research.

### E. Gossip for DLTs

Generally, improvements to the information propagation in the domain of distributed ledgers can be targeted in several layers of the protocol stack [14]. Firstly, data size and structure can be minimized or compressed, but this is not the focus of this research. As part of the network layer, both the gossip mechanism and the network topology should be optimized together.

In principle, a gossip protocol will often operate correctly on a great variety of underlying topologies as long as they are sufficiently connected [12], but the average rate at which new messages can be sent is roughly the inverse of the residency time. Therefore faster dissemination is beneficial even apart from the direct benefit of decreasing finality times of ledger transactions. Luca et al. observed that under the same protocol configuration, the most connected networks achieve a higher successful communication rate and a lower average delay, but at the cost of a greater volume of forwarded messages [15]. However, the choice of connections also matters, as various separate works recommend that peers should be choosing neighbors according to round-trip-times [14], [16], [17]. Some works propose the use of clustered topologies [16], these do however suffer from increased vulnerability to attacks on certain key nodes. The authors argue that this may not be detrimental, as gossip is usually far more resistant to node failures as necessary for DLTs, since the application level consensus can handle at most 50% failures.

Common improvements for gossip apart from optimizing the basic parameters include the use of message digests instead of full objects, for example as part of the proposed improvements to Hyperledger Fabric in [18]. Here direct propagation

is replaced with the slower two-stage digest exchange after the first few rounds of gossip, as this protocol takes the form of an infect-upon-contagion push model [11] with very high network coverage and in turn removes the now unnecessary pull component of HLF entirely.

Additionally, numerous works investigated the potential of adaptive gossip schemes. Huy et al. [4] propose an improvement to Bitcoin in which the probabilities of sending information to neighbors is based on previous message exchanges between the nodes, however, this is mainly applicable to environments where the number of neighbors can differ substantially across all peers. Perhaps more intriguingly, Rodrigues et al. present a gossip-based broadcast protocol that automatically adjusts the per-node emission rate [20], and [21] adapts the dissemination probability of messages at a node varies depending on the perceived communication performances.

Caching metadata about the gossip objects can also be helpful. Some approaches simply remember the directions of incoming duplicates and constrain infections to other neighbors [22], whereas other protocols pass the propagation path or a list of past gossip targets along with the messages. Even though gossip is very much a community process, as all the nodes are in some sense dependent upon the correct behavior of all other nodes [12], the reliance on control information and instructions received from other peers should be kept to an absolute minimum in the trustless DLT environment. There are some ideas to build trust by employing remote attestation such as Intel SGX, and although the direct overhead may not be significant, there are numerous drawbacks that make this a poor fit for most DLTs.

### *F. Healthor*

Healthor is a heterogeneity-aware flow-control mechanism developed in cooperation with IOTA that aims to use shared information about the health of network nodes to shift memory load from low-end or intermittently connected nodes away to more capable neighbors, thereby allowing them to keep participating in the consensus and more easily catch up with the network [2]. This is achieved by maintaining an outbox buffer for each neighbor, that is emptied at a rate dependent on the health of that peer. However, the exchange of health information may open the door to malicious behavior, which has not yet been fully investigated. Healthor varies the transmission rate between peers, whereas a gossip protocol varies which peers messages are shared with, therefore the two mechanisms will likely overlap in a minor way or not at all and can exist next to each other regardless of the chosen approach to gossip.

## III. Methodology

Multiple consecutive steps have been necessary to achieve the goal of optimizing the gossip layer for the IOTA system. First of all, a peer-to-peer network simulation was implemented with all relevant components, simplifying some to exclude irrelevant complexity. This required some reasoning about suitable parameters and assumptions to reflect the intended functioning of the system. The simulation served two purposes, as it empirically demonstrated how the gossip layer is influenced by system parameters and environments and later facilitated the comparison between gossip and flooding. The network topology and the peering component responsible for constructing it required special consideration due to its close relation to the gossip layer. The parameters that influence the topology are not yet precisely defined and may be adapted to best suit any eventually chosen gossip protocol, thus multiple variants were mathematically analyzed to reason about their broad impact and determine one that most closely fits the current design intention. After identifying potential failure and attack vectors through literature research, an initial understanding of the system resilience was gained using both the topology analysis and simulation.

These first results about the system powered by flooding then served to determine the relevant performance metrics that alternative gossip protocols need to consider and optimize for. On this basis, a suitable candidate for an alternative gossip approach was chosen by considering options presented in prior work. This proposal was then implemented to undergo a parameter sensitivity analysis using the already available simulation environment. Evaluation against flooding is vital to precisely determine the advantages and shortcomings of gossip, this comparison was driven by simulating the system under varying load conditions. Once the behavior of the gossip proposal was reasonably understood, it too was subjected to a similar resiliency analysis as flooding before. Finally, variations in topology including both previously discussed and freshly emerging possibilities are simulated in conjunction with gossip to identify recommendations to address its shortcomings.

Through these steps, a proposal for a new gossip protocol emerged, together with the understanding of the relevant parameters, and an idea of its resilience compared to flooding. Additionally, some recommendations can be made concerning beneficial network topologies and therefore the most relied-on system component, the peering mechanism. Further, lessons can be drawn to aid similar systems in the DLT space that share key characteristics.

## IV. Simulation Setup

This section serves to describe the setup of the simulation environment and the implementation of the relevant IOTA protocol components. The simulation is needed to generate data to evaluate the current functioning of the system and helps to gauge the differences caused by changing parameters.

### *A. Framework*

Peersim is a simulation library [26], written in Java, crafted specifically for peer-to-peer (P2P) systems. Its main goal is to aid in the design of efficient and scalable P2P network algorithms by offering tools for thorough testing and analysis. This powerful tool facilitates the development of customized network configurations and topologies, making it an ideal

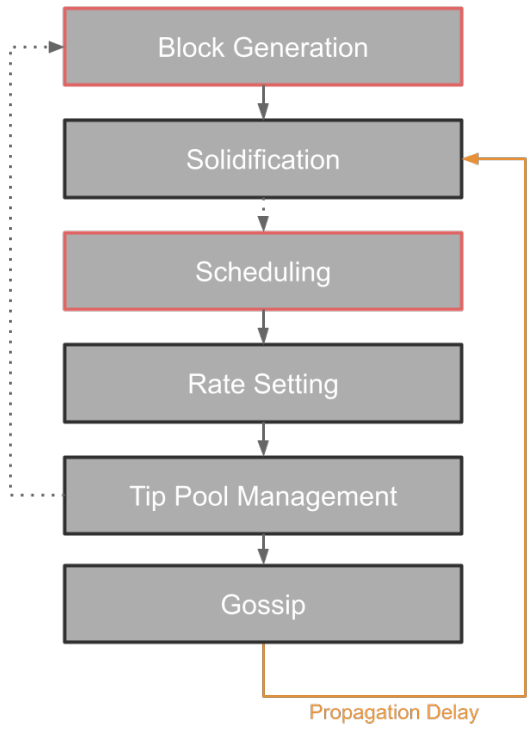


Fig. 1. The structure of the simulated protocol stack, where solid transitions happen immediately initiated through the events designated in red. Yellow arrows denote the introduction of delay through propagation to other nodes.

choice for networks composed of numerous homogeneous nodes.

In this project, each node is modeled as an identical protocol stack, effectively replicating the characteristics of the network. The event-driven simulation engine provided by Peersim is employed to model the network, owing to its ability to accurately mirror the asynchronous nature of the IOTA network.

The simulator operates on a discrete-time model with a resolution of 10 microseconds, equating to 100 ticks per millisecond. This high resolution ensures clear differentiation between events. One constraint of this simulation framework is its assumption of instantaneous processing. Although the IOTA protocol stack is relatively resource-intensive, this limitation is acceptable due to the minimum hardware requirements imposed on network nodes.

### B. Protocol Implementations

This study omits comprehensive modeling of the IOTA protocol stack's consensus mechanism, instead focusing on the gossip protocol. Given that the gossip protocol does not govern transaction approval or validation, but rather disseminates transaction data among network nodes, the consensus mechanism does not factor into performance metrics concerning block dissemination speed and network load. Further simplifying this model, we posit that all blocks issued are entirely valid and consensus-compliant, thus the opinion-based consensus will always agree and therefore does not influence the workings of other components [7].

The data flow in the simulated protocols is as follows: received blocks undergo a solidification process, ensuring that the blocks' past references are known locally or otherwise requesting unknown blocks from other peers. Once solid, these blocks are incorporated into databanks together with the locally issued new blocks.

Subsequently, blocks enter the scheduling component, which enforces the Proof of Stake (PoS) mechanism and controls block throughput. This output stream is instrumental in assessing network congestion levels and the rate setting component is adjusted accordingly. Blocks are then integrated into the local DAG representation and added to the tip pool, which contains blocks without successors. New blocks draw parent references from this pool such that expansion of the DAG from old blocks is avoided. Upon completion of this process, blocks can be relayed to neighboring peers via the gossip protocol.

*1) Block Generation:* With regards to issuing new blocks, nodes behave in one of three ways: they can be either silent, fair, or best-effort. Silent nodes refrain from issuing any blocks. Conversely, fair nodes issue blocks at a rate directly proportionate to their owned share of mana which guarantees this throughput and is enforced by the scheduler. Best-effort nodes go beyond that by leveraging idle network resources via the rate-setting component thereby pushing the network to operate near its maximum throughput. In fact, a single best-effort node could fully saturate the network resources if all other nodes were idle. This distinction of node behaviors is in line with previous research [5].

Nodes issuing blocks generate them at a temporally regular rate to fully exhaust their fair shares. Blocks contain a 32-byte identifier, a timestamp, a reference to the origin node, and references to older blocks taken from the tip pool determining the placement in the DAG. This pool only contains blocks that have passed the scheduling and have no known newer blocks referencing them, thereby ensuring unidirectional DAG growth and the eventual discarding of older blocks. Block sizes are normally distributed between 100 and 900 bytes and thereby accommodate most messages [7], value transactions, and smart contract interactions. Nonetheless, the specifications permit block sizes up to 32KB.

*2) Solidification:* This step ensures that all references contained in a block are locally known since otherwise the validity of blocks cannot be verified. This is achieved by buffering incoming blocks until the missing blocks are received. This may be expedited by sending requests for specific blocks to direct peers, a process that can be recursively repeated and allows nodes to rebuild a block's entire history. The solidification process can generally be understood as a correction mechanism for the gossip protocol, and should therefore be adapted to complement it.

*3) Scheduling:* The scheduler enforces the fair distribution of network resources while relying only on local information that is expected to mirror the state of the network as a whole.

This is achieved by employing a deficit round-robin scheduling algorithm with an input queue for each node in the network. Each queue is ordered by the block creation timestamp. The scheduler grants each queue with a deficit according to the mana held by the corresponding node, which allows for a block to pass once its size is exceeded by the deficit.

The scheduler provides an artificial bottleneck and therefore limits the theoretical throughput of the network [5]. When a queue grows beyond some predetermined threshold, the origin node consumes more resources than allowed and is consequently inherently limited by the forwarding behavior of his direct peers, provided they behave as specified.

*4) Rate Setting:* Since the size of the scheduling queues grants an approximate insight into the overall utilization of the network, it forms a basis upon which a best-effort node can judge unused throughput and adapt accordingly. The specifications intend this mechanism to function similarly to the TCP congestion control in an additive increase, multiplicative decrease (AIMD) approach which lends itself well to independent nodes without deliberate communication [4]. However, a scheduling deficit-based approach should be effectively identical for the purposes of this research. This allows nodes to issue new blocks when their scheduling queue has accrued sufficient deficit, directly utilizing the enforced fairness of the scheduler and the assumption that the observed state is representative of all nodes in the network.

