OPTIMUMP2P Demo: Fast and Reliable Gossiping in P2P Networks

Alejandro Bergasov, Aayush Rajasekaran, Nicolas Nicolaou, Kishori M. Konwar,
Santiago Paiva, Har Preet Singh, and Muriel Médard
{*nicolas,aayush,alejandro,kkonwar, santiago,harpreet, mmedard*}*@getoptimum.xyz*
Optimum, MA, USA

***Abstract*—Gossip algorithms are pivotal in the dissemination of information within decentralized systems. Consequently, numerous gossip libraries have been developed and widely utilized especially in blockchain protocols for the propagation of blocks and transactions. A well-established library is *libp2p*, which provides two gossip algorithms: *Floodsub* and *Gossipsub*. These algorithms enable the delivery of published messages to a set of peers. In this demo we aim to show the enhancement of the performance and reliability of *libp2p* by introducing OPTIMUMP2P, a novel gossip algorithm that leverages the capabilities of Random Linear Network Coding (RLNC) to expedite the dissemination of information in a peer-to-peer (P2P) network. Preliminary research from the Ethereum Foundation has demonstrated the use of RLNC in significantly improving block propagation time [2]. Here we present a demo for real-world environments that demonstrate the performance gains of OPTIMUMP2P over the *Gossipsub* protocol.**

I. INTRODUCTION

Low-latency gossip algorithms are essential for decentralized information dissemination and are widely used in blockchain systems to propagate blocks and transactions. However, blockchain implementations often operate on top of a permissionless, asynchronous, message-passing network, susceptible to unpredictable delays and node failures. Therefore, improper use of gossiping approaches may lead to high network overhead and congestion, high propagation latencies, and erroneous information propagation due to message alteration by malicious actors. *libp2p* [4], is one of the latest network communication frameworks and gossip algorithms used in modern blockchain solutions like Ethereum 2.0 [1]. *libp2p* adopts two different push gossiping algorithms: ***Floodsub*** and ***Gossipsub***.

Floodsub uses a flooding strategy where every node forwards messages to all of its neighbors. Although very efficient in discovering the shortest path and very robust in delivering a message to all the peers in the network, Floodsub suffered from bandwidth saturation and unbounded degree flooding.

Gossipsub is a successor to Floodsub, which addressed the shortcomings of the initial algorithm by organizing peers into topic-based mesh, network overlay, with a target mesh-degree D and utilizing control messages for reducing message duplication. Briefly, the Gossipsub protocol works as follows. A publisher selects D peers among its peers and broadcasts its message to them. Each peer receiving a message performs preliminary validation and rebroadcasts the message to another D peers. Peers exchange control messages such as IWANT, IHAVE or IDONTWANT to inform their peers about their status regarding the propagation of a particular message. These enhancements enabled Gossipsub to reduce bandwidth usage, but the introduction of the bounded degree D increased the number of hops a message required to reach distant peers, resulting in higher delivery latencies. Furthermore, similar to the Floodsub protocol, each peer forwards the full message to its peers even when other peers may already have received the full message, also suffering from (reduced compared to Floodsub) message duplication.

So is it feasible to implement a gossip protocol that is light in network usage and yet fast in information diffusion?

In this work, we demonstrate the enhanced performance of OPTIMUMP2P, an RLNC-based gossip protocol built on libp2p [3] against GossipSub. RLNC is a technique used in communication networks to enhance data transmission efficiency and robustness. In RLNC, data packets are encoded as random linear combinations of original packets over a finite field, typically $\mathbb{F}_{2^m}$. This approach allows intermediate nodes in the network to mix packets without decoding (aka *recode*), and the receiver can recover the original data by solving a system of linear equations once enough linearly independent combinations are received.

While the above results point to the potential benefit for using RLNC in gossip, in order for RLNC to be deployed in current decentralized systems, it requires the design of a full protocol. OPTIMUMP2P [5] is a novel gossip mechanism based on RLNC, which significantly enhances the spread of information across the network. Once a publisher publishes a message, the protocol splits it into RLNC-encoded fragments (shards) and sends a subset (or a linear combination of shards), rather than the full message, to each of its peers. In turn, peers do not simply forward the shards they receive; instead, they linearly combine them with previously received shards for the same message to generate and forward new shards to their own peers. This approach has dual benefit:

1) **Faster Network Coverage:** it allows each peer to reach more peers for the same amount of data sent in full message counterparts (e.g., Gossipsub), and
2) **Message Duplication Reduction:** as peers receive different shards in parallel from different peers which combine to decode the original message.

