Edge Acceleration of LiDAR Frame Transmission with In-network Machine Learning

Peng Qian
peng.qian@eng.ox.ac.uk
Department of Engineering Science,
University of Oxford
Oxford, United Kingdom

Changgang Zheng
changgang.zheng@eng.ox.ac.uk
Department of Engineering Science,
University of Oxford
Oxford, United Kingdom

Noa Zilberman
noa.zilberman@eng.ox.ac.uk
Department of Engineering Science,
University of Oxford
Oxford, United Kingdom

ABSTRACT

In real-time vehicle perception scenarios, ensuring timely and stable transmission of LiDAR data between vehicles and the network edge is crucial for accurate object detection. However, the inherent variability of wireless links, coupled with the added impact of vehicle mobility, leads to inevitable packet loss and latency jitter, compromising both the timeliness and accuracy of vehicle perception. To address this challenge, we introduce a packet duplication mechanism on dual wireless links, improving LiDAR frame transmission performance. The solution is driven by an integrated In-Network Machine Learning module at a programmable edge device that dynamically detects performance degradation and controls packet duplication. Through practical implementation and extensive evaluation, it is demonstrated that the proposed packet duplication function can effectively address uncertainties in LiDAR frame transmission, while achieving 50% reduction in transmission times.

KEYWORDS

In-network Machine Learning, P4, Edge computing, Vehicle perception

1 INTRODUCTION

The past decade has witnessed significant advancements in the field of autonomous driving, particularly in real-time vehicle perception, which relies on timely Light Detection and Ranging (LiDAR) data transmission and processing. On the one hand, due to the rapid development of deep neural networks, core tasks such as scene and object recognition, can now be performed in real-time with milliseconds-level latency. On the other hand, the deployment of new wireless communication technologies (e.g., the new 5G radio) lays a robust foundation for enabling Vehicle-to-Vehicle (V2V) and Vehicle-to-Infrastructure (V2I) communication. However, the inherent uncertainty of wireless networks in vehicular environments presents a significant challenge for ensuring the real-time delivery of LiDAR data, which remains an issue that has not been thoroughly evaluated and addressed.

A glance at recent works [3] [10] shows that while various Vehicle-to-Everything (V2X) frameworks have been deployed and tested across real-world networks, these studies primarily focused on reducing data size through redundancy elimination or vehicle-edge collaboration across different access network types. For instance, EMP [10] tackled visual redundancy by merging frames and offloading real-time perception tasks to the edge via 4G link, achieving vehicle recognition in milliseconds. Soar [3] introduced a WiFi-based approach that clusters adjacent WiFi access nodes to improve mobility support, enabling real-time LiDAR frame transmission to vehicles within campus area. However, a common limitation of these approaches is the lack of feasible solutions addressing sluggish and unstable LiDAR frame transmissions, which are often caused by unavoidable network fluctuations.

Inspired by these advancements in wireless network frameworks, towards a fast and robust LiDAR frame transmission performance, we propose an In-Network Machine-Learning (IN-ML) driven dual-link packet duplication mechanism. The mechanism has two key components: 1) Packet duplication on dual links. Each packet in a LiDAR frame can be directly cloned on sender side across both Wi-Fi and cellular links, then the receiver keeps only the first arrival replica of each packet, mitigating the uncertainties of wireless links. 2) A trainable in-network machine learning component detects network performance degradation in real-time, and dynamically activates the packet duplication function when needed. The two components leverage the computational power of programmable edge network devices, which have been widely proven to introduce negligible latency overhead when running machine learning tasks and various network optimization operations [11], [5]. This allows tasks to be executed on such a platform without adding additional latency, ensuring no impact on vehicle perception scenarios which set strict millisecond-level requirements. Through implementation and extensive evaluation with a public dataset and a vehicle perception algorithm, the proposed packet duplication function demonstrates both effectiveness and performance robustness, achieving around 50% reduction in transmission latency. This mechanism offers network operators the flexibility to plan network links and deploy them in scenarios requiring critical data uploads, key area monitoring, or urgent danger notifications.