*5) Tip Pool Management:* After being scheduled, blocks become fresh tips of the local DAG as long as no other known blocks reference them. These blocks remain in the tip pool until they are either referenced or expire after 30 seconds. The simulation is initialized with a set of 50 dummy blocks to provide a foundation for the DAG. The amount of tips referenced by new blocks depends on the current size of the tip pool and is chosen such that the size of the pool stays relatively constant. It is determined as follows: $ceil(log_{10}(|pool|))$, thus most blocks contain two references for a pool size between 10 and 100.

## V. Network Topology

The exact parameters that influence the topology are not precisely defined in the documentation, rather a general goal and guidelines are provided. Additionally, the network topology and peering component that constructs it have a close relation to the gossip layer and thus require detailed investigation. Therefore multiple variants are mathematically analyzed to determine their characteristics and the closest fit to the current design intentions. That topology is then used in the gossip protocol evaluation.

### A. Communication

Communication throughout the network is taking place using TCP over the internet. Peersim does not provide extensive link simulation capabilities, therefore we are employing a simplified model without transmission delay or failures. This is approximated by uniformly randomly placing nodes on a unit square as shown in Fig. 2, and determining the propagation delay between nodes based on the cartesian distance plus a small normally distributed variation ($\mu = 10, \sigma = 3$). This results in delays varying between 1ms and $sqrt(2)*100+20 \approx$ 160ms to model a worldwide P2P network [27]. The square is not meant to directly represent the geographic location of nodes, but also factors such as isolation and the quality of physical infrastructure. This approach is limited by not taking the clustering caused by population centers and continental divides into account.

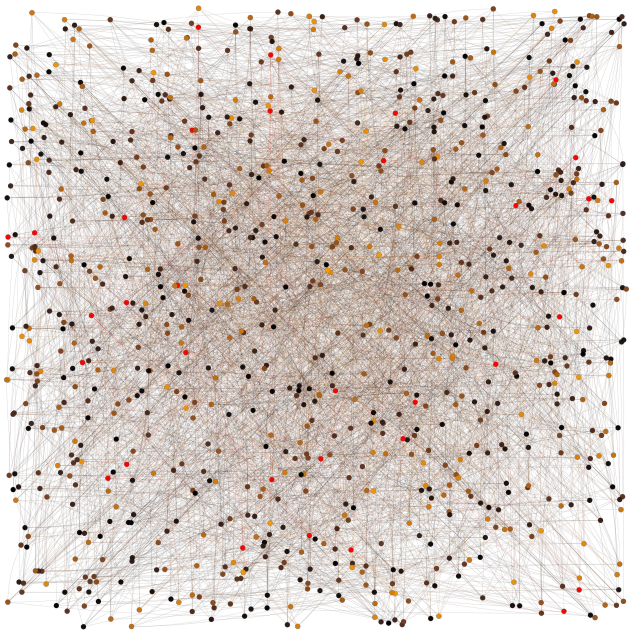

Fig. 2. Uniformly random positioning of nodes on the unit square to represent latency. For an explanation of colors see Fig. 3. Links between nodes represent peering connections, here according to the control baseline.

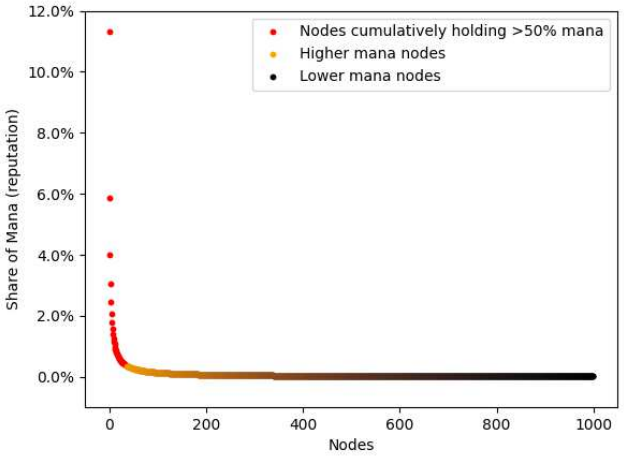


Fig. 3. Distribution of mana according to the Zipf Distribution, where the top 33 red nodes together hold half of the total mana and orange hold more than black nodes.

Flooding serves as a placeholder for a more optimized gossip protocol. Thus, blocks are immediately and individually forwarded to all direct peers after passing through the scheduler. This implies an average redundancy of 7 unnecessary copies received by each node.

### B. Peering Mechanism

Within the network, nodes are distinguished based on the amount of mana they possess. Representing each node's importance, mana corresponds to voting weight and assures an equivalent share of total throughput. The mana values follow a Zipf distribution dictated by a characteristic parameter, wherein the nth value is approximately inversely proportional to n as seen in Fig. 3. This assumption is in line with other research about the IOTA 2.0 network [5]. Further, no distinction is made between the access and consensus mana and the values are immutable for the length of the simulation which is much shorter than an epoch as defined in the specifications, the time unit that generally dictates the rate of change.

In the simulation, a total of 1 million mana is distributed among 1000 nodes with a Zipf exponent of 0.95. As a consequence, the top node holds 11.3% of the total mana, a larger share than the lower half of nodes combined. Contrary to that the top 10% of nodes hold 65% of mana. Half of the mana is shared among only 33 nodes, which therefore have the absolute majority of consensus voting power and throughput guarantee. In general, the owned mana ranges from 160 to 113300 across three orders of magnitude.

The topology of the network is generated closely following the specified auto-peering mechanism [6], ensuring that nodes holding similar amounts of mana are more likely to be direct neighbors. This is accomplished for each node by creating a candidate set comprising all nodes that hold similar amounts of mana. From this set bilateral peering connections are determined based on verifiably random salts, where half of the peers are explicitly requested while the other half is accepted from incoming requests. Each node aims to have 8 peers, thereby structuring the network into an 8-regular graph, although some may have fewer connections when all their candidates already have the full amount. For the simulation, the peering partners are initialized by finding the stable solution given the random values, after which the topology remains static throughout the simulation.

### C. Peering Options

This section investigates different approaches to finding the candidate set for the auto-peering mechanism and provides a cursory overview of the impact of the chosen parameters. All experiments are based on the same set of nodes in regard to placement on the unit square, random peering salts, and mana values. Further, all approaches target a degree of 8 throughout the entire network. Changing this parameter would have big implications on flooding-based message dissemination, as a bigger parameter would proportionally increase the overhead while smaller options were ruled out likely due to resiliency considerations. A control network is needed to serve as an evaluation baseline. It is constructed without respect to node reputation or latency, instead treating all nodes as candidates and forming a stable solution according to the previously discussed random salts.

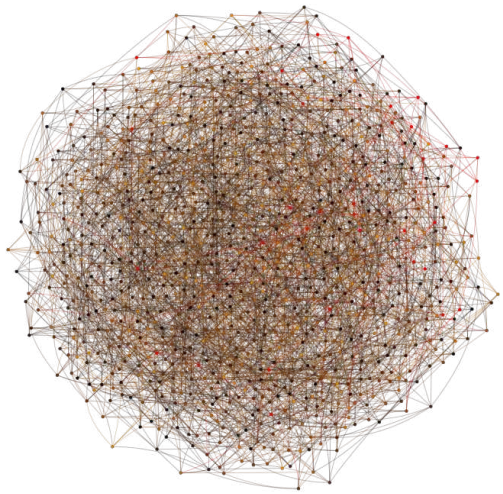

Fig. 4. Topology of the baseline network with random peering in physics-based representation. For an explanation of colors see Fig. 3. As expected, no clear structures emerge.

Fig. 4 shows a physics-based representation of the control network graph using the peering connections as unweighted springs to visualize the clustering and potential isolation of nodes. It can be observed that the control network forms one big interconnected cluster as expected.

The network exhibits a diameter of 5, indicating the maximum shortest path between any two nodes. On average, the shortest route to any other node encompasses 3.6 hops and traverses a distance of 1.35 units. This distance incurs a minimum propagation delay of 135ms, nearly equivalent to the distance between the most distant nodes on the unit square, which are separated by $sqrt(2) = 1.41$ units. The average propagation time for a message from any node to its farthest node is approximately 270ms. This represents a lower bound for new blocks to reach full network coverage through flooding.

*1) Mana Similarity:* The specifications suggest an approach based on linear mana similarity, where for any node the candidate set includes other nodes whose mana differs by a certain percentage. This means that candidate sets of high mana nodes are much smaller. Candidates are divided into an upper and lower set, depending on whether they hold more or less mana than the node itself. If either set is smaller than a certain threshold (here 16), it is padded to include the 16 nodes that have the closest mana values in that direction to

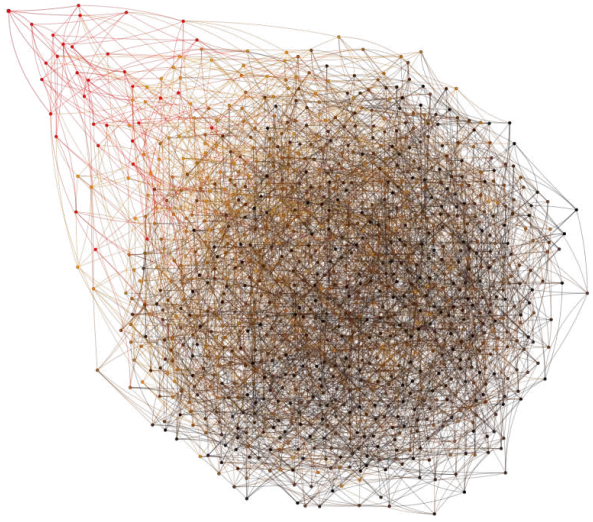

Fig. 5. Topology of network constructed with logarithmic mana similarity where $rho2$. For an explanation of colors see Fig. 3. Note the clear grouping and isolation of top nodes (red) and the gradient from orange to black nodes through the big cluster.

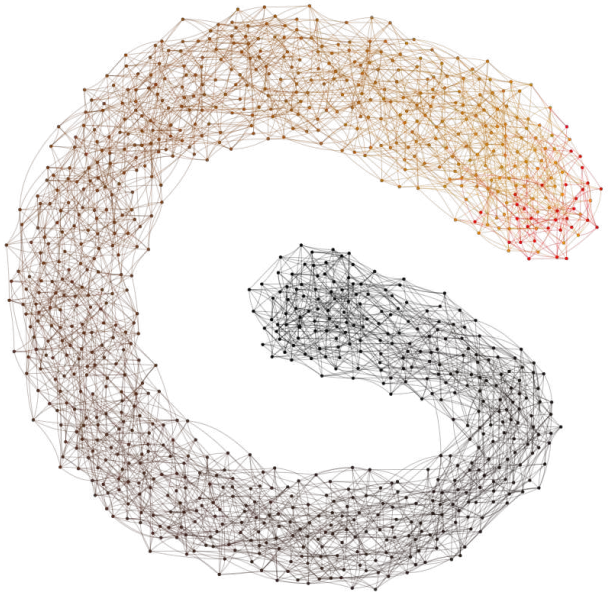

Fig. 6. Topology of network constructed with direct mana similarity, with 100 closest mana neighbors as candidates. For an explanation of colors see Fig. 3. The topology is very drawn out with a clear gradient from high to low mana nodes, requiring a very large amount of hops for communication.

offer sufficiently many peering options.

Candidate sets with similarity thresholds of 100%, 50%, 25%, 12.5%, and 6.75% were analyzed. Generally, it seems like this approach mainly affects the size of the cluster formed by the low mana nodes, and conversely the length of the tail containing the high mana nodes. These have a very small candidate set and therefore require far more hops and incur higher latencies to reach the bulk of the nodes. Nodes with high mana are regarded as vital for network health due to their anticipated major contribution of new blocks and significant voting power [5]. Despite this, these nodes tend to be the most distanced from the rest of the network. It is important to recognize that this isolation refers to communication delays, and does not inherently indicate a danger of being disconnected.