Essentially, OPTIMUMP2P allows peers to spread information *piece by piece*, as opposed to traditional gossip approaches

978-3-903176-75-1 ©2025 IFIP

that broadcast full information between any pair of peers. Overall OPTIMUMP2P is a new protocol implemented within *libp2p* decreases latency, enhances fault tolerance, and optimizes bandwidth usage. Here we present the OPTIMUMP2P protocol and summarize extensive experiments we conducted to compare its performance with Gossipsub. both in a simulation and real-world environments.

II. REAL WORLD DEMO: OPTIMUMP2P VS GOSSIPSUB

In this section we briefly describe the experimental evaluation of OPTIMUMP2P.

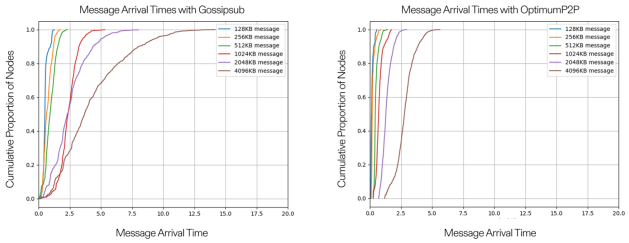

Fig. 1. Comparison of latency between OptimumP2P and Gossipsub by message size.

a) OPTIMUMP2P *Simulation Results:* We used the Ethereum tool Ethshadow[1] to simulate gossip in an Ethereum-like network. We built off the work of prior Ethereum research [2], running the same simulations as them with 1,000 nodes, 20% of which have incoming/outgoing bandwidth of 1Gbps/1Gbps and 80% of which have 50Mbps/50Mbps. The publisher always has 1Gbps/1Gbps in order to get consistent simulation results. The latencies between pairs of nodes are based on real-world geographic locations. We first ran a simple experiment, in which a single publisher publishes a single message. We varied message sizes from 128KB to 4096 KB, and observed significantly faster arrival times for all message sizes, as shown in Figure 1. We remark that our observed performance of Gossipsub matches the results in [2], which give us confidence in our reproducibility.

b) OPTIMUMP2P *Real World A/B Testing:* We performed side-by-side A/B testing of Gossipsub and OPTIMUMP2P, each deployed across 36 geographically distributed identical nodes in Google Cloud Platform (GCP) data centers (Figure 2), roughly mirroring the distribution of Ethereum validator nodes. In each test, nodes propagated large data blobs, simulating transaction blocks. A randomly selected node initiated each gossip round, and propagation was deemed successful once at least 95% of nodes received the message. We varied two main parameters: (i) the message size (from 4MB up to 10MB blobs), and (ii) the publish rate (ranging from isolated single-block sends to rapid bursts up to several messages per second). Both protocols were pushed to carry **100 messages** per run in some high-load scenarios to observe behavior under message bursts. A live demo of our A/B testing can be found in [7]. A video recording showcasing the usage of the demo is accessible in [6].

[1]https://ethereum.github.io/ethshadow/

Fig. 2. Geographic distribution of the 36 nodes used in each protocol.

The performance metrics recorded include the *propagation latency*, i.e., the time for 95% of the nodes to receive and reconstruct the message, and the *delivery ratio*, i.e., the fraction of nodes that obtained the message within a fixed time-bound. We also tracked the average per-message delay and its variance to gauge stability.

Propagation Latency and Scalability: In this demo, we show the average end-to-end propagation delay for 10MB messages under increasing publish rates, comparing RLNC-based OPTIMUMP2P and Gossipsub. In our demo, we will showcase example cases of publishing rates and message delivery latency through side-by-side comparisons of OptimumP2P and Gossip across 36 identically distributed nodes.

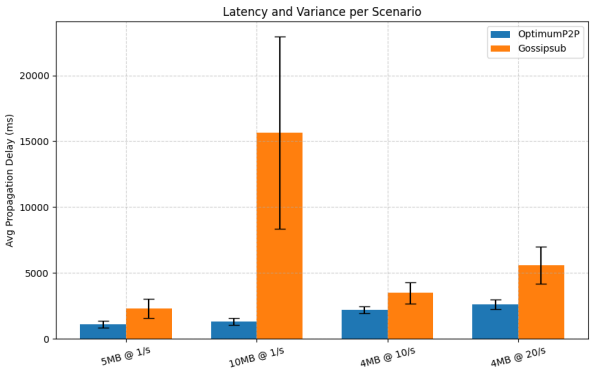

Fig. 3. Propagation delay comparison between OPTIMUMP2P and Gossipsub.

