Semantic-Aware Loss Recovery for Cross-Datacenter Model Training

Xingjian Zhang
Beijing University of Posts and Telecommunications
Beijing, Beijing, China
zhangxj2025@bupt.edu.cn

Yutong Zhao
Beijing University of Posts and Telecommunications
Beijing, Beijing, China
zhaoyutong@bupt.edu.cn

Jiaxue Liu
Beijing University of Posts and Telecommunications
Beijing, Beijing, China
jiaxueliu@bupt.edu.cn

Lizhuang Tan
Shandong Computer Science Center (National Supercomputer Center in Ji'nan)
Jinan, Shandong, China
tanlzh@sdas.org

Shangguang Wang
Beijing University of Posts and Telecommunications
Beijing, Beijing, China
sgwang@bupt.edu.cn

Yiran Zhang
Beijing University of Posts and Telecommunications
Beijing, Beijing, China
yiranzhang@bupt.edu.cn

Abstract

Cross-datacenter distributed model training is increasingly important, but WAN packet loss and long propagation delays make retransmission-based recovery expensive. Existing RDMA FEC schemes are application-unaware and uniformly protect all packets, wasting inter-datacenter bandwidth. In this paper, we present SMART, a semantic-aware loss recovery scheme for cross-datacenter model training. SMART distinguishes data-parallel (DP) gradients from pipeline-parallel (PP) activations using lightweight message-level tags; it selectively protects high-priority DP packets, gives earlier PP traffic stronger FEC, zero-fills only small profiled residual PP loss, and compensates tolerated low-priority DP losses. Simulation experiments with Qwen3-3B-derived workloads show that SMART reduces average and P99 flow completion times compared with retransmission-only recovery and uniform FEC, while preserving convergence close to the no-loss baseline.

CCS Concepts

• **Networks** → **Data center networks**; **Transport protocols**.

Keywords

Remote Direct Memory Access, Cross-Datacenter, Loss recovery

ACM Reference Format:
Xingjian Zhang, Yutong Zhao, Jiaxue Liu, Lizhuang Tan, Shangguang Wang, and Yiran Zhang. 2026. Semantic-Aware Loss Recovery for Cross-Datacenter Model Training. In *The 10th Asia-Pacific Workshop on Networking (APNet 2026), August 06–07, 2026, Singapore, Singapore.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3820441.3820458

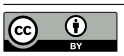
1 Introduction

As AI models continue to scale, training increasingly spans multiple datacenters to pool accelerators and power capacity [11]. However, once training traffic leaves the datacenter fabric, the communication environment changes fundamentally: cross-datacenter links have much longer propagation delay and non-negligible packet loss [6]. In this regime, conventional RDMA loss recovery based on retransmission becomes increasingly expensive, because even a small number of lost packets can trigger WAN-scale recovery delay and inflate both average and tail flow completion time.

Forward Error Correction (FEC) is a natural alternative because it can recover losses without waiting for a full retransmission round. Existing schemes for lossy cross-datacenter links, however, are still largely packet-centric and application-unaware [15]. They assign redundancy using a fixed coding configuration and implicitly assume that all packets deserve the same protection. This assumption is poorly matched to cross-datacenter training, where repair traffic competes for the same scarce WAN bandwidth as useful training traffic, and where the cost of over-protecting insensitive packets can be as harmful as under-protecting critical ones.

The key issue is that cross-datacenter training traffic is not homogeneous. In practice, the two dominant communication patterns are DP and PP, which exhibit different loss sensitivities. For DP, the importance of gradients is highly uneven: losing large-magnitude gradients is much more harmful than losing small ones, and prior work shows that small gradients can tolerate substantially higher loss rates than large gradients [12]. For PP, activation traffic is generally more loss-sensitive because it directly affects subsequent computation, but even within PP the required protection is depth-dependent: perturbations introduced at earlier pipeline boundaries are more damaging than those introduced later. These observations suggest that uniform packet-level protection is fundamentally inadequate for cross-datacenter training. Existing application-aware mechanisms, such as packet trimming, exploit the robustness of ML training by compressing gradients under congestion, primarily to mitigate intra-datacenter congestion rather than to recover long-haul RDMA losses [2].

Motivated by the above observations, this paper presents SMART, a 𝕊eℳantic-𝔸ware loss ℝecovery scheme for cross-datacenter model 𝕋raining. SMART uses lightweight message-level tags to distinguish DP gradients from PP activations. For DP, SMART identifies high-priority gradient packets and protects only them with



APNet 2026, Singapore, Singapore

ACM ISBN 979-8-4007-2664-4/26/08
https://doi.org/10.1145/3820441.3820458

FEC, while low-priority packets may be delivered unrecovered and treated through a tolerance-aware strategy. For PP, SMART protects all activation packets but allocates greater redundancy to earlier-layer traffic and less to later-layer traffic, and zero-fills residual loss only within a small configured bound; otherwise it triggers retransmission. To prevent unrecovered low-priority DP losses from biasing subsequent optimization, SMART further introduces an error-compensation mechanism that reinjects the corresponding residuals into later transmissions.

