Resource Scheduling Strategies for Cooperative Perception and Computing in the Internet of Vehicles

Zheng Xue

Department of Communication Engineering

Guangdong University of Technology

9 November 2025

廣東工業大學
Guangdong University of Technology

Outline

Background

C-V2X

- C-V2X utilizes existing cellular networks and the 5.9GHz frequency band, which is also used by the DSRC standard, allowing for comprehensive vehicle-to-network (V2N) communication alongside vehicle-to-vehicle (V2V), vehicle-to-infrastructure (V2I), and vehicle-to-pedestrian (V2P) communications.
- C-V2X employs both direct (PC5/sidelink) and indirect (Uu/network) communication methods.

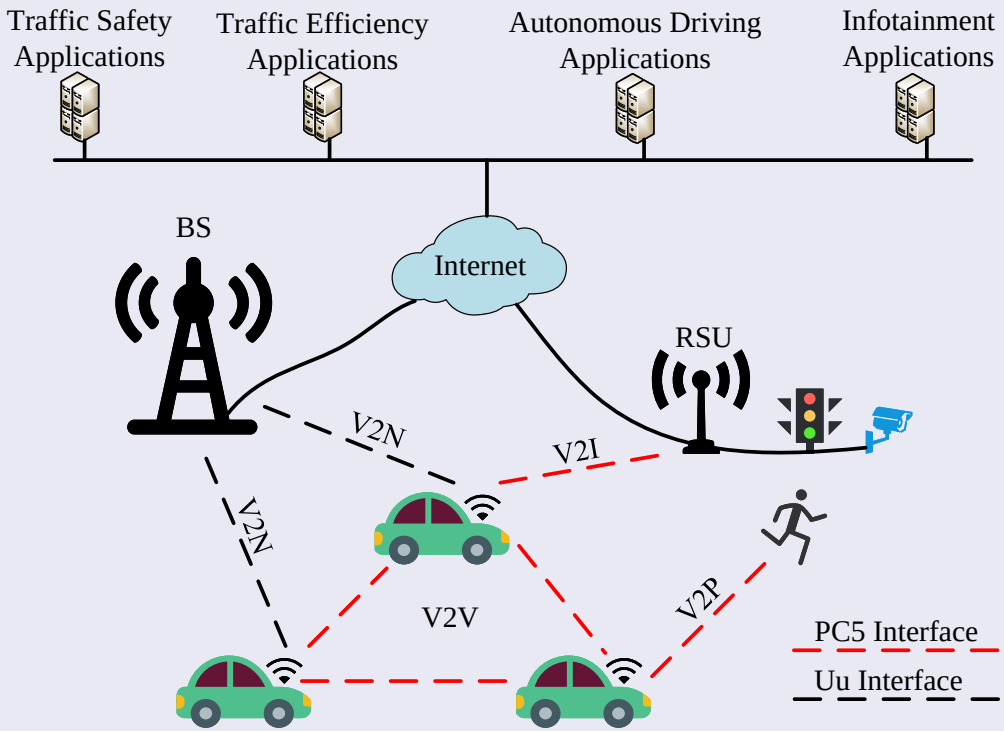


Figure 1: Types of V2X communication

C-V2X Evolution

- C-V2X, standardized in 2017, is defined by 3GPP standards (LTE-V2X: Release 14, 15; NR-V2X: Release 16, 17, 18, 19) and continues to evolve. Rel-19 is nearing its freeze point.

