Communication Challenges in Infrastructure-Vehicle Cooperative Autonomous Driving: A Field Deployment Perspective

Shaoshan Liu, Bo Yu
PerceptIn

Jie Tang
South China University of Technology. Corresponding author: *cstangjie@scut.edu.cn*

Yuhao Zhu
University of Rochester

Xue Liu
McGill University

***Abstract*—We have commercially deployed an infrastructure-vehicle cooperative autonomous driving system and verified its superiority over on-vehicle only autonomous driving systems. However, we have also identified network communication as the main technical roadblock for reliable cooperative autonomous driving. Hence, the goal of this article is to help the communications research community to better understand the real-world communication challenges in cooperative autonomous driving, as well as to introduce practical solutions for establishing a good baseline for further research and development efforts. Specifically, we introduce the real-world network communication challenges: first, the current mobile network bandwidth is constraining the uploading of raw sensing data, which is crucial for cloud-based applications such as deep learning model training and high-definition map generation *etc*. Second, the network latency jitters remain high for a considerable portion of a vehicle's trip, greatly impacting the reliability and safety of the operations of autonomous vehicles. To address these two challenges, we have developed, deployed, and verified two practical solutions. First, a sensing compression strategy to cope with the network bandwidth challenge. Second, an adaptive fusion engine to cope with the latency variation challenge.**

1. Introduction

In the past few years, we have developed and commercially deployed an infrastructure-vehicle cooperative autonomous driving system on public roads, and we have verified that the cooperative autonomous driving system significantly improves the safety of autonomous vehicles compared to the on-vehicle only autonomous driving

approach.

Specifically, the cooperative autonomous driving system extends the perception range from 70 meters (with the on-vehicle perception only) to over 1000 meters (by fusing perception results from multiple road-side perception units within a region, hence the whole system knows more of the environment). In addition, the disengagement rate measures how often a safety driver has to take over control of the autonomous vehicle in cases that the autonomous driving system cannot handle. With the deployment of the cooperative autonomous driving system, the disengagement rate has been dropped by more than 90% in multiple busy intersections in our deployment where the on-vehicle only perception fail to detect the blind spots.

However, we have also identified that network communication is a major technical roadblock for reliable cooperative autonomous driving deployments. Hence, the goal of this article is to help the communications research community to better understand the real-world communication challenges in cooperative autonomous driving, as well as to provide practical solutions so as to establish a good baseline for further research and development efforts. The contributions are fourfold:

- We introduce the cooperative autonomous driving application to the communications community.
- We explain two real-world network communication challenges for cooperative autonomous driving: first, the current mobile network *bandwidth* is constraining the uploading of raw sensing data, which is crucial for cloud-based applications such as deep learning model training and high-definition map generation *etc*. Second, the network *latency jitters* remain high for a considerable portion of a vehicle's trip, greatly impacting the reliability and safety of autonomous vehicles.
- We propose the sensing data compression strategy to address the network bandwidth challenge.
- We propose the adaptive fusion strategy to address the network latency variation challenge.

The rest of the article is organized as follows: in section 2 we provide the background on on-vehicle as well as cooperative autonomous driving. In section 3, we introduce the technical details of our commercial cooperative autonomous driving system. In section 4, we explain the real-world communication challenges in cooperative autonomous driving. In section 5, we delve into the sensing data compression strategy for surpassing the bandwidth limitation. In section 6, we delve into the adaptive fusion strategy for latency variation mitigation. Finally, we conclude and summarize future work in section 7.

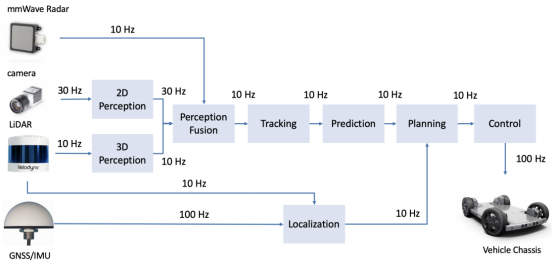

Figure 1. On-vehicle only autonomous driving system.