In principle, this isolation of top nodes is desired since it protects them from eclipse attacks due to the high amounts of mana required to peer with them directly. Additionally, short communication paths between these major contributors ensure a robust core of the network required for the ledger to function correctly without depending on the rest of the network. This desire motivates the exploration of different ways of finding good candidate sets. One such possibility is the consideration of a logarithmic similarity threshold instead to better fit the extreme distribution of mana, with

$$rho > log_e(mana_1/mana_2) \tag{1}$$

where $mana_1 < mana_2$. With a threshold of $rho > 2$, the structure retains the comet shape as demonstrated in Fig. 5, but more closely resembles the control network. This is reinforced by nearly identical shortest-path characteristics. Additionally, this network retains excellent resilience against eclipse attacks because every node holds less mana than all direct peers combined.

*2) Direct Similarity:* To remedy the limited size of candidate sets for high mana nodes, candidates could be chosen from a list of all nodes ordered by mana by selecting the nodes that have the least absolute difference and thus immediately precede or follow in the ordering.

$$abs(mana_1 - mana_2) \tag{2}$$

This results in a network mirroring a steady gradient of mana values without the unbalanced clustering observed in previous approaches. Larger candidate sets also increase the variance of the peering randomness for high mana nodes, and therefore likely contribute to resilience against eclipse attacks, but also decrease the cost of such an attack. With the nearest 10% of nodes considered candidates, the network shown in Fig. 6 becomes far more stretched out with a diameter of 22. This is also reflected by the 8.4 hops and 270ms of the average shortest path, whereas the farthest communications incur a 720ms propagation delay, far higher than the control network.

The clear gradient persists even for much larger candidate sets, as the network built with half the nodes being candidates shows similar characteristics to the control while being noticeably more segregated, i.e. intermediate nodes are more likely

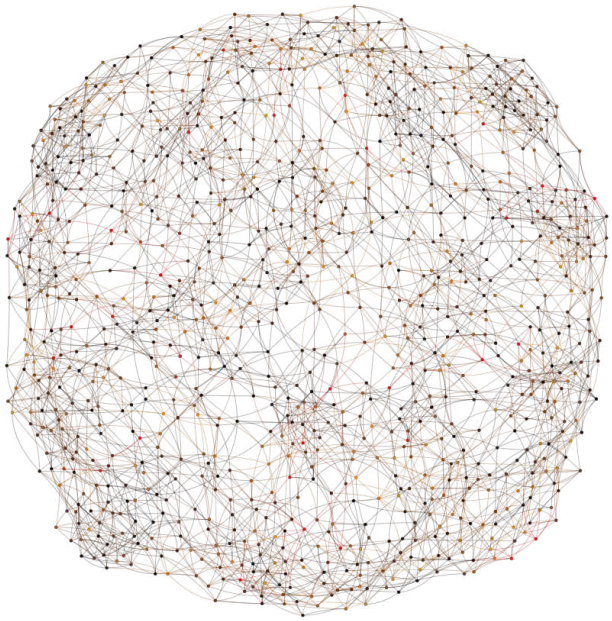

Fig. 7. Topology of network constructed by prioritizing peers with low latency. For an explanation of colors see Fig. 3. No visible mana gradient, but a clear structure where nodes are more likely to have common neighbors with their peers.

to be direct or indirect peers of nodes holding similar amounts of mana, while high mana nodes are less likely due to most of their candidates having far less mana.

It can be concluded that nodes having little mana benefit from much larger candidate sets causing the network to be much less spread out without sacrificing any security. On the other hand, high mana nodes require very small candidate sets to guarantee peering connections with nodes holding sufficient mana, of which there are very few due to the extreme distribution. Consequently, this approach is far less suitable than the logarithmic alternative discussed previously.

*3) Latency Priority:* Peering based on latency presents a completely different approach intended to minimize propagation speed directly. Round trip times can be evaluated locally at no additional overhead during the regular node verification processes undergone by all network participants. Fig. 7 shows a network that results from candidate sets containing the 50 closest nodes by latency. The visibly denser interconnections around the perimeter are a result of the latency modeling using the unit square.

The difference compared to previous approaches becomes immediately apparent as the diameter of 14 is substantially larger than the control, and the average shortest path experienced a tradeoff between the latency and the number of hops. Now the average is 60ms over 6.3 hops, and the furthest node is reachable in just 144ms, approximately half that of other network structures. As each hop incurs a processing delay, especially that of the scheduling component, there is likely a larger difference now between the performance during low and high load scenarios.

One concern with this structure is the potentially low cost of bisecting the network. Since mana similarity is not considered at all when choosing peers, it may be possible to disconnect two clusters of high mana nodes on opposite sides of the unit square by controlling a large number of low mana nodes in between at a considerably lower combined cost than eclipse attacks in previous topologies.

*4) Mix of Approaches:* A potential improvement over the logarithmic approach is therefore a combination of a mana similarity and a latency prioritizing peering mechanism. This can be achieved with little effort by combining the candidate sets. Two different mixes were attempted, one with the 20 closest nodes and 250 most similar by mana, and one with 50 and 100 respectively. The resulting topologies demonstrated that the advantages of both approaches come together to form a network with a very low diameter, short paths, and a low number of hops between nodes. Additionally, the network has reasonably good eclipse resilience, about two-thirds of the logarithmic similarity peering with $rho > 2$.

However, many different combinations and methods of mixing are possible but reserved for future work. For example, forming a union between two (larger) candidate sets may ensure the benefits of both. Similarly, nodes may be forced to select half of their peers from either set, maximizing both. The latter approach is less of a compromise and combines two optimized topologies at the cost of having only half of the peers belonging to either method, potentially reducing their impact.

### D. Comparison

Some approaches tend to produce drawn-out topologies with large diameters and therefore communication paths with more hops. These are worth minimizing when the processing at each node incurs a substantial penalty in relation to the propagation delay as is the case during periods with high system load. Similarly, the link latency of the shortest paths presents as a lower bound to achieving full dissemination coverage and therefore should be reduced where possible.

From Fig. 8 and 9 it becomes clear that any approach can result in graphs with similarly small diameters, while the link latency is expectedly lower by half for the approach targeting this metric. Interestingly, the mixed candidate sets are consistently on par with the best comparable alternatives.

Measures of centrality can provide insight into the dissemination behavior of communication networks [29]. Betweenness centrality equals the share of shortest paths that pass through a single node. While this is not very relevant for flooding, alternative gossip protocols that aim to reduce the message redundancy will see significantly higher loads at nodes with high betweenness centrality if the majority of data preferentially takes the shortest path. Some mana similarity topologies have nodes that are crossed by more than 25% of shortest paths. On the other hand, latency-based approaches tend to have more clusters, which are tightly nit groups with

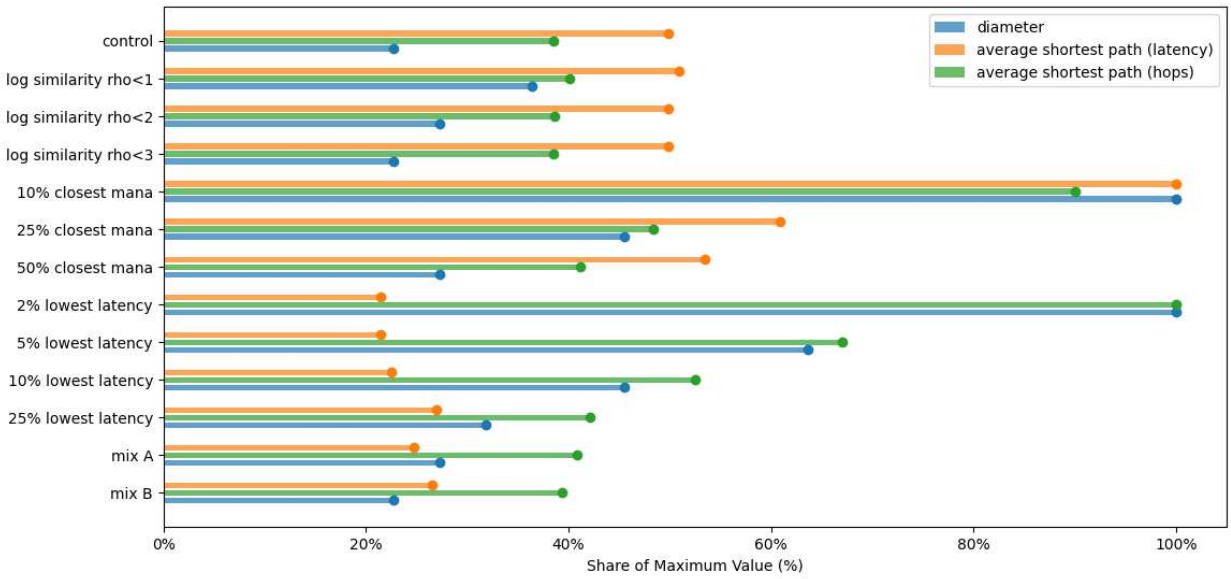


Fig. 8. Topology characteristics in relation to each other. The network based on 10% closest mana has the worst overall characteristics, while the mixed topologies approach is the best alternative for each characteristic, close to the diameter and path hops of the logarithmic similarity options and the path latency of latency-based topologies.

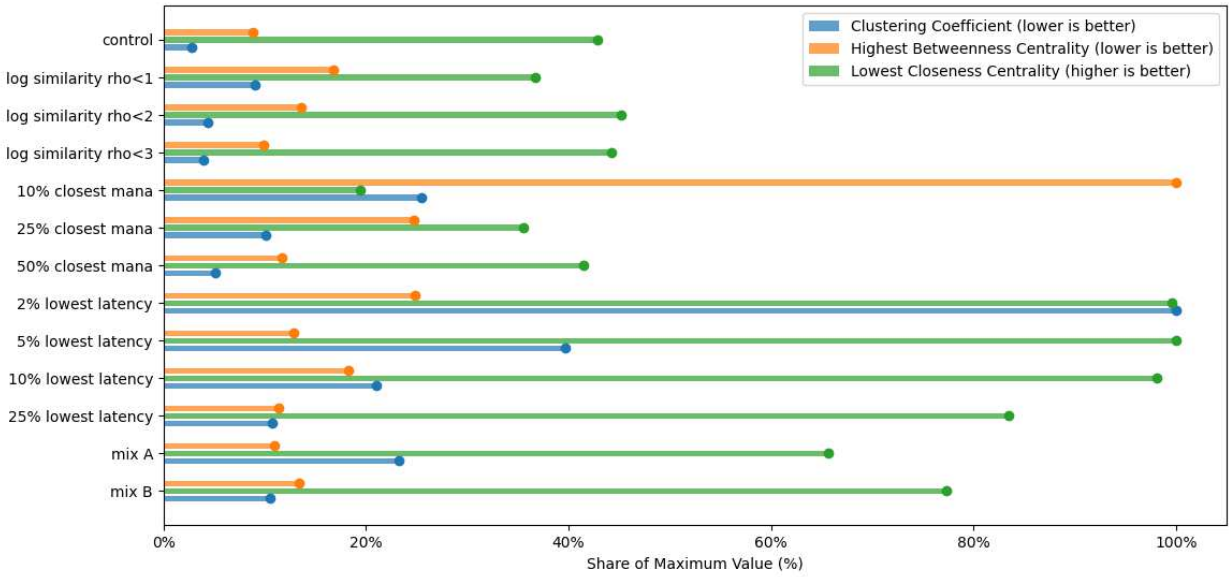


Fig. 9. Topology characteristics in relation to each other. Again, the mixed options combine good characteristics of other alternatives. The topology based on 10% closest mana has the worst characteristics, whereas logarithmic similarity is decent overall but suffers from below average closeness centrality. Latency-based topologies have substantially higher clustering, as seen in Fig. 7.

a relatively high density of ties at the expense of inter-group ties. Generally, networks with less clustering are more resilient towards attacks and failures [28]. However, these networks also have excellent closeness centrality values across all nodes, a measure of the distance between the node and the center of the network. If the network contains nodes with very low closeness, dissemination is expected to be slower.

### E. Consequences

It is fundamentally important for the network to have a similar degree for all nodes due to concerns about resiliency. Then, the maximum shortest path between nodes should be minimized to reduce the time it takes to reach all participants. In visual terms, this refers to shortening the tail of the topology. Additionally, these shortest paths should contain few hops to reduce the impact of delays incurred by other system components at each node.

