

HPPC: High-Performance Packet Cache for WAN Nodes with DDR Optimization

Shuangyan Wang*, Lei Liu*, Yifei Li*

*National Network New Media Engineering Research Center, Institute of Acoustics,
Chinese Academy of Sciences, Beijing 100190, China
School of Electronic, Electrical and Communication Engineering,
University of Chinese Academy of Sciences, Beijing 100049, China
wangsy@dsp.ac.cn, liul@dsp.ac.cn, liyf@dsp.ac.cn

Abstract—High-speed wide-area networks (WANs) exhibit large bandwidth–delay products (BDPs), so even modest random packet loss can drive congestion-controlled reliable transports far below line rate, motivating hop-local loss recovery at in-path devices. Realizing such hop-by-hop recovery requires buffering a sliding window of recently transmitted packets, yet commodity NICs and switches expose only limited on-chip SRAM, and naive use of off-chip DRAM incurs pointer chasing, poor row-buffer locality, and high read amplification. HPPC is a DRAM-resident packet cache that stores all buffered packets in a single circular buffer and maintains a compact, two-level flow-aware index, turning cache operations into predominantly sequential DDR accesses while elastically sharing capacity across flows. Implemented on a 100 Gb/s FPGA-based network card with dual DDR4 channels, HPPC sustains line-rate throughput for 1500-byte packets (≈ 8.3 Mpps) and, under 1% packet loss across more than 1,000 concurrent flows, maintains at least 94% of line-rate goodput on the protected WAN segment without requiring changes to TCP/RDMA endpoints.

Index Terms—Wide-area networks, packet caching, reliable transport, DDR, buffer management.

I. INTRODUCTION

MODERN cloud and AI workloads increasingly span geo-distributed datacenters interconnected by 100 Gb/s WAN links over tens to thousands of kilometers [1], [2]. These paths have large bandwidth–delay products (BDPs), so even modest loss rates and tens-of-milliseconds RTTs can push TCP/RDMA throughput far below line rate [3], [4]. Lossless Ethernet mechanisms such as PFC for RoCE are difficult to operate across wide-area networks and risk head-of-line blocking and congestion spreading [5], [6]. Meanwhile, datacenter congestion-control schemes such as DCTCP and HPCC are tuned for shallow-RTT intra-datacenter settings and retain a purely end-to-end control loop, leaving WAN feedback delay fundamentally unchanged [7], [8].

One line of work embeds reliability and selective retransmission into RDMA-capable NICs [9], enabling fast recovery in low-latency data centers [10], [11]. Over high-BDP WAN segments, however, each loss still incurs at least one full end-to-end RTT, and the per-flow packet state required for selective retransmission must fit into limited on-chip SRAM; with many concurrent flows and large windows, NIC memory is quickly exhausted and throughput collapses [10], [11]. A

complementary approach is to terminate ARQ hop-locally at in-path nodes or switches so that recovery operates over a short segment. Software-based link-local ARQ and P4-programmable dataplanes follow this direction, but sustaining 100 Gb/s typically requires substantial CPU budgets, specialized ASIC support, or both [12]–[14].

Hop-local recovery at 100 Gb/s therefore hinges on an in-network packet cache that can run at line rate. Yet high-speed packet caching is constrained by data-structure and memory-system limitations. Conventional designs maintain per-flow buffers or linked chains of packets, inducing pointer chasing and non-sequential DRAM accesses on lookup and eviction; these random accesses limit sustainable bandwidth and inflate tail latency [15], [16]. Per-packet indexing further encounters a memory wall: with ≈ 1500 B packets and ≈ 32 B index entries, a ≈ 210 Mbit on-chip SRAM can index only ≈ 1.2 GB of DRAM, leaving most of a 10 GB DRAM pool unused at packet granularity [15]. Moving the index to DRAM introduces at least one extra DRAM access per lookup or update, which is incompatible with 100 Gb/s timing margins [15]. A practical design must therefore avoid per-flow linked structures, minimize random memory accesses, and keep the latency-critical part of the index on chip [15].

