Scaling the Lightning Network with Practical Set Reconciliation*

Xingyu Chen *Boston University*
Boston, USA
chxy517@bu.edu

Anish Sinha *Red Hat*
Boston, USA
anishs@bu.edu

David Starobinski *Boston University*
Boston, USA
staro@bu.edu

Ari Trachtenberg *Boston University*
Boston, USA
trachten@bu.edu

June 18, 2026

Abstract

The Lightning Network (LN) utilizes gossip to share network topology, channel announcements and updates, and node announcements among its local constituents. Yet, our measurements show that this flooding-based gossip reconciliation is fundamentally inefficient. We propose, instead, to use set reconciliation protocols for sharing this information, and we systematically evaluate existing approaches under realistic network conditions. We further propose AdaptiveIBLT, a novel adaptive IBLT (Invertible Bloom Lookup Table) protocol with a partial-decoding enhancement. By simulating reconciliation in Core-Lightning and evaluating real gossip snapshots, we demonstrate the practical benefits of reconciliation in scaling gossip reconciliation from hours down to a few minutes.

Lightning Network, Set Reconciliation

1 Introduction

The Lightning Network (LN) is a prominent off-chain scaling solution for Bitcoin and other cryptocurrencies, designed to support fast and low-cost payments by executing transactions over bidirectional off-chain (multi-hop) payment channels

*A version of this article appeared in IEEE ICBC 2026

rather than on the more cumbersome blockchain [1]. To successfully compute payment routes, each node must maintain an up-to-date view of the network's channel graph, including channel announcements / updates, and node announcements.

Each LN node maintains a local database, commonly referred to as the `gossip_store`, which stores all known channels and node information. When a node connects to new peers or rejoins the network after downtime, it synchronizes its gossip state by receiving gossip messages from its peers. Today, this reconciliation process relies almost exclusively on flooding: peers repeatedly exchange gossip messages regardless of how much information is already common.

Several prior studies and engineering efforts [2–4] have observed that LN gossip converges slowly and consumes substantial bandwidth. Recent discussions among Lightning Network developers further highlight that these inefficiencies stem from deeper structural limitations of the current gossip protocol [5]. In particular, the code exhibits inconsistent synchronization, lack mechanisms to reliably detect stale or missing updates, and often rely on heuristic policies (*e.g.* peer selection, rate limiting, or periodic resynchronization). These limitations can lead to incomplete network views (*e.g.* failure to resurrect pruned channels) and unnecessary transmission of out-of-order or redundant messages. Such observations suggest that the inefficiency of LN gossip is not merely due to suboptimal broadcast strategies, but rather arises from the fundamental mismatch between broadcast-based dissemination and the underlying goal of state reconciliation across peers.

By contrast, we argue that the LN gossip mechanism is more naturally viewed as a *set reconciliation* problem. At any point in time, two LN peers maintain highly overlapping gossip datasets and only exchange their differences. From this perspective, repeated gossip flooding is unnecessary: peers can reconcile their local states directly by exchanging compact summaries that encode only set differences.

This problem naturally maps to the classical set reconciliation problem [6], in which two parties seek to compute the symmetric difference of their datasets while minimizing communication and computation. A rich literature proposes reconciliation protocols based on compact sketches—such as Bloom filters [7], characteristic polynomial interpolation (CPI) [6], and invertible Bloom lookup tables (IBLTs) [8]. However, much of this work assumes that the size of the set difference is known in advance, or else evaluates protocols under idealized network conditions.

In practice, neither assumption holds for LN gossip. The difference between peers' gossip states is highly dynamic and difficult to estimate, and reconciliation is performed over real TCP connections where latency, bandwidth, and round complexity strongly affect end-to-end performance. Although recent reconciliation protocols have saught to remove the need to know the set difference in advance, there has been little systematic evaluation of how such protocols

behave or how they compare against one another in practical settings.

1.1 Contributions

Our work addresses these gaps through a combination of system-level measurement, protocol design, and transport-aware evaluation. We make the following contributions:

- We perform packet-level measurements of the Lightning Network gossip protocol, showing significant redundancy and latency from its flooding-based gossip.
- We integrate the open-source GenSync [9] reconciliation framework into Core Lightning to conduct a unified, fair comparison of multiple reconciliation protocols.
- We propose AdaptiveIBLT, a dynamic, runtime-adaptive, reconciliation protocol designed for unknown difference sizes and challenging network conditions.

