Reputation-Based Data Carrying for Web3 Networks

Stokkink, Quinten; Ileri, Can Umut; Pouwelse, Johan

DOI
10.1109/LCN53696.2022.9843374

Publication date
2022

Document Version
Final published version

Published in
Proceedings of the 47th IEEE Conference on Local Computer Networks, LCN 2022

Reputation-Based Data Carrying for Web3 Networks

Quinten Stokkink, Can Umut Ileri, Johan Pouwelse
Delft University of Technology, The Netherlands
Email: {q.a.stokkink, c.u.ileri, j.a.pouwelse}@tudelft.nl

2022 IEEE 47th Conference on Local Computer Networks (LCN) | 978-1-6654-8001-7/22/$31.00 ©2022 IEEE | DOI: 10.1109/LCN53696.2022.9843374

***Abstract*—Web3 networks are emerging to replace centrally-governed networking infrastructure. The integrity of the shared public infrastructure of Web3 networks is guaranteed through data sharing between nodes. However, due to the unstructured and highly partitioned nature of Web3 networks, data sharing between nodes in different partitions is a challenging task. In this paper we present the TSRP mechanism, which approaches the data sharing problem through nodes auditing each other to enforce carrying of data between partitions. Reputation is used as an analogue for the likelihood of nodes interacting with nodes from other partitions in the future. The number of copies of data shared with other nodes is inversely related to the nodes' reputation. We use a real-world trace of Twitter to show how our implementation can converge to an equal number of copies as structured approaches.**

***Index Terms*—infrastructure, integrity, reputation, Web3, peer-to-peer, partitioning, split**

I. Introduction

The "Web" is turning anarcho-capitalist. Centrally-governed infrastructure is no longer required to share files (BitTorrent), to communicate (OppNets), to transfer money (Bitcoin), or even to identify at passport strength (TCID [13]). The "Web3" movement goes beyond classical peer-to-peer solutions, seeking to turn our digital world into collaboratively maintained infrastructure [16]. However, as individualistic users only opportunistically interact with others, the classical issue of network partitioning resurges. A partitioned (split) network is incompatible with even the weaker consistency models, e.g., eventual consistency [1]. We present an emergent algorithm for Web3 networks to probabilistically guarantee that data is shared between network partitions by using reputation.

The reality of Web3 networks does not fit the connectivity model of peer-to-peer networks. The traditional way of thinking of peer-to-peer networks is that at least one path exists between any two nodes, allowing a rumor-mongering protocol to deliver information to all nodes. However, we observe a shift away from this connectivity assumption. Even a relatively small network like that of Bitcoin, in the order of $10\,000$ nodes [9], cannot keep up with the vast volume of user interactions and is seeing temporary partitioning, using aggregated "off-chain" interactions (e.g., the lightning network [10]). Beyond Bitcoin, partitioning has also been observed for Web3 digital assets in the interaction graph of Non-Fungible Tokens [8]. Web3 connection graphs should not be thought of as relatively static networks that periodically update connections. Instead, we should consider opportunistic networks with highly mobile users that frequently migrate between network partitions.

Web3 networks find themselves in a precarious balance between nodes' ability to maintain their public infrastructure and lowering communication costs. On the one hand, all data should be available to all nodes, allowing them to identify when the infrastructure degrades and to correct it (used in Bitcoin for instance). On the other hand, if overlays are optimized too much and data is never shared between network partitions, these networks become vulnerable to attacks again. Instead of requiring over 50% of the network to be malicious in order to degrade the infrastructure, an attacker would require only over 50% of a partition's size. Some data has to periodically be shared between partitions.

The main problem that we address in this paper is the need for data sharing between partitions in split networks with highly mobile nodes. We leverage reputation mechanisms to select nodes that are likely to traverse partitions to carry—and share—data. A node's reputation reflects the willingness of other nodes to interact with it. Therefore, nodes with a high reputation are more likely to share data with others than low-reputation nodes and make for prime carriers to share data between partitions. In contrast, more nodes with lower reputation should be used to have an equal chance of data being shared. Our contributions are as follows:

- We detail the requirements for Web3 data sharing solutions, and how they can be attacked (Section II).
- We motivate the design of our *Timely Sharing with Reputation Prototype* (TSRP) in Section III.
- We apply our prototype to a real-world trace of Twitter to show how the number of copies converges to the number for structured networks (Section IV).