The main contributions of this work are as follows:

- We developed an IN-ML plugin featuring a trainable decision tree algorithm that dynamically detects transmission performance degradation and automatically activates packet duplication when necessary.
- We designed a packet duplication function that can be seamlessly deployed on programmable devices at network edges and within vehicles, enabling dual-link utilization to improve LiDAR frame transmission performance, with no additional header encapsulation required.
- We implemented and tested the proposed mechanism using public datasets and algorithms, under various network conditions. The demonstrated improvements and scenarios comparisons offer practical guidance for deploying such IN-ML based solutions, and tailoring them to various vehicular data transmission needs.

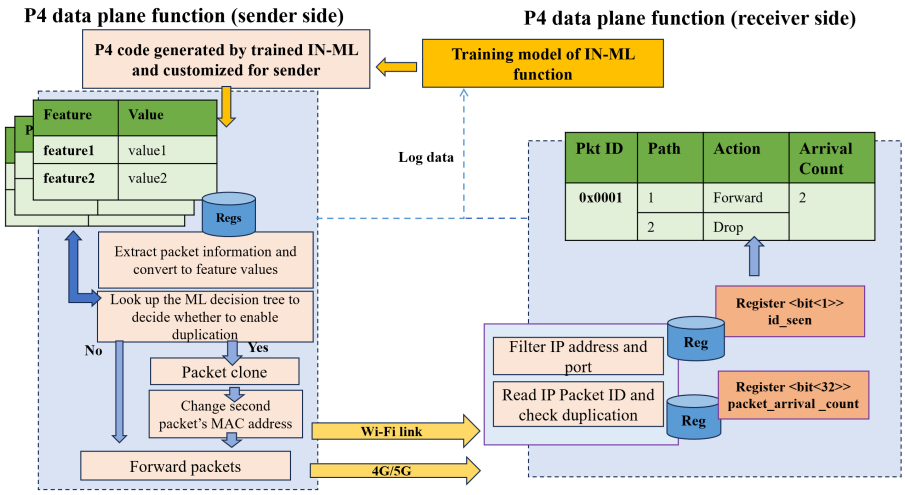


Figure 1: The proposed IN-ML driven packet duplication function pair

2 P4-BASED PACKET DUPLICATION EDGE FUNCTION DESIGN

In this section, we review the literature in two key areas: LiDAR frame size reduction techniques through cooperative vehicle perception frameworks, and the role of programmable network devices in enhancing network acceleration and resilience, which together form the foundation for our high-throughput, low-latency packet duplication strategy.

2.1 Cooperative Vehicle Perception Framework

Chen et al. [9] proposed a feature learning network to perform spatial feature fusion and voxel feature fusion, effectively reducing raw LiDAR point cloud frame data from 2 MB to less than 1 MB. By offloading object fusion and detection tasks to the network edge, the communication cost is significantly reduced while achieving a 10% improvement in vehicle perception precision within a 20-meter range and a 30% improvement at farther distances. Similarly, Zhang et al. [10] introduced EMP, an edge-assisted multi-vehicle perception system to aggregate raw sensor data from nearby vehicles. Using adaptive partitioning and bandwidth-aware algorithms, EMP reduced data transmission size and achieved real-time processing with low latency. Khan et al. [13] proposed VRF, which speeds up LiDAR frame transmission through a two-stage fusing process—first aligning frames to a common reference and then refining the alignment—ensuring centimeter-level precision. Luo et al. [8] proposed a scalable transmission strategy based on maximum cost flow theory, to enabled vehicles to upload LiDAR data to an edge server via 5G V2X multi-hop communication, creating a comprehensive and detailed perception view. In Soar[3], Shi et al. presented an innovative smart roadside infrastructure (SRI) system that enhanced autonomous driving via infrastructure assistance, leveraging WiFi for connectivity. Using off-the-shelf 802.11ac interfaces, Soar created a bi-directional multi-hop Infrastructure-to-Infrastructure (I2I) network and an Infrastructure-to-Vehicle (I2V) broadcast service, supporting data rates of up to 100 Mbps across up to 9 hops. Deployed across 18 lamppost nodes, Soar supports a diverse range of autonomous driving applications while achieving desirable real-time performance.