To address these constraints, we propose HPPC (High-Performance Packet Cache), a node-centric hop-by-hop reliability architecture for high-speed WAN segments. HPPC deploys, at each WAN node, a unified DDR-resident circular packet buffer, a compact two-level per-flow index, and a lightweight NACK-based repair protocol between paired nodes. Cache insertions and evictions are issued as sequential DDR bursts, eliminating per-flow heap management and fragmentation. The two-level index maps each (flow, sequence number) pair to a ring position in $O(1)$ time with a small on-chip footprint, enabling microsecond-scale fetch of missing packets on demand. Losses are repaired within a single node-to-node RTT, masking WAN losses from end hosts while remaining transparent to TCP/RDMA transports [15], [16]. Compared to NIC-only selective retransmission and P4-based link-local ARQ, HPPC decouples reliability from NIC SRAM and full end-to-end RTT by offloading bulk packet storage to commodity DDR while maintaining line-rate operation [10],

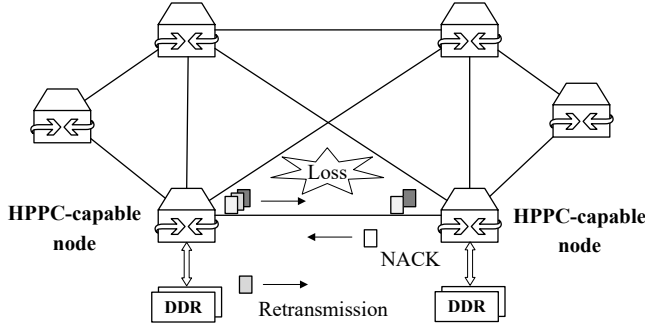


Fig. 1. HPPC deployment model.

[12], [14].

The main contributions of this work are threefold. (i) We design a ring-based packet store shared across all flows, together with a compact two-level per-flow index, that provides $O(1)$ lookup and turns cache I/O into sequential DDR accesses for high space and bandwidth efficiency (Section III). (ii) We formalize a model for WAN-segment packet caches, derive design requirements—BDP-driven ring sizing, RTT-constrained readback budget, and flow-scale indexing—and define metrics for sequential write bandwidth, read amplification, and space efficiency (Section II). (iii) We implement a 100 Gb/s FPGA node prototype with dual DDR4 banks and show that HPPC delivers near line-rate goodput under non-zero loss and scales to thousands of concurrent flows (Section IV), and we discuss limitations and deployment considerations in Section V.

II. SYSTEM MODEL & CACHE REQUIREMENTS

A. Model and Assumptions

Fig. 1 shows the HPPC deployment model. Two HPPC-capable nodes on the path (upstream and downstream) define a reliability segment, while end hosts and transport protocols (TCP/RDMA) remain unchanged. The upstream node forwards packets along the original path and, in parallel, clones them into a shared DDR-backed circular buffer (ring) while updating a two-level per-flow index. The downstream node monitors per-flow sequence continuity and emits NACK(flow, seq[/range]) when it detects gaps. Loss repair within the segment consists of an $O(1)$ index lookup followed by sequential reads from the ring to retransmit the missing packets. We assume that (i) NACKs are reliably delivered within the segment and (ii) both HPPC nodes share a consistent (FlowID, Seq) namespace.

HPPC uses two external DDR channels. A *data DDR* holds the unified circular packet buffer containing cloned payloads from all flows, and an *index DDR* holds metadata for the logical two-level index (L1 directory and L2 segment table). Only a small subset of L1 entries and control state are cached in on-chip SRAM; all bulk index data reside in the index DDR, so metadata traffic does not contend with payload bursts on

the data DDR. Together, the two DDR-backed nodes provide hop-local loss repair over the WAN segment.

We denote:

- link rate by R ,
- segment RTT by T_{seg} ,
- MTU by MTU ,
- ring slot size by S_{slot} ,
- number of concurrent flows by N_{flow} .

The bandwidth–delay product (BDP) of the segment is

$$\text{BDP}_{\text{seg}} = R \cdot T_{\text{seg}}, \quad (1)$$

which determines the amount of in-flight data that the ring and index must cover if HPPC is to repair losses within one segment RTT.

B. Cache Requirements

The ring must be large enough to cover the entire clone $\rightarrow$ loss detection $\rightarrow$ NACK return $\rightarrow$ readback $\rightarrow$ retransmit pipeline over one segment RTT, while leaving headroom for burst losses and control jitter:

$$C_{\text{ring}} \geq \alpha R T_{\text{seg}}, \quad (2)$$

where $\alpha \geq 1$ is a safety factor that accounts for additional buffering under bursty loss and control jitter. In practice, C_{ring} is provisioned in units of slots of size S_{slot} , aligned with the data DDR burst width.