We evaluate SMART in SimAI [13] using cross-datacenter PP and DP traffic derived from Qwen3-3B workloads [14]. In cross-DC training scenarios, SMART reduces the P99 flow completion time by up to 89.9% and 78.8% compared to Selective Repeat and LoWAR [15] in PP communication, and by up to 91.7% and 73.0% in DP communication, respectively. Besides, SMART's error-compensation mechanism preserves convergence close to the no-loss baseline even when low-priority gradient losses are tolerated.

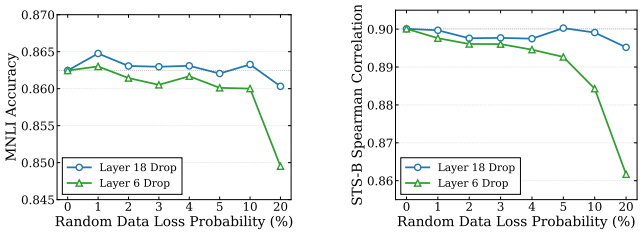
2 Background and Motivation

2.1 Background

As AI models continue to scale in parameter count and training demand, a single datacenter often cannot provide sufficient accelerators (or power/energy capacity) for timely training, motivating training jobs to span multiple datacenters. In practice, large-scale training combines multiple forms of parallelism, including Data Parallelism (DP), Pipeline Parallelism (PP), Tensor/Sequence Parallelism (TP/SP), and Expert Parallelism (EP). However, TP/SP/EP are typically characterized by large *and high-frequency* communication (often per-layer collectives), which is particularly costly over high-latency cross-DC links; therefore, these intra-layer parallelism modes are usually confined within each datacenter. As a result, when parallelism is extended across datacenters, it is most commonly realized by PP and DP.

PP partitions the model into S pipeline stages and streams *activations* forward (and corresponding activation gradients backward) between adjacent stages. If PP is extended across datacenters by placing different stages in different DCs while co-locating the same stage across replicas within each DC, then gradient synchronization can be confined within each DC and only inter-layer activations traverse cross-DC links. The resulting cross-DC traffic per step is approximately $V_{\mathrm{PP}} \approx K \cdot B_\mu \cdot A \cdot b_a$, where K accounts for forward/backward pipeline transfers, B_μ is the micro-batch size, A is the activation size per micro-batch at the stage boundary, and b_a is the activation precision in bytes; in practice, this stage-boundary activation traffic can be substantially smaller than exchanging a full set of gradients across DCs for large models.

DP accelerates training by replicating the full model on multiple workers and splitting the input mini-batch across replicas; after each iteration, replicas must synchronize *gradients* (or equivalently, model updates). Thus, the dominant DP payload is the full gradient tensor, whose size is proportional to the number of model parameters. For a model with P parameters and gradient precision of b bytes, the per-step synchronization volume is on the order of $V_{\mathrm{DP}} \approx P \cdot b$ per replica (up to constant factors depending on the collective). For example, GPT-3 (175B)[1] transfers at least ~ 175 GB of gradients per iteration under synchronous DP (batch size 3096), making gradient exchange the main contributor to cross-DC traffic.



(a) MNLI (b) STS-B

Figure 1: Performance results under random data loss probability at different pipeline partitions.

To reduce the cross-DC traffic introduced by gradient synchronization, NVIDIA advocates *hierarchical all-reduce (HAR)* [10] that aggregates gradients within each data center before global synchronization, reducing the WAN-facing communication to roughly $V_{\mathrm{HAR}} \approx V_{\mathrm{DP}}/N$, where N is the number of GPUs participating within each intra-DC group. Google proposes low-communication training DiLoCo[4], which decreases synchronization frequency by running many local updates and only occasionally averaging across sites, reporting up to 500× less communication than fully synchronous training while matching quality in their evaluated setting. These communication-aware DP variants also make DP a reasonable choice for cross-datacenter training by substantially reducing the WAN traffic required to maintain replica consistency.

2.2 Characteristics of DP and PP traffic

Cross-datacenter training traffic traverses long-haul WAN links that are fundamentally different from intra-DC fabrics: they typically exhibit higher propagation delay and more frequent packet loss events, and any retransmission-based recovery can incur large delay penalties. In this setting, the communication layer must not only pursue low average completion time, but also control tail latency and avoid stalling training progress. This motivates us to examine the tolerance characteristics of training traffic—i.e., when and what kind of loss can be safely absorbed by the learning process—so that loss recovery can be made selective rather than uniformly reliable for all data.

