Random Linear Network Coding for Scalable BlockDAG Propagation in Kaspa

Gordon Murray
linkedin.com/in/gordonmurray333

August 2025

Abstract

Kaspa is a high-throughput cryptocurrency that employs a blockDAG consensus protocol (GHOSTDAG) to achieve rapid block creation while retaining robust security. While consensus rules are efficient, the peer-to-peer propagation of block data and transactions remains a limiting factor, especially under high block rates. Random Linear Network Coding (RLNC) has been shown to improve bandwidth efficiency and resilience in wireless and overlay networks by transmitting coded combinations of data packets rather than raw duplicates. This paper explores the application of RLNC to Kaspa's networking layer, providing a theoretical framework, potential message formats, performance trade-offs, and an incremental deployment path. We argue that RLNC can significantly reduce redundant bandwidth, improve propagation robustness, and help Kaspa scale to orders of magnitude higher transaction throughput without altering its consensus protocol.

1 Introduction

Kaspa's blockDAG architecture allows multiple blocks to be produced and accepted in parallel, providing higher throughput compared to linear blockchains. However, parallelism also increases network load: many blocks contain overlapping sets of transactions, and multiple tips must be relayed simultaneously. Traditional gossip protocols forward these blocks redundantly, wasting bandwidth and creating propagation bottlenecks.

Random Linear Network Coding (RLNC) offers a potential solution. By transmitting linear combinations of data packets, peers can ensure that nearly every received byte is *innovative*, adding new information that contributes to decoding the original blocks and transactions. This property makes RLNC a natural fit for Kaspa's DAG, where multiple tips and redundant transactions are the norm.

2 Background

2.1 BlockDAG Consensus

Kaspa uses the GHOSTDAG protocol, which generalizes Nakamoto consensus to a Directed Acyclic Graph (DAG) of blocks. **Terminology note:** Unlike longest-chain protocols, Kaspa does *not* orphan blocks; all valid blocks are admitted to the DAG and later *ordered* by GHOSTDAG, which partitions blocks into blue/red sets for scoring while preserving them in the history. Blocks reference multiple parents, enabling high throughput and low confirmation latency. Propagation delay is

critical: slow relays reduce miner synchronization and increase ordering lag (and can lower a block's blue-set inclusion/score), even though Kaspa does not orphan blocks.

2.2 Random Linear Network Coding

RLNC transmits packets as random linear combinations of source packets over a finite field (typically $GF(2^8)$). Receivers collect coded packets until the system of equations becomes full rank, after which they decode all original packets simultaneously. RLNC improves throughput, resilience to packet loss, and efficiency in networks with overlapping information.

3 RLNC for Kaspa: Problem Fit

Kaspa's propagation layer faces three major challenges:

1. **Redundant transaction transmission.** Many tips share overlapping subsets of mempool transactions.
2. **Concurrent block bodies.** Multiple blocks are generated per second, often referencing similar transactions, stressing bandwidth.
3. **Global latency constraints.** To minimize ordering lag and maximize blue-set inclusion (blue score), block bodies must reach peers quickly and efficiently.

RLNC addresses these by:

- Mixing overlapping transactions into coded packets, avoiding duplicate relays.
- Allowing peers to reconstruct multiple block bodies from a smaller set of coded shards.
- Ensuring any coded shard is useful (*innovative*), reducing wasted bandwidth.

3.1 Gossip Limitations and RLNC

Bitcoin and other longest-chain protocols rely on a flooding-style gossip protocol in which nodes announce new transactions and blocks to all peers, leading to substantial redundancy. This causes two well-known issues: (i) excessive bandwidth use when many peers already possess the same payloads, and (ii) a latency–bandwidth tradeoff, since suppressing redundant gossip risks delaying propagation. In Bitcoin, these effects directly translate into orphaned blocks and wasted mining work.

Kaspa shares some of these network-layer inefficiencies, since its peer-to-peer layer also gossips transaction and block announcements. However, the blockDAG consensus model changes the outcome:

- **All blocks are accepted.** Kaspa does not orphan valid blocks; they are included in the DAG and ordered by GHOSTDAG. Thus, wasted mining work from stale blocks is avoided.
- **Ordering lag replaces orphaning.** Slow propagation still harms miners, because blocks that arrive late may be excluded from many future blocks' blue sets, reducing their blue score accumulation. The effect is softer than orphaning but still a competitive disadvantage.
- **Redundancy persists.** As in Bitcoin, overlapping transactions and multiple concurrent blocks cause significant duplicate transmission in the gossip layer.