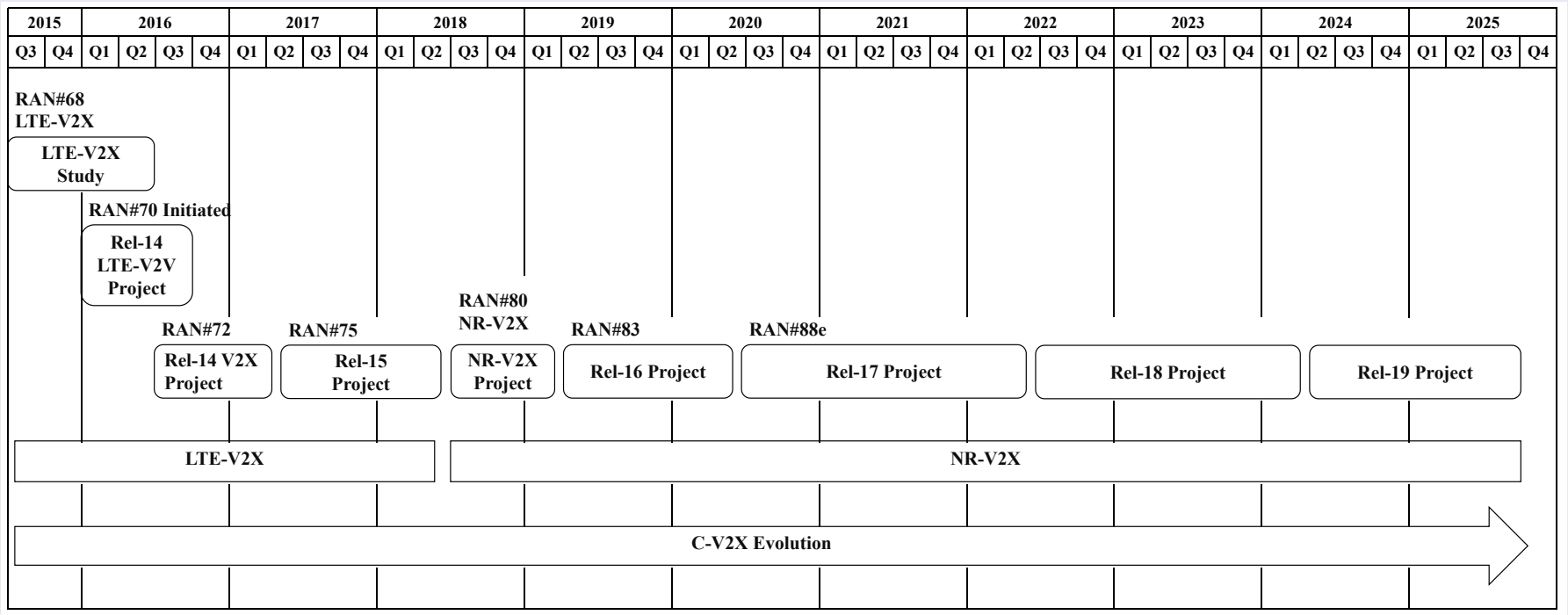


Figure 2: 3GPP C-V2X standard evolution timeline

Background

C-V2X-Based Cooperative Perception

- Green/red boxes denote ground truths and predictions; yellow/blue ellipses mark occluded and distant regions, addressing occlusion and long-range perception.