2. Background

Since its inception, autonomous driving has primarily relied on the on-vehicle processing systems [1], [2]. As shown in Fig. 1, these on-vehicle only autonomous driving systems usually rely on raw sensing data from mmWave radars, LiDARs, cameras, and GNSS/IMUs, and each sensor produces raw data at a different frequency. The processing components are invoked with different frequencies, performing computation using the latest input data and produce outputs to downstream components periodically.

The cameras capture images at 30 FPS and feed the raw data to the 2D Perception module, the LiDARs capture point clouds at 10 FPS and feed the raw data to the 3D Perception module as well as the Localization module, the GNSS/IMUs generate positional updates at 100 Hz and feed the raw data to the Localization module, the mmWave radars detect obstacles at 10 FPS and feed the raw data to the Perception Fusion module.

The results of 2D and 3D Perception are fed into the Perception Fusion module to create a comprehensive perception list of all detected objects. The perception list is then fed into the Tracking module to create a tracking list of all

detected objects. The tracking list then is fed into the Prediction module to create a prediction list of all objects. After that, both the prediction results and the localization results are fed into the Planning module to generate a navigation plan. The navigation plan then is fed into the Control module to generate control commands, which are finally sent to the autonomous vehicle for execution at 100 Hz.

The on-vehicle only autonomous driving approach, while widely experimented and adopted by many companies, faces fundamental reliability, safety, and accuracy challenges going forward. In detail, on-vehicle sensors such as the cameras and LiDARs have limited detection ranges and cannot handle occlusion problems. Worse, cameras are challenged by varying lighting conditions, and LiDARs can be confused by varying weather conditions such as fog and snowfall. Unusual, unfamiliar, and unstructured situations, such as accidents, road work can be hard to classify. Even if on-vehicle sensors could capture perfect information, the compute capability of a single vehicle would unlikely be sufficient for real-time processing as algorithms become increasingly complicated.

The limitations of on-vehicle only autonomous vehicles can be overcome by V2X (vehicle to everything) communication technology. In the assists of V2X communication, cooperative autonomous driving [3] could be achieved by obtaining augmented sensing and maneuvering information in real-time from surrounding vehicles and infrastructures, which will enhance safety, reliability and efficiency of autonomous driving. While discrete use cases (such as cooperative sensing and intersection management) demonstrating the advantages of cooperative autonomous driving have been explored in several studies [3], [4], there still lacks a real deployment of cooperative autonomous driving systems to reveal and address real design and implementation challenges [4].

As far as we know, our cooperative autonomous driving system is the first commercial infrastructure-vehicle cooperative autonomous driving system that combines intelligent infrastructures, dubbed Systems on Road (SoRs) [5], and autonomous vehicles, dubbed Systems on Vehicle (SoVs) [2], along with the intelligent transportation cloud system (ITCS) [6] into one integral and fully intelligent transportation system. Specifically, the SoV consists of an Industrial PC (IPC), a series of sensors, as well as the ECUs and the CAN bus connecting all components (the ECUs and the CAN bus are standard autonomous vehicle components). The SoR consists of similar but more advanced equipment. Different from the SoV, the SoR's IPC is configured with a high-end Nvidia 1080Ti GPU along with other sensors. The interactions between the SoVs and the SoRs are highly latency-sensitive, each SoR is equipped with a roadside unit (RSU), and each SoV is equipped with an on-board unit (OBU), the RSUs and OBUs communicate directly to facilitate the fusion of the SoR data and the SoV data on the vehicle-side.

3. Infrastructure-Vehicle Cooperative Autonomous Driving: An Overview

In this section, we present the technical details of our commercial cooperative autonomous driving system. Fig. 2 presents a detailed end-to-end illustration of our cooperative autonomous driving system. Starting from the left side, the sensing systems of the SoV and the SoR generate raw sensing data from mmWave radars, LiDARs, cameras, Global Navigation Satellite System (GNSS) receivers, and Inertial Measurement Units (IMUs), where each sensor produces raw data at its own frequency. The cameras capture images at 30 FPS, the LiDARs capture point clouds at 10 FPS, the GNSS/IMUs generate positional updates at 100 Hz. Next, the 2D Perception node (*e.g.* YOLO [7]) consumes raw images for object detection and scene segmentation. The 3D Perception node (*e.g.* PointPillars [8]) consumes raw LiDAR scans for object shape and type detection. The Perception Fusion node consumes outputs from the 2D Perception node, the 3D Perception node, as well as the mmWave radars raw data to create a comprehensive perception list of all detected objects. The Localization node is only needed on the SoVs (*e.g.* LiDAR odometry and mapping [9]). It consumes and fuses raw LiDAR scans as well as GNSS/IMU data for vehicle positional updates.