RLNC's role. By turning overlapping payloads into innovative coded shards, RLNC directly addresses the redundancy problem and reduces ordering lag. In Kaspa this does not prevent blocks from being accepted—since they always are—but it increases the chance that blocks achieve timely blue-set inclusion, improving fairness and synchronization across the DAG. Thus RLNC mitigates the same bandwidth limitations that Bitcoin faces, but in Kaspa the benefit manifests as lower ordering skew rather than lower orphan rate.

4 System Design

4.1 Architecture Overview

- **Headers** are transmitted unencoded to preserve immediacy and allow miners to recognize new tips quickly.
- **Block bodies and transaction batches** are grouped into coding windows.
- Peers generate coded shards by applying random coefficients over $GF(256)$ to the chunks of data in the window.
- Receivers decode once enough innovative shards have been collected, verifying results against Merkle commitments advertised alongside headers.

4.2 Windowing Strategy

A sliding window maintains the most recent W block bodies and mempool batches. Window size is a tunable parameter (e.g., 16–64 items) balancing decoding complexity and propagation efficiency.

4.3 Verification

Merkle roots for each block body and transaction batch are committed to in advance. After decoding, peers verify reconstructed data against these roots, discarding invalid data before acceptance.

4.4 Integration with Existing Protocol

RLNC is implemented as an optional extension at the P2P layer. Legacy nodes fall back to requesting block bodies directly, while upgraded nodes benefit from coded propagation. This ensures backwards compatibility and gradual adoption.

5 Performance Analysis

5.1 Bandwidth Efficiency

Without RLNC, redundant transaction overlap leads to near-quadratic redundancy in dense peer topologies. RLNC eliminates duplication by ensuring each coded shard carries useful information. Simulation studies in analogous networks show 20–40% bandwidth reduction.

5.2 Latency

Decoding introduces computational overhead (Gaussian elimination over $GF(256)$), but decoding complexity grows quadratically with window size and is tractable for typical parameters (e.g., 16–32 items of 2 MB each). Parallel GPU-accelerated decoding can further reduce latency.

5.3 Robustness

RLNC inherently provides resilience to packet loss and peer churn, as any innovative shard contributes to decoding. This robustness is valuable for Kaspa, where wide global peer distribution introduces heterogeneous bandwidth and latency conditions.

5.4 Finality Effects: A Simple Propagation-to-Depth Model

RLNC does not alter consensus rules, but it changes the *time profile of visibility* of a newly created block across the mining power. We model the fraction of global hashpower that has received and validated a block by time t as $A(t) \in [0,1]$ ("awareness"). Let λ denote the global block creation rate, and let $p_{\text{ref}} \in (0,1]$ be the probability that an aware miner includes (directly or indirectly) the block in its parent set in a way that contributes to the block's blue-set inclusion (this absorbs policy and DAG topology effects). Then the instantaneous expected rate at which the block accrues blue score is

$$\beta(t) \;\approx\; c\,\lambda\,A(t)\,p_{\text{ref}}\,, \tag{1}$$

where c is a constant of proportionality capturing that multiple contemporaneous blocks can contribute to a single block's blue score in a blockDAG.

Hence the expected blue score at time t is

$$B(t) \;=\; \int_0^t \beta(u)\,du \;\approx\; c\,\lambda\,p_{\text{ref}} \int_0^t A(u)\,du\,. \tag{2}$$

Practical finality is reached once $B(t)$ exceeds a depth threshold D_ϵ that bounds reordering probability by ϵ (the threshold depends on security parameters such as k and the observed adversarial hashpower).

Awareness dynamics. A common and analytically convenient model is an exponential approach to 1 with time constant τ (the "network mixing time"):

$$A_{\text{base}}(t) \;=\; 1 - e^{-t/\tau}\,. \tag{3}$$

Under RLNC, the total bytes needed to disseminate a window of items is reduced by an efficiency factor $\eta \in (0,1]$ (with $\eta < 1$ better), while decoding adds a startup overhead δ (seconds). A first-order approximation is

$$A_{\text{rlnc}}(t) \;=\; \begin{cases} 0, & t \le \delta, \\ 1 - \exp\big(-(t-\delta)/(\eta\tau)\big), & t > \delta. \end{cases} \tag{4}$$

The corresponding expected blue score is

$$B_{\text{rlnc}}(t) \;\approx\; c\,\lambda\,p_{\text{ref}}\Big[(t-\delta)_+ \;-\; \eta\tau\Big(1 - e^{-(t-\delta)/(\eta\tau)}\Big)\Big]\,, \tag{5}$$

where $(x)_+ = \max\{0, x\}$.

