UniFL: Enabling Loss-tolerant Transmission in Federated Learning

Zixuan Chen
zxchen20@fudan.edu.cn
Fudan University

Yifan Ruan
yfruan20@fudan.edu.cn
Fudan University

Sen Liu
senliu@fudan.edu.cn
Fudan University

Yang Xu*
xuy@fudan.edu.cn
Fudan University
Peng Cheng Laboratory

ABSTRACT

As Distributed Deep Learning (DDL) gains prominence, network constraints have emerged as a critical bottleneck impacting DDL performance. While state-of-the-art loss-tolerant (LT) transmission protocols enhance DDL efficiency, their application in federated learning (FL) environments is hindered by several challenges: (1) LT protocols necessitate client-side modifications, impractical in FL settings; (2) maintaining LT protocol transparency to senders compromises congestion control integrity; (3) LT protocols disrupt stream cipher, which is widely utilized in FL. To address these hurdles, this paper introduces UniFL, an innovative LT protocol tailored for FL applications. UniFL seamlessly integrates with FL architectures by preserving congestion control via a specialized speed limiter and adopting an advanced encryption technique that withstands packet loss, ensuring data integrity. UniFL is implemented within the NS3 for simulation evaluation. UniFL's efficacy is evaluated across diverse models and datasets, demonstrating substantial performance enhancements in FL operations. In detail, UniFL can bring up to 40x speedup than the original FL with widely used congestion control algorithms and achieves throughput close to the state-of-the-art LT while being transparent to the workers.

CCS CONCEPTS

• **Networks** → **Transport protocols**; • **Computer systems organization** → *Cloud computing*.

KEYWORDS

Deep Learning, Federated Learning, Distributed Training

ACM Reference Format:
Zixuan Chen, Yifan Ruan, Sen Liu, and Yang Xu. 2024. UniFL: Enabling Loss-tolerant Transmission in Federated Learning. In *The 8th Asia-Pacific Workshop on Networking (APNet 2024), August 3–4, 2024, Sydney, Australia.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3663408.3663432

*Corresponding author.

APNet 2024, August 3–4, 2024, Sydney, Australia

ACM ISBN 979-8-4007-1758-1/24/08…$15.00
https://doi.org/10.1145/3663408.3663432

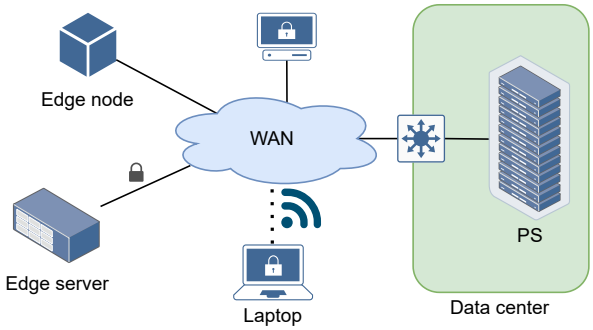

Figure 1: An overview of Federated Learning.

1 INTRODUCTION

In the realm of artificial intelligence (AI), deep learning (DL) has experienced unprecedented growth, establishing itself as a cornerstone technology across a myriad of applications ranging from image recognition [1] to natural language processing (NLP) [2]. This surge in adoption has necessitated advancements in distributed deep learning (DDL), a paradigm that leverages computational resources spread across multiple locations to tackle complex models and datasets. By harnessing the collective power of these distributed systems, DL endeavors can achieve scalability and efficiency, addressing the intensive computational demands of training sophisticated neural networks.

Federated Learning (FL) [3], a pivotal branch of DDL, revolutionizes data privacy and computational efficiency by training models on numerous devices while keeping data localized (see Figure 1). FL usually consists of one Parameter Server (PS) [4] and a large number of workers distributed throughout the Wide Area Network (WAN). Each worker only trains its unique data and synchronizes the model gradient with PS. This framework is characterized by three core features: autonomy, ensuring data and training remain with its owner; heterogeneity, accommodating diverse data, network, and computational capabilities; and transmission security, protecting data as it moves across networks. The significance of FL is evident in sectors such as healthcare, where it enables the development of predictive models without sharing sensitive patient data, and in finance, for fraud detection without compromising client confidentiality. By fostering collaboration without sacrificing privacy, FL marks a significant advancement in how we approach DL at scale.