Betweenness centrality should be approaching uniformity across all nodes, such that traffic is well distributed over many different paths and does not accumulate at some central nodes. Closeness centrality should be maximized for all nodes, particularly for the high-impact, high mana nodes such that they form the center of the system not only operationally but also topologically. Finally, it is vital for the high mana nodes to be direct peers with each other to be resilient against eclipse attacks.

## VI. System Performance with Flooding

Here the performance of the system with flooding-based message dissemination is investigated to properly understand the behavior, such that changes after the introduction of an optimized gossip layer can be correctly attributed to that change. Further, determining relevant performance metrics helps to understand what needs to be considered and optimized by alternative gossip protocols.

### A. Data Collection

The simulation requires an initial warmup period for it to attain full stability and saturation with data superseding the initialization artifacts. This stabilization process ensues in multiple phases, during which nodes initially activate at random times, later transitioning to their assigned issuing behaviors. Stability is evaluated based on the consistency of the standard deviations of control metrics such as issued messages, scheduling queue sizes, and tip pool size. The simulation was observed to stabilize approximately three minutes into the simulated timeframe.

| Peering | Logarithmic Similarity $rho < 2$ |
|---|---|
| Network Size | 1000 Nodes |
| Average Block Size | 500 Bytes |
| System Throughput Limit | 50KB/s or 100tps |
| Zipf Parameter | 0.95 |

TABLE I
Simulation Parameters

Following stabilization, data is compiled into batches, each containing all blocks created within 10-second intervals. Aggregated averages are collected across all blocks and nodes concerning the incoming and outgoing network load per node, as well as the overhead induced by solidification requests, quantified by the number of messages and redundancy of blocks.

Additionally, data on propagation speed is gathered, specifically the duration required to achieve 100% and 95% coverage, as well as the time taken to reach a group of nodes collectively possessing two-thirds of all mana. This information is recorded both in aggregate and for each individual block, enabling the reconstruction of the propagation behavior when paired with the reception timestamp at each node.

For each metric the minimum, mean, median, and maximum as well as 25th and 75 percentiles and standard deviations are determined.

### B. Block Propagation

The scheduling delay is accrued per hop, while the propagation delay is contingent upon the length of the path. Given this topology, the anticipated average propagation delay between nodes is approximately $1.35 * 100 + hops * 10 = 180$ms. Full coverage is expected to be under $2.5*100+hops*10 = 300$ms in all cases without considering scheduling delay.

Under very low network loads (10%), full coverage is achieved within 190ms to 250ms for all blocks. This range aligns with the topological observation as no scheduling delay is anticipated at such low loads. However, at higher congestion levels (99%), full coverage takes between 200ms and 1200ms, albeit with an average of 290ms and a standard deviation of 90ms. The high maximum is attributable to the scheduler operating near the maximum throughput, which can temporarily induce backlog. These outliers can be observed in Fig. 10, where it can also be seen that the majority of blocks originated from high mana nodes, and share a very similar dissemination pattern of a slow start followed by rapid spreading until the last few nodes again require more time. Blocks that are slow outliers often stagnate at some coverage percentage, then resume the usual pattern. This can likely be attributed to to the fact that the local state at the scheduler is not perfectly representative of the entire network, and thus two realities ”collide” at some frontier in the topology, causing the block to be delayed more than at previous nodes.

Lower coverage thresholds are consistently attained more swiftly. On average, 95% of nodes are reached within 250ms, while two-thirds of the mana is reached within 200ms. The delay introduced by the scheduler is observed to depend on the overall network congestion.

### C. Resiliency

The congestion control and scheduling component efficiently mitigates most standard attacks by filtering traffic at each node. Blocks are only forwarded after passing robust semantic validation, ensuring known and valid references, and confirming the fair behavior of the issuer. This framework

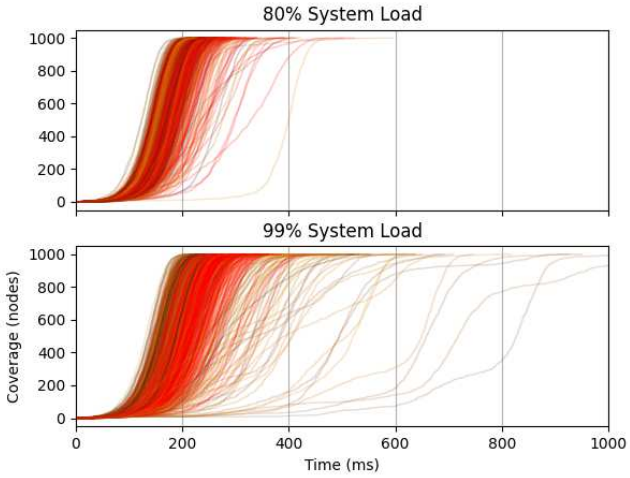


Fig. 10. Normalized block dissemination behavior where color indicates the mana of the origin node according to Fig. 3. The vast majority of messages originate from top mana nodes and disseminate well, with only very few outliers at high system load.

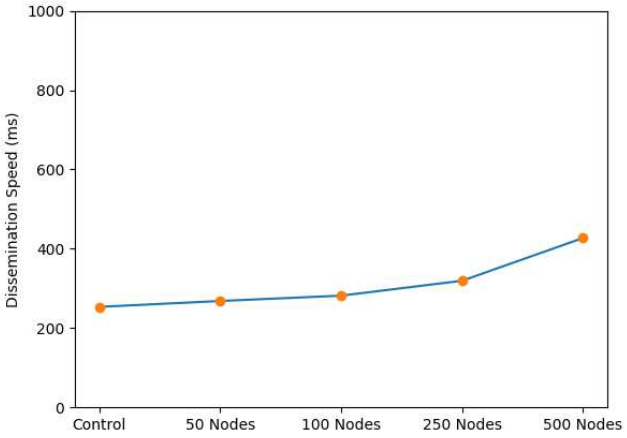


Fig. 12. Effect of non-participating nodes on the block dissemination speed. The network remains fully connected with half of the nodes affected and suffers less than 100% speed penalty.

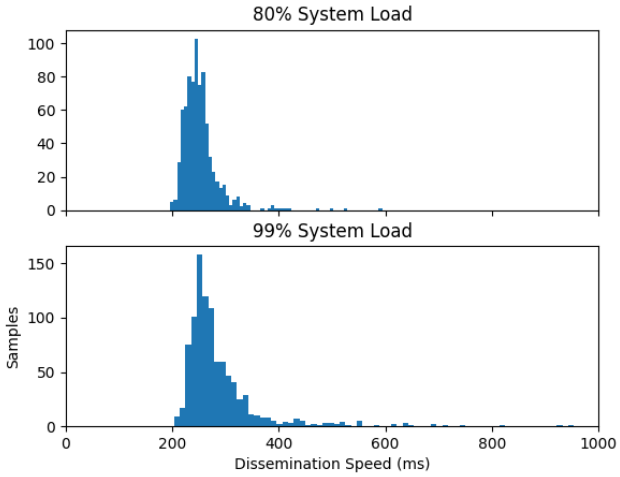


Fig. 11. Histogram of block dissemination speed seen in Fig. 10 to reach full network coverage. Note that the distributions are very similar, with only a few more outliers at high load.

restricts the impact of constructive attacks targeted at direct peers, such as introducing falsified, spoofed, or invalid data, spam, or chatterbox attacks.

Moreover, these attacks can often be effectively detected and traced due to the signature included in all blocks, enabling validation and potential exclusion of poorly behaving nodes from the network. Consequently, disruptions primarily arise from nodes that deviate from the specified block-forwarding behavior. Selfish nodes may selectively forward only some blocks, while failing nodes may generally be unable to participate. Here, the ledger itself offers a powerful tool against withheld data, since the relations between blocks are well documented. Therefore nodes can only be kept unaware of existing blocks if they never receive any descendants, since otherwise all intermediate blocks are discovered through the recursive solidification process.

Dividing the network into separate distinct parts would significantly impact the correct functioning of the IOTA ledger [9], representing a denial of service for some or all of the participants, since the consensus ensures that some relevant share of total mana actively participates. To achieve this, actors could deploy colluding silent nodes that halt propagation in an attempt to hinder full-scale data dissemination. However, approaches with random placement are not particularly effective as almost half the network becomes silent before the first individual nodes become isolated. Moreover, even extensive node failures have a small effect on the overall dissemination speed, which worsens by less than 100% even in the case of 500 colluding participants as shown in Fig. 12. While the average speed slightly increases, it remains consistently faster than the slowest blocks, which incur most of their penalty due to scheduling delays, not slower network routes or solidification processes.

Alternatively, an attacker could aim to bifurcate the network into two major segments by strategically placing nodes. Even if the peering could be influenced to this degree, the required effort is infeasible in terms of the cost of cumulative reputation across colluding nodes. Further, the computation of cost-effective bisections of the graph is itself an np-hard problem [30]. Furthermore, gossip can usually deal with more than 90% node failures, but the ledger cannot. Therefore gossip only needs to be resilient to the same degree as the application.

## VII. Gossip Design

The observations discussed throughout the previous sections form the basis to determine an alternative gossip approach. Different approaches presented in other prior work are compared to find a candidate with suitable characteristics.

### A. Objective

Flooding is a naive approach to disseminate blocks through a network and especially excels at propagating information reliably and unbeatably fast but at the cost of immense overhead in the form of redundant data. Avoiding this transmission of unnecessary copies reduces the network load per block and consequently facilitates a higher transaction throughput without requiring additional resources. However, redundancy remains a valuable asset in DLT environments as it is a fundamental contributor to their resilience. DLTs already provide the tools to facilitate this redundancy while avoiding unnecessary data transfer, since all blocks are identifiable by their unique id, which is also the cryptographic digest of its content. Optimally the gossip protocol would burden each node to a similar degree proportional to the number of blocks and regardless of the total network utilization.

Similarly, the propagation speed is not as critical as the consistency between blocks, origin nodes, and across different load scenarios, as long as it remains within a reasonably short amount of time. This consistency can then serve as a reliable basis to conform to the parameters and expectancies of other protocol components, such as the time frames for opinion setting and consensus voting. This also necessitates fast and infrequent solidification, which may otherwise cause some nodes to fall behind and hinder their proper participation. Therefore the gossip component should be highly probable to reliably work in a nominal environment without relying on solidification.

Epidemic protocols are a common alternative modeling information as the spread of disease in an epidemic, relying on randomness to provide excellent fault tolerance and scalability at the cost of redundant transmission similar to flooding. While this resiliency is a desired characteristic, it is not required to this extreme degree, since the IOTA application has a significantly more stringent requirement for correctly functioning nodes. Gossip protocols can commonly tolerate about 90% of node failures, whereas the ledger requires more than half of the nodes to participate actively to reach consensus. Therefore some reliability can be traded for other desirable characteristics.

### B. Push-Pull Gossip

The very popular Push-Pull gossip protocols combine the upsides of push and pull-based gossip to provide an excellent compromise between speed, efficiency, and resilience. In the push mechanism, a node forwards a new piece of information to a randomly selected peer, whereas in the pull mechanism, a node in need of information contacts a peer to request new information from it. Generally, the push mechanism performs better at the beginning of a gossip spread when only a few nodes have the information but struggle to reach the last few uninfected nodes in a large population of infected ones, a situation where pull gossip outperforms. Push-based protocols struggle in environments with high churn, which is often characteristic of DLT systems, and pull gossip additionally helps nodes recover from transient failures.

Following a mixed strategy, peers are offered a summary of available blocks, of which the unknown ones can then be specifically requested. This drastically reduces the overhead as redundancy is still available, but not propagated. Since blocks are not transmitted through individual messages but instead aggregated into rounds the total number of messages is now much lower, consistent, and predictable. The amount of messages now depends solely on the round interval and the size of the chosen subset of peers.