Together, these contributions provide a practical understanding of how reconciliation can fundamentally improve the efficiency of gossip dissemination in the Lightning Network.

1.2 Roadmap

The remainder of this paper is organized as follows. Section 2 provides background on Lightning Network gossip and surveys prior data reconciliation protocols. Section 3 presents an empirical study of Lightning Network gossip behavior, quantifying the communication overhead incurred by flooding-based dissemination. Section 4 presents our design of AdaptiveIBLT, including an enhancement via partial decoding. Section 5 evaluates reconciliation protocols within a unified GenSync benchmarking environment. Section 6 discusses our integration of set reconciliation into the Lightning Network and reports end-to-end measurements on gossip snapshots. Finally, Section 7 summarizes our key findings.

2 Background

2.1 Lightning Network Gossip Protocol

Gossip dissemination in LN follows a flooding-based design. When a node establishes a connection with a peer, it exchanges gossip messages and forwards newly received gossip to other peers. For example, when node A connects to

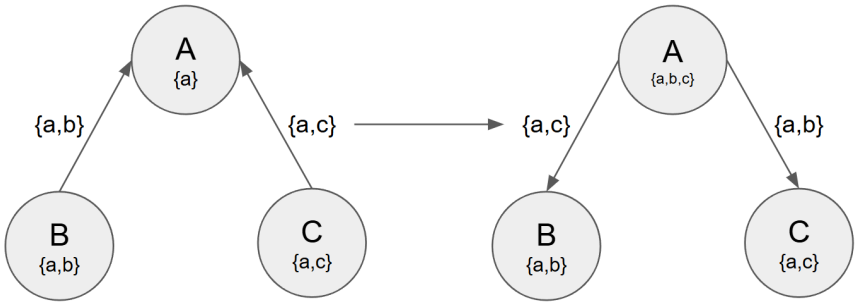

Figure 1: Flooding-based Gossip Dissemination in the Lightning Network. Nodes maintain sets and receive update gossips from peers. Here, peers redundantly forwards $\{a\}$, though they all already it.

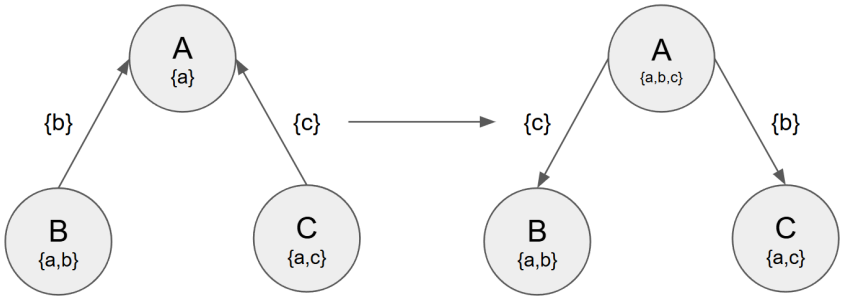

Figure 2: Example of an efficient set reconciliation workflow, that only communicates elements missing from other nodes.

node B and node C with partial data set (see Figure 1), it may receive gossip items that are already within its own store (e.g., $\{a\}$). To avoid infinite rebroadcasting, LN nodes track short-term message history using gossip timestamps and rate limits, but do not maintain an explicit representation of which peers already share common state.

In practice, when a node restarts or reconnects after downtime, it may need to replay a substantial amount of historical gossip to reconstruct a routing-ready view of the network. These inefficiencies have motivated engineering solutions such as Rapid Gossip Sync (RGS) [10], which distributes compact snapshots of the routing state to accelerate initial reconciliation. RGS relies on trusted snapshot providers and does not address the more general problem of peer-to-peer reconciliation under unknown differences.

Overall, LN has limited control over efficiency when peers possess highly overlapping routing state. For this reason, we explore alternative reconciliation mechanisms in this work.

2.2 Data Reconciliation Protocols

Data reconciliation enables the efficient computation of $(S_A \setminus S_B) \cup (S_B \setminus S_A)$ for two remote sets S_A and S_B with minimal communication. Figure 2 contrasts the gossip workflow with a set reconciliation–based approach. Instead of repeated broadcasts, nodes treat local state as sets and propagate only differences. In the illustrated example (Figure 2), node A reconciles separately with node B and node C to obtain missing elements, and then propagates only newly learned items. Reconciliation directly targets state divergence instead of relying on repeated broadcast.