Then, the interaction between the SoRs and the SoVs starts: the output of the SoR's Tracking node is fed to the SoV's Tracking node, which

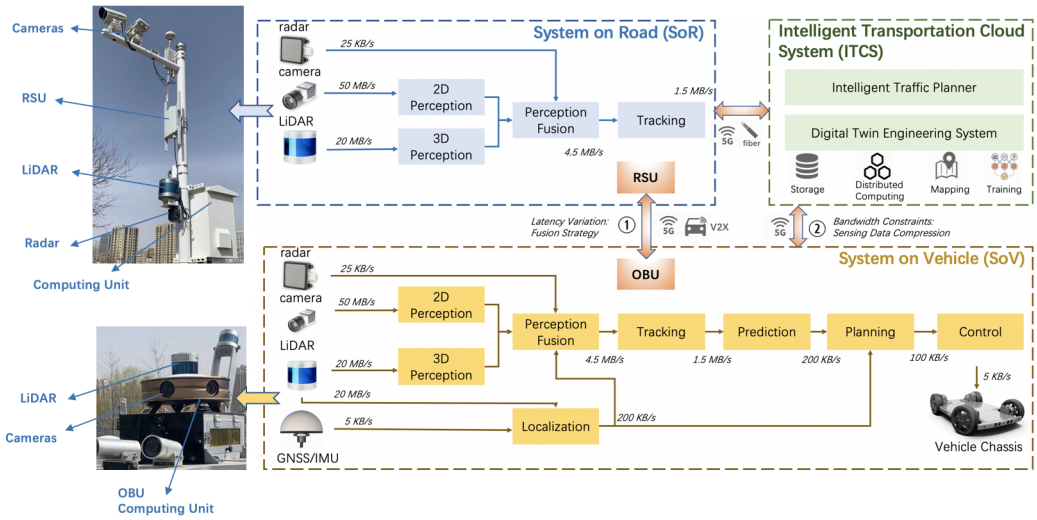

Figure 2. An overview of our cooperative autonomous driving system.

generates a comprehensive perception list as well as the historical trajectory of each detected object. Note that the interaction between the SoRs and the SoVs are highly sensitive to latency jitters. To ensure the reliability and safety of the vehicles, the SoV is designed to tolerate **up to 50 ms of latency** delay from the SoR.

At last, the SoV has comprehensive tracking data, which is fed to the Prediction node to generate the prediction list. Both the prediction list and the localization results are fed into the Planning node at 10 Hz to generate a navigation plan. The navigation plan then is fed into the Control node at 10 Hz to generate control commands, which are finally sent to the autonomous machine for execution at 100 Hz.

In addition to the latency-sensitive SoV-SoR interactions, the SoRs and the SoVs also communicate directly with the ITCS. The SoRs and the SoVs will upload the raw sensing data for cloud-based applications such as simulation, deep learning model training, and high-definite (HD) map generation [10]. The uploading of sensing data can tolerate latency variation but is bandwidth bound, it demands **at least 25 MB/s uplink bandwidth** for streaming essential raw data to the ITCS.

Three modes of communication are supported, C-V2X, commercial 5G network, and optical fiber network. The latency-sensitive communication between the SoRs and the SoVs can be done through C-V2X and commercial 5G network; the bandwidth-constrained communication between the SoVs and the ITCS can be done through commercial 5G network; and the communication between the SoRs and the ITCS can be done through commercial 5G network, or optical fiber network. We never experienced any bandwidth or latency problem with the optical fiber network, nonetheless, the C-V2X network does not provide sufficient bandwidth and the commercial 5G network contains too much latency jitters for the cooperative autonomous driving workloads.

4. Real-World Communication Challenges in Cooperative Autonomous Driving

In this section, we delve into the details of the communication challenges we have observed in our deployments: first, the current mobile network *bandwidth* is constraining the uploading of raw sensing data, which is crucial for cloud-based applications such as deep learning model training and high-definition map generation *etc*. Second, the network *latency jitters* remain high for a considerable portion of a vehicle's trip, greatly impacting the reliability and safety of autonomous vehicles.