A fundamental challenge confronting FL is the network variability inherent in distributed systems, which can severely impact

the efficiency of model training. This variability introduces latency, packet loss, and intermittent connectivity issues, especially when FL is deployed across geographically dispersed devices with fluctuating WAN conditions. Moreover, the presence of network heterogeneity may lead to some traffic experiencing long-term severe tail phenomena. Such network challenges can slow down the convergence of the learning process of models. In light of these challenges, the introduction of Loss-tolerant Transmission (LT) stands out as a pivotal solution for DDL across WAN. LT is designed to mitigate the adverse effects of network issues by allowing dynamic packet loss during data transmission, ensuring that the distributed training can maintain high levels of efficiency even in suboptimal network conditions. LT can also significantly reduce the tail latency of many-to-one traffic, especially in scenarios where FL operates over WAN.

However, due to the aforementioned features of FL, LT cannot be directly applied to FL scenarios. Firstly, current state-of-the-art (SOTA) LT protocols require modifications to both the sender and receiver, which is difficult to achieve in FL with its strong autonomy. To address this issue, our method avoids extensive changes to the sender by masking packet loss signals at the receiver. This leads to the second challenge: once packet loss signals are masked, all congestion control on the sender's side becomes ineffective. To tackle this, we adjust the receiver's response signals, achieving **unified** congestion control for all streams within one round of incast communication in **FL** (UniFL). Additionally, the LT mechanism can compromise encrypted transmission, especially traditional stream encryption methods. UniFL mitigates the impact of LT on link encryption through a specially designed control method that does not require modifications to the sender's program.

In summary, our study makes the following contributions.

(1) We provide a detailed discussion on the advantages and challenges of applying SOTA LT transmission in FL scenarios. Our analysis reveals that while network heterogeneity makes LT well-suited for FL, the principle of autonomy complicates its application.
(2) To address the requirement for autonomy in FL, we design an operation that masks packet loss signals at the receiver end. This approach allows the sender to maintain its protocol stack unchanged, preserving autonomy.
(3) Through extensive evaluation, we demonstrate that sender-unaware LT leads to ineffective congestion control. We develop a receiver-side rate limiter that unifies all congestion control protocols, addressing this issue.
(4) We thoroughly discuss how UniFL ensures the reliability and recoverability of streaming encryption and redesign a data encryption method to mitigate the impact of LT on encrypted transmission.
(5) We validate the performance of UniFL with large-scale simulation experiments. Our evaluations show that, compared to traditional FL, we can achieve a performance increase of approximately 100%. Furthermore, without sender awareness, we attain performance comparable to SOTA LT algorithms while ensuring fairness among all traffic flows.

The rest of the paper is organized as this. The background and motivation of this paper are described in § 2. The design concept of UniFL is introduced in § 3. § 4 presents extensive evaluation results. The discussions and future works are conducted in § 5. We conclude the paper in § 6.

2 BACKGROUND AND MOTIVATION

2.1 Distributed and Federated Learning

2.1.1 Distributed Deep Learning. The fundamental goal of DL is to refine deep neural networks (DNNs) for tasks such as object classification or generating specific outputs, achieved by adjusting network parameters. In data-parallel DDL, multiple workers simultaneously train on different segments of the dataset and undertake gradient or parameter synchronization with their colleagues. Systems utilizing a PS [4] deploy one or more PS(s) to orchestrate this synchronization. Each worker transmits its computed gradients to the PS, which then aggregates these gradients in a manner weighted by the ratio of the data subset each worker has processed compared to the entire dataset. The PS then consolidates these gradients to update the global parameters, redistributing them to the workers for continued training.

