# Access Control in Adversarial Environments for IoT-oriented Distributed Ledgers

Andrew Cullen, Pietro Ferraro and Robert Shorten
*Dyson School of Design Engineering*
*Imperial College London*
London, United Kingdom
{a.cullen19, p.ferraro, r.shorten}@imperial.ac.uk

William Sanders and Luigi Vigneri
*IOTA Foundation*
Berlin, Germany
{william.sanders, luigi.vigneri}@iota.org

***Abstract*—The surge of Internet of Things (IoT) applications requires distributed systems capable of securely exchanging messages and immutably recording data. While traditional blockchain architectures were not designed with IoT in mind, more recent Distributed Ledger Technologies (DLTs), which are not based on proof of work, can be considered as a solution to deal with the above requirements. However, without proof of work, these ledgers require an explicit way to manage the rate at which messages are issued and disseminated.**

**In this work, we present an access control scheme for IoT-oriented DLTs, that is the mechanism used to choose which messages can be written to the ledger. Our approach aims to efficiently exploit the available network resources (bandwidth, processing power) and to guarantee fair access depending on node reputation. While these concepts have already been touched by well-known areas of networking research, such as TCP and quality of service, in DLT networks this problem is harder since nodes cannot trust familiar feedback measurements, such as packet acknowledgements or congestion notifications. In this paper, we design a completely decentralised mechanism which involves a round robin-based scheduler and a TCP-inspired rate setter. Extensive simulations show that our approach provides fair access, guarantees that all honest nodes eventually receive the same messages, and makes sure that malicious nodes cannot degrade performance or affect security.**

## I. Introduction

The Internet of Things (IoT) paradigm is becoming increasingly prevalent in both industrial and academic communities. IoT devices are now ubiquitous, including smart home and personal devices, a vast array of urban and industrial sensors, and intelligent vehicles. Many of these IoT devices must immutably record data or conduct secure financial transactions, such as machine-to-machine payments. Centralised infrastructures fail to efficiently deal with such use cases, introducing privacy and data integrity concerns and leading to the emergence of monopolies in certain domains.

Blockchains [1] and, more generally, Distributed Ledger Technologies (DLTs) represent an attractive alternative to solve the aforementioned issues. A DLT is a *trustless* peer-to-peer network where nodes store a local copy of a database, called a ledger. Trustless means that nodes do not need to trust any other individual node, but must only trust that the system as a whole is functioning correctly. This is a fundamental property of DLTs which allows nodes to reach consensus on the state of the ledger without the aid of a central entity. In the case of blockchains, the first generation of DLTs, the ledger is stored in blocks, each one cryptographically linked to the previous one. To guarantee the security of the system, blocks should only be created one at a time, and the time between blocks should be reasonably large. This process is too slow and inefficient for IoT devices which would have to rely on larger, more centralised nodes to add information to the ledger such as Bitcoin's miners [2].

Recently, as a generalisation of the blockchain structure, it has been proposed that each new transaction is cryptographically linked to *two or more* existing transactions building a DAG structure for the ledger. Unlike blockchains where access to write into the ledger is limited at protocol level by the block size and by some resource based leader election mechanism (e.g., Bitcoin's proof of work), in DAG-based DLTs access is not intrinsically controlled and throughput is usually only bounded by the resources available. The envisioned performance improvements offered by DAGs suggest that they are more suitable for the IoT, but also requires an explicit access control in order to efficiently exploit the limited resources and be suitable for low-power devices. An example of such IoT-oriented DLTs is provided by IOTA [13], where the proportion of resources allocated to a node (i.e., the capability of issuing new transactions) is proportional to node's *reputation*[1] rather than its computational capabilities as in proof of work DLTs.

At a high level, in DAG-based DLTs nodes validate transactions, add them to the ledger, and run some consensus algorithm to ensure a consistent ledger is maintained. We refer to the set of these tasks as *ledger writing* bottleneck. The specifics of the ledger writing bottleneck vary across DLT implementations and may even vary from node to node. For example, in certain DLTs some nodes may do the most computationally heavy tasks, while other limited nodes, like IoT devices, perform lighter tasks while writing. The lack of an intrinsic access control in IoT-oriented DLTs introduces unique challenges, requiring transaction throughput to be explicitly

[1]Node's reputation is an uninflatable number which is assigned to each identity. A DLT can calculate this number according to node's behavior, its stake in the system or any other objective metric. In this work, the reputation system is used as an input parameter, and any further research is out of the scope of this paper.