Note that in our deployments, LTE V2X is used for broadcasting simple messages of vehicle states and 5G mobile network is used for transferring the large volume of perception data. While 3GPP (3rd Generation Partnership Project)

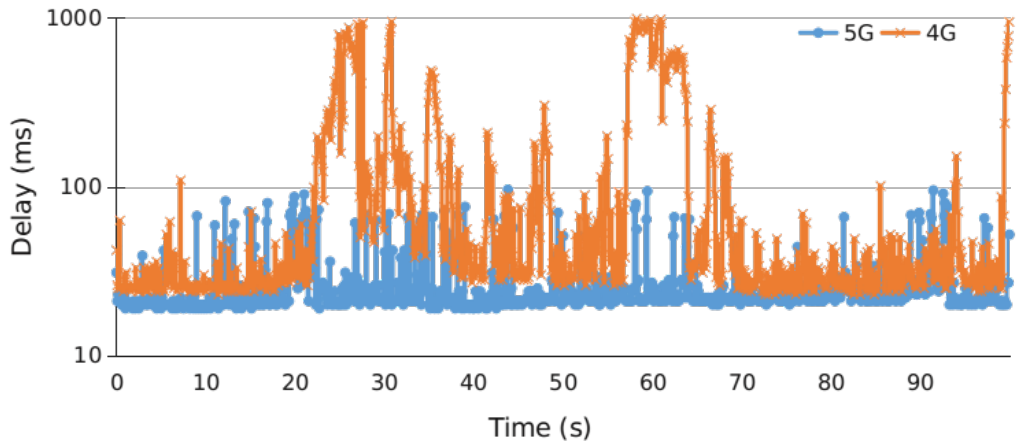

Figure 3. An excerption of 4G and 5G communication latency from our measurements on our deployment cite.

has recently completed 5G-NR release 16 [11] to enhance V2X direct communication, the state-of-the-art commodity OBU and RSU only support LTE V2X direct communication (specified by 3GPP release 14), of which communication capacity is insufficient for infrastructure based perception. Our field testing, in which ZTE's and Huawei's RSUs and OBUs are used to perform direct communication via the PC5 interface, verifies that although the latency of LTE direct communication (20 to 30 ms) is sufficient for fusing road-side tracking results on the vehicle side, the maximum bandwidth (200 KB/s) is far below the required data transmitting rate of the road-side infrastructure, even when only transmitting the results of the Tracking module demands 1.5 MB/s. Because of the limited bandwidth, direct communication is used in our real deployment to broadcast small sized packages for some latency-sensitive use cases, such as platooning.

In our deployment, commercial 5G mobile network is used for providing sufficient bandwidth and transfer road-side tracking results to the vehicle in real-time. In terms of bandwidth, the commercial 5G network can achieve 25 MB/s throughput for downloads, which is sufficient for fusing road-side perception results (1.5 MB/s) on the vehicle, and provides 9 MB/s throughput for uploads, which is sufficient for uploading autonomous vehicle's ego states.

To have a comprehensive understanding of the wireless communication network environment, we also tried with the commercial 4G LTE network, the test results indicate that the 4G LTE network provides 4 MB/s downlink bandwidth and 2.5 MB/s uplink bandwidth, which can satisfy the bandwidth requirements of the infrastructure assisted fusion system, but is far from meeting the bandwidth requirement of sending sensor data.

We conducted 4G and 5G latency measurement for several rounds on our deployment site, in which data from SoR is first transferred to a local edge server through direct fiber connections, then transferred to core networking using TCP/IP protocol, and finally sent to SoV through mobile network. Derived from experimental measurements, the 5G network has an average latency of 30 ms, which satisfies our requirements. However, latency jitters can be as large as 100 ms, which lead to serious frame dropping while performing the infrastructure-vehicle fusion operations. As shown by an excerption from our measurements, in Fig. 3, the 4G LTE network exhibits rather long latency and large jitters, which can not be used for fusing road-side information timely. rk exhibits rather long latency and large jitters, which can not be used for fusing road-side information timely.