In DP, gradient transmission exhibits a certain degree of loss tolerance, and the extent of tolerance is closely related to gradient magnitude. The underlying reason is that SGD-based training is approximate and inherently stochastic, so occasional missing gradient pieces can be treated as transient perturbations (e.g., implicitly zero-filled) while the optimization continues, and the resulting deviation can be amortized and corrected by subsequent update steps.

More specifically, this loss tolerance decreases as the gradient magnitude increases. Losing small-magnitude gradients is typically much less harmful, whereas losing large-magnitude gradients more directly discards critical update information and can significantly distort the effective update. MLT's experiments explicitly quantify this gap: to maintain similar convergence behavior, training can

tolerate more than 20% loss among small gradients, but only about 0.4% loss among large gradients [12]. This pronounced asymmetry suggests that DP traffic should not be protected uniformly, and that substantially stronger reliability protection should be allocated to large-magnitude gradients.

PP activation traffic is more loss-sensitive than DP traffic, but its sensitivity varies with pipeline depth. PP primarily communicates *activations* between adjacent pipeline stages, whose loss directly perturbs the intermediate representations used by subsequent computation. To quantify how this sensitivity varies with pipeline position, we conduct a preliminary packet-drop experiment on BERT-Large (24 Transformer layers, hidden size 1024)[3] using two downstream tasks, MNLI and STS-B. We emulate a two-stage pipeline-parallel setting and inject packet losses at two candidate partition points: an earlier split at Layer 6 and a later split at Layer 18, corresponding to cross-stage activation transfers after the first quarter and the first three-quarters of the model, respectively.

For each partition point, activation loss is applied at packet granularity. Each packet carries 2,048 FP16 activation elements, matching a 4 KB payload, so that the emulation is aligned with practical RDMA/InfiniBand MTU-scale transmission. We evaluate random packet-drop probabilities of 0%, 1%, 2%, 3%, 4%, 5%, 10%, and 20%, where 0% serves as the no-loss baseline. Model quality is reported by validation accuracy on MNLI and Spearman correlation on STS-B. As shown in Figure 1, the results indicate that PP activations have depth-dependent but nonzero loss tolerance. Compared with the later partition, the earlier partition tolerates less residual packet loss: its model quality degrades earlier and more noticeably as the drop probability increases. The later partition can tolerate a relatively larger residual-loss budget, but this tolerance is still limited and must be profiled conservatively. This suggests that PP traffic should not be treated with uniform reliability, and that earlier pipeline boundaries deserve stronger protection than later ones.

3 Design

Fig. 2 presents the overview of SMART, a semantics-aware FEC scheme for cross-datacenter model training. Motivated by the traffic characteristics observed in Sec. 2.2, SMART does not apply a uniform protection policy to all packets. Instead, it first identifies the traffic type and classifies packet priority according to traffic properties (Sec. 3.1). The priority information and traffic type information then determine the redundancy level used for FEC encoding, enabling differentiated protection for packets with different training importance.For DP, priority reflects gradient magnitude: SMART protects high-priority gradient packets with FEC, allows bounded low-priority losses(Sec. 3.2). Finally, to prevent errors introduced by unrecovered loss from accumulating over iterations, SMART incorporates an error-compensation mechanism for subsequent transmissions (Sec. 3.3).

3.1 Type Identification and Priority Classification

SMART relies on *message-level semantic tags*. Each message carries the traffic type and minimal context, such as tensor/operator identity or PP stage/layer index; the sender-side shim consumes this metadata before packetization, so applications need not manage packet layout or repair tags.

DP gradients priority classification. For DP, the payload of each gradient packet is a sequence of gradient values from the source tensor.

To encode magnitude information at the packet level, SMART adopts a strategy similar to gradient sparsification[7]. For each gradient packet p, we compute the mean magnitude of all gradient values carried in the packet:

$$\mu(p) = \frac{1}{|p|} \sum_{g \in p} |g|. \tag{1}$$

SMART then compares $\mu(p)$ against a threshold τ to determine whether the packet is important. If $\mu(p) \geq \tau$, the packet is classified as High-Priority (HP); otherwise, it is classified as Low-Priority (LP).

Rather than deriving τ from the full tensor, SMART estimates it from a small uniform sample of gradients drawn from the source tensor, so as to minimize overhead. Specifically, SMART computes τ as the cutoff corresponding to the top-K fraction within the sampled gradient magnitudes. By default, we set $K = 1\%$, meaning that τ approximates the magnitude of the largest 1% gradients.