II. Requirements analysis

We now derive requirements using three typical Web3 assumptions. Firstly, Web3 applications assume that over half of the network's nodes are not malicious. Secondly, nodes do not enter the system with prior trust for other nodes. Lastly, it is assumed that there exists no node with special authority.

Load-balancing over nodes is required to meet Web3's incentives. Without external regulation, peer-to-peer interactions lead to rich-get-richer dynamics [6], [15]. Therefore, if interactions are used to select nodes to carry data, nodes with more interactions become disproportionally burdened with data. To select nodes to carry data our proposal uses reputation, which is based on nodes' interactions. Therefore, simply selecting high-reputation nodes as carriers for data places an unequal burden on these nodes, effectively punishing them for a high reputation score. In turn, unequal burden leads

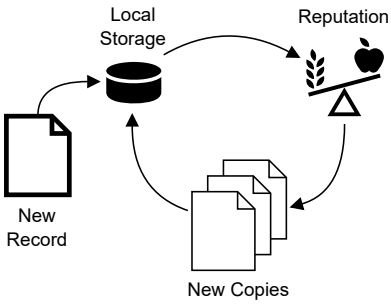

Fig. 1: New records and new copies from others are carried in a node's local storage between partitions based on reputation.

to unequal reciprocity between nodes, which is not compatible with peer-to-peer incentive mechanisms without payment [4]. Nevertheless, we do expect nodes with a high reputation to interact more frequently with other nodes and, therefore, to be more effective as carriers for data between partitions.

All nodes have a partial view of a Web3 system due to the lack of trust in data obtained from others. However, nodes do share logic that allows them to verify that the data they receive complies with a system's rules. This approach was pioneered by Bitcoin, in which every node adopts the longest chain of correctly hashed data. In a general sense, this corresponds to a Web3 system design pattern that consists of providing all nodes with an algorithm to verify the integrity of their common public infrastructure. However, nodes are never forced to accept data and may deviate from the shared logic.

Records are required to both be locally stored on nodes and be shared to avoid network partitioning in our system model. We use the term *record* to refer to all (possibly encrypted) data that is usable by a given Web3 application. Every instance of the same record is what we refer to as a *copy*. Both records proposed by a node and those copied from other nodes are stored in a node's *local storage*. The implementation of the local storage can be a simple database and may serve a higher-order data structure, like a blockchain.

Our system model consists of three components that form the feedback cycle shown in Fig. 1. First, any newly introduced record becomes part of the local storage of a node. A node decides what other nodes to share a record with based on its shared verification logic and by using its stored copies. Other nodes that receive a record verify that they should store a copy of the record in their own local storage using the same shared logic. The feedback cycle continues until nodes no longer want to, have to, or have space to, store a copy of the record.

The refusal to store records is the main attack a data sharing solution is required to resolve. Any node that does not store records should be detectable by any other node through shared verification logic. Nodes may then choose to not further interact with a node that does not store what it must store. We do not distinguish between nodes with malicious intent and non-malicious nodes that have simply run out of storage space, as both necessitate additional copies of a record in the network. However, specifically for our solution, the manner of calculating the change in reputation due to not storing a copy can differ per node. Nevertheless, the reputation of a node is still based on the copies it shares with other nodes.

III. Design

Our *Timely Sharing with Reputation Prototype* (TSRP) is designed to select nodes to carry copies of records based on nodes' reputation. Using both the identity and the reputation of a node, TSRP provides the shared logic to decide whether the node "must store" a given record (i.e., locally store a copy).

A. Overview

Every node using TSRP follows a simple two-step decision flow to determine whether a node (including the node itself) should act as a carrier for a given record. First, the distance between an identifier and a record is calculated using a distance function, which we call the *binding score*. Finally, the binding score is compared to the reputation of a node to decide whether it "must store" the given record.

The distance between an identifier and a record, i.e., the binding score, is used in a similar fashion to how DHTs decide which identifiers should store certain records. Typically, DHTs, like Kademlia [7], use XOR functions to bind nodes' identifiers to the records they should store. In particular, TSRP uses the Hamming distance, which is part of the class of symmetric XOR functions [2]. The binding score is used to provide the initial association of records with nodes and has nothing to do with the value, quality or importance of a record.

The binding score is made to resist index poisoning attacks, which consist of creating fake identities until one with a high binding score is found [12]. The score is grounded in values produced by a pseudo-random number generator (PRNG). The PRNG is seeded using both a record and a node identifier. Thereby, the generated values are always the same, regardless of which node evaluates it and when it is evaluated. Assuming attackers cannot pre-empt the creation of a record, this disallows index poisoning attacks through fake identities.