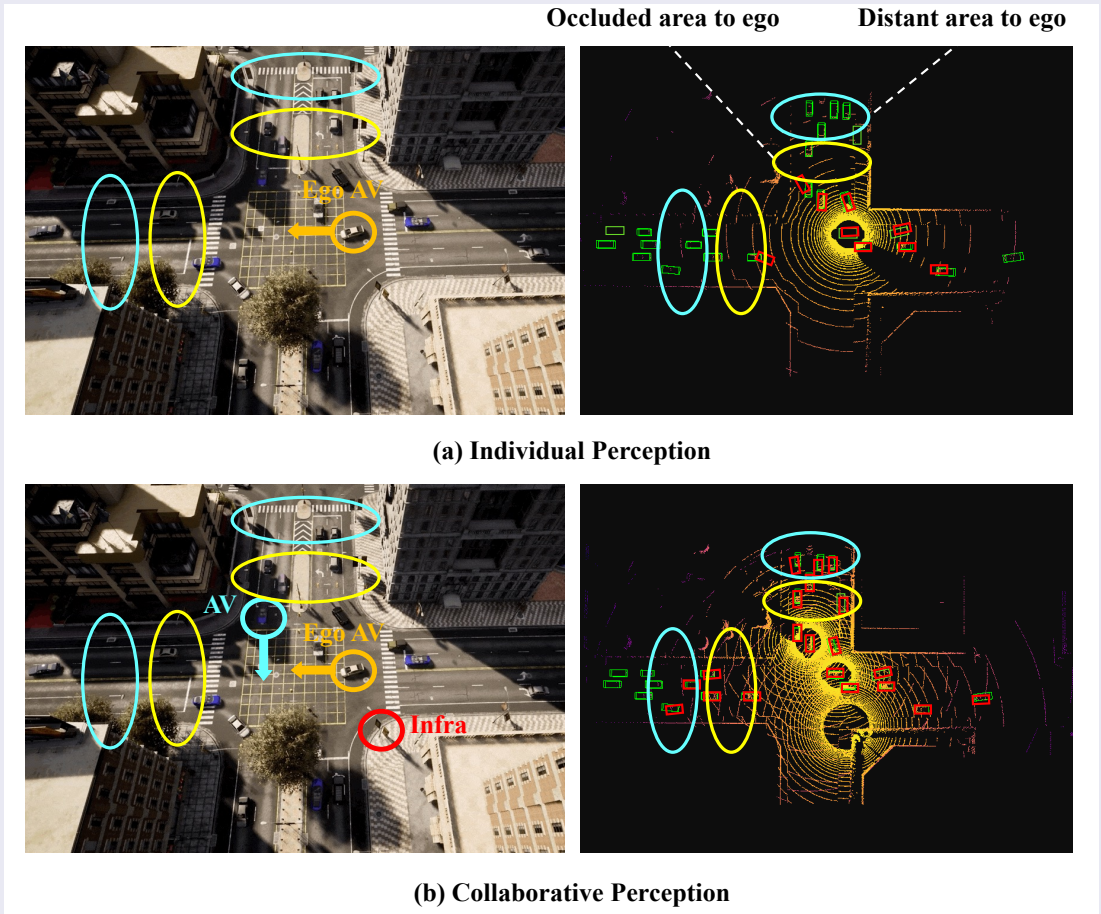


Figure 3: An example of (a) individual perception and (b) collaborative perception in autonomous driving.

Background

Cooperative Perception

- (a) shows individual perception without collaboration.
- (b-d) illustrate three cooperative perception paradigms: early (raw data fusion), intermediate (feature fusion), and late (result fusion).

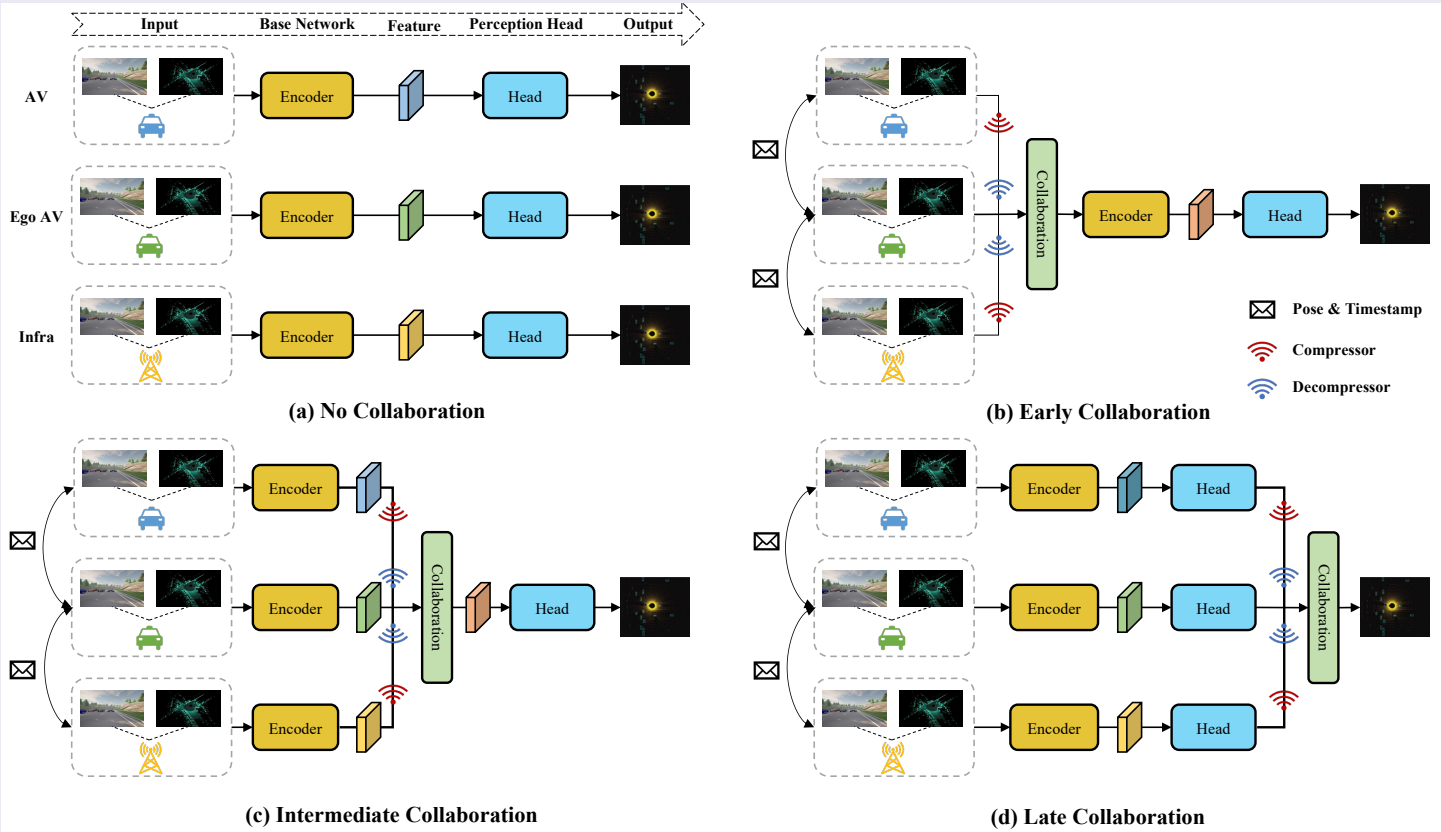


Figure 4: The collaboration scheme in the collaborative perception system.