PP activations priority classification. For PP activation packets, priority is determined by the layer order.Let the model have L layers, and let an activation packet p carry the layer index $\ell \in \{1, \ldots, L\}$.Earlier layers are more important, while later layers are less important.We use the normalized depth ratio $\rho(p) = \ell/L$ and a threshold k (as a fraction) to classify:

$$\texttt{prio}(p) = \begin{cases} \texttt{HP}, & \rho(p) = \ell/L \leq k, \\ \texttt{LP}, & \text{otherwise}, \end{cases} \tag{2}$$

i.e., activation packets from the first $\lceil kL \rceil$ layers are marked as high priority. Equivalently, when cross-datacenter PP traffic traverses only one layer boundary, SMART can be viewed as assigning different redundancy levels according to the layer index of that boundary.

3.2 Error Correction Encode and Decode

Error correction code. At the sender, SMART uses XOR codes as the basis for FEC coding. To tolerate burst losses, we introduce an interleaving depth c to generate multiple independent repair packets within a single coding block. Specifically, the system assigns each data packet to one of c parallel XOR sequences by computing the coding unit index j from the packet index i via modulo operation, $j = i \pmod{c}$. Each sequence independently maintains its own XOR sum and generates a dedicated repair packet once the coding block or message ends. This interleaved structure ensures that continuous burst losses of length up to c are dispersed across separate sequences, allowing each repair packet to recover a unique lost packet.

Based on the priority information, SMART applies differentiated redundancy to improve loss recovery efficiency. *For PP activations*, both HP and LP packets are protected by FEC to preserve the correctness of pipeline execution: HP (early-layer) activations are assigned higher redundancy and a stricter residual-loss bound, while LP (late-layer) activations use lower redundancy but still trigger

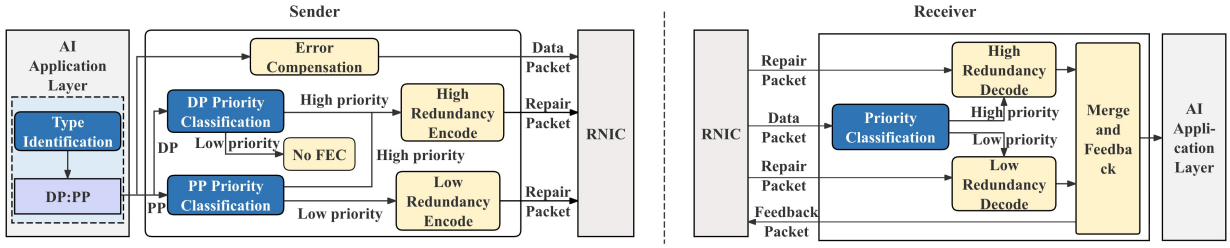


Figure 2: SMART overview

retransmission if the small configured tolerance is exceeded. *For DP gradients*, SMART leverages magnitude-dependent loss tolerance and adopts a K-sparsification-inspired *selective FEC* policy: only HP gradient packets enter the FEC encoder, whereas LP gradient packets carry *zero redundancy* and may be left unrecovered under the loss-tolerance bound, with their impact compensated by the error-feedback mechanism in Sec. 3.3.

To ensure precise recovery, we embed a Base Sequence Number and a bitmap into the repair packets. This bitmap explicitly indicates which data packets participated in the current FEC encoding.

Error correction decode. At the receiver, SMART first distinguishes *DP gradients* from *PP activations* using the message-level semantic tags and then applies different recovery strategies.

For DP gradients, SMART applies selective FEC decoding and prepares error feedback. In detail, upon receiving a DP repair packet, the receiver parses the embedded *Base Sequence Number* and *bitmap* to locate the protected data-packet indices and performs FEC decoding to recover missing high-priority DP packets. After decoding, the receiver reconstructs the DP stream in three cases: packets received intact are delivered directly, missing HP packets are recovered via FEC, and missing LP packets are zero-filled. If decoding is insufficient to recover the protected HP packets, the receiver triggers retransmissions to preserve the correctness of important gradient updates. For LP packets, the receiver does not request retransmission. Instead, it records, in a single bitmap, the indices of LP packets that remain unrecovered and are delivered as zeros, and feeds this bitmap back to the sender together with the corresponding coding-block identifier. The sender then uses this feedback to perform error compensation in Sec. 3.3.

For PP activations, SMART performs FEC decoding under a priority-aware tolerance mechanism. Before deployment, SMART profiles a small residual-loss bound for each PP priority class: HP uses the stricter bound, while LP may use a slightly looser but still small bound. At runtime, the receiver decodes PP repair packets using the Base Sequence Number and bitmap. If residual unrecovered packets exceed the class bound, SMART retransmits to preserve PP correctness; otherwise, it zero-fills the remaining packets and delivers the activation stream.