Classical reconciliation protocols roughly fall into two types: algebraic and filter-based. Algebraic approaches, such as Characteristic Polynomial Interpolation (CPI) [6] and its variants InteractiveCPI [11, 12], encode set differences using mathematical structures, incurring low communication cost but high computation complexity. Filter-based approaches trade accuracy or robustness for efficiency. For example, reconciliations based on Bloom Filters [7] admit false positives, and those based on Invertible Bloom Lookup Tables (IBLT) [8] require accurate estimates of set difference d.

Several adaptive and rateless designs have been proposed to remove the need for estimating d. METIBLT [13] improves decoding robustness by transmitting redundant coded IBLTs, at the cost of increased communication. RatelessIBLT [14] eliminates parameter tuning by streaming coded symbols until decoding succeeds, introducing high round complexity. CertainSync [15] provides deterministic guarantees. In this work, we introduce AdaptiveIBLT, which adjusts transmission size based on decoding feedback, balancing robustness and round efficiency without requiring prior knowledge of d.

Reconciliation has also been explored in high-throughput blockchain subsystems, particularly for transaction relay and mempool reconciliation [16, 17]. Practical systems such as Shrec [18] highlight that sketches with strong theoretical guarantees may become impractical once computation cost and transport-layer behavior are taken into account.

In this work, we build upon the GenSync [9], a standardized reconciliation framework, to evaluate modern reconciliation protocols under realistic network conditions and further implement AdaptiveIBLT.

3 Observations of LN Gossip Inefficiency

We conduct an empirical study of LN gossip behavior under typical operating conditions. We seek, seeking to quantify the magnitude of communication overhead during routine gossip.

We keep a standard Lightning Network node (default configuration) continuously connected to the public LN network. Each 10-minute run maintains active peers and normal gossip exchange while capturing TCP traffic and track-

ing growth of the local `gossip_store`. Across runs, the `gossip_store` grows at $\sim$ 500–1000 bytes/s from new channel and node data, although transport-layer traffic is much higher.

On average, only about *30%* of the TCP-level traffic (including both payload and headers) directly contributes to persistent `gossip_store` growth. The remaining *70%* consists of protocol overhead, including redundant gossip messages, control messages, timestamp-based filtering, and traffic required to maintain channel and node liveness. While absolute traffic volumes vary slightly across runs due to peer selection and network dynamics, the ratio between useful state growth and total transmitted data remains stable.

This persistent gap suggests that LN gossip inefficiency is a structural consequence of the flooding-based dissemination strategy. Even when peers already share highly overlapping gossip state, the protocol lacks a mechanism to exchange only differences. This motivates the exploration of reconciliation approaches that directly target difference exchange, rather than attempting to optimize message-level broadcast alone.

4 Adaptive-IBLT

AdaptiveIBLT generalizes the IBLT protocol with a runtime feedback loop through which the sender adjusts future IBLT transmission size based on decoding progress. In so doing, this design eliminates the requirement of an accurate estimate of the number of differences between peers.

Our protocol starts with standard IBLT-based reconciliation using an initial number of cells (default 0.5% of the set size). If decoding fails, the receiver notifies the sender, which doubles the cell count (default factor 2) and resends the encoding. This repeats until decoding succeeds.

We extend AdaptiveIBLT with *partial decoding* to reduce communication and improve decoding success in marginal failure cases. When IBLT decoding fails, some elements yet be recovered; we cache them and exclude them from later rounds, lowering cost. Experiments (Section 5) show reduced communication, especially near the decoding threshold, and improved latency when differences are hard to estimate.

5 Evaluation of Sync Protocols

We implemented AdaptiveIBLT within the `GenSync` benchmarking framework [9], together with a number of additional state-of-the-art reconciliation protocols. This unified framework allows us to compare the performance of the various data reconciliation protocols under the practical constraints. Our evaluation measures total time to reconcile (TTR) and communication cost across various workloads. *We conclude that no protocol is globally optimal.*