978-3-903176-32-4 

managed. Without access control, unwritten transactions could accumulate in queues causing delays and discrepancies between nodes' local ledgers. Moreover, overwhelmed nodes may drop transactions, potentially causing inconsistencies in the ledger. While our problem presents some similarities with congestion control for packet-switched networks, here three additional requirements make the problem challenging:

- *Consistency*: If a transaction is written by one correct node, it should be written by all correct nodes, within some delay.
- *Fairness*: All nodes should get a fair share of throughput, weighted by the nodes' reputation [3], achieving max-min fairness in the number of transactions issued by each node during congestion.
- *Security*: The above requirements should be retained even in adversarial environments where malicious nodes attempt to gain more than their fair share of resources.

In this paper we propose an access control scheme for DAG-based DLTs in adversarial environments which efficiently exploit all available resources (bandwidth, processing power, storage) while satisfying all the above requirements. Our proposal has two core components: (i) a TCP-inspired algorithm for decentralised rate setting; (ii) a scheduler based on Deficit Round Robin (DRR) [8]. The mechanism is based on the crucial idea that nodes can adapt their fair traffic share through (i) according to the varying traffic conditions. While this task would be unfeasible in traditional packet-switched networks, the overall congestion status of the DLT network can be accurately inferred as all transactions must be delivered to all nodes. Additionally, the scheduler in (ii) censors transactions issued by malicious or selfish nodes which are attempting to exceed their allowed throughput. Our scheme is also applicable to any distributed database replication architecture with the requirements listed above.

Following a brief summary of related work, in Section II, we provide a node model capturing the main bottleneck of writing transactions in Section III. Then, in Section IV, we present our access control scheme, introducing the rate setting algorithm and the DRR scheduler. After that, in Section V, we describe a simulator which demonstrates that the proposed scheme satisfies the requirements. In Section VI, we summarise our results and propose directions for future work.

## II. Related Work

Placing this work in the context of the existing literature is challenging, as access control for DLTs lies at the intersection of DLTs, and many topics within the networking community, such as Quality of Service (QoS), TCP, network security, gossip protocols. Specifically, our solution employs a scheduler and a decentralised rate setting algorithm, so we discuss some instances of these in the networking literature.

A similar algorithm, in spirit, to the one proposed in this paper, but unsuitable for our requirements, can be found in [4]. The authors analyse a number of epidemic algorithms and present a 'flow control' algorithm for a replicated database. Their algorithm addresses how nodes should adaptively control their update rate to avoid backlogs. However, the algorithm in [4] is unsuitable for the trustless DLT setting for the following reasons: firstly, congestion is detected by overflows of updates in the buffer, which could result in inconsistency across nodes; additionally, fairness is achieved in [4] through communication and agreement with neighbouring nodes, which is not a reasonable assumption in our trustless setting. Our problem is also closely linked to resource management in several network architectures, but our DLT setting presents new challenges, as well as opportunities. Namely, our DLT setting does not permit the assumption of trusted communication between nodes, rendering traditional means of detecting congestion, such as acknowledgements, vulnerable to attack.

From the perspective of achieving QoS in IP networks, the well-known differentiated services (DiffServ) [5] and integrated services (IntServ) [6] architectures offer opposing approaches: DiffServ offering approximate, but scalable and interoperable differentiated treatment of traffic; IntServ permitting more precise, though less scalable, control through explicit reservation of network resources. These solutions heavily rely on a backbone of trusted routers and are therefore unsuitable for our problem. A common thread for these and many modern QoS solutions is the use of packet scheduling at routers to protect honest network users from the congestion caused by malicious flows. Solutions range from the fine-grained and complex Weighted Fair Queueing (WFQ) [7] to the course-grained and efficient DRR [8]. FQ-CoDel [9] provides a combined packet scheduler and active queue management system to prevent high latency in packet-switched networks caused by excess buffering of packets, problem known as bufferbloat [10]. In various forms of TCP, nodes additively increase their packet rate until congestion is detected, at which point they multiplicatively decrease it. However, TCP requires some form of feedback from congested routers, either through acknowledgements [11], or explicit congestion notifications (ECNs) [12], which make them unsuitable for our trustless setting.