Joint Service Caching and Computation Offloading Scheme

System Model

- A generic vehicular network scenario consisting of moving vehicles, edge pools, and the cloud, where edge servers are interconnected and vehicles dynamically request task computation.
- Different service program caches dictate task offloading to specific computing nodes, categorized into six types.

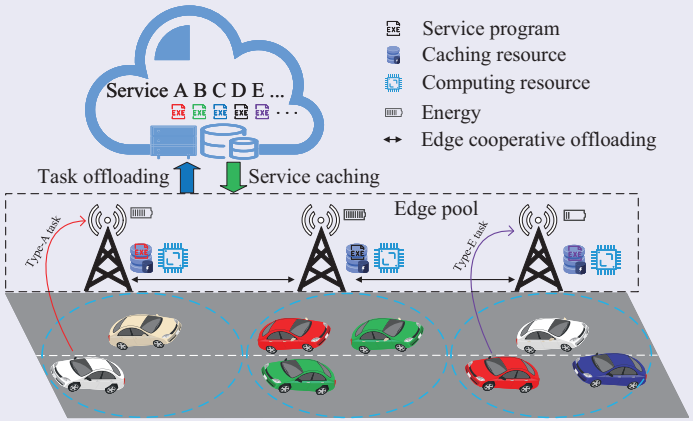


Figure 5: System illustration.

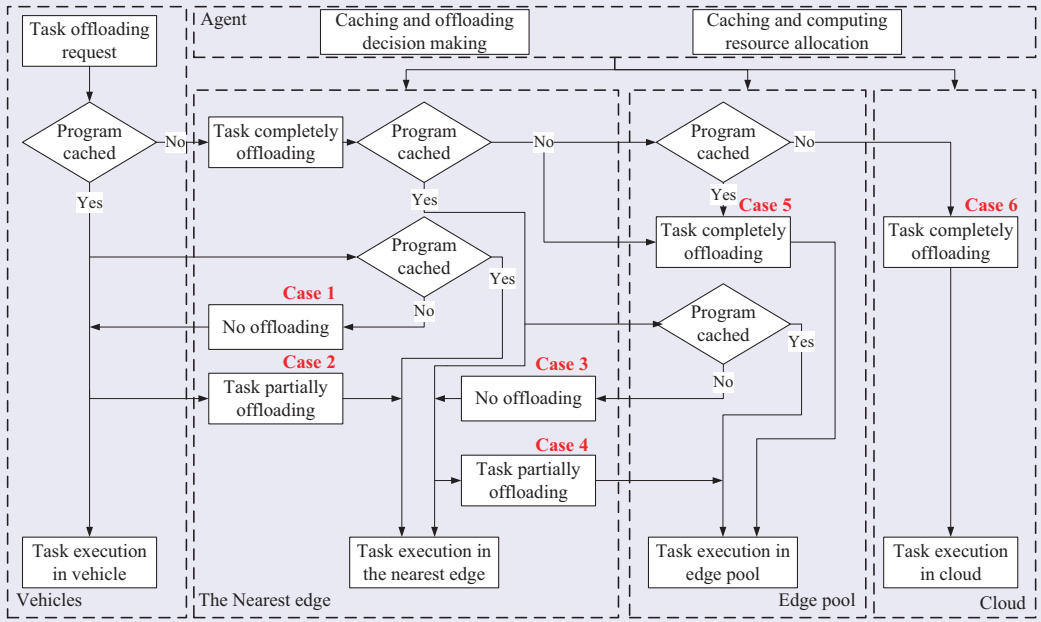


Figure 6: Flowchart of task offloading

Joint Service Caching and Computation Offloading Scheme

Service Caching and Task Offloading Model

- The caching cases correspond to the examples in the task offloading flowchart.
- Cases 1, 3, 5, and 6 do not require offloading ratio decisions.
- Cases 2 and 4 require offloading ratio decisions.
- The agent needs to make reasonable caching and offloading decisions.