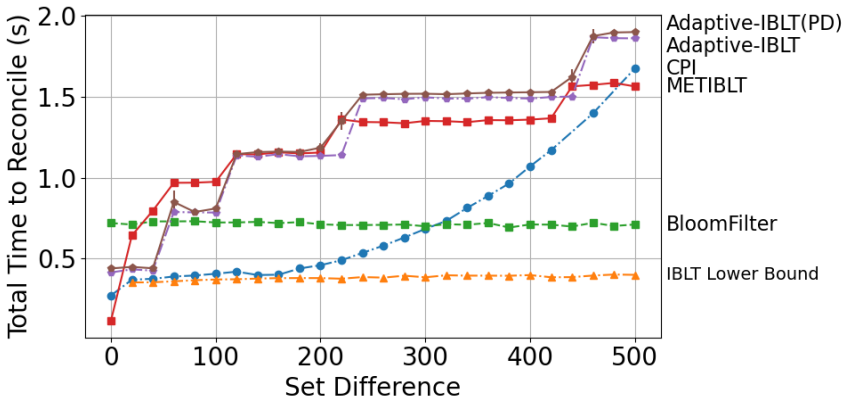

Figure 3: Total Time to Reconcile for some protocols.

Each reconciliation session runs between two local processes communicating over TCP sockets. We emulate realistic network conditions using the `mininet` [19] framework, which enables packet-level control of latency, bandwidth, and loss. Our experiments use 10 ms round-trip latency, and bandwidth caps of 18 Mbps, all with 0.1% packet loss (as in [9]), and each experiment is repeated 100 times on independently sampled sets of 32-byte strings with 95% confidence intervals.

Experiments were on an Intel Core i5-8400 CPU @ 2.80GHz and 16GB RAM, running Ubuntu 20.04 LTS. The `GenSync` framework is compiled with `g++ 11.4.0` `CMake 3.27`, `GMP 6.1.2`, and `NTL 10.5.0`. Protocol configurations and implementations are available at [20].

For each reconciliation run, we collect: *Total Time to Reconcile* (TTR) - The wall-clock time from protocol initiation to successful reconciliation; *Communication Cost* - The total number of bytes exchanged by both parties over the full reconciliation process, including all data and control.

5.1 Total Reconciliation Time

Figure 3 compares reconciliation protocols under 10,ms RTT, 18,Mbps bandwidth, and 0.1% packet loss. BloomFilter and IBLT (a "lower bound" assuming knowledge of d) maintain low TTR, with IBLT achieving the lowest latency. METIBLT has higher TTR at small d due to redundant coded transmissions but scales more gracefully as d increases. In contrast, AdaptiveIBLT may incur higher TTR at large d when doubling overshoots the true difference. Among interactive protocols (not shown), InteractiveCPI yields moderate TTR (20 seconds at $d = 500$) for small differences but exhibits sharp spikes (and high variance), while RatelessIBLT reaches up to 60 seconds TTR, partly due to high round complexity from sending one cell per round.

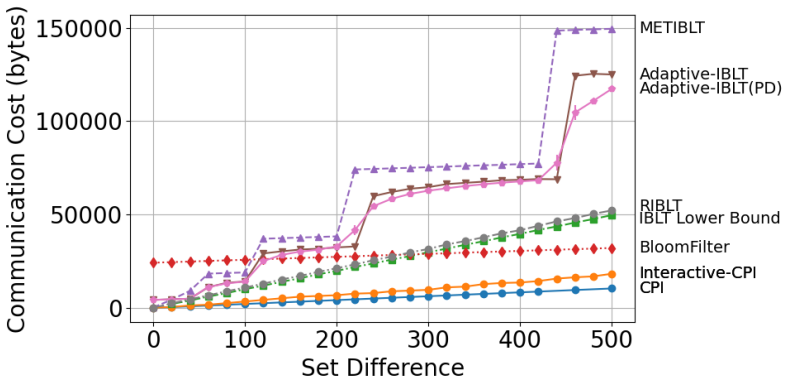

Figure 4: Communication Cost Comparison (Set Size = 10,000).

5.2 Communication Cost

Figure 4 presents total communication cost versus d. CPI and InteractiveCPI achieve the lowest communication overhead, with BloomFilter slightly higher and IBLT scaling linearly. CPI, BloomFilter, and IBLT require a known d.

AdaptiveIBLT incurs higher cost due to geometric retries when d is underestimated, while its partial decoding variant reduces this overhead by avoiding retransmission of already recovered elements. METIBLT has the highest communication cost, as its redundancy is independent of decoding success. RatelessIBLT achieves near-linear cost in d with minimal redundancy, but transmits over many rounds.

6 Implementation in Lightning Network

Lightning Network implementations rely heavily on timely broadcast of gossip messages to maintain an up-to-date view of the network topology. We suggest a more efficient design: performing a pre-connection gossip-state reconciliation using Gensync. In this architecture, a node executes a reconciliation protocol with trusted peers and can then proceed with standard Lightning connections with nearly identical gossip state. This avoids the large initial gossip flood required to "catch up".