## III. Node and Network Model

We denote the set of all nodes participating in the network as $\mathcal{M}$. Each node, $m \in \mathcal{M}$, has a set of neighbours, $\mathcal{N}_m \subset \mathcal{M}$, with which it communicates directly over a secure bidirectional channel. The data shared by nodes are referred to as *transactions* and can include, for example, signed updates of account balances, or IoT sensor data. A transaction is referred to as *disseminated* when it is present in all ledgers. Conversely, undisseminated transactions are those which have been created but are not yet disseminated. The rate at which node $i$'s transactions are disseminated is denoted $D_i$, and the dissemination rate of all transactions is denoted $D$. Dissemination rate can be thought of as a measure of network throughput in DLTs. $D$ is bounded, in the long run, by the writing bottleneck at nodes and will be the primary metric we use to evaluate the performance of our access control scheme. Figure 1 illustrates the model of a node, $m$, and

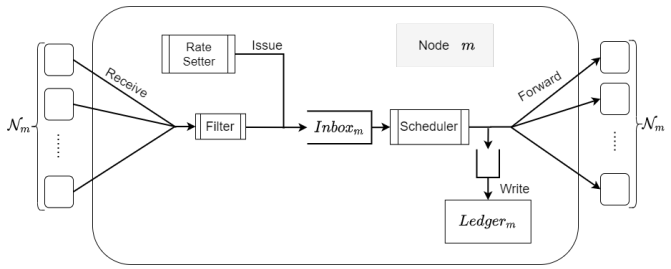


Fig. 1. Model for a node $m$.

its neighbours. Some of the key notation for the following section is outlined in Table I. Each node has an identity and an associated reputation. Examples of reputation systems which fit our model include reputation directly linked to a node's wealth, delegated forms of reputation, such as the *mana* system employed in the IOTA network [13], or permissioned DLTs with an elected consortium of reputable nodes [14]. Node $m$'s reputation is denoted by $rep_m$. The reputation distribution is assumed to be known by all nodes. Additionally, in this work, reputation is assumed not to vary with time.

The relevant operations performed on transactions, by nodes, are: *issuing*; *receiving*; *forwarding*; *writing*.

### A. Issuing Transactions

Transactions are issued and cryptographically signed by nodes, linking that transaction to the issuer. This means that a receiving node can identify the node from which any transaction originates. The rate at which node $m$ issues new transactions is denoted by $\lambda_m$. As we shall describe below, we impose a global limit, $\nu$, on transaction writing in order to ensure consistency, and each node is entitled to a minimum fraction of this, proportional to its reputation. In other words, node $m$ has an assured rate, $\tilde{\lambda}_m$, defined as:

$$\tilde{\lambda}_m = \frac{\nu \cdot rep_m}{\sum_{i \in \mathcal{M}} rep_i}. \tag{1}$$

Note that if all nodes always had transactions to issue, or could exchange trusted communications, the problem of access control would be trivial: specifically, each node could either set their issue rate to $\lambda_m = \tilde{\lambda}_m$, or coordinate a fair issue rate with their peers, as in [4]. In reality, many nodes are likely to be offline at various times, while other nodes have additional transactions to issue, resulting in under-utilisation of network resources if a fixed rate were to be used. Thus, in the presence of varying traffic conditions and a trustless, decentralised environment, managing transaction issue rate becomes considerably more difficult. To capture these varying traffic conditions, we define four modes of operation for nodes issuing transactions:

- *Inactive:* Not issuing any transactions i.e. $\lambda_m = 0$.
- *Content:* Issuing transactions at a fixed rate $\lambda_m \leq \tilde{\lambda}_m$. This is modelled as a Poisson process with rate parameter $\lambda_m$, which is a standard model for arrival processes.
- *Best-effort:* Issuing transactions at the highest rate possible under the current traffic conditions, without causing excessive congestion. This requires a node to use the rate setting algorithm, outlined in Section IV, to utilise unused network resources and adaptively set $\lambda_m > \tilde{\lambda}_m$.
- *Malicious:* Issuing transactions at a rate $\lambda_m \gg \tilde{\lambda}_m$, without concern for the congestion caused.