The reputation of each node is calculated individually by nodes it interacts with. As a result, the calculated reputations may differ wildly between nodes, depending on the node that calculates it. As the reputation of a node is a factor in deciding whether or not it should store a given record, nodes using TSRP may not come to the same decision on whether a particular node should store a particular record.

A node with a high reputation should be avoided when choosing carriers for records to achieve load balancing. In a peer-to-peer setting, the reputation of nodes is a reflection of the "work" they have done that has gone unreciprocated [14]. Therefore, selecting high reputation nodes to perform more "work", in the form of carrying records for TSRP, is not compatible with the incentives of Web3 applications (Section II). Instead, TSRP makes a node's reputation inversely related to the probability of it storing any record. New records are more frequently coupled to nodes with a low reputation.

B. Auditing

To detect nodes in the network that do not follow the shared storage logic, TSRP uses "audits". Each audit consists of a request for proof of an audited node storing the required records according to the storage logic, based on their estimated reputation and the auditor's known records. Audit proofs can be naively implemented by showing the record belonging to a hash, used within the scope of this paper. However, these proofs can also be implemented in the encrypted domain [18].

Failing an audit (i.e., not being able to show that a required record is stored locally) negatively affects the reputation of the audited node. Nodes that fail audits are punished through ostracization by setting their reputation score to 0 until they store the required records.

As Web3 users are egocentric and records quickly become stale, nodes are both expected and incentivized to remove records that are no longer interacted with. For example, in Bitcoin this staleness occurs after blocks have six descendants, at which point blocks are considered to be probabilistically finalized [11] and no longer require frequent auditing.

C. Formalization

In order to prove that our decision flow leads to availability of records through random carriers, we provide the concrete definitions of its elements. For any node i it can be calculated whether or not i should store a specific record d (a node may also decide this for its own identifier). To do so, the *binding score* $\mathcal{B}(d,i)$ is calculated. Basically, a node i must store a record d if its reputation $\mathcal{R}(i)$ is smaller than its binding score $\mathcal{B}(d,i)$ for d. If a node does not store what it must store, we consider it to be a malicious node. We formulate the store condition function S as follows:

$$S(d,i) \equiv \mathcal{R}(i) \leq \mathcal{B}(d,i) \tag{1}$$

This carrier-selection function S is deterministic. When given the same reputation $\mathcal{R}(i)$ and binding score $\mathcal{B}(d,i)$, all nodes will have the same output for S. However, nodes do not necessarily agree on the reputation $\mathcal{R}(i)$ of a node i.

The binding score $\mathcal{B}(d,i)$ is calculated for a record-identifier pair (d,i) in four transformations. First, the PRNG, seeded with the concatenation of the binary strings representing d and i, is used to generate a binary string of Q bits. Second, a uniformly distributed secure hash function H_Q is used on the binary string representing the node identifier i, shortening the representation without violating its probabilistic distribution. Third, the Hamming distance is calculated between the PRNG-generated binary string and the hash value (i.e., two binary strings of the same length Q). Lastly, the resulting distance is mapped to the interval $[0,1]$ with a mapping function Z. In short, these four transformations are formalized as follows:

$$\mathcal{B}(d,i) = Z(\Delta(H_Q(i), r(d,i))) \tag{2}$$

The mapping function changes the relationship between a node's reputation and its chance to store any particular record. For example, Z can be chosen to have a doubling of reputation lead to a quarter of the storage requirements. In the context of this paper, we use $Z(x) = x/(Q-D)$, where Q and D are integers where $Q > D \geq 0$. Both the length of the hash Q and the minimum storage factor D can be chosen to guarantee a minimum expected number of copies of a record.

D. Lower bound on available copies

We now discuss TSRP's probabilistic guarantee on the number of available copies of a record. Given our definition of TSRP and the function $Z(x) = x/(Q-D)$, we show that any record d has an expected lower bound of $n \times \frac{1}{2}^Q$ copies.