For this application, blocks would pass through the following stages: After scheduling, blocks are added to the gossip buffer. A subset of peers is chosen each round and offered a list of digests from the buffer. Every peer then responds with a list of the desired blocks, which are then supplied to them in full. After some time the blocks expire and are removed from the buffer. Solidification can be seamlessly supported by adding missing blocks to the request set. This further allows the independent optimization of solidification for the sake of catching up with the ledger, where blocks are only requested from certain peers according to outside agreements or resource availability.

Therefore, the gossip behavior depends on a number of parameters:

- Round interval (how frequently contact is initiated)
- Peer subset size (how many of the peers are contacted per round)
- Peer subset selection (how this subset is selected)
- Block offer expiry (span of time in which blocks have a chance to be offered)
- Block offer selection (how the offer set is selected)
- Solidification delay (how long before missing references are explicitly requested)

Each initiated contact encompasses a staggered three-phase push-pull procedure in both directions, thus resulting in an exchange of information. This combines two unidirectional gossip processes in 4 messages instead of 6 if they were independent. Compared to flooding, one major consequence of this three-stage exchange pattern is the subsequent decrease of propagation speed by a factor of three, and further delays depending on the round lengths and the additional processing.

### C. Topology Based Gossip

Topology-based gossip protocols rely on nodes selecting neighbors to form a structured overlay network. Common topologies include rings, grids, or more complex structures like hierarchical trees. In some scenarios, topology-based gossip can ensure faster or more consistent dissemination of information across all nodes and can reduce the number of redundant messages, leading to lower bandwidth consumption. Additionally, the systems scale more gracefully such that the average number of hops to disseminate information remains relatively low.

The benefits of the structured approach are not without trade-offs. Compared to random gossip, it may be less adaptable, especially when confronting network shifts or node failures. Dynamic environments, where nodes frequently join,

depart, or become inactive, pose challenges in maintaining the structured overlay, often leading to increased overhead. Ethereum serves as an example, illustrating that DLT systems can experience significant churn rates, amplifying the effort needed to maintain the topology. Such complications might compromise the consistency and fault tolerance required of DLT systems due to their decentralized nature and adversarial environments.

The structuring may introduce inherently new problems. If not designed carefully, certain nodes might become central points of control or failure, which goes against the decentralized nature of distributed ledgers. With the topology being publicly known, an adversary can exploit this knowledge to carry out targeted attacks, like eclipse attacks, where a node is isolated from the rest of the network by adversarial nodes, or DoS attacks to induce failure in that node and cascading consequences throughout the network.

One common strategy to address the challenges of structured gossiping relies on implementing a reputation system that can help nodes gauge the trustworthiness of their peers based on their past actions. Designing a robust and abuse-resistant reputation system is generally challenging, but already exists for IOTA as part of the consensus and Sybil protection protocols. It is therefore conceivable that it would be possible to create a purely topology-based gossip protocol for this system, but it was concluded that the benefits are not sufficient to fully balance out the risks. Additionally, such an approach would require the full redevelopment of the auto-peering protocol, which lies outside the scope of this work.

In many existing distributed ledger systems, especially public blockchains like Bitcoin or Ethereum, a combination of random and structured approaches has been used to balance efficiency, decentralization, and security. For Iota, the use of auto-peering to ensure that neighbors fulfill certain requirements about similarities regarding reputation or location could be considered a weak form of structuring. More advanced structured approaches are reserved for future work.

## VIII. Results

### A. Data Collection

The data collection mechanism described in Section VI-A is expanded to facilitate proper comparison between different runs by employing seeded randomization. This ensures that the network topology, mana distribution, node locations, and roles are identical between runs.

### B. Parameter Sensitivity Analysis

The sensitivity analysis serves to explore the impact of altering the identified parameters. For that purpose, an arbitrarily chosen base configuration is compared against variations thereof with one differing parameter. All simulations are set at 99% of the maximum throughput as this is the expected state of the system for the vast majority of the time. This load is achieved without best-effort nodes, as the impact of rate setting is explored in Section VIII-C1. The baseline is chosen with a round interval of 50ms and expiry after 6 rounds during which blocks are offered with 100% probability to a subset of 4 peers.

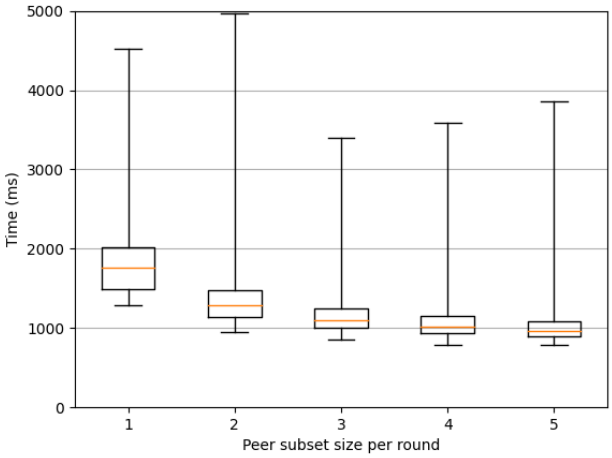


Fig. 13. Block dissemination speed for different peer subset sizes showing diminishing returns for higher values.

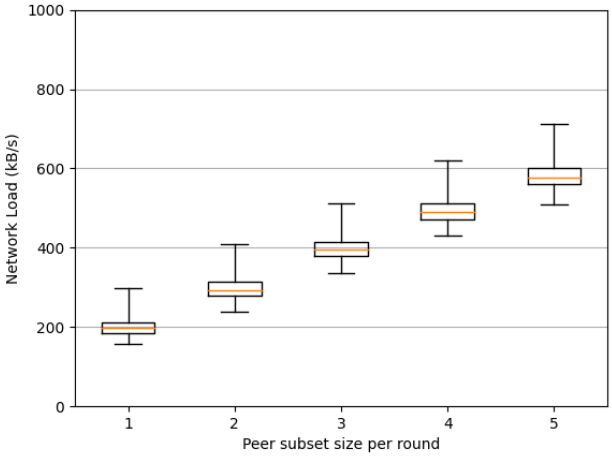


Fig. 14. Network load for different peer subset sizes showing a steady linear increase for higher values.

This configuration is expected to disseminate blocks at least three times slower than simple gossip due to the offer-request-response pattern. The round interval further introduces delays at each hop since blocks are no longer immediately forwarded. However, rounds that are too short would introduce unnecessary interactions and reach a point of diminishing return as they approach the frequency of issued blocks at 100 per second or every 10ms. Choosing too many neighbors per round may also create unnecessary interactions, while too few could lead to erratic behavior where some nodes receive new blocks significantly earlier than others.

Even though the block digests are much smaller than the full blocks, they would still cause an enormous amount of data to be propagated repeatedly if they expire too late and are thus

offered too frequently, whereas a fast expiry would require solidification far more often, especially in the presence of packet loss and short term disconnects which are not modeled here. This in turn delays the block dissemination, especially for large delays before solidification is initiated. However, if this delay is very small blocks may be requested and transferred in duplicate as part of simultaneous gossip exchanges. A low probability of a block being included in the offer causes longer delays before being forwarded, but too many offers again increase the interaction size unnecessarily.

*1) Peer Subset Size:* Contacting more peers per round invariably causes more contacts per second and consequently more contacts per block before expiry. However, as can be seen in Fig. 13 and 14 there is a tradeoff between the time needed for full dissemination and the network load caused by the higher overhead. With every additional outgoing contact per round, the cost in network load increases proportionally at diminishing benefits with regard to speed, which appears to be declining logarithmically. For 5 peers, the speed is only 80ms worse than the theoretical minimum of 3x flooding but only marginally faster than a set of 4 or even 3 peers.

*2) Block Selection Probability:* The baseline configuration includes an offer for all blocks in all contacts. Instead, blocks are offered only with a certain probability in order to counteract the overhead of a long expiry period. This way, blocks can be offered over a longer period of time while keeping the same amount of expected offers. As a direct consequence there is a longer delay before a block is first offered hence reducing dissemination speed, barring any other changes. As a possible solution, a simple adaptive block selection based on its age is put forward, titled "decay". Here, the selection threshold is set inversely proportional to the delay since scheduling time with a floor of 10%. Therefore, more recently scheduled blocks are more likely to be chosen. This approach results in an average probability of 55% over the lifetime of the block offer.

Fig. 15 and 16 show that the network load is directly proportional to the average selection threshold. This makes sense since the amount of offers is directly dependent on that probability. Here again, the speed benefits are diminishing at higher thresholds. Interestingly, the "decay" option disseminates messages 2.3% slower than the 100% offer probability, and at a 31% decrease in load, therefore providing a worthwhile alternative.

*3) Expiry Period:* Longer expiry periods simply increase the number of offers per block without influencing the contact frequency. As the speed is not dependent on repeated offers, it makes sense to see that it remains very stable for the first three runs with late expiry, then increases slightly by 10% for an expiry set at 50ms after scheduling as demonstrated in Fig. 17 and 18. On the flip side, this comes at a load saving of 27% over an expiry set at 150ms. At 25ms gossip starts to behave erratically with much lower speed and more variance. It seems that 4 expected contacts may not be sufficient for reliable timely full dissemination. This is likely due to the slower solidification process now having a significant impact since the average speed to reach 95% nodes is 50% lower than

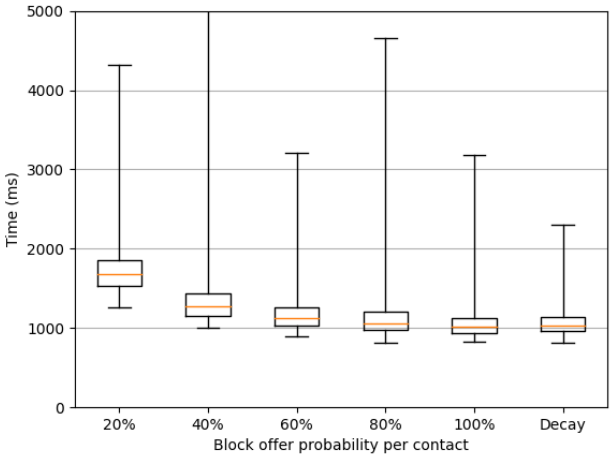


Fig. 15. Block dissemination speed for different block offer selection probabilities showing diminishing returns at higher values, and a close to optimal performance from the decay mechanism.

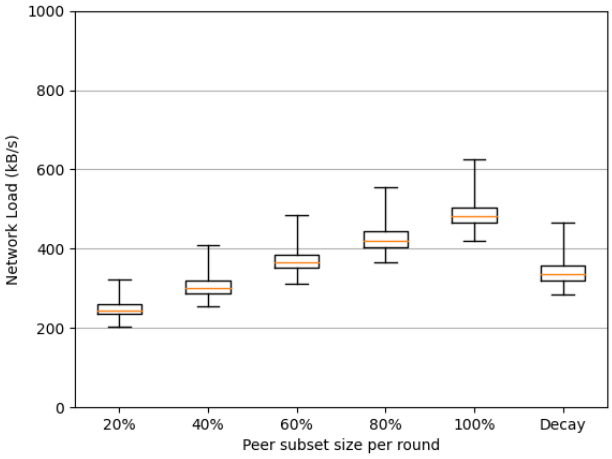


Fig. 16. Network load for different block offer selection probabilities showing a linear increase with higher values with the decay mechanism performing at about 50% subset size equivalent.

full coverage, albeit still 50% higher than equivalent coverage at 50ms expiry and with much slower maximum results.