2.1.2 Federated Learning. FL evolves DL by training DNNs on decentralized devices, thus prioritizing data privacy by keeping data localized. This approach not only mitigates privacy and security concerns but also leverages network heterogeneity and node autonomy to enhance learning from diverse data sources without centralization [5]. The network heterogeneity in FL allows the system to efficiently operate across devices with varying network speeds and conditions, optimizing the learning process in real-world scenarios [6]. Node autonomy in FL empowers individual devices to manage their data and compute resources, contributing to a more democratic and scalable learning model [3]. Additionally, the imperative for secure data transmission in FL is addressed through sophisticated encryption and privacy-preserving protocols, ensuring that data remains confidential and secure during the training process [7, 8]. Despite the challenges such as ensuring efficient communication in diverse network conditions [9] and maintaining secure data transmission [10], FL's design inherently supports scalable, privacy-preserving machine learning across a wide array of devices. These features make FL particularly valuable in applications requiring stringent data confidentiality and resource efficiency.

2.2 Loss-tolerant Transmission

As DDL demands for network transmission throughput increase, a domain-specific method of transmission protocol has been discussed repeatedly [11, 12]. This method, known as LT, seeks to enhance network throughput and reduce tail latency by allowing transmission protocols to mask certain packet loss signals. Taking the loss-tolerant transmission protocol (LTP) [11] as an example, it employs a "send once" mechanism on the sender's side to ensure that all gradients are sent at least once without guaranteeing the delivery of every packet. Subsequently, it allows for the retransmission of lost data packets when time permits. This approach can significantly boost throughput and reduce the tail latency of streams, particularly beneficial in DDL scenarios where lower tail latency effectively increases training throughput.

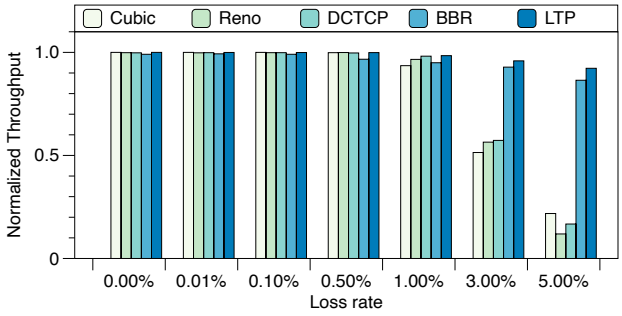

Figure 2: A comparison between LTP and other congestion control algorithms.

In the context of FL, widely employed over WAN, LT demonstrates superior efficacy. We evaluate the performance of LT's representative, LTP, against other TCP congestion control algorithms in real-world scenarios. Using IPerf3 as both a client and a server, with Network Emulator (Netem) [13] simulating the WAN link environment (1Gbps bandwidth, 40ms latency), the results, illustrated in Figure A, lead us to conclude that even the WAN-optimized Bottleneck Bandwidth and Round-trip propagation time (BBR) [14] congestion control protocol cannot surpass LTP in throughput. Moreover, LTP significantly reduces tail latency, allowing all participating clients to complete synchronization tasks simultaneously. However, integrating LT into FL presents considerable challenges.

2.3 Introducing LT into FL

2.3.1 Autonomy and Heterogeneity Present Challenges in LT Deployment. The introduction of LT into FL aims to enhance the performance of FL nodes across WANs. However, significant challenges emerge. Primarily, the high autonomy and heterogeneity of the FL nodes make it difficult to deploy the LT transmission protocol stack across all the worker nodes. For example, different workers might be managed by different entities, or architectural differences in devices (such as AMD64 vs. ARM) could prevent a unified deployment of the algorithm. Furthermore, if only a subset of devices updates to use the LT mechanism on the sender side, it could lead to these devices overly dominating bandwidth over those not using LT, or finishing transmissions prematurely and waiting for others to complete, resulting in prolonged idle states. We validate this problem in the next paragraph.