**Note:** The issue rate of a node can be capped to prevent certain malicious behaviour, such as spamming. The interested reader can refer to [15] for more details.

### B. Receiving and Forwarding Transactions

Nodes receive transactions from their neighbours, and also forward transactions to their neighbours, as illustrated in Figure 1. Each node maintains an inbox buffer, denoted $Inbox_m$, which contains transactions received from neighbours and issued by itself. Transactions are filtered before being added to the inbox, to prevent spam attacks [15]. We denote by $Inbox_m(i)$, the set of transactions issued by node $i$ in $m$'s inbox buffer. The size of each buffer is finite, and each node, $m$, should ensure that $Inbox_i(m)$ at other nodes $i \in \mathcal{M}$ does not become too large, as this could lead to excessive queuing delays. We assume that *flooding* is employed here for forwarding of transactions, meaning that all new information is forwarded to all neighbours (except the source node) without regard for whether or not neighbours already possess it. Hence, the stream of transactions from $m$ to $i$ consists of transactions issued by all nodes (except those received from $i$ itself). We refer to a sequence of transactions, issued by node $i$, as $i$'s flow. Gossip protocols can be employed in some DLT networks, to reduce forwarding overhead, but we defer the discussion of such optimisations to future work.

### C. Writing Transactions

When a node is informed of a new transaction, a series of actions must be taken to add the transaction to the node's local copy of the ledger. We refer to these steps as writing, and the details of what is required to write a transaction will depend on the underlying ledger structure, and details which are specific to the DLT implementation. We note that, fundamentally, if a transaction is to become part of the distributed ledger, it must be written by every node (while DLTs with sharding are an exception to this rule, our solution still applies to each single shard). For this reason, we impose a global transaction writing rate, $\nu$, to ensure consistency. Transactions from $Inbox_m$ are scheduled for writing at a deterministic rate, $\nu$, by the scheduling algorithm described in Section IV-A. Transactions are added to a writing buffer after they are scheduled, so although the time required to write transactions will vary, nodes must ensure that a writing rate of $\nu$ can be achieved in the long run to ensure this buffer from overflowing.

## IV. Access Control

Our access control scheme seeks to maximise resource utilisation while ensuring that the requirements laid out in Section I, namely consistency, fairness and security, are met. The two core components of our solution are a scheduling algorithm and a rate setting algorithm:

TABLE I
NOTATION FOR NODE AND NETWORK MODEL.

| | |
|---|---|
| $\mathcal{M}$ | set of all nodes in the network |
| $\mathcal{N}_m$ | set of all nodes that are neighbours of node $m$ |
| $D$ | total dissemination rate |
| $D_i$ | dissemination rate of $i$'s transactions |
| $\nu$ | global transaction writing rate |
| $rep_m$ | reputation of node $m$ |
| $\lambda_m$ | issue rate of node $m$ |
| $\tilde{\lambda}_m$ | assured issue rate of node $m$ |

*Scheduling:* The goal of this component is to schedule transactions, issued by each node $i \in \mathcal{M}$, at a rate proportional to $rep_i$. In other words, we wish to achieve weighted max-min fairness [3] in the writing rate across issuing nodes. This ensures that, for a node $m$, issuing transactions at an appropriate rate relative to its reputation, the aforementioned requirements will be met: namely, $m$'s transactions will not become backlogged at any node, so consistency will be ensured; $m$'s fair share of the network resources are allocated to it, guaranteeing fairness; malicious nodes sending above their allowed rate will not interrupt $m$'s dissemination rate, fulfilling the security requirement.