2.2 Network Optimization Powered by In-Network Machine Learning

The strict millisecond-level transmission time of LiDAR makes us focus on programmable network devices in this work, which not only provide flexible programmability but also offer high performance and low latency [5]. Devices such as Switch Application-Specific Integrated Circuits (ASICs), Network Interface Cards (NICs), and Field Programmable Gate Array (FPGA)-based systems now leverage the domain-specific language P4 [17] to directly define and tailor network protocols within the data plane, enabling advanced network functions and optimising resource utilization. This programmability not only supports custom protocol design but also opens opportunities for offloading complex computations onto network devices, allowing processing to occur entirely within these programmable units. For example, packet and flow classification tasks [4] have been successfully implemented on data-plane hardware using popular machine learning algorithms in a hybrid fashion. The evaluation demonstrated that near-optimal classification results are achievable while significantly reducing latency and server load. Similarly, the authors in [12] proposed a knowledge distill model to transfers knowledge from complex models to the binary decision tree, enabling indirect deployment within switches and improving efficiency for packet classification task. Reinforcement learning-based load balancing (e.g., QCMP [6]) achieves line-rate throughput with minimal latency by dynamically distributing traffic based on real-time conditions, preventing congestion and optimising performance. Leveraging programmable hardware, these algorithms enable real-time detection, classification, and inference with faster response times than traditional centralized monitoring.

After reviewing these two types of studies, we find that existing cooperative frameworks focus mainly on the design and deployment of specific networks and encoded frames, lacking solutions

for network uncertainty and protocol behaviour [20]. This motivates us to explore this issue using high-performance, low-latency programmable devices. While packet duplication has been applied across protocol layers, a purely Layer 3 solution remains unexplored. Compared to Layer 2-based 5G duplication [21], which is limited to specific access methods and lacks dynamic configuration, application-driven flexibility, and customizability, our approach offers greater adaptability. Similarly, multipath transport-layer solutions [22] require additional connection management, congestion control algorithms, and redundant kernel packet processing, introducing significant overhead. In contrast, our IN-ML-driven Layer 3 approach eliminates these inefficiencies while complementing existing methods by enabling seamless integration with various underlying access technologies and adapting to diverse upper-layer application requirements.

3 P4-BASED PACKET DUPLICATION FUNCTION DESIGN

In this section, we describe the proposed mechanism consisting of two main components. The first is a dual-link packet duplication function, and the second is a network condition detection function leveraging an IN-ML plugin (e.g., Planter [11]) to dynamically sense and infer if the current network link is experiencing congestion or instability, subsequently triggering the packet duplication function.

In this work, the following scenario is considered: the vehicle acts as the sender of LiDAR data, while the network edge server serves as the receiver. Assuming in a given area, there is only one vehicle able to capture the real-time image, while other vehicles cannot observe the area due to obstructions. Therefore, it is crucial to ensure that the data from this vehicle in the unobstructed position is delivered promptly and reliably. This guarantees that the considered area can be comprehensively identified at the edge. The information exchange model is that the data-sending function is deployed on the vehicle, while the receiving program operates on the network edge server.

The rationale behind the dual-link packet duplication function, implemented using the P4 language on a programmable device, lies in enhancing data transmission reliability. By replicating packets and sending them simultaneously across multiple links, the packet is ensured to reach its destination through the best link conditions at any given moment. In our design, this packet replication functionality is divided into two components: the sender-side function and the receiver-side function (see Fig. 1).

3.1 Sender-Side Function Design

The packet duplication function in P4 relies on the clone operation, which uses a globally defined session in the switch. This session allows cloned packets to be identified and appropriately processed throughout the pipeline.