Cases for caching		Task offloading ratio			
		Task execution in vehicle	Task execution in the nearest edge	Task execution in edge pool	Task execution in cloud
Case 1	$c^V_{v,k}(t)=1, c^E_{e,k}(t)=0, \sum_{i=1,i\neq e}^{N_E} c^E_{i,k}(t)\geq 0$	$1-o^V_{v,k}(t)=1$	$o^V_{v,k}(t)=0$	0	0
Case 2	$c^V_{v,k}(t)=1, c^E_{e,k}(t)=1, \sum_{i=1,i\neq e}^{N_E} c^E_{i,k}(t)\geq 0$	$1-o^V_{v,k}(t)$	$o^V_{v,k}(t)$	0	0
Case 3	$c^V_{v,k}(t)=0, c^E_{e,k}(t)=1, \sum_{i=1,i\neq e}^{N_E} c^E_{i,k}(t)=0$	$1-o^V_{v,k}(t)=0$	$o^V_{v,k}(t)(1-o^E_{e,k}(t))=1$	$o^V_{v,k}(t)o^E_{e,k}(t)=0$	0
Case 4	$c^V_{v,k}(t)=0, c^E_{e,k}(t)=1, \sum_{i=1,i\neq e}^{N_E} c^E_{i,k}(t)>0$	$1-o^V_{v,k}(t)=0$	$o^V_{v,k}(t)(1-o^E_{e,k}(t))$	$o^V_{v,k}(t)o^E_{e,k}(t)$	0
Case 5	$c^V_{v,k}(t)=0, c^E_{e,k}(t)=0, \sum_{i=1,i\neq e}^{N_E} c^E_{i,k}(t)>0$	$1-o^V_{v,k}(t)=0$	$o^V_{v,k}(t)(1-o^E_{e,k}(t))=0$	$o^V_{v,k}(t)o^E_{e,k}(t)=1$	0
Case 6	$c^V_{v,k}(t)=0, c^E_{i,k}(t)=0, \sum_{i=1,i\neq e}^{N_E} c^E_{e,k}(t)=0$	0	0	0	1

Figure 7: Task offloading ratio of different caching cases

Computation Delay

- Local task execution delay

$$T_{v,e,k}^{local}(t) = c_{v,k}^{V}(t)(1 - o_{v,k}^{V}(t))\frac{\lambda d_k}{f^V}.$$

- Delay of task computation at edge nodes

$$T_{v,e,k}^{edge}(t) = c_{e,k}^{E}(t)o_{v,k}^{V}(t)(1 - o_{e,k}^{E}(t))\frac{\lambda d_k}{f^E}.$$

- Transmission delay for uploading tasks to the edge pool

$$T_{v,e,k}^{uppool}(t) = (1 - c_{v,k}^{V}(t))\varphi(\sum_{i=1,i\neq e}^{N_E} c_{e,k}^{E}(t))o_{v,k}^{V}(t)o_{e,k}^{E}(t)\frac{d_k}{R_{edge}}.$$

- Processing delay of task k requested by vehicle v within the range of edge node e during time slot t:

$$T_{v,e,k}^{total}(t) = max\{T_{v,e,k}^{local}(t), T_{v,e,k}^{up}(t) + max\{T_{v,e,k}^{edge}(t), T_{v,e,k}^{uppool}(t) + T_{v,e,k}^{pool}(t)\}\}$$
$$+ (1 - c_{v,k}^{V}(t))(1 - \varphi(\sum_{e=1}^{N_E} c_{e,k}^{E}(t)))(\frac{d_k}{R_{v,e}(t)} + \frac{d_k}{R_{cloud}}),$$

- Average task processing delay during time slot t:

$$T_d(t) = \frac{1}{N_E}\sum_{e=1}^{N_E}\frac{1}{N_e^{task}(t)}\sum_{v=1}^{N_e^{task}(t)} T_{v,e,k}^{total}(t).$$

Joint Service Caching and Computation Offloading Scheme

Problem Formulation

- The main goal of our work is to develop a joint computation offloading and service caching scheme aimed at minimizing long-term average task processing time.
- This is a long-term MINLP problem and NP-hard.