Each cloned packet consumes exactly one address entry (AE) in the index. Let $\mathbb{E}[\text{pkt size}]$ denote the average packet size. The number of AEs that must be retained is on the order of

$$N_{\text{AE}} \gtrsim \frac{C_{\text{ring}}}{\mathbb{E}[\text{pkt size}]}, \quad (3)$$

which determines the size of the L2 segment pool and the footprint of page/subpage structures in the index DDR. L1 capacity scales with N_{flow} , but is small compared to L2 and is partially cached on chip.

Let B_{data} and B_{idx} be the sustained effective bandwidth of the data DDR and index DDR, respectively. On the data DDR, sequential clone writes and repair reads must satisfy

$$R_{\text{clone}} + \text{RA}_{\text{data}} \cdot R_{\text{repair}} \leq B_{\text{data}}, \quad (4)$$

where RA_{data} (read amplification) is the ratio of bytes read from the ring to bytes retransmitted. On the index DDR, metadata writes for AEs and metadata reads for repair must satisfy

$$R_{\text{idx,wr}} + \text{RA}_{\text{idx}} \cdot R_{\text{idx,rd}} \leq B_{\text{idx}}, \quad (5)$$

where RA_{idx} is the average number of index bytes read per repaired byte. HPPC is designed to achieve

$$\text{RA}_{\text{data}} \approx 1, \quad \text{RA}_{\text{idx}} = O(1), \quad (6)$$

through range-aware NACK coalescing and a pointer-free two-level index, so that each NACK range yields a bounded number of index DDR accesses and long, aligned bursts on the data DDR.

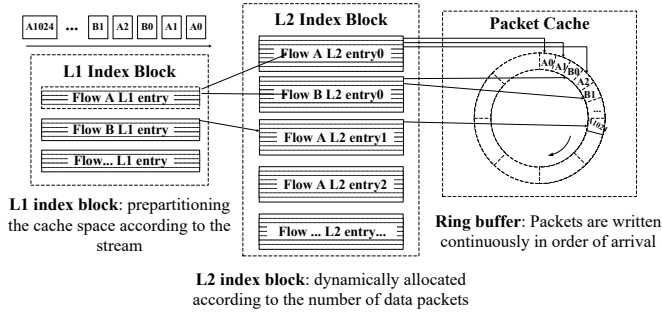


Fig. 2. HPPC packet cache organization.

III. HPPC CACHE ARCHITECTURE

Fig. 2 gives an overview of the HPPC packet cache. All packet payloads are stored in a unified DDR-backed ring buffer, while a flow-aware two-level index (L1 and L2) maps per-flow sequence numbers to ring locations.

A. Ring Layout in the Data DDR

HPPC stores packets from all flows in a single circular buffer (ring) of capacity C_{ring} in the data DDR, with a fixed slot size S_{slot} . On the write path, the upstream HPPC node forwards each packet on the original path and, in parallel, streams its payload into the ring using burst-aligned sequential writes; it simultaneously emits a compact address entry (AE) recording the starting ring slot index and packet length. The ring operates as a cyclic overwrite: a single write pointer advances by the number of slots consumed by each cloned packet, wrapping at the end of the allocated region and overwriting the oldest bytes.

The ring maintains no per-packet free list or garbage collector. Instead, overwritten references are detected and invalidated at lookup time by the index (Sec. III-B). Because packets are not segregated by flow, the data DDR sees a single flow-agnostic sequential writer, which maximizes burst efficiency and avoids allocator churn and pointer chasing that commonly hamper per-flow or linked-list caches [15].

B. Two-Level Index in the Index DDR

HPPC decouples data placement in the ring from flow metadata via a logical two-level index stored in the index DDR, with a small cache of hot entries on chip. For each active flow f , L1 stores the flow key, the maintained sequence window $[S_{\text{min}}, S_{\text{max}}]$, pointers to the head and tail of its L2 segment chain (segment IDs), and a generation/epoch tag. The total L1 footprint scales with N_{flow} .