We run a vanilla Lightning node connected to a highly connected peer (`1ML.com node ALPHA`) for 1 hour to populate its gossip store, then take a snapshot. A second snapshot is taken ∼20 minutes later, reflecting 19 minutes 35 seconds of gossip-based flooding. The snapshots are 22.26,MB (77,862 entries) and 23.61,MB (82,068 entries). We run each set reconciliation protocol 20 times on these snapshots and measure synchronization time relative to the flooding time.

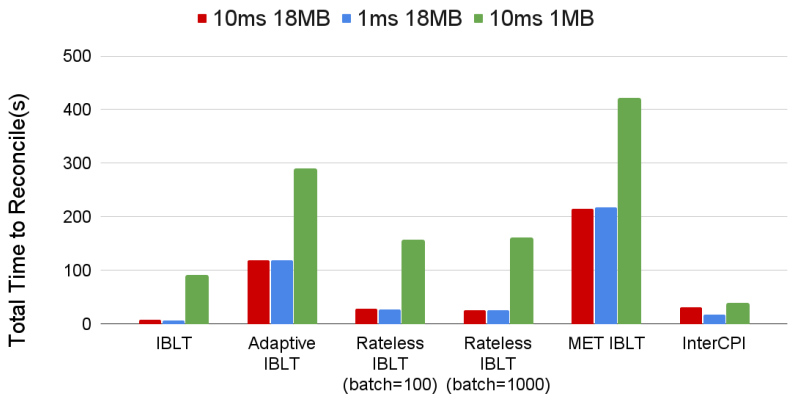

Figure 5: TTR comparison of reconciliation protocols across varying network bandwidths, evaluated on two gossip snapshots collected 20 minutes apart.

As shown in Figure 5, IBLTSync, AdaptiveIBLT, RatelessIBLT, METIBLT, and InteractiveCPI all outperform the existing gossip-based reconciliation with respect to TTR, as they all take less than 19 minutes to complete successfully. CPISync (not shown in the figure) takes approximately 40 minutes and is the only protocol to perform worse than the gossip dissemination, due to the long computation time.

We also compare the communication costs of each reconciliation protocol. Generally, IBLT-based protocols have much higher communication costs: METIBLT has the highest at 28 MB; AdaptiveIBLT and RatelessIBLT are slightly lower at 22 MB and 18 MB repectively; IBLT serves as the baseline of 10 MB;. The CPI and InteractiveCPI protocols have the lowest communication costs at 1.5 and 1.8 MB respectively.

7 Conclusion

This paper revisits Lightning Network gossip through the lens of set reconciliation. Our packet-level measurements of flooding-based gossip showcase a persistent inefficiency: although the rate of growth of new data is fairly low, we observed significant amounts of low-level network traffic, suggesting structural redundancy in the current approach.

Motivated by these observations, we conduct a systematic, fair evaluation of reconciliation protocols in GenSync under realistic network conditions, focusing on those without prior knowledge of the set difference. Our results show protocol choice depends heavily on network constraints: redundancy-heavy designs (METIBLT) trade bandwidth for robustness, while highly interactive designs (RatelessIBLT and other multi-round protocols) suffer latency amplification

under nontrivial RTTs. These findings reinforce that asymptotic communication complexity is insufficient to predict real performance.

To this point, we introduced AdaptiveIBLT, a simple runtime-adaptive IBLT-based protocol, enhanced by partial decoding to reuse intermediate progress across rounds. Empirically, AdaptiveIBLT provides a robust performance trade-off across diverse workloads and network settings. By evaluating on real gossip snapshots, we showed that reconciliation can reduce gossip convergence time by orders of magnitude.

Future work may include integrating additional protocols such as CertainSync [15] into the evaluation framework in order to provide a broader comparison across rateless and deterministic designs. We also envision studying reconciliation under more diverse churn and adversarial conditions, and exploring more designs to provide both fast startup and peer-to-peer freshness without relying on trusted providers.

Acknowledgments

This work was supported in part by the National Science Foundation under Grant CNS-2210029 and by the Red Hat Collaboratory at Boston University.

References

[1] A. M. Antonopoulos, O. Osuntokun, and R. Pickhardt, *Mastering the Lightning Network*. Sebastopol, CA: O'Reilly Media, 2021, chapter 11: Gossip and the Channel Graph.