In summary, disappointingly, the field experimental results demonstrate that the real-world wireless networks do not simultaneously provide low latency, high bandwidth, and low jitter, which significantly hurts the timely arrivals of

the infrastructure-side data and the uploading of raw sensing data. While we have little control over the commercially deployed communication networks, in the next two sections, we delve into two techniques, one to cope with the bandwidth limitation problem to facilitate continuous sensing data uploading , and the other to cope with the latency jitter problems to enable reliable real-time infrastructure-vehicle cooperation.

5. Practical Solution 1: Sensing Data Compression for Surpassing Bandwidth Limitation

In this section, we explain how we address the bandwidth limitation challenge with the sensing data compression strategy. We upload raw sensing data, including images and LiDAR-generated point clouds, which are used for HD map creation, deep learning model training, and simulation workloads. In particular, point cloud data require continuous transmission to assist HD ma construction; point cloud data are large in size and thus challenge the bandwidth limit of the communication technologies. Over the years, we have developed and deployed new point cloud compression algorithms to ease the bandwidth burden [12].

Today's LiDARs generate hundreds of thousands of points each frame, amounting to up to 25 MB of raw data per second, which leads to about 1 TB of data assuming 10 hours of operation time per vehicle. Worse, an HD map is typically updated at least weekly, thus requiring *continuous*, as opposed to one-off, collecting point cloud data from the vehicles. However, vehicles have only cellular accesses, part of whose bandwidth have to be used to communicate with the RSUs (and potentially other vehicles in the future). With more autonomous vehicles on the road in the same locality, point cloud data transmission will undoubtedly congest the base stations.

Hence, **our main challenge is to compress at least 25 MB/s of raw LiDAR data with only 9 MB/s of uplink bandwidth available**. Point cloud compression must also be done in real-time, i.e., match the data generation rate, in order to ease local storage pressure.

The main idea of our compression system is to exploit redundancies both within a point cloud (spatial) and across point clouds (temporal). Spatially, many surfaces in the real-world are planes (e.g., walls and ground); even non-plane surfaces could be approximated by a set of planes. Temporally, consecutive point clouds share a great chunk of overlapped areas of the scene; thus, the same set of planes could be used to encode points across point clouds. While intuitive, exploiting spatial and temporal redundancies in real-time is challenging due to the irregular/unstructured point cloud and the compute-intensive plane fitting process.

Fig. 4 provides a high-level overview of our system, which consists of three main blocks: range image conversion, spatial encoding, and temporal encoding. Given a sequence of consecutive point clouds, we differentiate between two point cloud types: key point cloud (K-frame) and predicted point cloud (P-frame). A sequence has only one K-frame and the rest is P-frames. P-frames are transformed (both translation and rotation) to K-frame's coordinate system using the IMU measurements. After transformation, each point cloud is converted to a range image for subsequent computations. The range image not only provides an initial compression to the original point cloud, but also provides a structured representation of the (unstructured) point cloud that is hardware-friendly.

We then spatially encode K-frame by fitting planes; the fitted planes in the K-frame are then (re-)used to temporally encode P-frames, greatly improving the overall compression rate and speed. In order to be robust against transformation errors, which might be introduced due to noisy IMU observations, we propose a set of techniques that compensate the sensor noise and preserve the encoding quality.

In the end, after spatial and temporal encoding, most of the tiles in the range images are plane-encoded; the unfit tiles are left in what we call residual maps. The planes and the residual maps are then further compressed by a lossless compression scheme (e.g., Huffman encoding [13]) to generate the final encoded data. Overall, our compression method achieves up to 90× compression rate, significantly outperforming MPEG's LiDAR point cloud compression standard [14].