*4) Round Interval:* Changes to the round interval may seem very similar to changing the peer subset per round, and while they have comparable consequences for the number of block offers, they differ regardless. The subset size provides a certain guarantee to the diversity of chosen peers, which becomes evident when considering the extremes of choosing all or only one peer per round. Therefore it provides a weak security guarantee with regard to the peers a node is actively exchanging information with. The round interval then serves as a balancing parameter and also distributes network traffic over time. Contact is initiated at the start of each round and then distributed over time merely as a consequence of the propagation delay between nodes, potentially leading to traffic

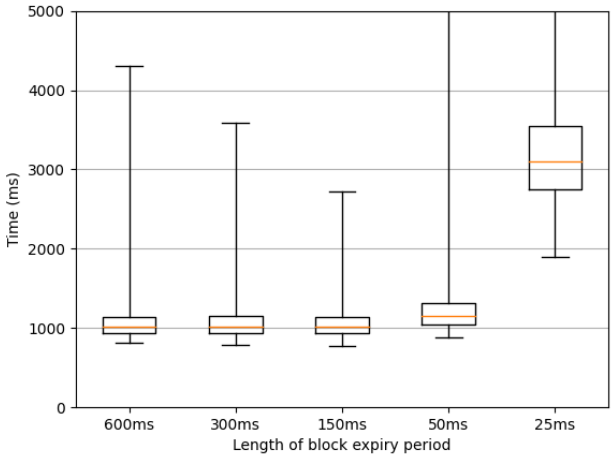


Fig. 17. Block dissemination speed for different block offer expiry periods, showing very little difference until the period becomes shorter than 50ms.

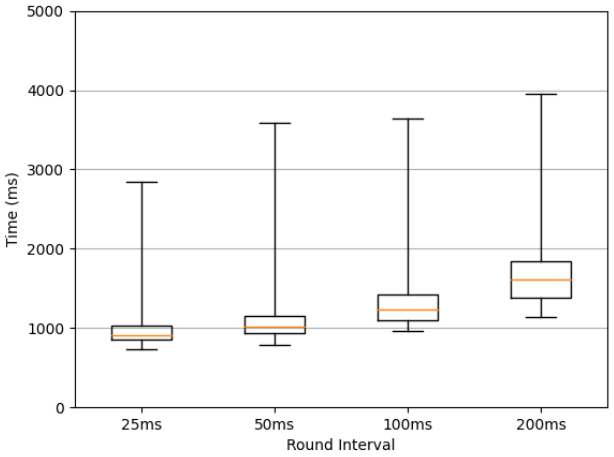


Fig. 19. Block dissemination speed for different round intervals showing a steady increase with longer intervals.

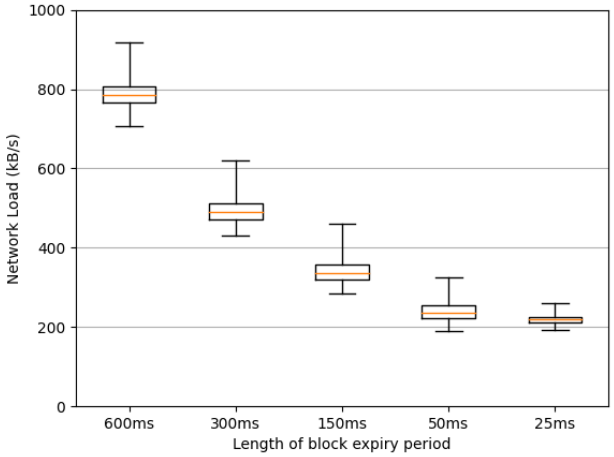


Fig. 18. Network load for different block offer expiry periods, showing a steady decrease with shorter periods.

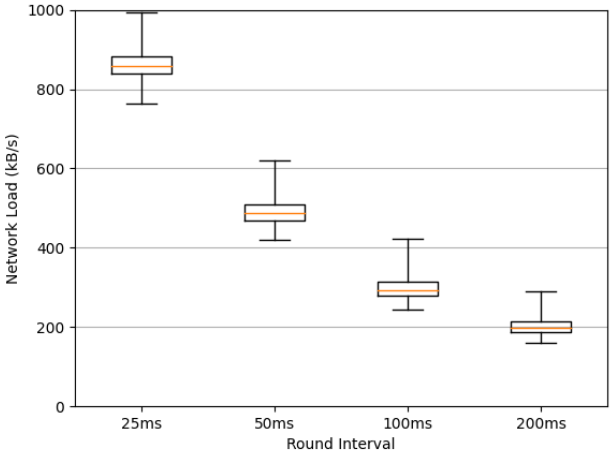


Fig. 20. Network load for different round intervals showing a steady decrease with longer intervals.

spikes for large round intervals.

It is therefore no surprise to observe similar results as well, shown in Fig. 19 and 20. A round length of 25ms results in the most frequent contacts so far, therefore it is not surprising that the dissemination speed is the best yet, at only 20ms worse than 3x flooding. The speed decreases and variance increases with longer rounds. The load is inversely linearly proportional to the round length, where the 25ms experiment requires only 10% less than flooding on average, and 5% more for some nodes.

*5) Further Experiments:* The configuration that served as a baseline for the parameter sensitivity analysis was chosen before any new insights were gained, necessitating the creation of a new, improved baseline for the remaining experiments. Experiments from the previous sections are combined with the goal of improving both the incurred network load and the dissemination time. The comparison is summarized in Fig. 21 and 22 with the listed configurations in Table II. It can be seen that halving the peer subset size drastically reduces the network load, but in turn, increases the dissemination time by over 30%. A better alternative is a decaying offer selection, which reduces the load to a lesser degree but does not compromise on speed. Prioritizing the round interval over the peer subset, both can be halved while keeping the same performance. Then the expiry is reduced by 33% which improves the network load by about 20% without a significant effect on the speed.

Finally, it can be seen that these gossip parameters compare to flooding in a balanced trade-off, a two-thirds reduction in incurred network load at the cost of tripling the dissemination speed.

| ID | Round interval (ms) | Offer expiry (ms) | Offer selection (%) | Solidification delay (ms) | Peer subset size |
|---|---|---|---|---|---|
| PP1 | 50 | 300 | 100 | 150 | 2 |
| PP2 | 50 | 300 | 100 | 150 | 4 |
| PP3 | 50 | 300 | decay | 150 | 4 |
| PP4 | 25 | 300 | decay | 150 | 2 |
| PP5 | 25 | 200 | decay | 100 | 2 |

TABLE II
PARAMETER CONFIGURATION OF ADDITIONAL EXPERIMENTS

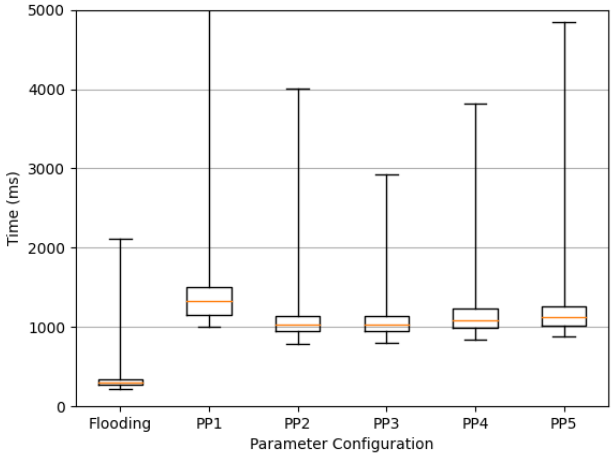


Fig. 21. Block dissemination speed for the configurations in Table II, showing that flooding is substantially faster than either gossip alternative, among which PP1 is the slowest.

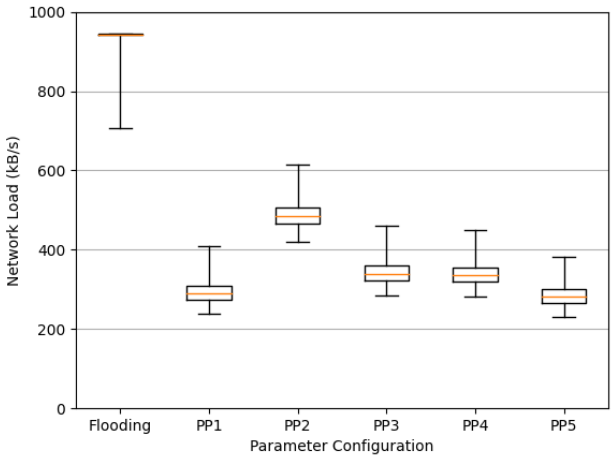


Fig. 22. Network load for the configurations in Table II, showing that flooding is significantly worse than either gossip alternative, and PP5 is on par with PP1 as the best performing option.

### C. Impact of Loads and Rate Setting

The previous section explored the behavior of the gossip protocol at high, fairly distributed system loads, which is expected to represent the normal state of the system. However, it is equally important to investigate other load scenarios to ensure the correct functioning in all situations in case these expectations are incorrect or the system evolves in the future. The scheduling component defines the limit of data throughput, thus the system load is stated as a percentage of that theoretical maximum. Even high loads are expected to be a fraction of the actual available network bandwidth.

A direct comparison between naive flooding and gossip at 10%, 80%, and 99% fairly distributed traffic in Fig. 23 and 24 clearly shows that the load increases proportionally to the system load for both but to a lesser degree for gossip, which requires fewer network resources by about one-third. The number of messages is not shown here, but it mirrors the same trend for flooding since they directly correlate. For gossip, it only depends on the chosen parameters and is therefore identical for each load level.

For flooding, nodes with fewer peers incur a proportionally lower load, but for gossip, the variance between nodes is explained by the fact that data passes preferentially along the shortest paths, and thus the load is dependent on the centrality of the node in the network topology. This causes some nodes to differ as much as 30% from the average in either direction, whereas for flooding nearly all nodes require the same, maximum network load.

It becomes clear that gossip is more efficient using the available resources across all load levels. The median dissemination speeds are quite constant across varying loads, where gossip performs approximately 4 times worse than flooding. The variance and maximum speeds increase noticeably at higher loads since the system intermittently reaches the throughput limit.

*1) Rate Setting:* Rate setting allows nodes to use the idle throughput capacity such that the system always operates near the scheduling limit. For these simulations, it is assumed that all nodes still issue fairly at 10% of their potential, and the remainder is distributed among 10 and 100 randomly determined best-effort nodes respectively.

Fig. 25 and 26 provide an overview of the dissemination behavior of individual blocks normalized to the same start time. With only fair nodes, the distributions are mostly similar, with increasingly many outliers to the right. When there are few best-effort nodes, there is a distinct larger second peak around 3600ms for gossip, this behavior is

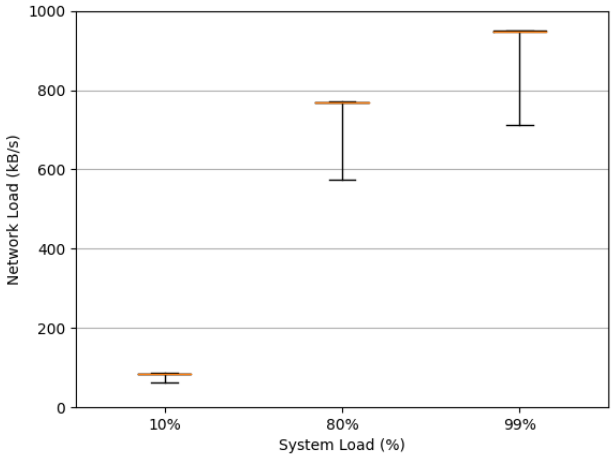


Fig. 23. Network load for flooding under varying system loads, showing a proportional increase. The vast majority of nodes share the same maximum value, the only lower outliers are nodes with fewer peers.

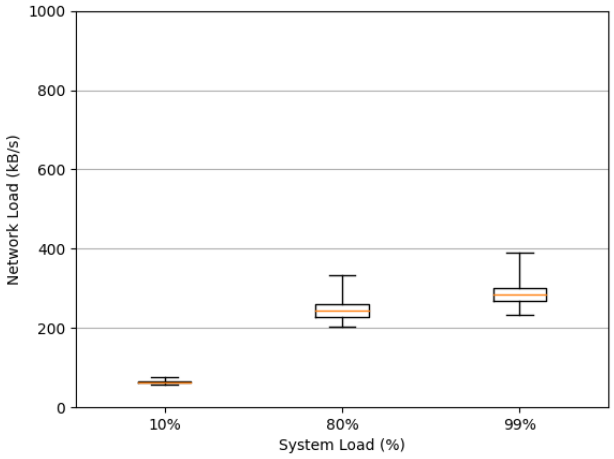


Fig. 24. Network load for gossip under varying system loads. The load increases proportionally but is also distributed across all nodes with a difference of as much as 75%.