A global flag variable on the programmable switch is defined to dynamically determine whether packet cloning should be activated, based on current network conditions and application requirements. The conditions for enabling packet duplication are as follows: (1) the destination IP address and port of incoming packet must match those of the LiDAR frame processing program running on the receiving server, and (2) based on specific features extracted at the time of packet arrival, a predefined group of lookup tables in the IN-ML inference module must confirm that packet duplication is required by setting the corresponding flag.

Once packet duplication is enabled, a clone function will be called for packets directed to the target destination. This function binds a unique clone session ID to the packet and specifies the clone type as `CloneType.I2E` (Ingress-to-Egress). The session ID is linked to an egress port on the secondary link, allowing duplicated packets to be routed over a different path.

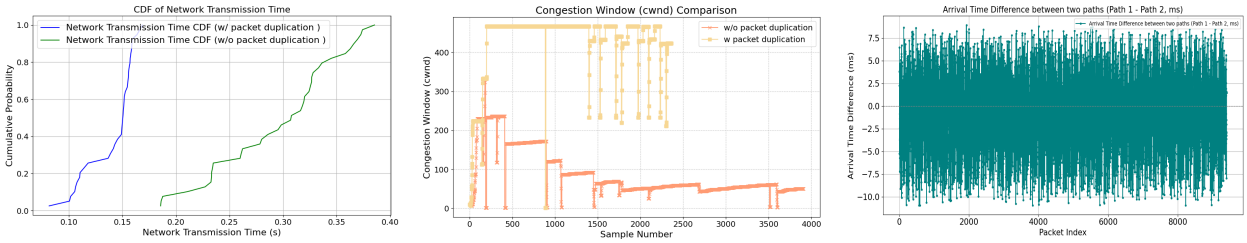
As the cloned packet enters the egress stage, it can be identified by examining the `standard_metadata.instance_type` field. At this point, additional modifications are applied to ensure proper handling and differentiation: the destination MAC address is updated according to the egress port configuration for the secondary link, and the Type of Service (ToS) field is set to a specific value. This ToS field modification enables cloned packets to be easily distinguished from original packets and allows the network to process them differently.

3.2 Receiver-Side Function Design

The primary function of the receiver-side system is to monitor incoming LiDAR data packets, ensuring that only one copy of each unique packet is retained by discarding duplicates that arrive later.

In order to track duplicated packets and their arrival time, we designed two registers, `id_seen`: to track if a specific packet ID has been seen before, and `packet_arrival_count`: to counts the arrival instances of packets with a particular ID. When a valid IPv4 packet with a traced source IP and non-zero identification field arrives, the code reads its IP ID into a metadata structure `meta.ip_id`. Then the arrival count of this packet will be increased by one. If another packet with the same packet ID arrives later, the receiver will lookup the `id_seen` register with the packet ID, and if its value is one, this replica will be marked as drop through a metadata flag. At the same time, the register `packet_arrival_count` will increase one accordingly. It is worth mentioning that since the IP packet ID is a 32-bit non-negative integer, in a LiDAR frame streaming process, this 32-bit space can be quickly exhausted due to the high volume of packet transmissions, causing the sequence numbers to restart from the beginning. This leads to multiple packets with the same packet ID arriving on the two links, although they actually belong to different packets, resulting in incorrect identification as delayed duplicate packets. In our case, with only two links, we can reset the state of a specific packet ID using a 1-bit `id_seen` variable. However, once this function is deployed to a multi-link scenario, we need to leverage the `packet_arrival_count` to record the number of packet arrivals, allowing for sequence number wraparound detection.

Moreover, the sender-side and receiver-side functions described above represent logical deployment positions, tailored for unidirectional flows, such as transmitting LiDAR frames. For acknowledgments returned by the network entity receiving the LiDAR data, this sender/receiver pair approach can also be applied. Packet duplication in both directions can be easily integrated into a unified code solution. To simplify and clarify the description, we omit the details on packet duplication pipeline for acknowledgments.