Therefore, UniFL addresses the issue of LT protocol not being deployable on all devices simultaneously through a protocol design that remains transparent to the sender. Specifically, UniFL masks the ACK signals for lost packets, leading senders to believe that packets have been successfully delivered. This approach effectively maintains transparency of the LT transmission protocol for the sender and preserves one of LT's fundamental benefits: maintaining high network throughput. However, this introduces a further challenge: the normal function of the sender's congestion control is compromised.

The loss of congestion control functionality can cause more severe problems, including excessive malignant switch buffer occupancy, unpredictable bulk packet loss, and extremely unfair network

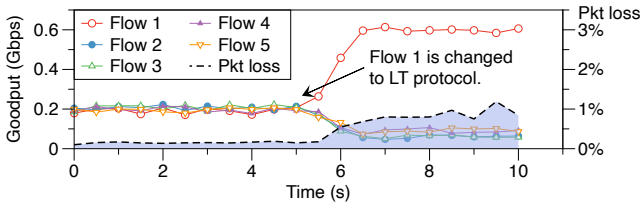

Figure 3: One worker uses LT while others use Cubic.

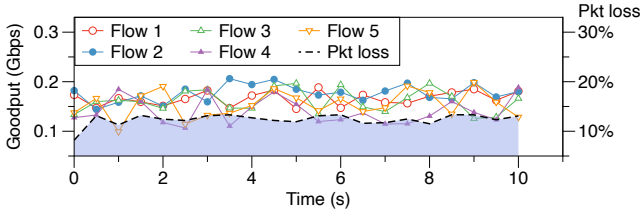

Figure 4: Mask all workers' packet loss signal.

sharing. We evaluate these challenges through a simple experiment. We construct a simple many-to-one topology with five workers and one PS, where the five workers were connected to the switch closest to the PS through a WAN link with 30-50ms delay and 0.1% packet loss, using the Cubic congestion control algorithm by default, with a bandwidth of 1Gbps. During one round of communication, all workers sent fixed-size data to one worker simultaneously, simulating the synchronization scenario in FL.

We conduct two sets of experiments to validate: (1) modifying one worker to use the LT transmission protocol while all other workers use Cubic; (2) masking packet loss signals without altering the worker senders. We assess the overall system changes under these conditions. The results of validation (1) are shown in Figure 3. Modifying the protocol stack of one worker to LT without congestion control led to it monopolizing almost all the bandwidth and caused some degree of packet loss, confirming one of the previously mentioned challenges. The results of validation (2) are shown in Figure 4. Masking all packet loss signals on the PS side, although all streams reached a nearly similar throughput, resulted in an extremely high packet loss rate (over 10%) in the link. This led to significant performance wastage and also indicated high switch buffer occupancy.

To counteract the inefficiency of congestion control algorithms resulting from the masking of packet loss signals, UniFLemploys a multifactor rate limiter to restore the functionality of sender-side congestion control at the receiver end. This aspect will be elaborated upon in the following sections.

2.4 Modifications in Secure Transmission Protocols

In FL, ensuring secure transmission is a fundamental goal. Beyond the commonly used homomorphic encryption techniques in FL, in traditional networking scenarios, when transmitting sensitive data over WANs, a combination of asymmetric like Rivest–Shamir–Adleman (RSA) algorithm and symmetric encryption methods is often employed. This involves the secure exchange of asymmetric keys,

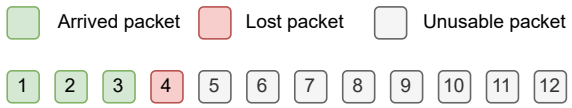

Figure 5: Packet loss will break the encryption/decryption process in the stream cipher.

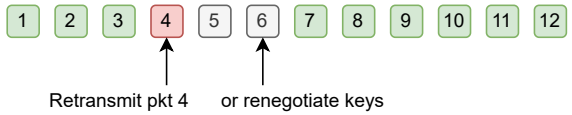

Figure 6: Retransmit/renegotiate keys can solve the problem but the efficiency is low.

followed by the use of one party's public key to encrypt a set of symmetric keys for content encryption and decryption (key negotiation). Once key negotiation is complete, the symmetric key is used by both parties to perform stream encryption of the transmitted content. This encryption and decryption could use the algorithm like Advanced Encryption Standard (AES)-based, packet-level Cipher Block Chaining (CBC) algorithm.