By definition of S, nodes store records if $\mathcal{R}(i) \leq \mathcal{B}(d,i)$. Clearly, the smallest number of nodes will store a copy of a given record d if all reputations $\mathcal{R}(i)$ are equal to the maximum reputation of 1. By expansion of $\mathcal{B}(d,i)$, we derive $1 \leq Z\left(\Delta\left(H_Q\left(i\right), r\left(d,i\right)\right)\right)$. The smallest number of copies are now stored if the function Z is minimized, by setting D to 0. Our storage function is now simplified to $Q \leq \Delta\left(H_Q\left(i\right), r\left(d,i\right)\right)$. As both $H_Q\left(i\right)$ and $r\left(d,i\right)$ are bit strings of length Q, the function can only be satisfied if the distance between both strings is maximized (equal to the chance that both strings are exactly equal). Because its inputs, r and H_Q, are both uniformly distributed and independent, it follows that the Hamming distance Δ follows a binomial distribution $B(Q, \frac{1}{2})$. The chance to store any record r *on a particular node* i is given by $P(H_Q\left(i\right) = r\left(d,i\right)) = P(X = Q \mid X \sim B(Q, \frac{1}{2})) = \binom{Q}{Q} \frac{1}{2}^Q (1-\frac{1}{2})^0 = \frac{1}{2}^Q$. Finally, all n nodes together are expected to store at least $n \times \frac{1}{2}^Q$ copies.

E. Estimating reputation

A node may estimate its own reputation incorrectly for a variety of reasons. For example, due to the differing reputation mechanisms employed by other nodes or their malicious behavior, views of a node's reputation may be inconsistent. In these cases, a "correct" estimation of a node's reputation does not exist. We therefore scale the current reputation estimate of a node's own reputation estimation by the amount of interaction rejections by other nodes. Upon a successful interaction, the scalar for the node's own reputation estimation increases and upon a rejection the scalar decreases.

IV. Real-world Twitter trace

We apply TSRP to a real-world trace of Twitter to show that the emergent storage of TSRP does not necessarily lead to many more copies of records in comparison to storage schemes for structured peer-to-peer networks. We replay the trace using our 64-physical-core machine (clocked at 2.4 GHz) with 528 GB DDR4 RAM (clocked at 2933 MHz). To provide a benchmark for the number of emergent copies of TSRP, we choose two representative solutions for respectively DHTs and data allocation: Kademlia [7] and CRUSH [17].

We replay the Twitter data set in such a way that data in shared between users, instead of data being distributed by a server to users. We use a Twitter data set that covers a flash crowd of tweets about the Higgs boson [3]. We construct records by mapping the "retweets", "mentions", and "replies",

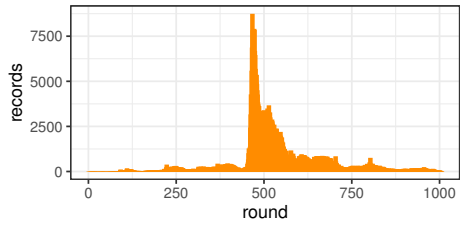

Fig. 2: Number of records per round in the Twitter data set.

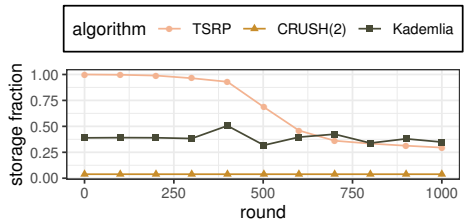

Fig. 3: The average storage fraction (i.e., the copies per record per node) per round for the Twitter data set.

between users to records shared between nodes. Through this construction, the trace that we replay contains 563069 records between 304691 nodes captured over 604621 seconds (roughly a week). We share records between nodes every ten minutes, to form the total of 1008 rounds of these record exchanges, for which we show the number of records per round in Fig. 2.

The source of reputation for the Twitter data set comes from the number of "followers" per Twitter user. Reputations are calculated using the "simplified version" of the Distributed EigenTrust algorithm [5] (normalized toward the median, a localized trust value of 1 if there was a record between nodes and 0 otherwise, and using a "forget factor" of 0.2). The follower count from the data set serves as the "a priori trust value" for each node. As the trace is played, this simplified EigenTrust algorithm causes the median of all the reputations of nodes in the network go to 1 and the average to near 0.72.

The visualization of the number of copies per record per node is given in Fig. 3, with a data point sampled every 100 rounds. As users build reputation in the network, we see that TSRP stores a similar number of records on average to Kademlia. In the final state of this experiment, in the 1008^{th} round, only 29.5% of the network stores a particular record on average with an average reputation score of 0.72. This is a 70.5% decrease from the starting state, where 100% of all nodes stored a particular record on average. Considering the number of stored records, TSRP successfully achieves the same result as Kademlia.

