EFFICIENT GOSSIP-BASED PROTOCOL IN THE NEO BLOCKCHAIN NETWORK

A.S. Shaleva, V.V. Korkhov[a]

Saint Petersburg State University, 7/9 Universitetskaya nab., St. Petersburg, 199034, Russia

E-mail: [a] v.korkhov@spbu.ru

Peer-to-peer (P2P) networks, together with other distributed systems, must meet strict efficiency requirements in terms of the information dissemination process. Rapid message transmission affects the entire functionality of the network, including data processing, attack resistance, topology changes, etc. In this article, we look at the most common network protocols based on gossip, explore and compare several approaches to improve the information dissemination model. We describe the Neo blockchain network protocol, which extends gossip algorithms to distribute messages over a P2P network. Based on the simulation results, we build a model of the Neo blockchain network protocol and evaluate the effectiveness of the Neo network protocol in terms of reactive message processing in several scenarios, propose protocol improvements and evaluate the resulting performance gain.

Keywords: Gossip protocol, P2P network, blockchain

1. Introduction

Distributed peer-to-peer systems offer a reliable, scalable and powerful alternative to traditional client-server architecture. This model assumes that end users act as a client and as a server, sharing resources in a peer-to-peer style. The P2P approach is free from central server failures and performance bottlenecks from load balancing issues. However, building an efficient, scalable, and low-cost P2P application layer protocol is still not a trivial task. At the moment, epidemic algorithms are the most effective and reliable solution for the development of P2P protocols for disseminating information in large dynamic systems. These algorithms are based on the idea of spreading infectious diseases and are used for a variety of purposes, from monitoring distributed systems to using in blockchain broadcast protocols.

One of the most important factors in the functioning of distributed P2P networks, affecting the efficiency of the network, is the delay in the propagation of network messages [6]. This includes message processing delays, up to and including broadcasting delays. Latency has a direct impact on frame non-delivery probability [4], which is a key performance metric in distributed P2P networks.

Another important problem is the problem of congestion of the network with broadcast messages throughout the network [4] or the problem of high traffic load. The large amount of synchronization information on the network results in low efficiency of the bandwidth protocol, which can slow down data processing. Consider the case of blockhain networks, where the creation of a large number of duplicate messages increases the block acceptance time, which directly affects the latency of transaction processing.

2. Related work

Gossip-based algorithms are a well-known and widely studied area of research. One of the most recent papers summarizing the known theoretical results for various gossip-based protocols is [11]. The study explores a knowledge-based approach to the problem of gossip in a multi-agent system, and the result is a theoretical basis for the reach of epistemic gossip protocols. It should be noted that gossip algorithms are very robust in a probabilistic sense. The article [6] briefly summarizes the known mathematical estimates of basic gossip spreading models, which include the probability of an atomic infection, the proportion of the infected process, and an estimate of the infection latency.

A wide range of approaches to optimizing gossip algorithms can be analyzed. Our research is inspired by the ideas presented in [2,3,4]. The paper [4] proposes a new protocol for real-time N-to-N multicast communication, which provides significantly less traffic load with higher performance improvements in lower latency networks. The article [2] describes a simulation model for evaluating the effectiveness of the push-gossip protocol. Based on the simulation results, the optimal number of peers to which the current peer should forward a message (also known as a fork) is calculated as a function of the number of nodes in the network. The paper [3] proposes a reliable and fast protocol for loosely consistent knowledge of process group membership information in all participating processes.

3. Aggregation problem

Most gossip optimization algorithms (such as those presented in [1] and [2]) require local peer-to-peer access to global network information. This is a separate issue known as the aggregation problem [7] and refers to a set of functions that provide access to such distributed system components as network size, load average and uptime, network map, and so on. There are a number of existing approaches to aggregating gossip-based information over distributed P2P networks, which can be divided into two groups. The first uses an epoch-based solution to compute a global score from an initial set of predefined local values [7]. Network members exchange aggregated information by repeating aggregation periods; as soon as the estimate converges on the scale of the entire system, a new era begins with fresh initial values, which makes the algorithm resistant to dynamic aging of estimates. Another approach uses continuous data aggregation without regard to epoch [8] and takes into account the local measurements of the nodes in each round. This approach does not require initial measurements and performs aggregation and exchange cycles between nodes continuously.