*Rate setting:* The rate setting seeks to allow *best-effort* nodes (see Section III) to issue at a rate above their assured rate, $\tilde{\lambda}_m$, without causing excessive congestion elsewhere, which could cause a violation of the consistency requirements. We also wish to maintain weighted max-min fairness among *best-effort* nodes, so that nodes can claim a portion of the available dissemination rate proportional to their reputation.

### A. Scheduling Algorithm

Nodes in our setting are capable of more complex and customised behaviour than a typical router in a packet-switched network, but our scheduler must still be lightweight and scalable due to the potentially large number of nodes requiring differentiated treatment. It is estimated that over 10,000 nodes operate on the bitcoin network[2], and we expect that an even greater number of nodes are likely to be present in the IoT setting. For this reason, we adopt a scheduler based on DRR[3] [8].

The scheduling algorithm is described in Algorithm 1. Transactions are scheduled at a maximum rate $\nu$, and the only other parameter, the quantum $Q_i$ should be set proportional to the correpsonding node's reputation $rep_i$.

### B. Rate Setting Algorithm

If all nodes always had transactions to issue, the problem of rate setting would be very straightforward: nodes could simply operate in *content* mode i.e. at a fixed, assured rate, $\tilde{\lambda}_m$. The scheduling algorithm ensures that this rate is enforceable and that increasing delays or dropped transactions are only experienced by misbehaving node. However, it is highly unlikely that all nodes will always have transactions to issue,

[2]https://bitnodes.io/

[3]DRR is implemented in Linux as a part of the FQ-CoDel packet scheduler, and efficiently supports up to 65535 separate queues [9].

**Algorithm 1** DRR Scheduler

*Repeat for $i \in \mathcal{M}$ in a round robin cycle:*
1: **if** $|Inbox_m(i)| > 0$ **then**
2: $DC_m(i) \leftarrow DC_m(i) + Q_i$
3: **if** $DC_m(i) \geq 1$ **and then**
4: Schedule a transaction from $Inbox_m(i)$
5: $DC_m(i) \leftarrow DC_m(i) - 1$
6: Wait $\frac{1}{\nu}$ seconds
7: **end if**
8: **end if**

and we would like *best-effort* nodes to better utilise network resources, without causing excessive congestion and violating requirements.

Our rate setting algorithm, for *best-effort* nodes, is inspired by TCP — each node employs additive increase, multiplicative decrease (AIMD) rules to update their issue rate in response to congestion events [16]. However, in the trustless DLT setting, the traditional means of responding to congestion is compromised. For example, malicious nodes could attempt to deflate the issue rate of their neighbours by not sending acknowledgements, or sending illegitimate congestion notifications. We recall, however, that in distributed ledgers, all transaction traffic passes through all nodes, contrary to traffic typically found in packet switched networks and other traditional network architectures. Under these conditions, local congestion at a node indicates congestion elsewhere in the network. This observation is crucial, as it presents an opportunity for an access control scheme based entirely on local traffic.

Recall that when a node $m$ issues a transaction, it is added to its inbox buffer to be scheduled. Node $m$'s own transactions in its inbox, $Inbox_m(m)$, are then scheduled at a rate which depends on the other traffic present in the buffer. We observe that the length of $Inbox_m(m)$ gives an estimate of congestion in node $m$'s traffic, not only at its own inbox buffer but at $Inbox_i(m)$ for all properly behaving nodes $i \in \mathcal{M}$, within some network delay.

Algorithm 2 outlines the AIMD rules employed by each node to set their issue rate, and the parameters of the rate setting algorithm are outlined in Table II. Each node sets their own local additive-increase parameter based on the global increase rate $A$, and their reputation. An appropriate choice of $A$ ensures a conservative global increase rate which does not cause problems even when many nodes increase their rate simultaneously. Nodes wait $\tau$ seconds after a multiplicative decrease, during which there are no further updates made, to allow the reduced rate to take effect and prevent multiple successive decreases. Waiting after decreases is common in implementations of AIMD algorithms, such as sliding window flow control in TCP [11]. The rate is updated each time a transaction is scheduled which at rate $\nu$ when the inbox is not empty. At each update, node $m$ checks how many of its own transactions are in its inbox buffer and responds with a multiplicative decrease if this number is above a threshold, $L_m$, which is proportional to $m$'s reputation. If the number of

transactions in $Inbox_m(m)$ is below the threshold, $m$'s issue rate is incremented by its local increase parameter $\alpha_m$.