However, introducing LT into FL presents significant challenges for traditional encrypted transmission. This is primarily because traditional stream encryption, aiming to ensure key security, employs stream ciphers for encrypting communications with continuous data packets. In essence, the encryption and decryption process for each packet requires knowledge of the previous packet's data. Consequently, packet loss can have a profound impact. For instance, consider a sequence of packets sent from workers to the PS, as illustrated in Figure 5. Suppose packets 1-3 arrive at the PS normally, but packet 4 is lost. Under such circumstances, all subsequent packets following packet 4 will be undecipherable by the recipient, which is unacceptable. Although we can avoid this issue, as shown in Figure 6, by triggering retransmission signals or renegotiating keys, these solutions have drawbacks. For example, triggering a packet loss signal requires an RTT to activate and can lead to reduced throughput. Renegotiating keys also takes time and makes all packets sent during the key negotiation period unusable.

To address this issue, we design a loss-tolerant secure transmission protocol to support encrypted transmission for UniFL. This protocol is capable of tolerating the loss of some data packets without compromising the integrity of data encryption and decryption. The details of this algorithm will be discussed in subsequent sections.

3 UNIFL DESIGN DETAILS

In this section, we present the design details of UniFL. We will discuss how UniFL addresses the issues outlined.

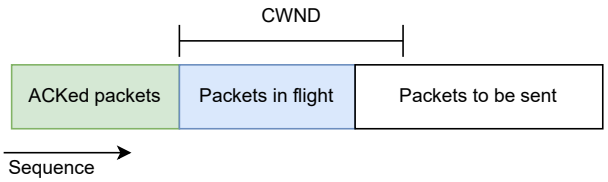

Figure 7: Modeling the congestion window (CWND) and the packet in flights.

3.1 Masking Packet Loss Signals

As discussed earlier, one of the major challenges of introducing LT into FL is the inability to modify all workers. Therefore, UniFL opts to modify the receiver-side software to mask packet loss signals, achieving the LT objective of allowing packet loss.

Specifically, UniFL masks the loss of packets to the sender by generating fake acknowledgement (ACK) responses. When the receiver detects out-of-order packet sequences, it skips sending acknowledgments for the missing packets. Instead, it combines the ACK for the lost packet with that of the next successfully received packet into a single SACK, creating a "fake ACK" signal for the sender, thereby achieving the goal of LT transmission. To illustrate UniFL's approach, consider an example where we need to transmit packets with IDs 1 through 10, and the packet with ID 5 is lost. The sequence received by the endpoint would be *..., 4, 6, 7...* In this case, upon receiving packet 6, UniFL's receiver would issue both ACK5 and ACK6 together as a single SACK. This method allows us to implement an LT strategy that is completely transparent to the sender.

The handling of such lost packets is detailed in the LTP paper and will not be elaborated on here.

3.2 Unified Congestion Control Signals

Due to the masking of packet loss signals, all congestion control algorithms maintained by senders will become ineffective. In such scenarios, it becomes necessary to manage congestion control signals at the receiver end to prevent senders from accelerating excessively, which could fill up the buffer space in the switches along the link and result in substantial packet loss. To address this issue, UniFL employs a multifactorial rate limiter, controlling the size of the sending window at the receiver end (PS) to regulate the rate of the sender (worker).

3.2.1 Congestion Window Modeling. Take traditional congestion control algorithms, which use packet loss as a congestion signal, as an example. As illustrated in Figure 7, when the congestion window (CWND) value exceeds the Packets in Flight (PIF), the sender increases speed. Conversely, if CWND is less, the sender halts for a while, effectively slowing down. Based on this phenomenon, we can achieve the goal of limiting the sender's rate by modifying the window value in the ACKs returned by the receiver.