However, these approaches rely on the reliability of information and are unable to cope with errors when nodes behave arbitrarily [6]. Thus, applying these algorithms to unreliable distributed systems requires some additional assumptions and constraints. One way to solve this problem is to detect malicious nodes and exclude their data from the results of the aggregation service. However, this issue is beyond the scope of the article, but it should be taken into account when applying the mentioned algorithms in unreliable distributed systems.

4. Neo network protocol

Neo is an open source, community driven blockchain platform [10]. The gossip protocol is used in Neo for a number of purposes: creating and maintaining a local peer representation of the network structure, exchanging messages related to consensus, discovering new peers, exchanging network metadata (i.e. current chain height, etc.) and, finally, distribution of blocks, transactions and other payloads. In this study, we mainly focus on block propagation from the consensus service to all peers, since the network protocol mechanism is practically the same for all payload messages.

Consider the case of block propagation, the beginning of the block distribution process is the moment the block is received by the consensus service. The consensus nodes then receive, validate, and store the block generated by the consensus service and forward it to a set of connected peers. There are three main stages in the Neo gossip algorithm that determine the rules for distributing blocks across the network: push, pull, and recovery.

The network protocol has a large impact on the entire process of the network, which can be described in two ways. First, the protocol directly affects the level of network bandwidth consumption, which is one of the key metrics for assessing the effectiveness of the network. Large payloads and a large number of synchronization-related messages result in a high traffic load, which creates load balancing problems and contention. Moreover, the large number of large payloads on the network results in a higher cost of operating the broadcast protocol.

5. Proposed improvements

Based on the shortcomings of the Neo protocol, we propose several protocol enhancements:

Introducing the f_out setting: The current scheme of the Neo protocol assumes that a peer either transmits received blocks to all peers to which it is connected, or does not transmit blocks at all. Although the Neo node can be configured with a maximum and minimum number of peers, it still lacks the f_out parameter, that is, the number of randomly selected peers to which the current peer should forward a message. The absence of the f_out parameter causes the network to flood with synchronization messages for large max_peers (maximum number of peers a peer sends messages to in the standard protocol) and lack of connectivity, resulting in slower block processing for small max_peers. At the same time, the proposed f_out setting can be significantly smaller than the maximum number of connected peers of a node, which permits to change the load of network traffic regardless of the connection speed of the nodes.

Gossip source randomization: According to the Neo consensus protocol, blocks created by the consensus service are first passed to nodes selected from the standby committee list, which has rather limited capacity. These leader nodes are the ones who start spreading, so their network utilization is several times (up to max_peers) higher than that of regular peers. To smooth out this difference, we'll set the leader's f_out to 1. We expect this improvement to distribute the initial propagation load across all regular peers.

Using infect-forever model for the push-phase: The Neo protocol currently uses the infect-and-die model for the push phase of the gossip algorithm. We propose to replace it with the infect-forever model [5] modified to use the stop condition. Peers are expected to forward blocks whenever they receive them, until a stop condition is met. To implement the stop condition, a counter r, initialized to 0, is appended to each new block. The first time a peer receives this block, it increments the counter and continues transmitting the incremented block in the standard way. The next time a peer receives the same block, it simply forwards the block to the f_out peers, selected at random from the set of connected peers. Block propagation stops when r reaches the pre-configured TTL time-tolive value, which is the same for all peers on the network. This improvement is highly dependent on the f_out parameter and, when combined with f_out, is expected to balance the load between peers initiating gossip rounds and others.

Table 1 overviews the proposed improvements and their expected effect.