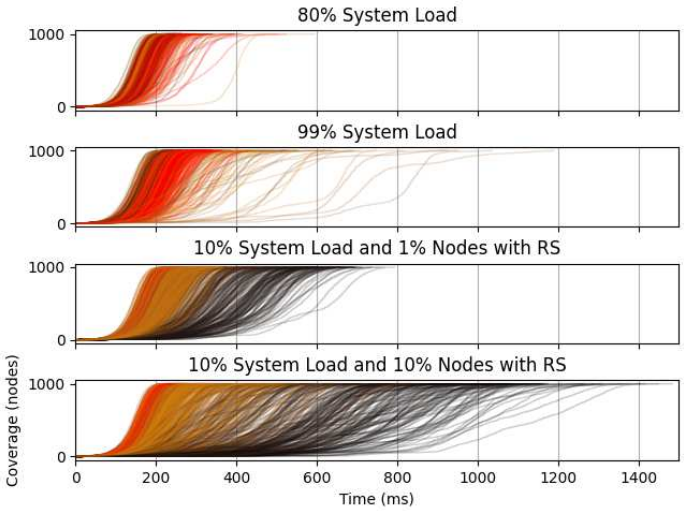


Fig. 25. Block dissemination behavior with flooding under varying system loads showing vastly more outliers with rate setting. However, blocks from high mana nodes generally disseminate the fastest. Origin nodes are color-coded according to Fig. 3.

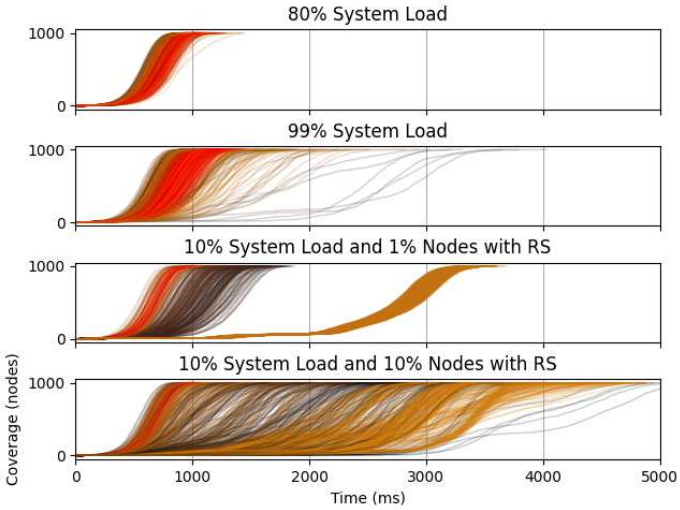


Fig. 26. Block dissemination behavior with gossip under varying system loads showing much more variance with rate setting. Origin nodes are color-coded according to Fig. 3.

caused by the scheduler since best-effort nodes have an average accumulated scheduling deficit of 0, whereas fair nodes have accrued maximum deficits and thus have their blocks scheduled faster, as described already in another paper (https://arxiv.org/abs/2005.07778) in Fig. 11. The variance between best-effort nodes is proportionally identical between flooding and gossip, as dissemination is slower by a factor of 4.

Generally, blocks require significant time to reach the first few nodes, then coverage increases exponentially before slowing down again for the final few. This makes intuitive sense since the dissemination behaves like a wave spreading through the network and gaining a significantly larger frontier over time. For slow outliers, especially in rate-setting scenarios, the dissemination seems to be delayed after succeeding in some initial coverage. This may be because the local representation of scheduling deficits is not quite accurate when compared to most other nodes, and the incurred delays caused by this inaccuracy are large when the system already operates at the maximum throughput.

Further separation of the nodes by issuing behavior in Fig. 27 shows that the outliers are exclusively issued by best-effort nodes, such that fair nodes are guaranteed to have speedy dissemination of their blocks. A similar prioritization can be observed for very high mana nodes, which reliably do not suffer from slow dissemination. This too can be attributed to the scheduling component as found in (https://arxiv.org/abs/2005.07778).

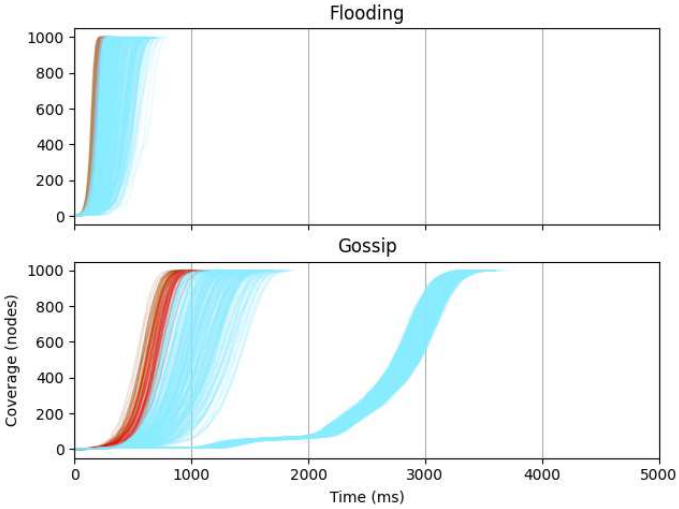


Fig. 27. Block dissemination behavior at 10% system load and 10 best-effort nodes designated as origin in blue. Fair nodes do not suffer from best-effort nodes using rate setting, and their blocks are consistently disseminated as fast as possible.

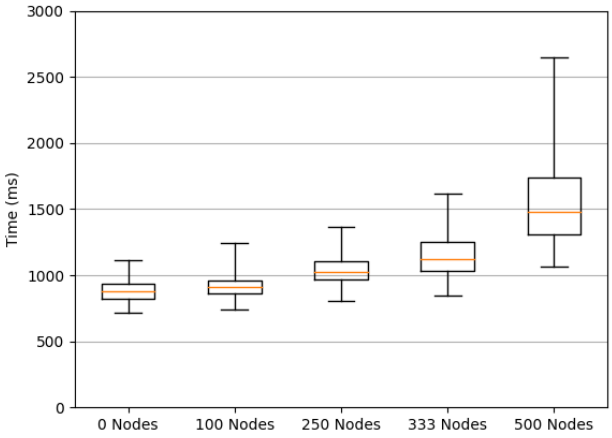


Fig. 28. Block dissemination speed to 95% of nodes at 80% system load where some randomly selected nodes are silent. The speed decreases steadily with higher numbers of affected nodes, and the variance between blocks increases as well.

### *D. Failure and Attack Resilience*

This section aims to show the impact of silent nodes on the performance of gossip with the new baseline parameters. Nodes that are not actively or correctly participating in information dissemination are the most straightforward and least defensible risk vector for a dissemination protocol. The base causes can differ wildly, from DoS attacks on these nodes to misconfiguration or potentially malicious voluntary non-participation. For simplicity, we assume here that the affected nodes do not forward any blocks except their own, which they continue to issue and therefore operate selfishly.

With randomly selected silent nodes, the system remains perfectly operational and well connected until one-third of nodes are affected, from that point onward a very small minority of nodes can not be reached reliably anymore. Dissemination to other nodes works well, increasing by 30% for 333 silent nodes and by 77% when half of all nodes are affected, as seen in Fig. 28.

The network load is reduced with higher failure rates, but singular anomalous nodes suffer under much larger loads, nearing the levels caused by flooding. This is likely related to the few very poorly connected nodes in these scenarios and their peers which are now the effective suppliers of all blocks for nearly all their peers.

Selecting silent nodes in a more coordinated manner by their betweenness or closeness value in the network topology heightens the impact on the system. Collusion by betweenness centrality, defined as the share of all shortest paths passing through that node, is increasing dissemination times by a further 35%. Much less effective proved collusion by closeness, taking into account the average distance to all nodes, which performs only marginally worse than random selection. This latter approach isolates far fewer nodes and affects mostly

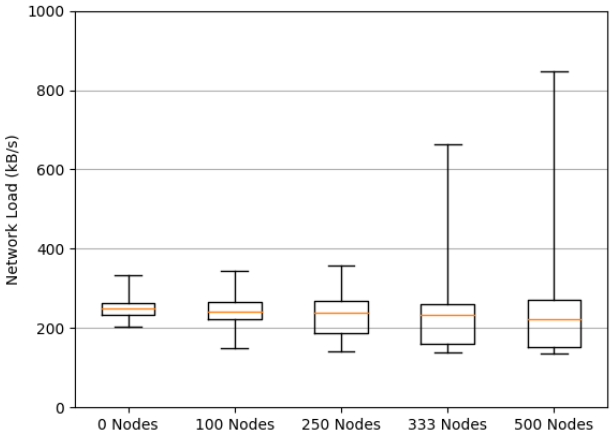


Fig. 29. Network load per node at 80% system load where some randomly selected nodes are silent. Generally, the load decreases with fewer participating nodes, but some nodes incur much higher loads to balance out failed propagation paths elsewhere.

those routes going through the densest, most well-connected part of the network where easy alternatives can be found.

*1) Impact of Gossip Parameters:* Can less optimistic gossip parameters lessen the impact of silent nodes on the system? The obvious candidates for these changes are shorter solidification delays, longer expiry periods, and higher offer probabilities, each improving the chance that blocks are offered more reliably in parts of the network where active nodes are poorly connected. However, Fig. 31 shows that the actual impact is very minimal. Therefore it seems unlikely that the consequences of silent nodes can be attributed to the chosen gossip parameters in a significant way, and instead depend mostly on the network topology and general gossip strategy.

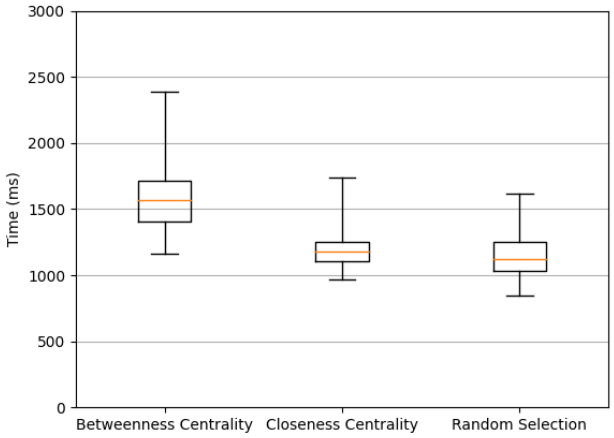


Fig. 30. Block dissemination speed to 95% of nodes at 80% system load with 333 silent nodes selected according to various collusion strategies. Nodes that are crossed by many shortest paths have a higher impact on performance.

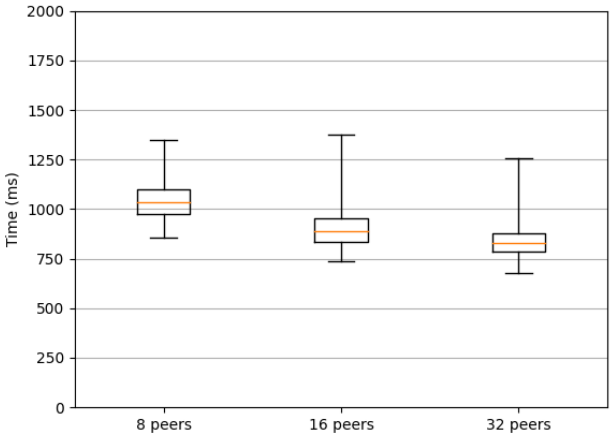


Fig. 32. Block dissemination speed with topologies of higher degrees, showing clear benefits from more peering connections.