V. Conclusion

Web3 networks need to be approached differently. Structure cannot be imposed over these networks. There is no authority to govern nodes and users should be assumed to be individualistic and egocentric. Connections between users are opportunistic, making a Web3 network prone to partitioning. Users do not wish to serve each other data, but rather police each other to uphold the integrity of their shared public infrastructure. We have shown that records can be shared efficiently between partitions without breaking the Web3 incentives for cooperation. Our TSRP prototype allows users to cooperatively maintain the integrity of records in a Web3 system.

Availability and Acknowledgements

A Python implementation of the "shared carrier-selection logic" described in this paper is available at https://gist.github.com/qstokkink/fecfe070d801538834ff90aa45ddb130. The authors would like to thank Jan S. Rellermeyer and Aaron Y. Ding for their comments on an earlier version of this work. This work was funded by NWO/TKI grant BLOCK.2019.004.

References

[1] P. Bailis and A. Ghodsi, "Eventual consistency today: Limitations, extensions, and beyond," *Communications of the ACM*, vol. 56, no. 5, pp. 55–63, 2013.

[2] E. Blais, J. Brody, and B. Ghazi, "The information complexity of hamming distance," in *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2014)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2014.

[3] M. De Domenico, A. Lima, P. Mougel, and M. Musolesi, "The anatomy of a scientific rumor," *Nature Scientific reports*, vol. 3, p. 2980, Oct. 2013. [Online]. Available: https://doi.org/10.1038/srep02980

[4] M. Feldman and J. Chuang, "Overcoming free-riding behavior in peer-to-peer systems," *ACM sigecom exch's*, vol. 5, no. 4, pp. 41–50, 2005.

[5] S. D. Kamvar, M. T. Schlosser, and H. Garcia-Molina, "The eigentrust algorithm for reputation management in p2p networks," in *12th international conference on World Wide Web*. ACM, 2003, pp. 640–651.

[6] D. Kondor, M. Pósfai, I. Csabai, and G. Vattay, "Do the rich get richer? an empirical analysis of the bitcoin transaction network," *PloS one*, vol. 9, no. 2, 2014.

[7] P. Maymounkov and D. Mazieres, "Kademlia: A peer-to-peer information system based on the xor metric," in *International Workshop on Peer-to-Peer Systems*. Springer, 2002, pp. 53–65.

[8] M. Nadini, L. Alessandretti, F. Di Giacinto, M. Martino, L. M. Aiello, and A. Baronchelli, "Mapping the nft revolution: market trends, trade networks, and visual features," *Nature Scientific reports*, vol. 11, no. 1, pp. 1–11, 2021.

[9] S. Park, S. Im, Y. Seol, and J. Paek, "Nodes in the bitcoin network: Comparative measurement study and survey," *IEEE Access*, vol. 7, pp. 57 009–57 022, 2019.

[10] J. Poon and T. Dryja, "The bitcoin lightning network: Scalable off-chain instant payments," 2016.

[11] T. Rocket, M. Yin, K. Sekniqi, R. van Renesse, and E. G. Sirer, "Scalable and probabilistic leaderless bft consensus through metastability," *arXiv preprint arXiv:1906.08936*, 2019.

[12] M. Steiner, T. En-Najjary, and E. W. Biersack, "Exploiting kad: possible uses and misuses," *ACM SIGCOMM Computer Communication Review*, vol. 37, no. 5, pp. 65–70, 2007.

[13] Q. Stokkink, G. Ishmaev, D. Epema, and J. Pouwelse, "A truly self-sovereign identity system," in *2021 IEEE 46th Conference on Local Computer Networks (LCN)*, 2021, pp. 1–8.

[14] Q. Stokkink, A. Stannat, and J. Pouwelse, "Foundations of peer-to-peer reputation," in *Proceedings of the 1st International Workshop on Distributed Infrastructure for Common Good*, 2020, pp. 25–30.

[15] K. Taeuscher, "Uncertainty kills the long tail: Demand concentration in peer-to-peer marketplaces," *Electronic Markets*, vol. 29, no. 4, pp. 649–660, 2019.

[16] S. Voshmgir, *Token Economy: How the Web3 reinvents the Internet*. Token Kitchen, 2020, vol. 2.

[17] S. A. Weil, S. A. Brandt, E. L. Miller, and C. Maltzahn, "Crush: Controlled, scalable, decentralized placement of replicated data," in *2006 ACM/IEEE Conference on Supercomputing*. IEEE, 2006, pp. 31–31.

[18] Q. Zheng and S. Xu, "Secure and efficient proof of storage with deduplication," in *Proceedings of the second ACM conference on Data and Application Security and Privacy*. ACM, 2012, pp. 1–12.