Table 1. Summary of the proposed improvements

Optimization	**Description**	**Expected result**
Introducing fan-out setting	Ability to tune the number of peers to forward new block to	Significant reduction of both network traffic consumption and frame non-delivery probability
Randomization of the source gossiper	Fan-out restriction for the group of consensus peers	Reduction of the bandwidth utilization level for the consensus peers
Using infect-forever model for the push phase	Block retransmission capability restricted by the TTL parameter	Decrease of the frame non-delivery probability rate and latencies

6. Experiments

We evaluate the proposed improvements to the NeoGo gossip module and compare the results with the original version of the module implementation. Inspired by [1], we mainly focus our assessment on the three key metrics described above: block non-delivery probability, block propagation latency, and traffic load (also called bandwidth utilization), and how these metrics correlate with each other.

We base our experiment environment on NeoBench [9], an open source test tool for stress testing Neo nodes. We created a network of 101 peers, where one peer is an RPC node accepting transactions from a client, four peers are consensus nodes creating and approving blocks, and the rest of the peers are regular nodes storing blocks and distributing network messages. All nodes are deployed in a cluster equipped with 2 Intel® Xeon® E5-2690 v4 2.60GHz processors and 256GB of RAM, and run in Docker containers that share the same percentage of CPU cycles and are grouped into a network. A test tool with an RPC client simultaneously sending pre-generated transactions to the RPC host and collecting statistics runs in a separate Docker container.

Here we present some examples of improvement results. Figure 1 shows the probability of non-delivery of frames for the original version of the protocol and for the optimized version with randomized gossip source. Figure 2 shows the bandwidth utilization for these protocol versions.

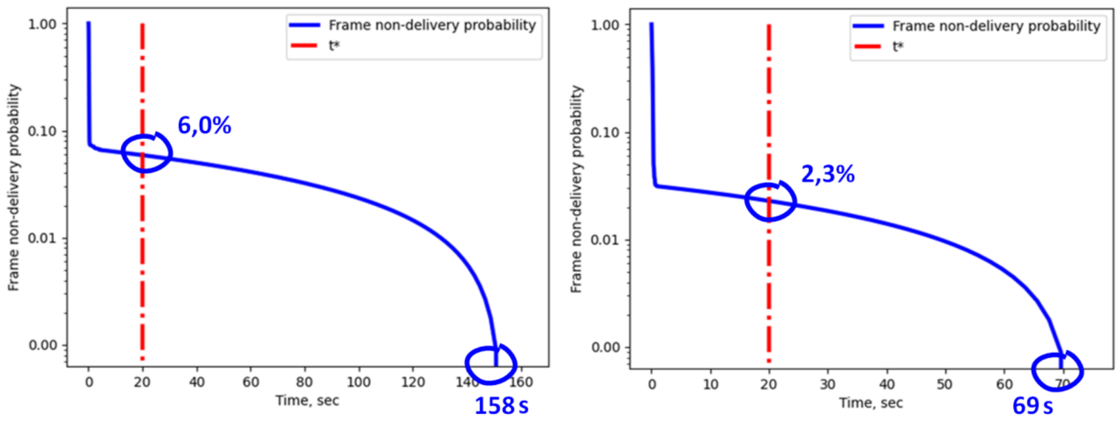

Figure 1. Frame non-delivery probability for original gossip module (left) and module with the source gossiper randomization (right)