3.3 Error Compensation

The previous two components of SMART improve recovery efficiency by reducing the amount of protected data. In particular, selective protection concentrates FEC bandwidth on the most important DP gradient packets, while low-priority packets remain unprotected. Although this design significantly improves bandwidth efficiency, it may introduce residual errors when low-priority packets are lost.

Under severe loss conditions, these unrecovered packets can accumulate into noticeable distortion in the transmitted gradients, which may degrade model performance or even hinder convergence. To mitigate this issue, SMART introduces an error compensation mechanism.

The key idea is to track the residual error caused by unrecovered packet loss and reinject this error into subsequent transmissions. By maintaining and feeding back the accumulated residual, SMART compensates for the missing gradient information over time, preserving training stability while retaining the bandwidth efficiency of selective protection.

Receiver-side feedback. In SMART, the error-compensation mechanism is triggered only by loss-induced error. Specifically, after DP decoding, high-priority packets that are successfully recovered through FEC are directly delivered and do not generate compensation feedback. In contrast, if a low-priority packet is not recovered and is delivered as zero, it introduces residual loss into the communicated gradient. Therefore, the receiver only records the indices of such unrecovered packets in a feedback bitmap B_c and sends it back to the sender. This design keeps the receiver logic simple and restricts compensation to the only error source considered in the current design, namely tolerated packet loss.

Sender-side error compensation. The sender maintains an error memory vector e_t with the same logical shape as the communicated tensor. The role of e_t is to store the residual introduced by tolerated packet loss that has not yet been compensated.Before transmission at iteration t, it forms the corrected payload

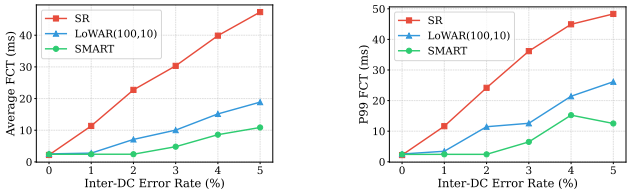
$$p_t = x_t + e_t. \tag{3}$$

After receiving the bitmap B_c, the sender treats each marked index as a loss-induced missing entry. Since these packets were delivered as zero at the receiver, the corresponding residual error equals the transmitted value itself. Therefore, the error memory is updated as

$$e_{t+1}[i] = p_t[i], \quad i \in B_c, \tag{4}$$

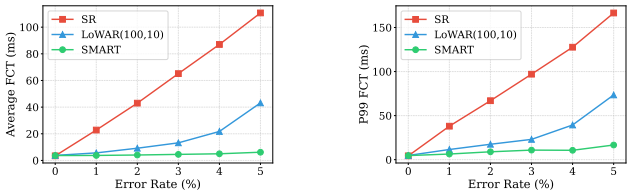
and

$$e_{t+1}[i] = 0, \quad i \notin B_c. \tag{5}$$



(a) Average FCT (b) P99 FCT

Figure 3: Average and P99 FCT of PP traffic under different inter-DC error rates.



(a) Average FCT (b) P99 FCT

Figure 4: Average and P99 FCT of DP traffic under different inter-DC error rates.

The updated error memory is then injected into the next iteration:

$$p_{t+1} = x_{t+1} + e_{t+1}. \tag{6}$$

In this way, SMART compensates only for tolerated packet loss. The receiver feeds back only a lightweight bitmap of unrecovered positions, and the sender reinjects the corresponding residual loss into the next transmission, preventing tolerated loss from biasing subsequent gradient updates.

3.4 Implementation Discussion

SMART is intended to run as a host-side shim in framework-level communication middleware, e.g., a PyTorch hook or NCCL network plugin, not as an application-level packet API. The middleware marks each message with its traffic type and minimal context; SMART then performs chunking, priority classification, FEC, and feedback below that boundary, so applications need not know packetization. In RDMA, data and parity can use ordinary data-QP work requests, while metadata and feedback bitmaps use a reliable control QP or out-of-band channel; no NIC firmware change or new packet format is required. Only SMART-enabled communicators use this semantic recovery path, while legacy traffic uses the existing stack.

4 Evaluation

4.1 FCT Performance Evaluation

We evaluate SMART in SimAI[13] using cross-datacenter training traffic derived from Qwen3-3B[14] workloads. The simulated topology consists of two interconnected data centers, each organized as a $k = 4$ Fat-tree. Each data center contains 16 servers, and the two sites are connected by a 400 Gbps inter-DC link with 400 μs propagation delay. All intra-DC links run at 100 Gbps with 1 μs delay. Unless otherwise specified, packet losses are injected only on the inter-DC link, while all intra-DC links are loss-free.