The compressed point clouds also provide a new avenue for video compression through

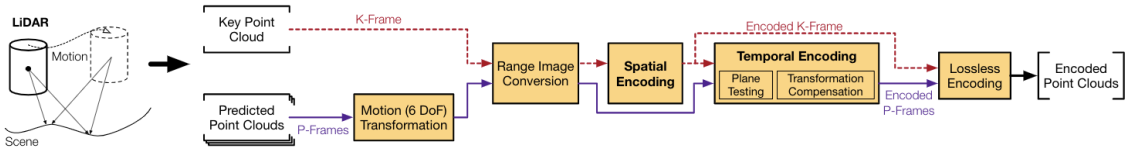

Figure 4. Overview of our compression system, which compresses a sequence of consecutive point clouds. All the points clouds are converted to range images to accelerate the compression speed. We spatially encode the key point-cloud (K-frame) in the sequence, typically the middle one. The spatial encoding results of the K-frame are then used to temporally encode the rest of the point clouds, which we call predicted point clouds (P-frames).

through signal reconstruction and enhancement. In particular, sparse point clouds can be seen as possessing accurate depth information about key points in an image. We demonstrated an ultra-lightweight ($< 10\mu s$) algorithm to estimate the pixel flow across frames using the sparse depth information, which is then used to deblur, denoise, and supersample videos [15]. In this way, the vehicles can afford to capture videos in a much lower resolution and at a much lower rate, relying on the point cloud-assisted enhancement pass to reconstruct the original video signals.

6. Practical Solution 2: Adaptive Fusion for Latency Variation Mitigation

In this section, we explain how we address the latency variation challenge with the adaptive fusion strategy. After receiving the SoR's tracking data, the SoV's OBU will fuse the data with its own tracking module's output. We call this process *tracking fusion*. Due to the variation in network latency, SoR's data often arrives non-deterministically, deviating from the expected arrival time by as much as 100 ms. However, there is only a very small time window (in our case 50 ms) that the SoV could wait for the SoR data arrival, since the entire system must operate in real-time to ensure safety. Thus, if the SoV cannot receive the data from the SoR on time, the SoV has no choice but to skip the tracking fusion process; instead, the vehicle would have to track other vehicles without the global information from the SoR, potentially degrading the task accuracy and, by extension, safety. In our field tests, we find that in the worst case consecutive frames from the SoR are discarded by the SoV. As a result, we observed a **deadline miss ratio as high as 8%** of the tracking fusion task, greatly impacting the reliability and safety of autonomous vehicles.

One way to address the latency jitter issue is if the semantics of the communicated data is inherently tolerant to delay. In particular, we could offload the tracking task to the SoR (which does not add much load to the SoR given that the SoR is equipped with a more powerful computing unit compared to the SoV). An implication of offloading tracking to SoR is that trajectory prediction will have to be offloaded to the SoR as well, since prediction depends on the tracking data. We could then communicate the prediction results from the SoR to the SoV. We call this process *prediction fusion*, as opposed to tracking fusion, since it is the prediction results that get fused.

Prediction fusion is less sensitive to network latency jitters and more tolerant to network congestion — because of two key reasons. First, the amount of data required for prediction fusion (200 KB/s) is less than that of tracking fusion (1.5 MB/s). Second, the semantics of prediction results dictate that they are more tolerant to network delay, because each prediction result contains the predicted trajectory of a few seconds into the future (5 seconds in our case). Even if one prediction packet is dropped or arrives late, the previous prediction packet still has sufficient information in the predicted trajectory to cover a 5-second temporal span.

However, fusing prediction results introduces more noise for the planning module compared to fusing the tracking results. Assuming the current time is T, compared to the actual trajectory at T+1 that should have arrived but did not, we now

resort to the predicted trajectory at T+1 (from the predicted packet), which is less accurate. The more prediction packet drops, the less accurate our downstream algorithm will be.

This trade-off between response time and accuracy motivated us to develop an adaptive fusion engine to dynamically choose a fusion method to achieve the best SoV performance under different situations. The adaptive fusion engine monitors the network condition in real time, measured by the number of dropped SoR fusion frames within a sliding window. If the number of dropped frames crosses a threshold, the adaptive fusion engine selects the planning fusion method, otherwise, it stays with the tracking fusion method.

To implement adaptive fusion, we developed a host component on the SoV side (OBU) and an agent component on the SoR side (RSU). The RSU and the OBU use almost the same software stack, including PHY layer, MAC layer, IP layer, and Control layer. We added an adaptive fusion layer above the Control layer; the OBU monitors the network conditions on the SoV side and makes adaptive fusion decisions accordingly. The OBU notifies the RSU as to whether to switch a different fusion approach to improve the fusion efficiency. The field test results verify that the adaptive fusion method has significantly improved autonomous driving's safety and reliability by **decreasing the fusion miss ratio from 8% to less than 1%**.