Time-to-depth comparison. Define the time to reach depth D_ϵ by $T = \inf\{t : B(t) \ge D_\epsilon\}$. For t beyond a few mixing constants ($t \gtrsim 3\tau$), $A(t) \approx 1$ and blue score accumulates at roughly constant slope $c\lambda p_{\text{ref}}$. Thus a practical approximation is

$$T_{\text{base}} \;\approx\; 3\tau \;+\; \frac{D_\epsilon}{c\,\lambda\,p_{\text{ref}}} \quad \text{and} \quad T_{\text{rlnc}} \;\approx\; \delta \;+\; 3\eta\tau \;+\; \frac{D_\epsilon}{c\,\lambda\,p_{\text{ref}}}. \tag{6}$$

Therefore, the improvement in *practical time to finality* satisfies

$$\Delta T \;=\; T_{\text{base}} - T_{\text{rlnc}} \;\approx\; 3(1-\eta)\,\tau \;-\; \delta\,. \tag{7}$$

Equation (7) makes the trade-off explicit: RLNC helps whenever $3(1-\eta)\tau > \delta$, i.e., when bandwidth savings dominate decoding overhead. In high-throughput or bandwidth-constrained regimes (large τ), RLNC yields a meaningful reduction in the wall-clock time to reach depth D_ϵ; in CPU-bound environments (large δ), careful engineering (e.g., sparse coding, GPU decoding) keeps δ small enough to preserve the advantage.

Variance and synchronization. Beyond the mean, RLNC also reduces the *variance* of propagation delays across peers, tightening the distribution of awareness times and shrinking the window during which different miners see materially different DAG views. This concentrates the blue-score growth trajectory $B(t)$ across nodes, further stabilizing perceived confirmation times.

5.5 Latency–Bandwidth Trade-off in Kaspa at 10 bps

Kaspa currently operates at approximately 10 blocks per second. At this high block rate, transaction overlap between blocks is significant, with many concurrent tips carrying the same transactions. This amplifies the benefit of coding: redundant payloads consume bandwidth in naive gossip but are eliminated in RLNC.

Let W denote the window size (blocks mixed), S the average block size, and ρ the average overlap fraction of payloads between blocks. Then the expected uncoded union size is

$$U \approx S\big[1 + (W-1)(1-\rho)\big], \tag{8}$$

and the RLNC efficiency factor is

$$\eta \approx \frac{U}{WS}. \tag{9}$$

For example, with $W = 10$ and $\rho = 0.6$, we obtain $\eta \approx 0.46$, corresponding to a 54% reduction in transmitted bytes.

Combining this with the finality model of Section 5.4, RLNC reduces practical time to depth when

$$3(1-\eta)\tau > \delta, \tag{10}$$

i.e. when the savings from reduced mixing time τ exceed the decode overhead δ. With $\tau \approx 80\,\text{ms}$ and $\eta = 0.46$, RLNC can tolerate δ up to $\sim 130\,\text{ms}$ before losing advantage—easily achievable with optimized decoding.

5.6 Parameterization for Kaspa's ASIC-Dominated Network

Kaspa is now a purely ASIC-mined network. Propagation improvements affect miners through blue-set inclusion: slow propagation does not orphan blocks, but it reduces their chance of timely inclusion in the blue set, diminishing blue score accumulation. RLNC must therefore keep its decoding overhead δ much smaller than the 100 ms inter-block interval.

Practical relay parameters for Kaspa's current regime are:

- **Mode:** Systematic RLNC over block bodies (send some uncoded chunks first).
- **Window size W:** 8–12 blocks ($\approx$1 s of history).