Workloads and traffic patterns. We evaluate both PP and DP communication patterns derived from Qwen3-3B training. For PP, we use a two-stage pipeline deployment across the two data centers, where only the activation exchange of the first half of the model across the stage boundary traverses the inter-DC link. This produces sequential point-to-point cross-DC flows. For DP, we use a 32-way data-parallel configuration and model gradient synchronization as cross-DC collective communication, including AllGather and ReduceScatter phases.

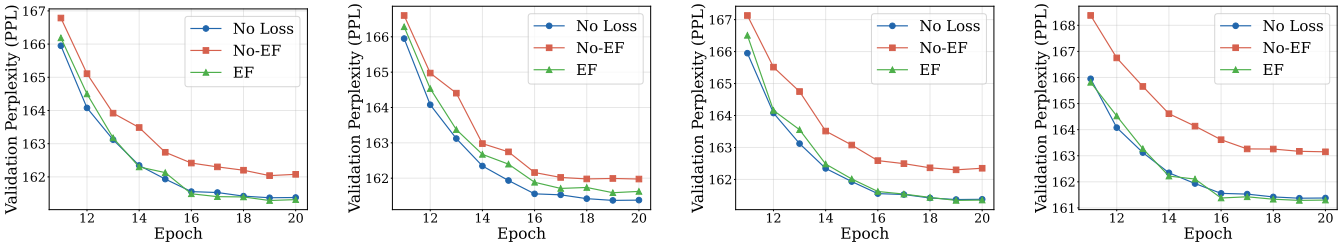
Baselines and configuration. We compare SMART with two representative baselines: Selective Repeat (SR)[9] and LoWAR[15]. SR represents a retransmission-based loss recovery mechanism without proactive redundancy, while LoWAR represents a uniform-FEC baseline that applies the same redundancy configuration to all protected packets; specifically, (100,10) means that every 100 data packets are protected by 10 parity packets. For a fair comparison, all schemes use the same packetization and interleaved XOR coding basis.

We explicitly distinguish the reliability objectives of these schemes. SR and LoWAR target packet-level reliability: lost packets are expected to be recovered before delivery, through retransmission in SR and through uniform FEC plus retransmission when FEC recovery is insufficient in LoWAR. SMART instead targets training-level reliability. For DP traffic, SMART enforces immediate recovery only for high-priority gradient packets. Low-priority gradient packets are not retransmitted when they remain unrecovered; instead, they are zero-filled, recorded through a feedback bitmap, and compensated at the sender in later iterations. For PP traffic, SMART follows a stricter policy than DP: activation packets are protected by FEC, and retransmission is triggered if the residual loss exceeds the profiled tolerance; only small residual losses within the configured bound are tolerated.

For DP traffic, it protects only the top 1% largest gradient packets, while the remaining 99% of packets follow the tolerance-and-compensation path. For PP traffic, we evaluate SMART under the above two-stage setting with the cross-DC boundary placed in the first half of the model, where the corresponding packets are protected with high redundancy. Based on offline experiments, the residual-loss tolerance is set to 1%.

Metrics. We use FCT as the primary metric and report both average FCT and P99 FCT. Average FCT reflects overall communication efficiency, while P99 FCT captures tail behavior that is particularly important for synchronous training. In cross-datacenter training, even a small number of delayed flows can directly stall iteration progress, making tail reduction as important as improving the average.

Overall effectiveness. Fig. 3 and Fig. 4 show the average and P99 FCT under PP and DP traffic, respectively. Across both traffic patterns and all tested error rates, SMART consistently achieves the lowest average and tail latency. Compared with SR, SMART avoids the large recovery delay caused by retransmission over long-haul WAN links. Compared with LoWAR(100,10), SMART still delivers lower FCT, showing that simply adding uniform redundancy is less effective than allocating protection according to training semantics.



(a) 0.5% loss (b) 1% loss (c) 2% loss (d) 5% loss

Figure 5: Zoomed validation perplexity (PPL) curves of LSTM training under different loss rates.

Overall, the results show that SMART is an effective loss-recovery mechanism for cross-datacenter training. Across both PP and DP traffic patterns and all tested inter-DC error rates, SMART consistently achieves the lowest average and P99 FCT among all schemes. The improvement over SR mainly comes from avoiding WAN-scale retransmission delay after packet loss, while the advantage over LoWAR(100,10) indicates that uniform redundancy allocation remains inefficient under heterogeneous training traffic. By allocating protection according to the loss sensitivity of different packets, SMART makes better use of scarce inter-DC bandwidth and improves both average communication efficiency and tail performance.