(a) Comparison of network transmission time (b) Comparison of Congestion Control Window (CWND) (c) Comparison of arrival time difference on dual links

Figure 2: Comparison between different congestion control algorithms

3.3 Detection of network congestion by IN-ML

As depicted in Fig. 1, a group of feature lookup tables are deployed at the sender-side switch For IN-ML. The tables are used to match extracted features from incoming packets, determining if network congestion is detected.

Two identified key features are inter-arrival time and the number of duplicate ACKs between consecutive application responses. For the first feature, in our framework we assume the following communication pattern for LiDAR frame transmission: an HTTP client on the vehicle side actively uploads the latest LiDAR frame to an edge server using an HTTP POST request, and an HTTP response is sent back from the edge server once the frame is fully received. Thus, on the frame sender side, the timestamp of each HTTP response can be recorded, and the inter-arrival time between consecutive responses is calculated and stored as the first feature. To constrain the value range, a right shift of 10 bits and a bitwise AND operations are applied.

The second feature is the number of duplicate ACKs observed between two HTTP responses. For each ACK packet from the receiver, the ACK number, which indicates the number of successfully received bytes, is extracted from the TCP header. At the transport layer, if a packet is likely to be lost, the receiver will repeatedly send ACKs with the same acknowledgment number to notify the sender. This feature thus serves as an indicator of likely packet loss or reordering within a specified time period, as observed by the receiver.

Additionally, the frame transmission time is recorded on the receiver side. Transmission times exceeding a threshold (e.g., 150 ms) are labeled as 1 (congestion experienced), while those below the threshold are labeled as 0. This dataset containing two types of features and corresponding label is then loaded into the Planter [11] IN-ML framework for offline training, using a decision tree model that ultimately generates the corresponding P4 data plane code for deployment on the sender side. This P4 code includes the necessary lookup table, runtime table entry insertion commands, and pipeline functions to assess congestion in real time based on the input features.

4 IMPLEMENTATION AND EVALUATION RESULTS

We set up a LiDAR frame transmission and processing system consisting of a client laptop and a server PC. The client, running Ubuntu 22.04, uses a RESTful client to post KITTI frames [16] and hosts an emulated network with a two-link P4/BMv2 topology in Mininet. The server, operating in an Ubuntu 20.04/WSL environment on Windows 11 with an NVIDIA RTX 3090 GPU and 64GB of memory, runs a RESTful server application to process LiDAR frames and infer object types and positions using the PointPillars [7] algorithm.

4.1 Network transmission with and without packet duplication on dual link

A basic test is conducted based on the optimal network conditions measured in [1], [2]. Specifically, both links are set to have identical properties: 300 Mbps bandwidth, 30 ms end-to-end delay with a 5 ms standard deviation, and a 0.05% packet loss ratio. This configuration allows us to compare the performance of LiDAR data transmission over a single link versus dual links. The client continuously sends a point cloud frame 50 times and records the transmission time. As shown in Fig.2(a), the transmission time for a LiDAR frame can be reduced by approximately 50%, with a significant improvement on performance robustness. To understand this improvement, we analyse the TCP layer's congestion window (cwnd) behaviour, shown in Fig.2(b). Loss-based and rate-based mechanisms temporarily reduce their rates and adopt a conservative approach to gradually re-probe bandwidth when packet loss or delay fluctuations occur. Consequently, such signals continuously constrain the transport layer's transmission rate. For instance, the cwnd curve without packet duplication exhibits multiple window reductions, slower recovery speeds, and lower new limits after each rate drop. With dual links enabled, each packet can consistently arrive via the link with the lowest delay, significantly reducing the likelihood of simultaneous packet loss on both links. Fig.2(c) illustrates the arrival time differences for packets across the two links, showing that packets alternately arrive first on each link. Without one of the links, the transport layer would perceive a packet delay fluctuation of approximately ±10 ms. However, with both links active, such fluctuations are nearly eliminated, preventing unnecessary rate adjustments.