**Algorithm 2** AIMD Rate Setter (Best-effort Mode)

*Initialise node $m \in \mathcal{M}$:*

$\alpha_m \leftarrow A \cdot rep_m / \sum_{i \in \mathcal{M}} rep_i$

*Repeat each time a transaction is written (rate $\nu$):*

**if** $|Inbox_m(m)| > L_m$ **then**

  $\lambda_m \leftarrow \lambda_m \cdot \beta$

  Wait $\tau$ seconds for next update

**else**

  $\lambda_m \leftarrow \lambda_m + \alpha_m$

**end if**

TABLE II
SCHEDULER AND RATE SETTER PARAMETERS.

| | |
|---|---|
| $A$ | global additive increase parameter |
| $\beta$ | global multiplicative decrease parameter |
| $\tau$ | wait time parameter |
| $L_i$ | inbox length threshold for node $i$ ($\propto rep_i$) |

## V. SIMULATIONS

A simulator was built in Python to test our access control scheme [17]. We present results from two sets of 100 Monte Carlo simulations: the first set with only honest nodes; and the second set with malicious behaviour.

Our test network consists of 15 nodes, arranged in a random 4-regular graph i.e. each node has 4 random neighbours. Channel delays are random between 50 ms and 150 ms. We first consider a group of five nodes in each of the non-malicious modes of operation. Each group of five nodes has the reputation distribution $\{3, 2, 1, 1, 1\}$. We identify nodes by a letter representing their mode of operation and a number representing their reputation. For example, if a node is denoted *B2*, this is a best-effort node with reputation 2. Simulations begin at time $t = 0$. Content nodes, in these simulations, issue at their full assured rate $\lambda_i = \tilde{\lambda}_i$. Best-effort nodes begin the simulation issuing at their assured rate, and start to increase their rate, with the rate setting algorithm, after 10 seconds of the simulation. The parameters used for the access control scheme are given in Table III.

TABLE III
ACCESS CONTROL PARAMETERS.

| **Scheduler** | | **Rate Setter** | | | |
|---|---|---|---|---|---|
| $\nu$ | $Q_i$ | $A$ | $\beta$ | $\tau$ | $L_i$ |
| 10 | $rep_i$ | 0.1 | 0.5 | 2 | $2 \cdot rep_i$ |

### A. Honest Environment

Figure 2 shows: the dissemination rate of each node; the dissemination rate, scaled by assured issue rate, of each node; and the number of undisseminated transactions for each node. These plots reveal that fairness is achieved, and that the unutilised network resources are fairly distributed among the best-effort nodes. The content nodes' rates converge to their assured issue rate, as expected, and as each group has equal reputation, the best effort nodes converge to twice their assured rate, consuming the unused resources left by the inactive nodes. The plot of undisseminated transactions demonstrates consistency, because the protocol drops no transactions and the number of undisseminated transactions is bounded. Figure 3 shows that the overall dissemination rate approaches the maximum writing rate, and the mean delay is bounded. Although counter intuitive, the dissemination rate often exceeds the maximum writing rate of nodes $\nu$ because, although transactions must pass through every node, they have different start and end points. Figure 4 demonstrates that delays are kept low.