3.2.2 Rate Limiter. The adjustments UniFL makes to congestion control are primarily aimed at preventing excessive congestion

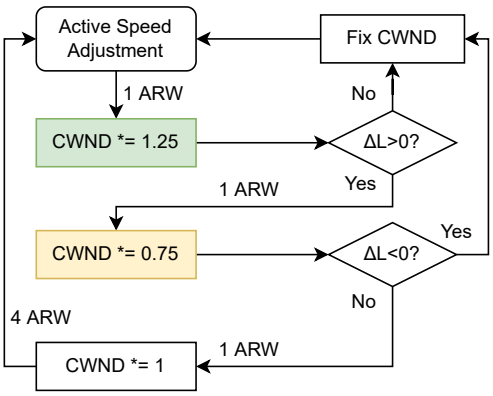

Figure 8: UniFL's congestion control algorithm flowchart.

packet loss due to too high a sending rate while ensuring that congestion control remains unaffected by inevitable non-congestion packet losses. Therefore, rate control is mainly based on the receiver's packet loss signal (L) and its change (ΔL). In other words, UniFL's congestion control algorithm (f_{cc}) aims to make the current connection's bandwidth usage (B) approach the link's upper limit (B_0), but not exceed B_0.

UniFL does not intervene in the speed increase during the initial phase of connection establishment. Upon detection of packet loss signals, UniFL does not directly decelerate but instead samples the packet loss rate over time windows, known as Loss Sampling Windows (LSW). When the packet loss rate remains constant across three consecutive LSWs, UniFL considers the congestion window (CWND) and packets in flight to have reached a relative equilibrium, transitioning into the active rate adjustment phase. This phase operates on the basis of Active Rate Adjustment Windows (ARW).

As shown in Figure 8, in the first ARW, the receiver attempts to increase the congestion window to 1.25 times its original size. If this adjustment does not result in an increase in the change in loss rate ($\Delta L > 0$), then the increase will be retained. If there is no increase in the window, in the next ARW, the congestion window will be adjusted to 0.75 times its original size. If this adjustment leads to a decrease in the loss rate ($\Delta L < 0$), the adjustment will also be retained. The four subsequent ARWs will maintain the current rate. This approach makes sure that $f_{cc}(B) \rightarrow B_0$. This approach bears some resemblance to the BBR rate adjustment methodology.

3.3 Loss-tolerant Stream Cipher

We design a secure transmission protocol that tolerates packet loss. In essence, UniFL still employs an encryption algorithm based on AES-CBC. The difference is that UniFL only performs stream en/decryption within a packet while being independent between packets. A 16-byte initialization vector (IV) is added to the start of each packet's payload for encryption and decryption. Although this approach incurs an additional overhead of about 1.1% per packet (16/1472 bytes), it prevents packet loss from affecting the encryption and decryption process while ensuring that the encryption algorithm remains sufficiently secure. Due to space constraints, this description is kept brief.

In summary, we introduce LT into FL to improve network throughput and solve its problems in device heterogeneity, device autonomy, congestion control, and data transmission security. We evaluate the performance of UniFL in the following sections.

4 EVALUATION

4.1 Simulation Setup

4.1.1 Cluster Configuration. To evaluate the performance of UniFL against existing solutions, we implement a preliminary UniFL protocol stack and a generic federated learning application (including PS and worker) in NS3 [15]. We implement a federated learning cluster with 64 workers and one PS, where the PS randomly selects 8 nodes for gradient synchronization in each training round. The GPU performance of the workers is equivalent to that of one RTX 3090. Workers connect to the PS through a simulated WAN environment, with each worker-to-PS link having a latency distribution of 20-60ms and bandwidth ranging from 0.6Gbps to 0.8Gbps, distributed randomly.

4.1.2 Baselines. For performance comparison, we compare UniFL against common TCP congestion control algorithms, including New Reno (Reno) [16], Cubic [17], and BBR [14]. In addition to these standard congestion control protocols, we also compare it with LTP [11].