At **1 msg/s**, both protocols achieve sub-second latencies; however, Gossipsub exhibits a slightly higher delay (1.0s) than OPTIMUMP2P (0.8s) due to the inefficiencies inherent in gossip-based redundancy and bandwidth usage.

At **10 msg/s**, Gossipsub's delay increases to 2.5s, while OPTIMUMP2P maintains a lower delay of 1.5s. Under the highest rate of 20 msg/s, Gossipsub's latency sharply degrades

to 4.0s, signaling network saturation and queuing delays. In contrast, OPTIMUMP2P maintains delivery within 1.8–2.0s.

These results demonstrate that **RLNC enables superior scalability** in terms of propagation latency. The coded approach makes more efficient use of network capacity, avoiding redundant transmissions, whereas Gossipsub's performance significantly declines in situations with bursty, high-throughput demands.

We further evaluated scalability under increasing message sizes at a fixed publish rate of **1 msg/s** testing both 5MB and 10MB payloads (Figure 4).

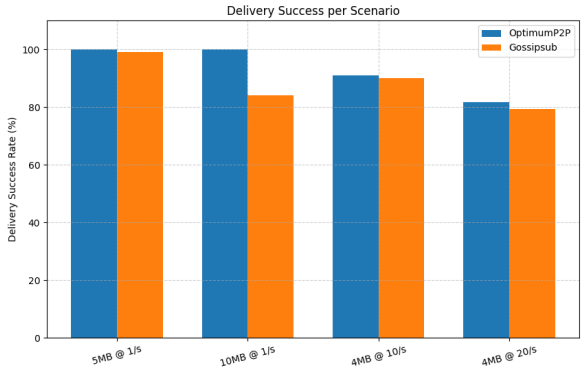

Fig. 4. Scalability comparison between OPTIMUMP2P and Gossipsub.

For 5MB blocks, OPTIMUMP2P achieved 100% delivery with an average latency of 1116ms (std = 262ms). Gossipsub delivered 99/100, with a delay of 2293ms and higher variance (std = 712ms). For 10MB blocks, Gossipsub's performance deteriorated sharply—only 84/100 messages were delivered, with an average delay of 15.6s and significant variability (std = 7.3s). OPTIMUMP2P again delivered 100%, with latency held to 1302ms and low deviation (std = 270ms).

These results demonstrate that OPTIMUMP2P remains strong and efficient as message sizes grow, consistently achieving low latencies. In contrast, Gossipsub becomes unreliable under larger payloads, succumbing to congestion and protocol overhead.

III. CONCLUSIONS

In this demo, we will present a dynamic performance comparison of OPTIMUMP2P, a gossip protocol that leverages RLNC to accelerate information propagation in peer-to-peer (P2P) networks, against Gossipsub.

Acknowledgments: The computing resources for the deployment of the experimental work in this paper are supported in part by the Startups Cloud Program by Google.

REFERENCES

[1] FOUNDATION, E. Ethereum 2.0 networking specification. https://github.com/ethereum/consensus-specshttps://github.com/ethereum/consensus-specs, 2023. Accessed: 2023-10-10.

[2] FOUNDATION, E. Faster block/blob propagation in ethereum. https://ethresear.ch/t/faster-block-blob-propagation-in-ethereum/21370/1, 2025. Accessed: 2025-03-10.

[3] HO, T., MÉDARD, M., KÖTTER, R., KARGER, D. R., EFFROS, M., SHI, J., AND LEONG, B. A random linear network coding approach to multicast. *IEEE Transactions on Information Theory 52*, 10 (2006), 4413–4430.

[4] LABS, P. libp2p: A modular network stack for peer-to-peer applications. https://libp2p.io/, 2023. Accessed: 2023-10-10.

[5] NICOLAOU, N., OBI, O., RAJASEKARAN, A., BERGASOV, A., BEZOBCHUK, A., KONWAR, K. M., MEIER, M., PAIVA, S., SINGH, H. P., VISHWANATH, S. S. S., AND MEDARD, M. Optimump2p: Fast and reliable gossiping in p2p networks, 2025. Arxiv URL: https://arxiv.org/abs/2508.04833.

[6] OPTIMUM. Consensus 2025. https://x.com/get_optimum/status/1925580354825638111https://x.com/get_optimum/status/1925580354825638111, 2025. Accessed: 2025-09-17.

[7] OPTIMUM. P2p comparison rlnc vs gossipsub. https://ui-abtester.getoptimum.io/https://ui-abtester.getoptimum.io/, 2025. Accessed: 2025-09-17.