4.2 Convergence Study

Experimental setup. To verify that tolerating low-priority gradient loss with error feedback does not harm convergence, we evaluate SMART on WikiText-2[8] using a two-layer LSTM[5] language model with 512-dimensional embeddings and hidden states, dropout 0.3, and tied input/output embeddings. We use the GPT-2 tokenizer, a block size of 256, batch size 32, Adam with initial learning rate 10^{-3}, cosine annealing to 10^{-6} over 20 epochs, and gradient clipping with maximum norm 1.0.

We compare three settings: *Baseline*, *No-EF*, and *EF*. *Baseline* performs standard training without injected loss. *No-EF* injects consecutive low-priority gradient loss without compensation. *EF* denotes the variant with *error feedback*, in which the top 1% gradients are protected and the residual error caused by missing low-priority gradients is accumulated and fed back into later iterations. We evaluate loss rate of 0.5%, 1%, 2%, and 5%. We use validation PPL as the evaluation metric because PPL is the standard measure for language modeling and directly reflects the model's predictive quality on the validation set. A lower PPL indicates better next-token prediction performance and provides a sensitive indicator of whether tolerated gradient loss affects optimization and final convergence behavior.

Convergence analysis. Figure 5 shows the zoomed validation PPL curves in the late training stage. Across all loss rates, *No-EF* consistently converges to a worse plateau than the baseline, while *EF* remains close to the baseline, indicating that error feedback effectively mitigates the optimization bias caused by tolerated low-priority loss.

At 1% loss, *No-EF* reaches a best validation PPL of 161.98, while *EF* improves it to 161.59, close to the baseline value of 161.37. At 2% loss, *No-EF* degrades to 162.30, whereas *EF* achieves 161.34, nearly matching the baseline. At 5% loss, the gap becomes larger: *No-EF* reaches 163.15, while *EF* achieves 161.29, reducing the gap to the baseline by about 1.86 PPL.

These results show that the main issue is not the tolerated low-priority loss itself, but the residual error left uncompensated. With error feedback, SMART preserves convergence behavior close to the baseline while allowing bounded low-priority gradient loss.

4.3 Evaluation Scope and Limitations

Evaluation scope and scalability. Our evaluation is based on simulation rather than a production deployment. The FCT experiments use DP and PP traffic derived from Qwen3-3B in SimAI, and the convergence study uses a two-layer LSTM on WikiText-2. Thus, the results show how SMART behaves in the tested settings, but they do not cover every model, cluster size, or pipeline split. We expect SMART to remain useful at larger scale for three reasons. First, larger models produce more DP gradient traffic across datacenters, so avoiding unnecessary protection for low-priority gradients can save more bandwidth. Second, deeper pipelines create more PP stage boundaries, so giving stronger protection to earlier boundaries and lighter protection to later ones becomes more useful. Third, with more nodes, a few slow cross-datacenter transfers can delay an iteration, so reducing recovery delay is still important. In practice, the priority thresholds and FEC redundancy levels should be configured for each model, pipeline split, and network condition.

Loss rates and loss type. We use 0%–5% inter-DC loss in the FCT experiments and 0.5%–5% low-priority gradient loss in the convergence study to cover the no-loss case, low-loss cases, and short degraded periods. This range should be viewed as a preliminary evaluation rather than a claim that production inter-region links usually have several-percent steady loss. In well-managed production networks, inter-region links are expected to be close to loss-free most of the time, but transient packet drops can still occur and may stall synchronous training. SMART is expected to provide meaningful benefits when the loss rate is high enough to make WAN retransmission or uniform FEC introduce visible recovery cost, but still within the configured FEC and loss-tolerance range. When loss is near zero, all schemes have little recovery work, so SMART's benefit becomes small; when loss is far beyond the configured range, recovery alone is unlikely to be sufficient. In our evaluation, the injected losses are generic packet-loss events on the inter-DC link after packetization, and we do not assume a specific root cause.

Congestion-related loss. The current evaluation injects generic packet-loss events on the inter-DC link and does not distinguish whether such losses would be caused by packet corruption or congestion-induced drops. If loss is caused by congestion, extra FEC parity may further load the WAN. SMART reduces this risk by avoiding uniform redundancy, but in deployment its FEC level should be coordinated with rate or congestion control and reduced under persistent congestion. Thus, SMART is a semantic recovery layer for transient inter-DC loss, not a replacement for WAN congestion control.

5 Conclusion

SMART makes cross-datacenter loss recovery training-aware by analyzing the loss-tolerance characteristics of different parallel traffic in cross-datacenter training and introducing semantics-aware differentiated protection and compensation for DP and PP traffic, thereby reducing both average and tail FCT while preserving model training convergence close to the no-loss baseline.