$$\min_{\{c^V_{v,k}(t), c^E_{e,k}(t), o^V_{v,k}(t), o^E_{e,k}(t)\}} \quad \sum_{t=1}^{t_{end}} \gamma^{t-1} \left(\frac{T_d(t)}{T_{dmax}} \right) \tag{1}$$

$$s.t.$$

$$c^V_{v,k}(t) \in \{0,1\}, \forall v \in V, \forall k \in K, \forall t \in \{1,2,\ldots,t_{end}\}; \tag{2}$$

$$c^E_{e,k}(t) \in \{0,1\}, \forall e \in E, k, t; \tag{3}$$

$$0 \leq o^V_{v,k}(t) \leq 1, \forall v, k, t; \tag{4}$$

$$0 \leq o^E_{e,k}(t) \leq 1, \forall e, k, t; \tag{5}$$

$$0 \leq \gamma \leq 1; \tag{6}$$

$$\sum_{k=1}^{N_K} c^V_{v,k}(t)\theta_k \leq S^V_v, \forall v, t; \tag{7}$$

$$\sum_{k=1}^{N_K} c^E_{e,k}(t)\theta_k \leq S^E_e, \forall e, t. \tag{8}$$

Joint Service Caching and Computation Offloading Scheme

Problem Formulation Based on DRL

- DRL is employed to learn vehicle demands and resource states, making dynamic offloading and caching decisions in a discrete-time Markov Decision Process (MDP).
- **State Space:** Each edge node collects vehicular network information at time slot t, forming the global state:

$$\mathrm{s}_e(t) = \{[\mathrm{I}(t)]^{\rho_{max} \times N_K}, [\mathrm{c}^V(t)]^{\rho_{max} \times N_K}, [\mathbb{B}(t)]^{\rho_{max} \times N_K}, [\boldsymbol{\gamma}(t)]^{\rho_{max} \times N_K}, [\mathrm{c}^E(t)]^{\rho_{max} \times N_K}\},$$

$$s_t = \{s_1(t), s_2(t), ..., s_{N_E}(t)\}.$$

- **Action Space:** The agent determines task offloading ratios and service caching decisions:

$$\mathrm{a}_e(t) = \{[\mathrm{c}^E(t)]^{\rho_{max} \times N_K}, [\mathrm{o}^V(t)]^{\rho_{max} \times N_K}, [\mathrm{o}^E(t)]^{\rho_{max} \times N_K}\}.$$

$$a_t = \{a_1(t), a_2(t), ..., a_{N_E}(t)\}.$$

- **Reward Function:** The agent receives feedback based on task latency:

$$r_t = r(s_t, a_t) = -(\frac{T_d(t)}{T_{dmax}}).$$

Joint Service Caching and Computation Offloading Scheme

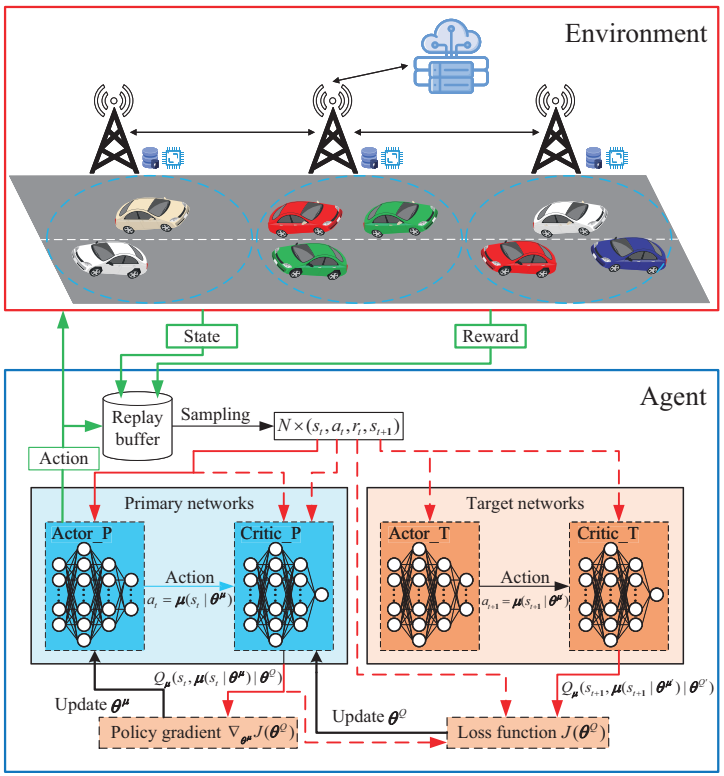


Figure 8: Framework of DDPG algorithm

Deep Deterministic Policy Gradient Algorithm (DDPG)

- Primary networks, target networks, replay buffer.
- Primary networks and target networks both consist of actor and critic networks.
- Loss function of the primary critic network:

$$J(\theta^{Q}) = \frac{1}{N}\sum_{j=1}^{N}(y_j - Q_{\mu}(s_j, \mu(s_j|\theta^{\mu})|\theta^{Q}))^2.$$