L2 maintains, for each flow f , a chain of fixed-size segments; each segment contains exactly K AEs. Given a query (f, Seq) , HPPC computes

$$\Delta = \text{Seq} - S_{\text{min}}, \quad (7)$$

$$\text{seg_id} = \left\lfloor \frac{\Delta}{K} \right\rfloor, \quad (8)$$

$$\text{ae_idx} = \Delta \bmod K, \quad (9)$$

and then indexes the corresponding AE to obtain the ring slot index for that packet—an $O(1)$ lookup with no list traversal. Each AE encodes a ring slot index, packet length, and generation/epoch bits. Segments and AEs are physically laid out in the index DDR via an internal page/subpage/data mapping, which is transparent to this logical interface.

Let M_f denote the number of active L2 segments owned by flow f , and let $M_{\text{tot}} = \sum_f M_f$ be the total number of allocated segments, bounded by a global pool cap M_{max} . Two global queues in the index DDR manage this pool: a free FIFO and an oldest-full FIFO that tracks segment insertion order. When a flow f needs an additional segment, HPPC dequeues from the free FIFO; if it is empty, HPPC evicts the globally oldest full segment (which may belong to any flow g), unlinks it from g 's L2 chain by updating g 's L1 head, invalidates its AEs via generation/epoch tags, and reassigns the segment to f . This mirrors the ring's cyclic overwrite at the index layer, avoids per-flow reclamation and fragmentation, and lets each flow's footprint adapt to its recent write intensity under the global cap M_{max} . An AE is considered valid only if its generation/epoch matches the current L1 entry and the referenced ring slot has not yet been overwritten; otherwise, it is treated as expired, and the packet is no longer recoverable.

C. Readback Path (Range-Aware Coalesced Bursts)

Upon receiving a NACK from the downstream node, HPPC performs $O(1)$ L1+L2 lookups to resolve the corresponding ring slot indices, merges adjacent AEs that map to contiguous slots into longer read windows, and issues aligned burst reads from the ring to retransmit the missing packets. NACK parsing, index lookup, range coalescing, DDR read scheduling, and packet reconstruction are fully pipelined, with backpressure ensuring that repair traffic respects configured bandwidth caps.

D. Complexity and Resource Usage

On the write path, each packet incurs one L1 access (typically hitting the on-chip cache), one AE append into the tail L2 segment (index DDR write), and one burst-aligned sequential write into the data ring (data DDR access). On the repair path, each missing packet incurs an $O(1)$ L1+L2 lookup (index DDR read), range-coalesced sequential reads from the data ring, and a constant-time AE validity check. The data ring capacity C_{ring} is provisioned to cover one RTT's worth of in-flight data with headroom; the index DDR is sized to store N_{AE} AEs plus segment metadata under the global cap M_{max} , while on-chip SRAM holds only a small cache of L1 entries and control FIFOs.

Clone traffic and readback traffic run on decoupled AXI channels and are rate-limited (e.g., readback to 20–40% of the effective data DDR bandwidth) to protect the advancing ring tail; index DDR accesses are similarly bounded to avoid interfering with other controller clients. Compared to linked-list or chain-based caches, which induce pointer chasing and random DDR accesses on insert and lookup/replacement, HPPC's flow-agnostic sequential writes and range-coalesced reads match DDR's burst "sweet spot." At the same time, the

two-level index provides strict $O(1)$ addressing and elastic per-flow capacity under a global cap, without requiring the entire index to reside on chip [15].

IV. IMPLEMENTATION AND EVALUATION

We prototype HPPC on commercial Semptian MA-X1 FPGA accelerator cards, each based on a Xilinx Virtex UltraScale+ XCVU9P device. Each card exposes two external DDR4 channels populated with Micron MT40A512M16HA-083E devices: one is configured as the *index DDR* (DDR_idx) for the logical two-level index (L1 directory and L2 segment table), and the other as the *data DDR* (DDR_data) for the unified circular packet buffer (ring). Our testbed uses two such MA-X1 cards as HPPC-capable in-path nodes installed in two servers. Traffic is generated and measured using an Ixia XM100GE4CXP 4×100 Gb/s test module, and a dedicated WAN emulator on one of the links injects controllable packet loss between the two HPPC nodes, as illustrated in Fig. 3. Unless stated otherwise, all experiments use this split-memory configuration. This section first contrasts HPPC with a conventional packet cache on the same hardware platform, and then evaluates loss tolerance on a WAN-like 100 Gb/s segment. Comparisons to P4 and SR-NIC designs are analytical and use published results as qualitative reference points rather than head-to-head baselines.