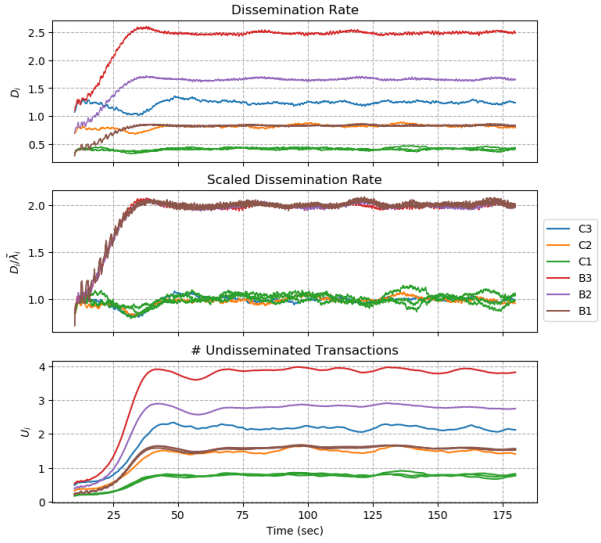


Fig. 2. Simulation set 1: dissemination rate, scaled dissemination rate, and number of undisseminated transactions over time (moving average over a ten second window).

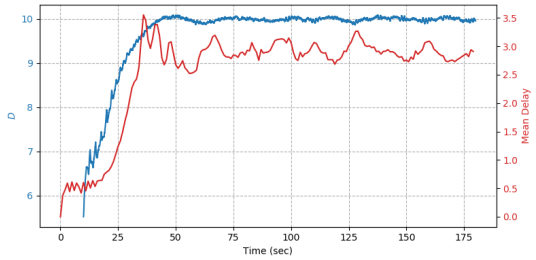


Fig. 3. Simulation set 1: dissemination rate (moving average over a ten second window) and mean transaction delay.

### B. Adversarial Environment

The remaining requirement to test is security. In the second set of simulations, one of the B1 nodes is switched from best-effort to malicious mode and issues transactions at five times its assured rate with all other parameters unchanged. Figure 5 shows that the dissemination rate of the malicious node is

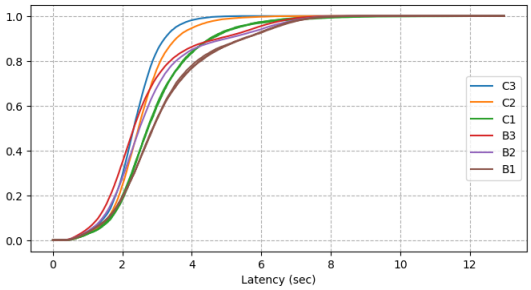


Fig. 4. Simulations set 1: CDF of transaction latency.

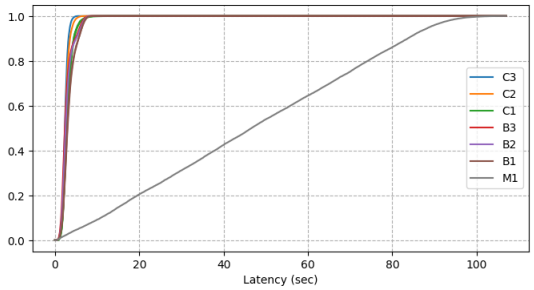


Fig. 6. Simulation set 2: CDF of transaction latency.

only slightly higher than the best-effort nodes. However, the number of undisseminated transactions issued by the malicious node is constantly growing as a result of large backlogs in its transactions in the buffers of non-malicious nodes. Excessively full inbox buffers detect malicious behaviour, and actions (such as blacklisting) can be taken to remove the malicious node from the network. Dealing with detected attackers is implementation-specific and will be a subject of future work. Additionally, Figure 6 shows that the malicious node only delays its own transactions. This set of simulations shows that a malicious actor can not violate the requirements of fairness and consistency, as this type of malicious behaviour is easily detectable. In other words, the security requirement is also satisfied.

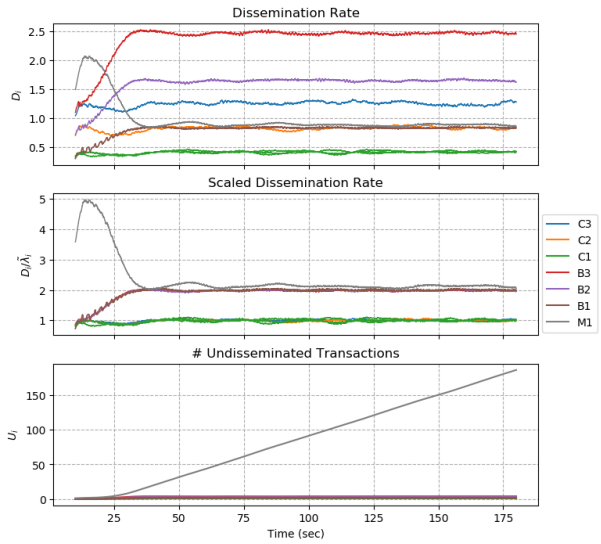


Fig. 5. Simulation set 2: dissemination rate, scaled dissemination rate, and number of undisseminated transactions over time (moving average over a ten second window).