4.2 Impact of different frame sizes

Next, we evaluate the performance of three frame sizes: the 1802 KB frame, representing the raw LiDAR data; the 480 KB frame, which has undergone redundancy removal; and the 120 KB frame, which is further down-sampled using farthest-point sampling. The measurement results (see Fig.3) indicate that, although reducing frame

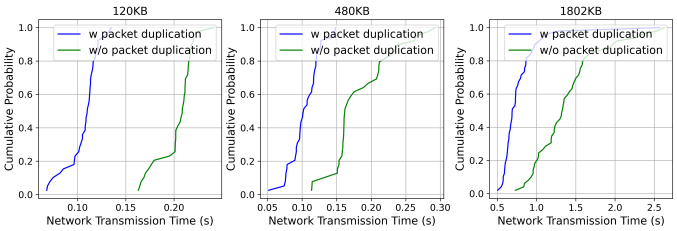


Figure 3: Comparison between different frame sizes

size effectively decreases transmission time, considerable fluctuations and poor transmission times still occur under challenging network conditions. For the 1802 KB frame, there is no doubt that this raw LiDAR frame cannot be transmitted within an acceptable latency range, although the dual link packet duplication can reduce its transmission time by more than 50%. Down-sampling the frame further to 120 KB with dual-link packet duplication reduces transmission time to around 100 ms but may require extra processing on the vehicle and edge to mitigate potential object detection accuracy loss from sparser point density. Notably, packet duplication consistently reduces transmission time and stabilizes performance when bandwidth is sufficient, regardless of frame size.

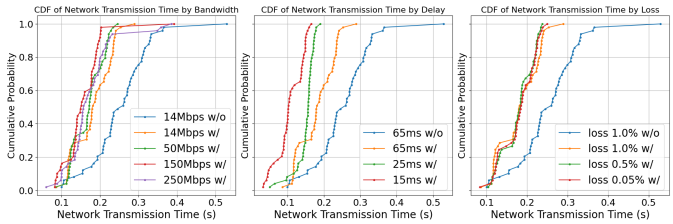


Figure 4: Comparison of encoded frame transmission with (w/) and without (w/o) packet duplication

We consider further encoding the frame to reduce the size to less than 100 KB. In [10], the raw LiDAR frame can be pre-processed on the client side before uploading, performing ground removal, partitioning, and encoding in a sequence. This helps to reduce its LiDAR frame size to around 38 KB. The baseline link is 14 Mbps bandwidth, 30 ms delay with 10 ms variation, and 1% packet loss ratio, to emulate a practical high mobility network [10],[1],[2]. To investigate the impact of bandwidth, delay, and packet loss, we vary one parameter at a time while keeping the other parameters fixed within each group of experiments. Fig. 4 shows the Cumulative Distribution Function (CDF) comparison of transmission time with and without packet duplication. The key conclusion from these subfigures is that while enabling packet duplication under the same link conditions provides varying degrees of improvement in transmission time, further increasing bandwidth or reducing packet loss does not yield additional performance gains. In contrast, a continuous reduction in delay results in a sustained and significant decrease in transmission time.

This can be attributed to the smaller size of the encoded frame, which can be transmitted within a few rounds. In this context, due to the instability of network performance in mobile environments, congestion control algorithms are unable to saturate all available bandwidth for the TCP connection. Additionally, since the number of packets is relatively small, even if a large number of packets are lost in a single round, the reduction in rate only occurs once, and retransmissions can typically be completed within a few rounds. In contrast, reliable transport layer protocols inherently rely on a rate adjustment mechanism based on sending and acknowledgments in rounds, making any reduction in link delay more impactful for overall rate improvement. Therefore, in practical network deployments, such as at an accident-prone, obstructed intersection, using 5G as a backup link should prioritize adjusting the base station antenna's orientation to minimize obstructions between the target vehicle and correctly configuring uplink and downlink resource allocation, as implied by the evaluation results of practical 5G networks [2] [19].

Another observation is that the compressed frame requires 41.3 ms for preprocessing and an additional 19.12 ms for edge-side decoding. Enabling packet duplication saves at least 50 ms of transmission delay, partially offsetting the processing overhead with faster network transmission. This provides a means to counteract the additional processing delays when deploying complex packet encoding strategies.