A. Structural Comparison with a Chain-Based Packet Cache

On the same MA-X1 FPGA card and clock domain, we implement two cache organizations that share identical 100 Gb/s MAC/PCS blocks, parsers, and NACK-handling logic. HPPC writes all payloads sequentially into a unified ring in DDR_data and stores indexing metadata (L1 per-flow directory and L2 segments) in DDR_idx; every cloned packet triggers one burst-aligned write to DDR_data and one address-entry append to DDR_idx, while repair operations perform $O(1)$ L1+L2 lookups in DDR_idx and range-coalesced burst reads from DDR_data. The chain-based cache instead maintains per-flow linked lists of packet buffers with both payloads and per-packet pointers stored in a single DDR channel, so inserts and lookups require pointer chasing and non-sequential accesses on that channel.

Using the same synthetic traffic generator on the FPGA, HPPC sustains long aligned bursts on DDR_data and keeps the read amplification for repair

$$RA_{data} = \frac{\text{bytes read from DDR_data for repair}}{\text{bytes retransmitted on the wire}} \quad (10)$$

close to 1, because missing packets within a NACK range are merged into contiguous read windows. DDR_idx traffic remains well below its maximum bandwidth, since each NACK range triggers only a bounded number of L1/L2 accesses. The chain-based cache, in contrast, issues many short, scattered reads on DDR_data and exhibits significantly higher RA_{data} , as each repair follows per-flow pointers to locate the corresponding buffer. These observations are consistent with the design intent in Sec. III: HPPC aligns cache I/O with DDR burst

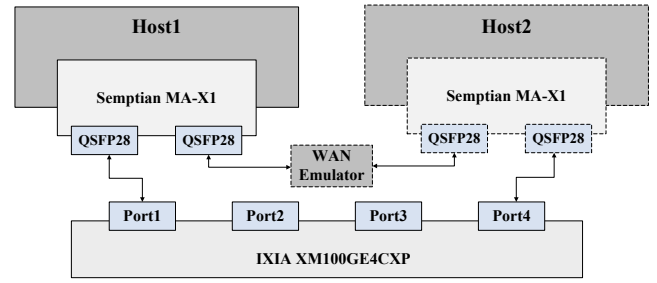


Fig. 3. Experimental setup for HPPC evaluation.

behavior, whereas per-flow chains are dominated by random accesses.

A single-level linear index provides a useful analytical baseline. One could store all payloads in the same unified ring as HPPC, while maintaining a flat per-packet index array in fast memory. This eliminates per-flow lists but still incurs two memory transactions per repair: one random access to fetch the index entry and one to fetch the payload. With 1500-byte packets and 32-byte per-packet index entries, the fast-to-slow memory ratio is about 1:46 [15]: a 210-Mbit SRAM can index only ≈ 1.2 GB of DRAM at packet granularity, leaving most of a 10 GB pool unused; moving the index itself to DRAM would double the number of DRAM transactions on the critical path. HPPC instead keeps the full index in DDR_idx but organizes it as fixed-size L2 segments with a global oldest-full FIFO, so that range-aware NACKs resolve runs of packets with a bounded number of DDR_idx accesses and long, coalesced bursts from DDR_data.

B. Loss Tolerance on a 100 Gb/s WAN-like Link

We next deploy HPPC in the testbed of Fig. 3, where two MA-X1/XCVU9P-based FPGA WAN cards act as HPPC-capable in-path nodes and the WAN emulator injects independent random loss between them. The Ixia XM100GE4CXP drives 100 Gb/s traffic across this emulated wide-area segment and reports wire-level throughput. The RTT across the lossy segment is fixed to a WAN-like value in the emulator configuration; other parameters (line rate, firmware, offloads, clocks) are held constant to avoid confounding factors such as time-synchronization skew.

Traffic consists of constant-bit-rate streams that attempt to saturate 100 Gb/s, with fixed packet sizes of 256, 512, 1024, 1500, 2048, and 4096 B. For each preset loss rate, swept from $10^{-5}\%$ to 5%, we record wire-level goodput at the HPPC nodes and verify residual loss at the receiver. The measured goodput curves for all packet sizes are shown in Fig. 4; the plotted values are taken directly from the combined FPGA and Ixia measurements.