## VI. Conclusions

We have presented an access control scheme for DLTs that utilises node resources optimally and minimises delay, satisfying the requirements of fairness, consistency and security. Our scheme presents an alternative to the expensive and wasteful PoW. We have evaluated its performance with extensive Monte Carlo analysis using a detailed agent-based Python simulator. Future research will address a broader range of simulation scenarios, as well as deployment of our algorithm on a DLT test network, such as the IOTA Foundation's GoShimmer network [18].

## References

[1] Nakamoto, S., "Bitcoin: A peer-to-peer electronic cash system", Available: https://bitcoin.org/bitcoin.pdf, 2008
[2] Cullen, A. and Ferraro, P. and King, C. and Shorten, R., "On the Resilience of DAG-based Distributed Ledgers in IoT Applications", *IEEE Internet of Things Journal*, 2020
[3] Hahne, E. L., "Round-robin scheduling for max-min fairness in data networks", *IEEE Journal on Selected Areas in communications*, Vol. 9, No. 7, pp. 1024–1039, 1991
[4] Van Renesse, R. and Dumitriu, D, and Gough, V. and Thomas, C., "Efficient reconciliation and flow control for anti-entropy protocols", in *Proceedings of the 2nd Workshop on Large-Scale Distributed Systems and Middleware*, pp. 1–7, 2008
[5] Grossman, D *et al.*, "New terminology and clarifications for diffserv", *RFC 3260*, 2002
[6] Bernet, Y. *et al.*, "A framework for integrated services operation over diffserv networks", *RFC 2998*, 2000
[7] Demers, A. and Keshav, S. and Shenker, S., "Analysis and simulation of a fair queueing algorithm", *Internetworking: Research and experience*, Vol. 1, No. 1, pp. 3–26, 1990
[8] Shreedhar, M. and Varghese, G., "Efficient fair queuing using deficit round-robin", *IEEE/ACM Transactions on networking*, Vol. 4, No. 3, pp. 375–385, 1996
[9] Hoeiland-Joergensen, T. and McKenney, P. and Taht, D. and Gettys, J. and Dumazet, E., "The flow queue codel packet scheduler and active queue management algorithm", *RFC 8290*, 2018
[10] Gettys, J. and Nichols, K., "Bufferbloat: Dark buffers in the internet", *ACM Queue*, Vol. 9, No. 11, pp. 40–54, 2001
[11] Postel, J. *et al.*, "Transmission control protocol", STD 7, *RFC 793*, 1981
[12] Floyd, S. and Jacobson, V., "Random early detection gateways for congestion avoidance", *IEEE/ACM Transactions on networking*, Vol. 1, No. 4, pp. 397–413, 1993
[13] Popov, S. *et al.*, "The Coordicide", Available: https://files.iota.org/papers/20200120_Coordicide_WP.pdf, 2020
[14] Androulaki, E. *et al.*, "Hyperledger fabric: a distributed operating system for permissioned blockchains", *Proceedings of the Thirteenth EuroSys Conference*, pp. 1–15, 2018
[15] Vigneri, L. and Welz, W., "On the Fairness of Distributed Ledger Technologies for the Internet of Things", in *IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, 2020
[16] Corless, M. and King, C. and Shorten, R. and Wirth, F., "AIMD dynamics and distributed resource allocation", Philadelphia, PA, USA: SIAM, 2016
[17] *DLT Congestion Control Simulator*, Available: https://github.com/cyberphysic4l/DLTCongestionControl, 2020
[18] *GoShimmer*, IOTA Foundation. Available: https://github.com/iotaledger/goshimmer, 2020