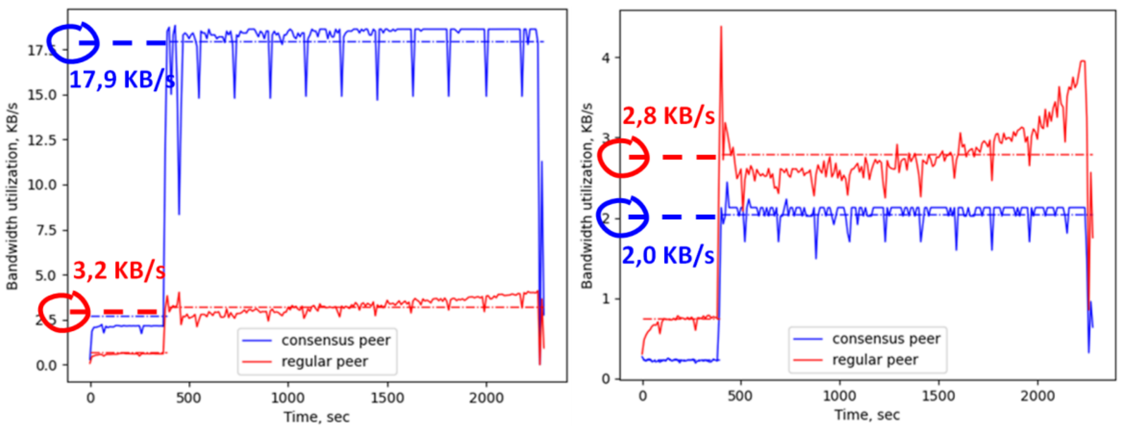

Figure 2. Bandwidth utilization using the original gossip module (left); Bandwidth utilization using module with the source gossiper randomization (right).

7. Conclusions

In this article, we have improved the Neo gossip layer. Reducing its latency allowed us to maintain a solid foundation for the consensus protocol to work. The improved protocol also demonstrates significant improvements in network bandwidth consumption, avoiding load balancing issues and lowering the cost of the broadcast protocol.

The current work considers the case of an "ideal" network state without loss of network packets, data corruption, and message transmission delays caused by an imperfect inter-node connection. This can mean better target performance than if you were deploying a network in a production environment. While these experiments allow us to estimate the expected efficiency gains, we still need to evaluate the proposed protocol improvements in a real physically distributed network. This can be achieved either by simulating network packet corruption and message loss using network simulation tools, or by evaluating targets in a production environment.

References

[1] Berendea, N., Mercier, H., Onica, E., Riviere, E.: Fair and Efficient Gossip in Hyperledger Fabric, in 2020 IEEE 40th International Conference on Distributed Computing Systems (ICDCS), Singapore, Singapore, 2020 pp. 190-200. doi: 10.1109/ICDCS47774.2020.00027

[2] Vanin, A., Bogatyrev, V.: Push-gossip protocol efficiency with network topology propagation. Proceedings of the 10th Majorov International Conference on Software Engineering and Computer Systems (MICSECS-2018), CEUR Workshop Proceedings 2019, Vol. 2344

[3] Das, A., Das, A., Gupta, I., Motivala, A.: SWIM: scalable weakly-consistent infection-style process group membership protocol. In Proc. 2002 Intnl. Conf. Dependable Systems and Networks (DSN ’02), 303--312

[4] Luk, V.WH., Wong, A.KS., Lea, CT. et al. RRG: redundancy reduced gossip protocol for real-time N-to-N dynamic group communication. J Internet Serv Appl 4, 14 (2013). https://doi.org/10.1186/1869-0238-4-14

[5] Koldehofe, B.: Simple Gossiping With Balls and Bins. Studia Informatica Universalis. (2004) 3. 43-60.

[6] Eugster, P. T., Guerraoui, R., Kermarrec, A., Massoulie, L.: Epidemic information dissemination in distributed systems, in Computer, vol. 37, no. 5, pp. 60-67, May 2004, doi: 10.1109/MC.2004.1297243.

[7] Jelasity, M., Montresor, A., Babaoglu, 0.: Gossip-based aggregation in large dynamic networks. ACM Trans. Comput. Syst. 23, 3 (August 2005), 219–252. DOI:https://doi.org/10.1145/1082469.1082470

[8] Rapp V., Graffi K.: Continuous Gossip-Based Aggregation through Dynamic Information Aging, 22nd International Conference on Computer Communication and Networks (ICCCN), 2013, pp. 1-7, doi: 10.1109/ICCCN.2013.6614118.

[9] NeoBench, https://github.com/nspcc-dev/neo-bench

[10] Neo blockchain platform, https://neo.org/

[11] Apt, K., Grossi, D., Hoek, W.: When Are Two Gossips the Same? Types of Com-munication in Epistemic Gossip Protocols. 2018, http://arxiv.org/abs/1807.05283