For packets ≥ 1500 B, HPPC sustains near line-rate goodput in the operational regime (up to 0.1% loss). In Fig. 4a, at 0.1% preset loss, the 2048/4096 B curves remain essentially at 100 Gb/s, while the 1500 B curve stays around 97 Gb/s. Even at 1% loss, 2048/4096 B still deliver ≈ 96 –99 Gb/s

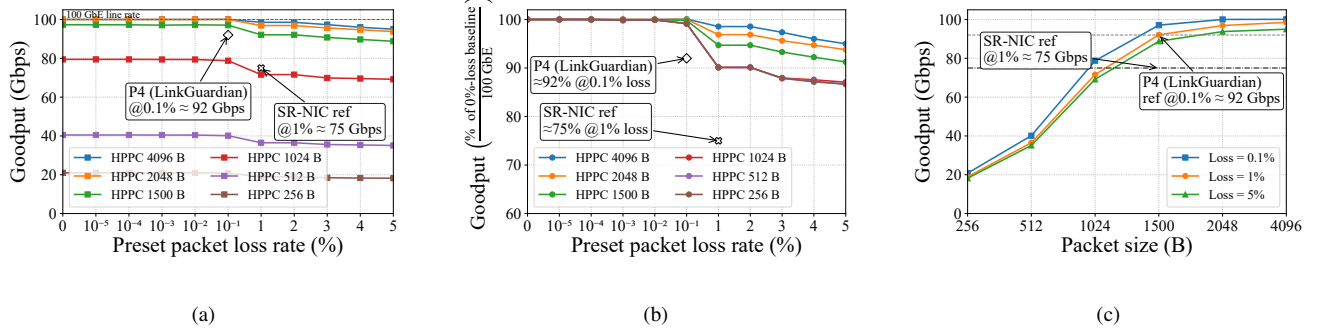


Fig. 4. HPPC goodput on a WAN-like 100 Gb/s segment. (a) Goodput vs. preset loss for multiple packet sizes. (b) Goodput normalized to each size's 0%-loss baseline. (c) Goodput vs. packet size at three fixed loss levels.

and 1500 B remains above 94 Gb/s. The 1024 B curve drops more quickly, and the 256/512 B curves plateau around 18–40 Gb/s. Auxiliary packet-per-second counters from the XM100GE4CXP module show that these plateaus coincide with the maximum packet rate the MA-X1/XCVU9P design can sustain, indicating a packet-rate bottleneck rather than cache misses or DDR contention.

Fig. 4b normalizes each curve to its own 0%-loss baseline. The curves for 1500–4096 B nearly overlap up to 0.1% loss and then exhibit similar degradation as loss increases, consistent with HPPC's design: clone writes are always sequential, and repair reads are long bursts independent of packet size, so loss mainly affects the fraction of DDR bandwidth devoted to readback. Fig. 4c fixes the loss rate (0.1%, 1%, 5%) and scans packet size. At 0.1% loss, all sizes ≥ 1500 B sit close to line rate; at 1% loss, goodput remains above 94 Gb/s; at 5% loss, large-packet goodput is still limited primarily by available DDR bandwidth rather than by the index organization.

The markers in Fig. 4 overlay published reference points from P4 and SR-NIC designs, but the corresponding experiments are not performed on our hardware. For P4, we take a 100 Gb/s LinkGuardian-style design [12] as a representative dataplane loss-masking system implemented in switches. Its reported throughput at 0.1% loss is around 92 Gb/s; in Fig. 4a, this point lies below the HPPC curves for packet sizes ≥ 1500 B, and in Fig. 4b it corresponds to roughly 92% of line rate. This reflects resource differences: switch ASICs rely on on-chip buffers and recirculation in a deeply pipelined dataplane, whereas HPPC uses off-chip DDR with a ring-plus-index structure tailored to burst-friendly access. The overlay should therefore be read as “HPPC can keep 100 Gb/s near line rate under similar loss regimes when placed at a node,” not as a cycle-accurate speedup over any specific P4 SKU.

For SR-NICs, IRN-like designs [10], [11] implement selective-repeat reliability in end-host NICs, tracking gaps and retransmitting from on-NIC buffers. Published evaluations report that, under WAN-like RTTs and many flows, throughput at 1% loss drops to around 75% of line rate. The cross marker in Fig. 4b places this point at 1% loss and 75% normalized goodput, while HPPC at the same loss level maintains 94–99%

for packet sizes ≥ 1500 B. This gap arises because SR-NICs must store per-flow reorder windows and outstanding packets in scarce on-chip memory and operate with full-path feedback delay. In contrast, HPPC terminates a hop-local repair loop at a memory-rich node.