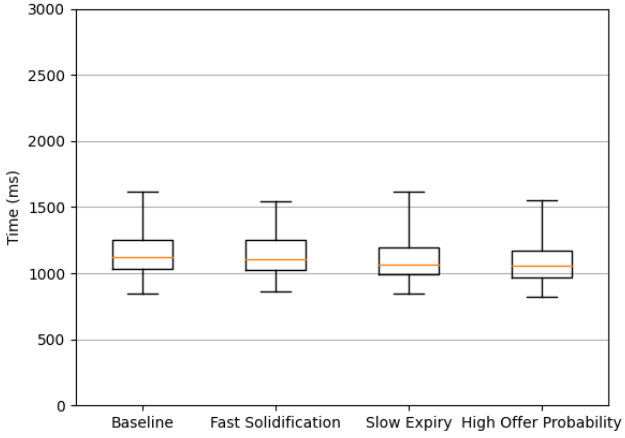


Fig. 31. Block dissemination speed to 95% of nodes at 80% system load where 333 random nodes are silent, with varying gossip parameters. There is little difference, with faster offers performing best.

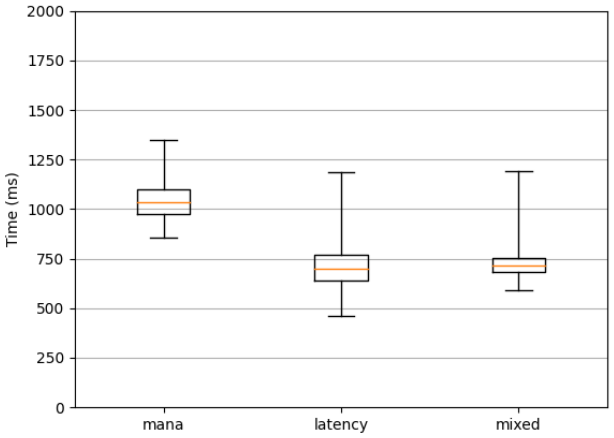


Fig. 33. Block dissemination speed with topologies formed of different candidates. Mixed candidate sets perform as well as latency-based ones in average and worst cases while improving substantially on mana-based peering.

### *E. Topology Parameters*

During the search for the network topology configuration that most accurately represents the current intent in the specifications in section V, other potentially more advantageous alternatives were presented. Through the adoption of gossip even more options become available that previously were too ingrained in flooding to consider. These strategies contribute to reducing the speed penalty that the gossip protocol incurs over flooding and additionally strengthen the resilience of the network.

Since gossip takes a subset of peers each round, the total number of active peering connections is less relevant than it was for flooding. It does however have a major impact on resilience, as it directly represents the cost of eclipsing any node and further interconnects the network. Simply choosing 16 or 32 peers instead of 8 from the same set of candidates improves the dissemination time of blocks by over 20% without incurring any additional cost in load as shown in Fig. 32. This is likely due to the average path latency between nodes decreasing since more potential paths are available now, even when the peering decision does not take latency into account.

Changing the similarity threshold of the specified peering mechanism has very little positive effect, as the current topology is already very well connected, only slightly isolating high mana nodes as intended by the specifications. Instead, the speed decreases if the threshold is chosen too low, creating longer propagation routes between high and low mana nodes.

However, applying different peering options from Section V, especially those that directly consider latency, results in extensive further benefits, as dissemination is 33% faster. Interestingly, peering based on a mixed candidate set is even faster than strictly considering only the latency, as the network now contains routes with low latency as well as those containing few hops, which reduces the delay incurred by the congestion control components at each node.

## IX. Conclusions

This research aims to optimize the gossip layer of the IOTA 2.0 network by exploring the current system, identifying a fitting gossip protocol, and evaluating its performance and resilience. Investigating the current system performance with flooding reveals the strong traffic filtering impact of the scheduling component on the dissemination behavior. Push-Pull gossip emerges as a good fit, partially due to available message digests and easy database lookups. Its evaluation within the IOTA system shows that it behaves similarly to flooding and can therefore be transparent to other protocol parts. It offers a worthwhile alternative and improves upon flooding concerning efficiency, incurring a substantially lower network load while remaining sufficiently resilient.

The switch to gossip also introduces the ability to tune its behavior while gaining increased independence from other protocol parts compared to flooding. As a consequence, the peering mechanism now shows substantial potential to be further optimized, in turn weakening the speed deficiency of gossip, for example by increasing the number or changing the selection criteria of peers. We observe that the network topology defines the limits of gossip protocol potential but also embodies the mechanism by which its advantages can be fully exhausted and its weaknesses remedied.

## X. Future Work

### A. Mixed Peering

Following the initial experiments in Section V-C4, more peering mechanisms and candidate set mixing strategies should be investigated.

### B. Improved Link Simulation

One shortcoming of the simulation setup used throughout this project is the simplified communication layer based on the unit square approach as described in Section V-A. More accurate modeling of the Internet including communication failures, intermittent connectivity, and higher churn as is often characteristic of DLT networks could be useful to discover potential additional design requirements and more accurately judge the performance of the gossip protocol. Some gossip parameters like the expiry period are chiefly aimed to provide resilience against these types of communication failures and could only be optimized here to a limited degree.

### C. Structured Gossip

As briefly discussed in Section VII-C structured gossip may provide further advantages by constructing an optimized overlay network. This likely sacrifices resilience to some degree as some nodes become more essential than others.

### D. Smart Peer Subset Selection

Finally, we propose a better way to select the peer subset. Instead of random selection, nodes can choose those peers that profit most from being contacted. This difference can be ascertained without relying on external information, simply through the data inherently produced by the gossip protocol. By requesting offered blocks through the three-phase exchange nodes necessarily share which specific blocks were previously unknown to them. Peers can then choose to prioritize contact with nodes that requested a higher share of offered blocks over the recent past. However, this would likely incur a higher load at the chosen nodes since they are selected more frequently by their peers. In turn, there may be the potential for exploitation or disruption by requesting blocks that are not needed, thereby influencing the knowledge base upon which contact decisions are made. Finally, such an approach may encounter problems in environments with high churn, which remains to be properly investigated.


## Acknowledgment

I am extremely grateful to my supervisors Dr. Maarten Everts and Prof. Dr. Ir. Maarten van Steen for their continuous assistance and patience. I am also grateful to Luigi Vigneri of the IOTA Foundation for his support and insightful discussions. Lastly, I would like to recognize my friends who motivated me to complete the final bits in a reasonable time.


## References


[1] Popov, S., Moog, H., Camargo, D., Capossele, A., Dimitrov, V., Gal, A., ... & Attias, V. (2020). The coordicide. Accessed Jan, 1-30.

[2] Theis, J., Vigneri, L., Wang, L., & Trivedi, A. (2020, December). Healthor: Protecting the Weak in Heterogeneous DLTs with Health-aware Flow Control. In Proceedings of the 4th Workshop on Scalable and Resilient Infrastructures for Distributed Ledgers (pp. 5-8).

[3] Popov, S., & Buchanan, W. J. (2021). Fpc-bi: Fast probabilistic consensus within byzantine infrastructures. Journal of Parallel and Distributed Computing, 147, 77-86.

[4] Vigneri, L., Welz, W., Gal, A., & Dimitrov, V. (2019, May). Achieving fairness in the tangle through an adaptive rate control algorithm. In 2019 IEEE International Conference on Blockchain and Cryptocurrency (ICBC) (pp. 146-148). IEEE.

[5] Cullen, A., Ferraro, P., Sanders, W., Vigneri, L., & Shorten, R. (2021). Access control for distributed ledgers in the internet of things: A networking approach. IEEE Internet of Things Journal.

[6] Müller, S., Capossele, A., Kuśmierz, B., Lin, V., Moog, H., Penzkofer, A., ... & Welz, W. (2021, September). Salt-based autopeering for DLT-networks. In 2021 3rd Conference on Blockchain Research & Applications for Innovative Networks and Services (BRAINS) (pp. 165-169). IEEE.

[7] https://github.com/iotaledger/IOTA-2.0-Research-Specifications

[8] https://github.com/ethereum/devp2p

[9] https://files.iota.org/papers/ISC_WP_Nov_10_2021.pdf

[10] Manevich, Yacov, Artem Barger, and Yoav Tock. "Endorsement in Hyperledger Fabric via service discovery." IBM Journal of Research and Development 63.2/3 (2019): 2-1.

[11] Eugster, Patrick T., et al. "Epidemic information dissemination in distributed systems." Computer 37.5 (2004): 60-67.

[12] Birman, Ken. "The promise, and limitations, of gossip protocols." ACM SIGOPS Operating Systems Review 41.5 (2007): 8-13.

[13] Kiffer, Lucianna, et al. "Under the hood of the ethereum gossip protocol." International Conference on Financial Cryptography and Data Security. Springer, Berlin, Heidelberg, 2021.

[14] Antwi, Robert, et al. "A Survey on Network Optimization Techniques for Blockchain Systems." Algorithms 15.6 (2022): 193.

[15] Serena, Luca, et al. "Simulation of dissemination strategies on temporal networks." 2021 Annual Modeling and Simulation Conference (ANNSIM). IEEE, 2021.

[16] Qiu, Haoran, et al. "A Geography-Based P2P Overlay Network for Fast and Robust Blockchain Systems." IEEE Transactions on Services Computing (2022).

[17] Santiago, Carlos, and Choonhwa Lee. "Accelerating message propagation in blockchain networks." 2020 International Conference on Information and Communication Technology Convergence (ICTC). IEEE, 2020.

[18] Berendea, Nicolae, et al. "Fair and efficient gossip in hyperledger fabric." 2020 IEEE 40th International Conference on Distributed Computing Systems (ICDCS). IEEE, 2020.

[19] Vu, Huy, and Hitesh Tewari. "An efficient peer-to-peer bitcoin protocol with probabilistic flooding." International Conference for Emerging Technologies in Computing. Springer, Cham, 2019.

[20] Rodrigues, Luis, et al. "Adaptive gossip-based broadcast." International Conference on Dependable Systems and Networks. No. CONF. 2003.

[21] D'Angelo, Gabriele, Stefano Ferretti, and Moreno Marzolla. "Adaptive event dissemination for peer-to-peer multiplayer online games." arXiv preprint arXiv:1102.0720 (2011).

[22] Burmester, Mike, Tri Van Le, and Alec Yasinsac. "Adaptive gossip protocols: Managing security and redundancy in dense ad hoc networks." Ad Hoc Networks 5.3 (2007): 313-323.

[23] Delgado-Segura, Sergi, et al. "Analysis of the bitcoin utxo set." International Conference on Financial Cryptography and Data Security. Springer, Berlin, Heidelberg, 2018.

[24] Bentov, Iddo, et al. "Proof of activity: Extending bitcoin's proof of work via proof of stake [extended abstract] y." ACM SIGMETRICS Performance Evaluation Review 42.3 (2014): 34-37.

[25] Pretre, Baptiste. "Attacks on peer-to-peer networks." Dept. of Computer Science Swiss Federal Institute of Technology (ETH) Zurich Autumn (2005).

[26] Montresor, A., & Jelasity, M. (2009, September). PeerSim: A scalable P2P simulator. In 2009 IEEE Ninth International Conference on Peer-to-Peer Computing (pp. 99-100). IEEE.

[27] Mao, Y., Deb, S., Venkatakrishnan, S. B., Kannan, S., & Srinivasan, K. (2020, July). Perigee: Efficient peer-to-peer network design for blockchains. In Proceedings of the 39th Symposium on Principles of Distributed Computing (pp. 428-437).

[28] Loguinov, D., Kumar, A., Rai, V., & Ganesh, S. (2003, August). Graph-theoretic analysis of structured peer-to-peer systems: routing distances and fault resilience. In Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications (pp. 395-406).

[29] Wang, F., Moreno, Y., & Sun, Y. (2006). Structure of peer-to-peer social networks. Physical Review E, 73(3), 036123.

[30] Bui, T. N., Chaudhuri, S., Leighton, F. T., & Sipser, M. (1987). Graph bisection algorithms with good average case behavior. Combinatorica, 7, 171-191.