7. Summary and Future Work

The infrastructure-vehicle cooperative autonomous driving approach is the promising way to make autonomous driving feasible. However, it faces communication challenges: the network bandwidth is constraining the uploading of raw sensing data, and the high real-world network latency jitters greatly impact the reliability and safety of autonomous vehicles. While currently we have limited control over commercial wireless communication networks, we have developed, deployed, and verified two practical solutions: a sensing compression strategy to cope with the network bandwidth problem, and an adaptive fusion engine to cope with the latency variation problem. It is worth noting that the proposed two practical solutions are not limited to our systems, but are general and can be applied to any implementation of the infrastructure-vehicle cooperative autonomous driving system.

The goal of this article is to inspire the development of a highly reliable cooperative autonomous driving communication network with bandwidth guarantees and minimum latency variations. There are potential areas of research to achieve these goals: First, we can adjust the priority in the commercial wireless communication networks to provide the highest priority to autonomous machines connections as they are safety-critical. Second, a virtual communication backbone that builds on top of both the LTE or 5G commercial networks as well as the direct communication C-V2X networks can improve the reliability of autonomous machine communication through intelligent scheduling.

REFERENCES

1. S. Liu, L. Li, J. Tang, S. Wu, and J.-L. Gaudiot, "Creating autonomous vehicle systems," *Synthesis Lectures on Computer Science*, vol. 8, no. 2, pp. i–216, 2020.
2. B. Yu, W. Hu, L. Xu, J. Tang, S. Liu, and Y. Zhu, "Building the computing system for autonomous micromobility vehicles: Design constraints and architectural optimizations," in *2020 53rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*. IEEE, 2020, pp. 1067–1081.
3. L. Hobert, A. Festag, I. Llatser, L. Altomare, F. Visintainer, and A. Kovacs, "Enhancements of v2x communication in support of cooperative autonomous driving," *IEEE communications magazine*, vol. 53, no. 12, pp. 64–70, 2015.
4. S. Liu, L. Liu, J. Tang, B. Yu, Y. Wang, and W. Shi, "Edge computing for autonomous driving: Opportunities and challenges," *Proceedings of the IEEE*, vol. 107, no. 8, pp. 1697–1716, 2019.
5. S. Liu, B. Yu, J. Tang, and Q. Zhu, "Invited: Towards fully intelligent transportation through infrastructure-vehicle cooperative autonomous driving: Challenges and opportunities," in *Proceedings of the 58th Design Automation Conference*, 2021.
6. S. Liu, J. Tang, C. Wang, Q. Wang, and J.-L. Gaudiot, "A unified cloud platform for autonomous driving," *Computer*, vol. 50, no. 12, pp. 42–49, 2017.
7. J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 779–788.

8. A. H. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang, and O. Beijbom, "Pointpillars: Fast encoders for object detection from point clouds," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 12 697–12 705.
9. J. Zhang and S. Singh, "Loam: Lidar odometry and mapping in real-time." in *Robotics: Science and Systems*, vol. 2, no. 9, 2014.
10. D. Li, J. Tang, and S. Liu, "Brief industry paper: An edge-based high-definition map crowdsourcing task distribution framework for autonomous driving," in *2021 IEEE 27th Real-Time and Embedded Technology and Applications Symposium (RTAS)*. IEEE, 2021, pp. 453–456.
11. "3GPP R16 (accessed on August 2021)." [Online]. Available: https://www.3gpp.org/release-16
12. Y. Feng, S. Liu, and Y. Zhu, "Real-time spatio-temporal lidar point cloud compression," in *2020 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE, 2020, pp. 10 766–10 773.
13. D. E. Knuth, "Dynamic huffman coding," *Journal of algorithms*, vol. 6, no. 2, pp. 163–180, 1985.
14. S. Lasserre, D. Flynn, and S. Qu, "Using neighbouring nodes for the compression of octrees representing the geometry of point clouds," in *Proceedings of the 10th ACM Multimedia Systems Conference*, 2019.
15. Y. Feng, P. Hansen, P. N. Whatmough, G. Lu, and Y. Zhu, "Fast and accurate: Video enhancement using sparse depth," *arXiv preprint arXiv:2103.08764*, 2021.