Acknowledgments

This work is supported in part by the National Key Research and Development Program of China (No. 2024YFB2907000), and by the National Natural Science Foundation of China (NSFC) under Grant No. 62302055, and in part by the Fundamental Research Funds for the Central Universities, and in part by the Shandong Provincial Natural Science Foundation under Grant No.ZR2024LZH006. Corresponding author: Yiran Zhang.

References

[1] Tom B. Brown, Benjamin Mann, Nick Ryder, et al. 2020. Language Models are Few-Shot Learners. arXiv:2005.14165 [cs.CL] https://arxiv.org/abs/2005.14165

[2] Xiaoqi Chen, Shay Vargaftik, and Ran Ben Basat. 2024. When ML Training Cuts Through Congestion: Just-in-Time Gradient Compression via Packet Trimming. In *Proceedings of the 23rd ACM Workshop on Hot Topics in Networks (HotNets)*. 177–185. doi:10.1145/3696348.3696880

[3] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. arXiv:1810.04805 [cs.CL] https://arxiv.org/abs/1810.04805

[4] Arthur Douillard, Qixuan Feng, Andrei A. Rusu, Rachita Chhaparia, Yani Donchev, Adhiguna Kuncoro, Marc'Aurelio Ranzato, Arthur Szlam, and Jiajun Shen. 2024. DiLoCo: Distributed Low-Communication Training of Language Models. arXiv:2311.08105 [cs.LG] https://arxiv.org/abs/2311.08105

[5] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Comput.* 9, 8 (Nov. 1997), 1735–1780. doi:10.1162/neco.1997.9.8.1735

[6] M. Khalilov, S. Shen, M. Chrapek, et al. 2025. SDR-RDMA: Software-defined reliability architecture for planetary scale RDMA communication. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC)*. 1223–1239.

[7] Yujun Lin, Song Han, Huizi Mao, Yu Wang, and William J. Dally. 2020. Deep Gradient Compression: Reducing the Communication Bandwidth for Distributed Training. arXiv:1712.01887 [cs.CV] https://arxiv.org/abs/1712.01887

[8] Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2016. Pointer Sentinel Mixture Models. arXiv:1609.07843 [cs.CL] https://arxiv.org/abs/1609.07843

[9] Radhika Mittal, Alexander Shpiner, Aurojit Panda, Eitan Zahavi, Arvind Krishnamurthy, Sylvia Ratnasamy, and Scott Shenker. 2018. Revisiting network support for RDMA. In *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication* (Budapest, Hungary) *(SIGCOMM '18)*. Association for Computing Machinery, New York, NY, USA, 313–326. doi:10.1145/3230543.3230557

[10] NVIDIA. 2024. *Turbocharge LLM Training Across Long-Haul Data Center Networks with NVIDIA NeMo Framework*. NVIDIA Developer Blog. https://developer.nvidia.com/blog/turbocharge-llm-training-across-long-haul-data-center-networks-with-nvidia-nemo-framework/ Accessed: 2026-03-11.

[11] F. Strati, P. Elvinger, T. Kerimoglu, et al. 2024. ML training with cloud GPU shortages: Is cross-region the answer?. In *Proceedings of the 4th Workshop on Machine Learning and Systems (EuroMLSys)*. 107–116.

[12] Hao Wang, Han Tian, Jingrong Chen, et al. 2024. Towards domain-specific network transport for distributed DNN training. In *Proceedings of the 21st USENIX Symposium on Networked Systems Design and Implementation (NSDI)*. USENIX Association, 1421–1443.

[13] Xizheng Wang, Qingxu Li, Yichi Xu, Gang Lu, Dan Li, Li Chen, Heyang Zhou, Linkang Zheng, Sen Zhang, Yikai Zhu, Yang Liu, Pengcheng Zhang, Kun Qian, Kunling He, Jiaqi Gao, Ennan Zhai, Dennis Cai, and Binzhang Fu. 2025. SimAI: Unifying Architecture Design and Performance Tuning for Large-Scale Large Language Model Training with Scalability and Precision. In *22nd USENIX Symposium on Networked Systems Design and Implementation (NSDI 25)*. USENIX Association, Philadelphia, PA, 541–558. https://www.usenix.org/conference/nsdi25/presentation/wang-xizheng-simai

[14] An Yang, Anfeng Li, Baosong Yang, et al. 2025. Qwen3 Technical Report. arXiv:2505.09388 [cs.CL] https://arxiv.org/abs/2505.09388

[15] Tianyu Zuo, Tao Sun, Shuyong Zhu, et al. 2024. LoWAR: Enhancing RDMA over lossy WANs with transparent error correction. In *Proceedings of the IEEE/ACM 32nd International Symposium on Quality of Service (IWQoS)*. Guangzhou, China, 1–10. doi:10.1109/IWQOS61813.2024.10682853