- Policy gradient of the primary actor network:

$$\nabla_{\theta^{\mu}} J(\theta^{Q}) = \frac{1}{N}\sum_{j=1}^{N}[\nabla_a Q(s, a|\theta^{Q})|_{s=s_j, a=\mu(s_j|\theta^{\mu})} \times \nabla_{\theta^{\mu}} \mu(s|\theta^{\mu})|_{s=s_j}].$$

- Target network parameter updates:

$$\theta^{Q'} \leftarrow \tau\theta^{Q} + (1-\tau)\theta^{Q'},$$
$$\theta^{\mu'} \leftarrow \tau\theta^{\mu} + (1-\tau)\theta^{\mu'},$$

Simulation Results

- As vehicle density increases and more tasks are requested, the total processing delay increases. The DDPG scheme achieves better performance.
- The agent can effectively avoid making caching decisions that result in maximum processing delays, so the DDPG scheme can reduce task processing delays.

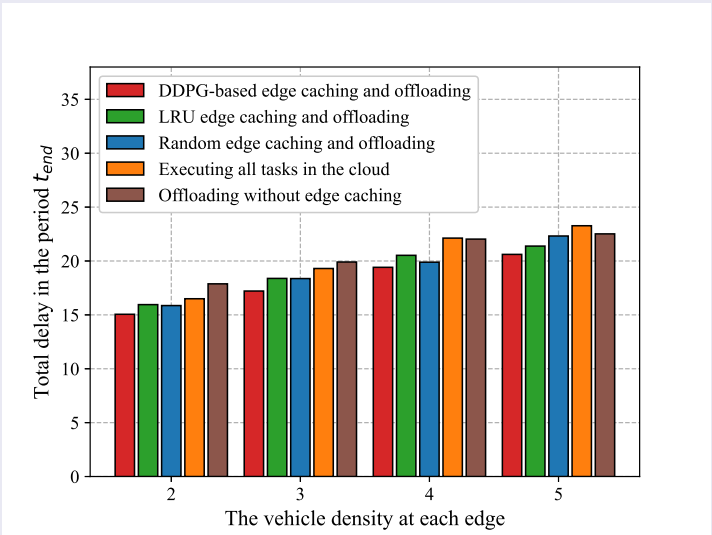


Figure 9: The effect of the vehicle density ρ on the total task processing delay.

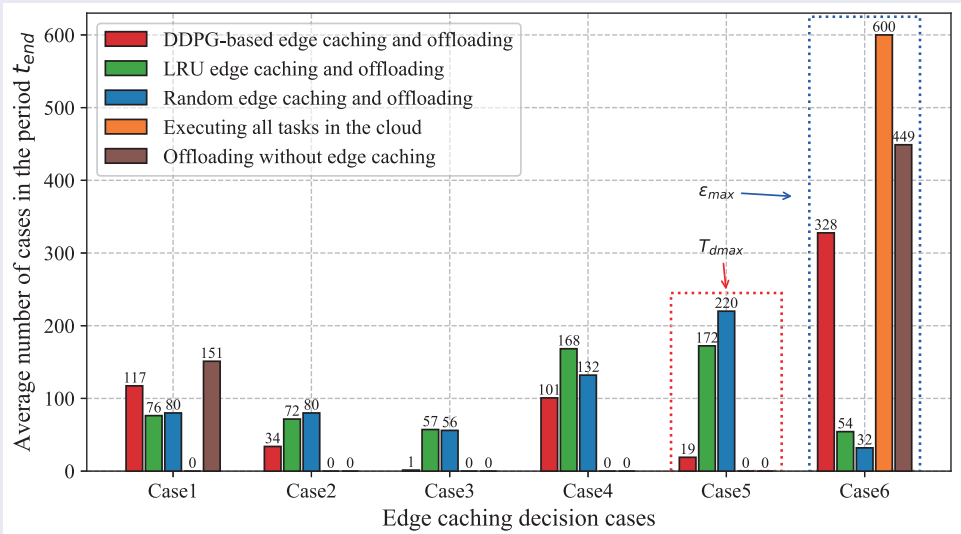


Figure 10: The average number of edge caching decision cases in the period t_{end} with $\rho = 5$.

Simulation Results

- As the size of task inputs increases, and assuming constant computing capabilities for each device, the cumulative task processing delay increases.
- As the size of task inputs increases, task transmission time and energy consumption increase, leading to higher cumulative energy consumption.
- As edge nodes' computing frequencies increase, their processing capabilities improve, leading to reduced cumulative task processing delays, except in cloud execution and no edge caching scenarios where delays remain unchanged due to lack of edge computations.