- **Chunk size:** 1–2 KB per coded unit.
- **Coefficient density:** Sparse (8–16 non-zeros) to allow near-linear decoding.
- **Systematic ratio:** 60–70% uncoded chunks, remainder coded shards.
- **Decode target:** <5 ms on relay CPUs using AVX2/AVX-512 optimized GF(256) arithmetic.
- **Deployment:** First in private pool relay overlays (FIBRE-style), then general P2P.

These settings ensure that δ remains in the few-millisecond range, well below the inter-block interval. Bandwidth savings of 40–60% are achievable in practice, reducing propagation skew and making blue-set inclusion more equitable across miners, especially those with weaker connectivity.

6 Security Considerations

- **Spam resistance:** Commitments (Merkle roots) prevent peers from flooding the network with undecodable or incorrect mixes.
- **Denial-of-service:** Encoding/decoding overhead must be capped per peer to prevent adversarial rank inflation.
- **Consensus neutrality:** RLNC affects only propagation; headers and consensus rules remain unchanged.

7 Deployment Roadmap

1. **Phase I:** Erasure-coded block relay. Each block body is distributed with RLNC shards instead of raw chunks.
2. **Phase II:** Transaction batch gossip. Mempool transactions are disseminated in RLNC-coded batches.
3. **Phase III:** Multi-tip coding. Sliding-window RLNC spans multiple tips simultaneously, further reducing redundancy.

Each phase is optional and backward-compatible, enabling progressive rollout.

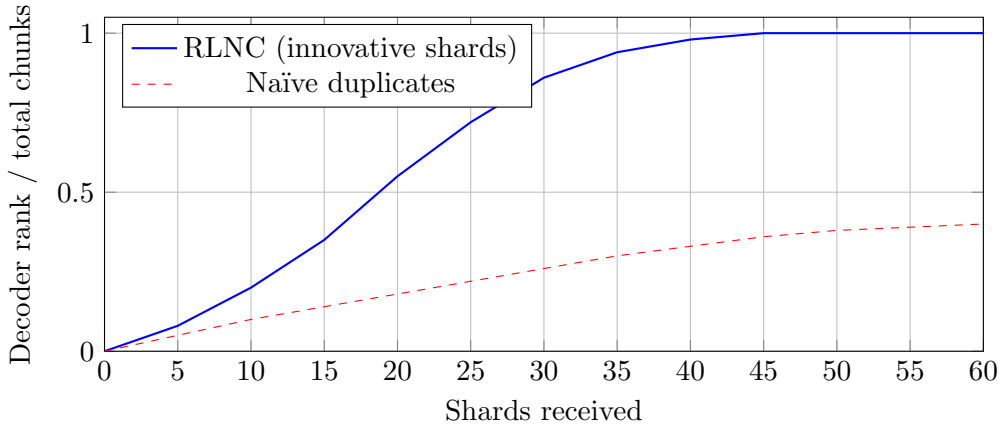

Figure 1: Conceptual rank growth: with RLNC, most received shards are innovative, driving rank quickly to full; naive gossip wastes bandwidth on duplicates that do not increase rank.

8 Figures

9 Algorithms

Algorithm 1: RLNC shard encoding over $GF(256)$

input : Window W with chunk matrix $B \in GF(256)^{m \times n}$; chunk size m
output : Shard $(\mathbf{c}, \mathbf{y})$ with $\mathbf{y} = \mathbf{c}^\top B$
Sample $\mathbf{c} \in GF(256)^n$ with non-zeros per desired density
for $j \leftarrow 1$ **to** m **do**
  $y_j \leftarrow 0$
for $i \leftarrow 1$ **to** n **do**
  for $j \leftarrow 1$ **to** m **do**
    $y_j \leftarrow y_j \oplus (c_i \otimes B_{ji})$
return $(\mathbf{c}, \mathbf{y})$

10 Related Work

- **Graphene / Compact Blocks:** Efficient transaction set reconciliation for Bitcoin and Ethereum. RLNC can be seen as a generalization providing probabilistic guarantees of innovation.
- **FIBRE (Bitcoin):** High-speed relay using UDP and forward error correction; RLNC extends this approach to multi-item coding.
- **RLNC in wireless networks:** Proven throughput and resilience gains in multicast and broadcast scenarios.