4.1.3 Workloads. We utilize a variety of deep learning models commonly used in FL scenarios, including ResNet-152 and VGG-16. The training was based on the CIFAR-10 dataset and continued until no accuracy improvement was observed after 10 epochs.

4.1.4 Metrics. Our evaluation primarily focuses on **throughput** and **top-1 accuracy**. Throughput reflects UniFL's ability to enhance training rates, while Top-1 accuracy assesses the impact of UniFL's introduction of the LT mechanism on training.

4.2 Evaluation Results

Figures 9 and 10 showcase the performance of various congestion control algorithms during the training of ResNet152 and VGG16 models. It's evident that UniFL, without any modifications to the sender, achieves throughput nearly matching that of LTP. Moreover, UniFL significantly outperforms traditional congestion control algorithms, achieving up to 40 times the throughput of Cubic in scenarios with high link packet loss rates.

Additionally, we compare the top-1 accuracy using different algorithms. UniFL achieves accuracy levels comparable to those of congestion control algorithms that do not experience packet loss. Even in the highest packet loss scenario (1%), the accuracy loss is only about -0.2%.

5 DISCUSSION AND FUTURE WORKS

5.1 Dynamic Setting of LT Thresholds

UniFL's approach of uniformly not retransmitting all lost packets may lead to substantial data loss when the link's non-congestion

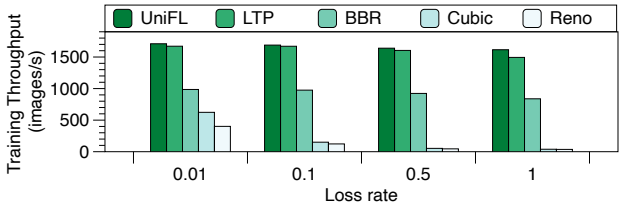

Figure 9: Comparison of throughput with ResNet152.

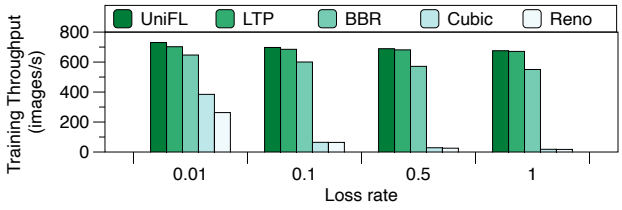

Figure 10: Comparison of throughput with VGG16.

packet loss rate is high. Although this was not replicated in our experiments, there remains a potential risk. To address this issue, we plan to implement an algorithm to control the number of lost packets. Retransmission signals will still be triggered in cases of excessive retransmissions to prevent excessive packet loss. We are currently designing this algorithm and assessing its feasibility.

5.2 Further Evaluation of Encryption/Decryption Algorithms

Currently, we have specifically designed encryption and decryption algorithms for introducing LT into FL, validating their feasibility and correctness, but without a detailed evaluation of the encryption algorithm's security and the potential overhead. Therefore, in future work, we are modeling and analyzing its performance against common malicious tactics from multiple perspectives, including its resistance to content leakage under active and passive attack scenarios.

6 CONCLUSION

This paper introduces UniFL, a system that adapts LT from traditional DDL from the data center to FL scenarios. We thoroughly explore potential challenges in deploying LT within FL, including addressing device heterogeneity and the difficulty of modifying all sending endpoints. Based on this, UniFL is proposed to mask congestion signals on the sender side, achieving the deployment of LT in FL. Despite achieving this, LT still encounters issues with ineffective congestion control. UniFL devises a rate limiter that controls the rate based on receiver-side modeling of link performance and implements the rate limiter through the ACKs. Additionally, UniFL has developed a loss-tolerant security protocol to overcome the challenges faced by traditional stream encryption algorithms, which do not support partial packet loss. Through preliminary simulation experiments under realistic FL workloads, UniFL has demonstrated the ability to achieve throughput comparable to the state-of-the-art LTP without modifying the sender's protocol stack. Moreover, compared to traditional congestion control algorithms, UniFL can achieve up to 40x increase in throughput.