Overall, the figures position HPPC as complementary to both approaches. P4-based loss masking operates in switches with limited off-chip memory; SR-NICs operate at end hosts with tight on-chip state budgets. HPPC occupies a middle ground: a DDR-rich node that uses a ring-plus-two-level-index organization to provide scalable hop-local recovery while remaining transparent to TCP/RDMA endpoints and requiring no changes to the switching fabric.

C. Summary and Threats to Validity

The MA-X1/XCVU9P prototype with dual DDR4 memories, driven by the Ixia XM100GE4CXP, demonstrates that a split-DDR HPPC cache—a sequential ring in `DDR_data` plus a two-level index in `DDR_idx`—can sustain near line-rate goodput on 100 Gb/s under realistic non-zero loss for packets ≥ 1500 B, with graceful degradation as loss increases. Qualitative comparisons against a chain-based cache and a single-level linear index highlight that HPPC's structure better matches DRAM capacity and bandwidth constraints.

Several limitations remain. All measurements are obtained on a single FPGA platform with DDR4 memory and synthetic CBR traffic, fixed RTT, and independent random losses; real WAN traces with correlated losses, flow churn, and mixed packet sizes may stress the index segment pool and NACK processing differently. The P4 and SR-NIC points in Fig. 4 are drawn from external publications [10]–[12], not from our testbed, so they serve as qualitative reference markers rather than quantitative benchmarks. We also do not implement the single-level linear-index strawman; its drawbacks are inferred from memory footprints and DRAM transaction counts. Despite these caveats, the results indicate that DDR-optimized ring-and-index caching at WAN nodes based on commercial MA-X1/XCVU9P hardware with DDR4 memory is a promising building block for hop-local reliability on high-speed WAN paths.

V. DISCUSSION AND LIMITATIONS

HPPC is primarily aimed at high-bandwidth WAN-like segments where the bandwidth–delay product is large, random losses are non-negligible, and in-path nodes can host ample off-chip memory while switches and end hosts are comparatively constrained. In this regime, the split-DDR design—a sequential DDR_data ring plus a two-level index in DDR_idx—lets the node act as a memory-rich reliability anchor that terminates a hop-local repair loop within one segment RTT and offloads packet caching from NICs and switches. As shown in Sec. IV, under such conditions HPPC keeps a 100 Gb/s link near line rate for packets ≥ 1500 B at non-zero loss while remaining transparent to TCP/RDMA endpoints.

A key design question is where the index should reside. Implementing the full two-level index on chip would minimize lookup latency but is infeasible at 100 Gb/s and the target flow scale given on-chip SRAM capacity and timing margins. HPPC therefore keeps only a compact L1 per-flow directory (plus a small hot-entry cache) in on-chip memory and places the full L2 segment table in DDR_idx on a channel isolated from DDR_data traffic. Measurements in Sec. IV-B indicate that this split yields low read amplification and high burst efficiency: repair traffic incurs a bounded number of DDR_idx accesses, long coalesced reads from DDR_data, and no observable starvation of the forwarding path.

HPPC complements existing dataplane-based and NIC-based reliability mechanisms rather than serving as a universal replacement. In low-RTT local fabrics, SR-NICs and lightweight P4-based loss masking may already suffice [10], [12], and additional hop-local caching at WAN nodes may bring limited benefit. On inter-datacenter and edge–cloud paths with large bandwidth–delay products and non-trivial random loss, however, the results in Sec. IV-B suggest that a DDR-rich in-path node can sustain higher goodput than reported P4 and SR-NIC baselines at similar loss rates by trading ASIC pipeline and on-chip buffer pressure for abundant off-chip memory and a DRAM-friendly ring-and-index organization.

This study has several limitations. All measurements are obtained on a single MA-X1/XCVU9P platform with DDR4 memory and synthetic constant-bit-rate traffic, fixed RTT, and independent random losses; real WAN traces may exhibit correlated losses, flow churn, and mixed packet sizes that stress the segment pool and NACK processing differently. Our experiments cover only a single HPPC segment between two in-path nodes, and do not explore multi-hop deployments, asymmetric routes, or interactions between multiple HPPC segments. Comparisons to P4 switches and SR-NICs rely on published results from heterogeneous hardware and traffic setups [10]–[12], and the overlaid markers in Fig. 4 are intended as qualitative reference points rather than quantitative benchmarks. We also do not implement the single-level linear-index strawman; its drawbacks are inferred analytically from memory footprints and DRAM transaction counts (Sec. IV-A). Exploring ASIC or SmartNIC realizations of HPPC, validating performance under real WAN traces and alternative hardware,

and studying tighter integration with routing, forward error correction, and congestion control (e.g., HPCC/DCQCN-style schemes [6], [8]) are important directions for future work.