11 Conclusion

Random Linear Network Coding offers a compelling upgrade path for Kaspa's propagation layer. By reducing redundancy, improving resilience, and enabling efficient multi-tip dissemination, RLNC

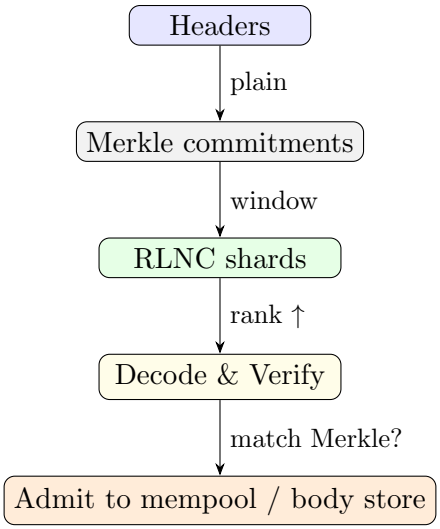

Figure 2: Decode pipeline: headers stay plain; commitments define the window; coded shards raise rank until decoding succeeds; reconstructed chunks are verified against commitments before admission.

aligns naturally with Kaspa's blockDAG architecture. Importantly, RLNC is consensus-neutral, requiring no protocol fork, and can be deployed incrementally.

12 Future Work

- Prototype implementation in Kaspa's Rust P2P stack.
- Simulation studies under realistic network topologies.
- GPU-accelerated decoding benchmarks.
- Integration with transaction reconciliation protocols (e.g., IBLT + RLNC hybrid).

References

[1] S. Katti, H. Rahul, W. Hu, D. Katabi, M. Médard, and J. Crowcroft. *XORs in the Air: Practical Wireless Network Coding.* SIGCOMM 2006.

[2] Y. Sompolinsky and A. Zohar. *Phantom: A Scalable BlockDAG Protocol.* IACR ePrint 2018.

[3] P. Bissias, G. Andresen, B. Levine, and S. Katkuri. *Graphene: Efficient Interactive Set Reconciliation Applied to Blockchain Propagation.* SIGCOMM 2018.

[4] M. Corallo. *Relay Networks and FIBRE.* Bitcoin Core Docs, 2016.

13 RLNC and DAGKNIGHT

DAGKNIGHT is Kaspa's planned consensus upgrade designed to achieve optimal latency, maximizing throughput and confirmation speed in the blockDAG setting. It strengthens the

guarantees of GHOSTDAG by providing faster finality and improved security margins in high-throughput environments. A key challenge for DAGKNIGHT will be ensuring that all blocks—produced at very high rates—are rapidly visible to all miners and validators, so that their ordering and finality decisions are made on complete information.

How RLNC can help:

- *Reduced ordering lag:* RLNC accelerates block-body propagation, ensuring that DAGKNIGHT's fast confirmation logic has access to all recent blocks when computing votes and ordering.
- *Maintained parallelism:* Because DAGKNIGHT accepts and orders all blocks in parallel, efficient dissemination of many concurrent tips is essential. RLNC prevents bandwidth saturation when hundreds of parallel blocks are relayed globally.
- *Resilience under stress:* In bursty high-throughput scenarios (e.g., sudden transaction floods), RLNC guarantees that any received shard contributes to decoding, reducing the chance of missing data that would delay DAGKNIGHT's confirmation pipeline.

Thus, RLNC is a natural companion to DAGKNIGHT: while DAGKNIGHT enhances consensus-level scalability, RLNC strengthens the underlying network layer, ensuring that the protocol's assumptions about low-latency and near-synchronous block availability are met in practice.

A Appendix: Blue Score, k-Clusters, and RLNC's Impact

Kaspa's GHOSTDAG consensus orders blocks by partitioning them into a *blue set* and a *red set*, where the blue set approximates a k-cluster—a set of blocks with bounded pairwise distance. Each block is assigned a *blue score* representing its accumulated weight in the consensus order.

- Blocks that propagate quickly are more likely to be included in the blue set of subsequent blocks, thereby contributing to the main ordering and achieving higher blue score.
- Slow-propagating blocks are more likely to be excluded (red) in many future blocks' cluster calculations, reducing their effective weight and delaying their confirmation depth.

Impact of RLNC: By improving bandwidth efficiency and ensuring that block bodies are received quickly, RLNC reduces propagation skew between peers. This increases the probability that new blocks see a more complete DAG when selecting parents and constructing their blue sets. Consequently, RLNC indirectly improves fairness in blue score accumulation and enhances the throughput that the k-cluster can sustain without consensus degradation.