ACKNOWLEDGEMENTS

This work is sponsored by the Key-Area Research and Development Program of Guangdong Province (2021B0101400001), the National Natural Science Foundation of China (62172108), the Major Key Project of PCL, and the Natural Science Foundation of Shanghai (23ZR1404900). We sincerely appreciate the anonymous reviewers for their valuable and constructive feedback.

REFERENCES

[1] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al. 2023. Segment anything. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 4015–4026.

[2] OpenAI. 2023. GPT-4 Technical Report. arXiv:2303.08774 [cs.CL]

[3] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. 2017. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*. PMLR, 1273–1282.

[4] Mu Li, Li Zhou, Zichao Yang, Aaron Li, Fei Xia, David G Andersen, and Alexander Smola. 2013. Parameter server for distributed machine learning. In *Big learning NIPS workshop*, Vol. 6. 2.

[5] Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. 2021. Advances and open problems in federated learning. *Foundations and trends® in machine learning* 14, 1–2 (2021), 1–210.

[6] Tian Li, Anit Kumar Sahu, Ameet Talwalkar, and Virginia Smith. 2020. Federated learning: Challenges, methods, and future directions. *IEEE signal processing magazine* 37, 3 (2020), 50–60.

[7] Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. 2017. Practical secure aggregation for privacy-preserving machine learning. In *proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. 1175–1191.

[8] Ziyao Liu, Jiale Guo, Wenzhuo Yang, Jiani Fan, Kwok-Yan Lam, and Jun Zhao. 2022. Privacy-preserving aggregation in federated learning: A survey. *IEEE Transactions on Big Data* (2022).

[9] Felix Sattler, Simon Wiedemann, Klaus-Robert Müller, and Wojciech Samek. 2019. Sparse binary compression: Towards distributed deep learning with minimal communication. In *2019 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 1–8.

[10] Keith Bonawitz, Hubert Eichner, Wolfgang Grieskamp, Dzmitry Huba, Alex Ingerman, Vladimir Ivanov, Chloe Kiddon, Jakub Konečnỳ, Stefano Mazzocchi, Brendan McMahan, et al. 2019. Towards federated learning at scale: System design. *Proceedings of machine learning and systems* 1 (2019), 374–388.

[11] Zixuan Chen, Lei Shi, Xuandong Liu, Xin Ai, Sen Liu, and Yang Xu. 2023. Boosting distributed machine learning training through loss-tolerant transmission protocol. In *2023 IEEE/ACM 31st International Symposium on Quality of Service (IWQoS)*. IEEE, 1–10.

[12] Hao Wang, Han Tian, Jingrong Chen, Xinchen Wan, Jiacheng Xia, Gaoxiong Zeng, Wei Bai, Junchen Jiang, Yong Wang, and Kai Chen. 2024. Towards {Domain-Specific} Network Transport for Distributed {DNN} Training. In *21st USENIX Symposium on Networked Systems Design and Implementation (NSDI 24)*. 1421–1443.

[13] Linux. [n. d.]. netem - Network Emulator. https://man7.org/linux/man-pages/man8/tc-netem.8.html

[14] Neal Cardwell, Yuchung Cheng, C Stephen Gunn, Soheil Hassas Yeganeh, and Van Jacobson. 2016. Bbr: Congestion-based congestion control: Measuring bottleneck bandwidth and round-trip propagation time. *Queue* 14, 5 (2016), 20–53.

[15] web:ns3 2011-2024. ns-3 Network Simulator. https://www.nsnam.org/.

[16] Sally Floyd, Tom Henderson, and Andrei Gurtov. 2004. *The NewReno modification to TCP's fast recovery algorithm*. Technical Report.

[17] Sangtae Ha, Injong Rhee, and Lisong Xu. 2008. CUBIC: a new TCP-friendly high-speed TCP variant. *ACM SIGOPS operating systems review* 42, 5 (2008), 64–74.