4.3 Impact of different congestion control algorithms

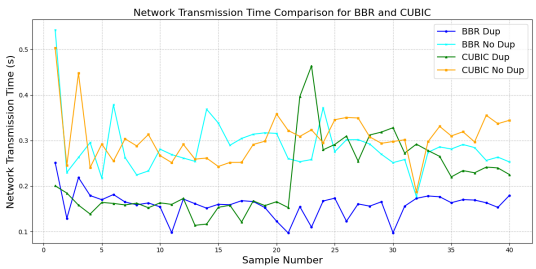


Figure 5: Comparison between different congestion control algorithms with and without packet duplication

We also compared the impact of different congestion control algorithms, as shown in Fig. 5. Intuitively, because the loss-based CUBIC [14] algorithm primarily relies on packet loss as a congestion signal, it tends to more frequently reduce its transmission rate in response to packet loss events. In contrast, BBR, which adjusts future transmission rates based on estimated optimal sending rates for each round, demonstrated relatively better performance. With packet replication enabled, BBR achieved further improvements in overall performance.

Given BBR's known limitation of relatively poor fairness [18], caution is advised when considering its use in multiple flow scenarios. That said, in cases where multiple vehicles can aggregate data from the entire scene through V2V communication for processing and transmission to one frame and transmit it on single connection, or when key learned features are transmitted, BBR can still be effective and suitable for mitigating network uncertainty.

Threshold (ms)	Accuracy	Recall	F1	Precision	No. of Table entry (w/o and w/ direct mapping)
100	0.97	0.99	0.99	0.98	19751 vs 31
150	0.93	0.97	0.96	0.94	19750 vs 31
200	0.79	0.77	0.81	0.86	17914 vs 31
250	0.76	0.77	0.77	0.77	17916 vs 31
300	0.76	0.80	0.76	0.73	17916 vs 31

Table 1: IN-ML parameter settings and impact

4.4 Impact of IN-ML algorithm settings

The impact of parameters in the IN-ML decision tree model is evaluated (see in Table 1). When emulating network conditions for generating training data, the consensus is that a significant gap remains between the network conditions in vehicular networks used for LiDAR transmission and those in ideal networks. To practically account for varying network link conditions and their impact on LiDAR transmission performance, we configured a baseline combination of 300 Mbps bandwidth, 10 ms latency, 1 ms jitter, and 0.01% packet loss as the lower bound for performance timing. Conversely, a combination of 14 Mbps bandwidth, 160 ms latency, 10 ms jitter, and 1% packet loss represents the worst-case performance scenario. The generated transmission time samples (around 5000 samples) ranged from tens of milliseconds to several seconds. As the above tests have demonstrated that our duplication mechanism is particularly effective in improving performance when the packet transmission delay exceeds 100 ms, then we recommend to set the threshold larger than 100 ms. Once the threshold is set to 100 ms or 150 ms, high model accuracy, precision, F1 score and recall results can be obtained, while threshold larger than 200 ms will have relatively lower model accuracy, implying ranging 100 ms to 200 ms would be a proper threshold for our model and settings. Regarding the number of table entries in the P4 environment, we use a direct mapping mechanism to reduce the count from over ten thousand entries to fewer than a hundred. Specifically, this approach maps logical operations and the tree into a sequence of depth tables and a decision table. Each depth table applies logical operations to determine the output of a branch at that depth, which then serves as input for the next depth until the leaf node is reached.

5 CONCLUSION

In this work, we proposed an IN-ML-driven packet duplication function to enhance the transmission performance of LiDAR frames in autonomous driving. By monitoring network conditions and extracting features within programmable devices, the trained IN-ML model can dynamically determine if packet duplication is necessary. The proposed packet duplication function was implemented leveraging Planter [11] to automate ML model training and data plane code generation.