ACKNOWLEDGMENT

This work was supported by the project “Research on Wide-area RDMA Transport Acceleration based on SEANet”, an independently deployed oriented project of the Institute of Acoustics, Chinese Academy of Sciences (Grant No. JCQY202407).

REFERENCES

- [1] S. Jain, A. Kumar, S. Mandal *et al.*, “B4: Experience with a globally-deployed software defined WAN,” in *Proc. ACM SIGCOMM*, Hong Kong, China, Aug. 2013, pp. 3–14.
- [2] C.-Y. Hong, S. Kandula, R. Mahajan *et al.*, “Achieving high utilization with software-driven WAN,” in *Proc. ACM SIGCOMM*, Hong Kong, China, Aug. 2013, pp. 15–26.
- [3] T. V. Lakshman and U. Madhow, “The performance of TCP/IP for networks with high bandwidth-delay products and random loss,” *IEEE/ACM Trans. Netw.*, vol. 5, no. 3, pp. 336–350, Jun. 1997.
- [4] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose, “Modeling TCP Reno performance: A simple model and its empirical validation,” *IEEE/ACM Trans. Netw.*, vol. 8, no. 2, pp. 133–145, Apr. 2000.
- [5] C. Guo, M. Zhu, L. Hu *et al.*, “RDMA over commodity Ethernet at scale,” in *Proc. ACM SIGCOMM*, Florianópolis, Brazil, Aug. 2016, pp. 202–215.
- [6] Y. Zhu, H. Eriksson, M. Mercian *et al.*, “Congestion control for large-scale RDMA deployments,” in *Proc. ACM SIGCOMM*, London, U.K., Aug. 2015, pp. 523–536.
- [7] M. Alizadeh, A. Greenberg, D. Maltz *et al.*, “Data center TCP (DCTCP),” in *Proc. ACM SIGCOMM*, New Delhi, India, Sep. 2010, pp. 63–74.
- [8] H. H. Liu, F. Lu, C. Li *et al.*, “HPCC: High precision congestion control,” in *Proc. ACM SIGCOMM*, Beijing, China, Aug. 2019, pp. 44–58.
- [9] Y. Duan, H. Ni, X. Zhu *et al.*, “A survey of reliable multicast error recovery mechanisms,” *Network New Media Technology* (in Chinese), vol. 11, no. 5, pp. 1–7, 40, 2022.
- [10] P. Li, L. Zhu, Z. Zhang *et al.*, “Revisiting network support for RDMA,” in *Proc. ACM SIGCOMM*, Budapest, Hungary, Aug. 2018, pp. 313–326.
- [11] D. Firestone, A. Putnam, S. Mutlu *et al.*, “Azure accelerated networking: SmartNICs in the public cloud,” in *Proc. USENIX NSDI*, Renton, WA, USA, Apr. 2018, pp. 51–66.
- [12] J. Zhang, Y. Cui, B. Li *et al.*, “Masking corruption packet losses in datacenter networks with link-layer retransmission,” in *Proc. USENIX NSDI*, Boston, MA, USA, Apr. 2023, pp. 1–17.
- [13] W. Kong, X. Zhang, and J. Liu, “An efficient method for link-local retransmission in a datacenter network,” *Electronics*, vol. 13, no. 7, pp. 1–16, 2024.
- [14] P. Bosshart, D. Daly, G. Gibb *et al.*, “P4: Programming protocol-independent packet processors,” *ACM SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 3, pp. 87–95, Jul. 2014.
- [15] M. Yu, A. Greenberg, D. Li *et al.*, “Fast in-network key-value stores with NetCache,” in *Proc. ACM SOSP*, Shanghai, China, Oct. 2017, pp. 121–136.
- [16] F. Li, M. Huang, K. Chen *et al.*, “MemC3: Compact and concurrent Memcache with dumber caching and smarter hashing,” in *Proc. USENIX NSDI*, Lombard, IL, USA, Apr. 2013, pp. 371–384.