[2] N. Gögge, E. Rohrer, and F. Tschorsch, "On the routing convergence delay in the lightning network," 2022. [Online]. Available: https://arxiv.org/abs/2205.12737

[3] J. Harvey, "Gossip observer: Monitoring gossip propagation in the lightning network," https://github.com/jharveyb/gossip_observer, 2025.

[4] X. He, Y. Cui, and Y. Jiang, "An improved gossip algorithm based on semi-distributed blockchain network," in *2019 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC)*, 2019, pp. 24–27.

[5] Lightning Network Developers, "Ln summit 2024 notes: Summary and commentary," https://delvingbitcoin.org/t/ln-summit-2024-notes-summary-commentary/1198, 2024.

[6] Y. Minsky, A. Trachtenberg, and R. Zippel, "Set reconciliation with nearly optimal communication complexity," *IEEE Transactions on Information Theory*, vol. 49, no. 9, pp. 2213–2218, 2003.

[7] B. H. Bloom, "Space/time trade-offs in hash coding with allowable errors," *Communications of the ACM*, vol. 13, no. 7, pp. 422–426, July 1970.

[8] M. T. Goodrich and M. Mitzenmacher, "Invertible Bloom lookup tables," in *49th Annual Allerton Conference on Communication, Control, and Computing*. IEEE, 2011, pp. 792–799.

[9] N. Boškov, A. Trachtenberg, and D. Starobinski, "Gensync: A new framework for benchmarking and optimizing reconciliation of data," *IEEE Transactions on Network and Service Management*, 2022.

[10] Lightning Dev Kit Team, "Rapid gossip sync: Efficient graph synchronization for lightning nodes," https://lightningdevkit.org/blog/announcing-rapid-gossip-sync/, 2022.

[11] Y. Minsky and A. Trachtenberg, "Scalable set reconciliation," in *Proceedings of the Annual Allerton Conference on Communication, Control, and Computing*, vol. 40, no. 3. The University; 1998, 2002, pp. 1608–1617.

[12] D. Starobinski, A. Trachtenberg, and S. Agarwal, "Efficient PDA synchronization," *IEEE Trans. on Mobile Computing*, vol. 2, no. 1, January-March 2003.

[13] F. Lázaro and B. Matuz, "A rate-compatible solution to the set reconciliation problem," *IEEE Transactions on Communications*, vol. 71, no. 10, pp. 5769–5782, 2023.

[14] L. Yang, Y. Gilad, and M. Alizadeh, "Practical rateless set reconciliation," in *Proceedings of the ACM SIGCOMM 2024 Conference*, ser. ACM SIGCOMM '24. Association for Computing Machinery, 2024, p. 595–612. [Online]. Available: https://doi.org/10.1145/3651890.3672219

[15] T. Keniagin, E. Yaakobi, and O. Rottenstreich, "CertainSync: Rateless set reconciliation with certainty," Cryptology ePrint Archive, Paper 2025/623, 2025. [Online]. Available: https://eprint.iacr.org/2025/623

[16] N. Boskov, X. Chen, S. Simsek, A. Trachtenberg, and D. Starobinski, "Out-of-band transaction pool sync for large dynamic blockchain networks," *International Journal of Network Management*, vol. 34, no. 5, p. e2265, 2024.

[17] A. P. Ozisik, G. Andresen, B. N. Levine, D. Tapp, G. Bissias, and S. Katkuri, "Graphene: efficient interactive set reconciliation applied to blockchain propagation," in *Proceedings of the ACM Special Interest Group on Data Communication*, ser. SIGCOMM '19. Association for Computing Machinery, 2019, p. 303–317.

[18] Y. Han, C. Li, P. Li, M. Wu, D. Zhou, and F. Long, "Shrec: bandwidth-efficient transaction relay in high-throughput blockchain systems," in *Proceedings of the 11th ACM Symposium on Cloud Computing*, ser. SoCC '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 238–252. [Online]. Available: https://doi.org/10.1145/3419111.3421283

[19] B. Lantz, B. Heller, and N. McKeown, "A network in a laptop: rapid prototyping for software-defined networks," in *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks*, ser. Hotnets-IX. New York, NY, USA: Association for Computing Machinery, 2010. [Online]. Available: https://doi.org/10.1145/1868447.1868466

[20] X. Chen *et al.*, "Gensync: A network-aware set reconciliation framework," https://github.com/nislab/gensync, 2026.