Using tests spanning different packet sizes, network conditions, and transport-layer congestion control protocols, we demonstrated that the transmission time of LiDAR frames is significantly reduced and stabilized. This validates the effectiveness of the approach, providing a new network deployment strategy for various scenarios in vehicular perception, such as critical information notifications and key object image uploads.

6 ACKNOWLEDGMENT

This research was partly funded by EU Horizon SmartEdge (101092908, Innovate UK 10056403). For the purpose of Open Access, the author has applied a CC BY public copyright license to any Author Accepted Manuscript (AAM) version arising from this submission.

REFERENCES

[1] Hassan, Ahmad, et al. "Vivisecting mobility management in 5G cellular networks." Proceedings of the ACM SIGCOMM 2022 Conference. 2022.
[2] Narayanan, Arvind, et al. "A first look at commercial 5G performance on smartphones." Proceedings of The Web Conference 2020. 2020.
[3] Shi, Shuyao, et al. "Soar: Design and Deployment of A Smart Roadside Infrastructure System for Autonomous Driving." ACM MobiCom 2024.
[4] Zheng, Changgang, et al. "IIsy: Hybrid In-Network Classification Using Programmable Switches." IEEE Transactions on Networking 2024.
[5] Zheng, Changgang, et al. "In-Network Machine Learning Using Programmable Network Devices: A Survey." IEEE Communications Surveys & Tutorials, 2023.
[6] Zheng, Changgang, et al. "QCMP: Load Balancing via In-Network Reinforcement Learning." ACM SIGCOMM FIRA workshop, 2023.
[7] Lang, Alex H., et al. "Pointpillars: Fast encoders for object detection from point clouds." Proceedings of IEEE/CVF CVPR 2019.
[8] Luo, Guiyang, et al. "Edgecooper: Network-aware cooperative lidar perception for enhanced vehicular awareness." IEEE Journal on Selected Areas in Communications 2023.
[9] Chen, Qi, et al. "F-cooper: Feature based cooperative perception for autonomous vehicle edge computing system using 3D point clouds." IEEE/ACM SEC conference 2019.
[10] Zhang, Xumiao, et al. "Emp: Edge-assisted multi-vehicle perception." ACM Mobicom conference 2021.
[11] Zheng, Changgang, et al. "Planter: Rapid prototyping of in-network machine learning inference." ACM SIGCOMM Computer Communication Review 54.1 (2024): 2-21.
[12] Xie, Guorui, et al. "Empowering in-network classification in programmable switches by binary decision tree and knowledge distillation." IEEE/ACM Transactions on Networking 32.1 (2023): 382-395.
[13] Khan, Kaleem Nawaz, et al. "VRF: Vehicle Road-side Point Cloud Fusion." ACM Mobisys conference 2024.
[14] Ha, Sangtae, et al. "CUBIC: a new TCP-friendly high-speed TCP variant." ACM SIGOPS operating systems review 42.5 (2008): 64-74.
[15] Cardwell, Neal, et al. "BBR: congestion-based congestion control." Communications of the ACM 60.2 (2017): 58-66.
[16] Geiger, Andreas, et al. "Vision meets robotics: The kitti dataset." The International Journal of Robotics Research 32.11 (2013): 1231-1237.
[17] Bosshart, Pat, et al. "P4: Programming protocol-independent packet processors." ACM SIGCOMM Computer Communication Review 2014.
[18] Hock, Mario, et al. "Experimental evaluation of BBR congestion control." IEEE ICNP conference. 2017.
[19] Qian, Peng, et al. "Remote production for live holographic teleportation applications in 5G networks." IEEE Transactions on Broadcasting 2022.
[20] Cui, Jiaxun, et al. "Coopernaut: End-to-end driving with cooperative perception for networked vehicles." IEEE/CVF CVPR conference 2022.
[21] Aijaz, Adnan. "Packet duplication in dual connectivity enabled 5G wireless networks: Overview and challenges." IEEE Communications Standards Magazine 3.3 (2019): 20-28.
[22] Li, Ming, et al. "Multipath transmission for the internet: A survey." IEEE Communications Surveys & Tutorials 18.4 (2016): 2887-2925.