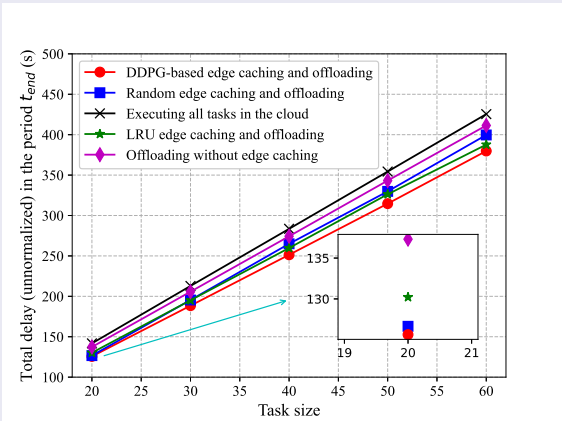


Figure 11: The total delay versus task size with $\rho = 5$.

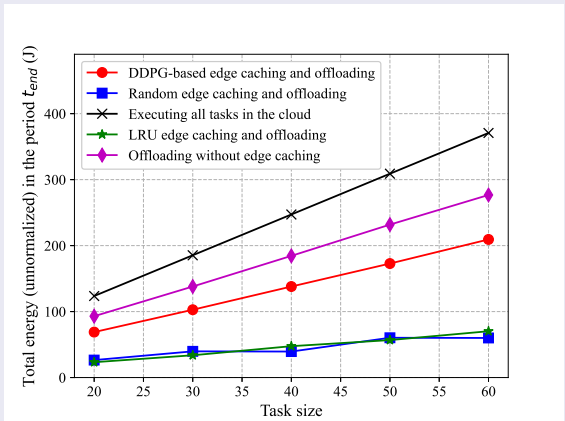


Figure 12: The total energy versus task size with $\rho = 5$.

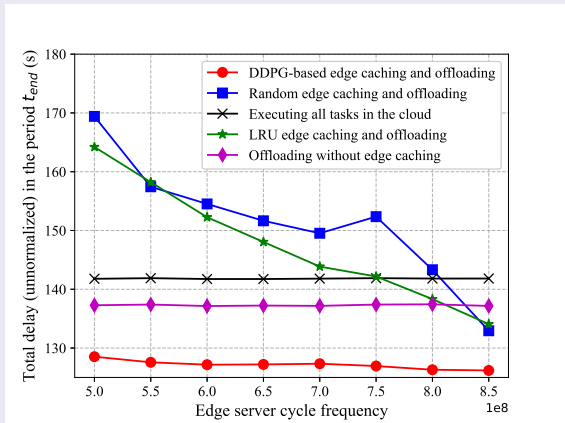


Figure 13: The total delay versus edge sever cycle frequency with $\rho = 5$.

Cooperative Perception Task Offloading Scheme

System Model

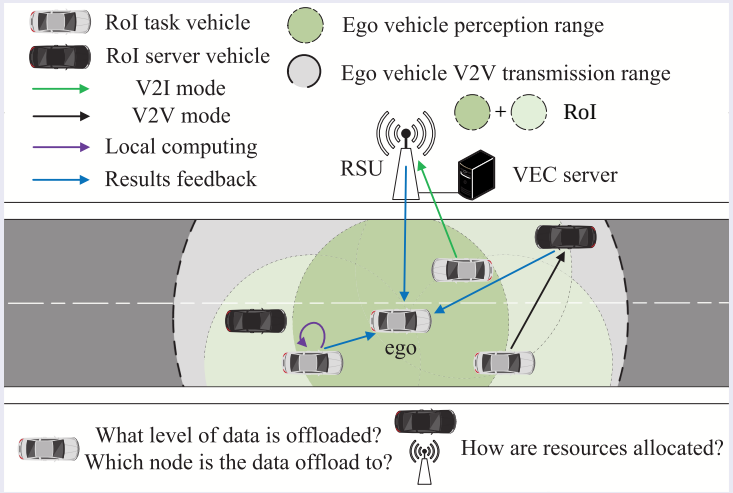


Figure 14: An illustration of the VEC system model.

- RoI task vehicle

$$\Omega_v^{RoI} = \{v_i; \forall v_i \in C_{ego}^{sensor} \cap C_{ego}^{V2V} \cup v_{ego}\}.$$

- RoI server vehicle

$$\Omega_{server}^{RoI} = \{v_j; \forall v_j \notin \Omega_v^{RoI} \cap C_{v_i}^{V2V} \cap C_{ego}^{V2V}, v_i \in \Omega_v^{RoI}\} \cup \{r\},$$

System Overview

- Vehicles in the RoI set collect raw sensor data I_{raw}, process it for initial feature extraction to get I_{fea}, and then use a detection model to produce object data I_{obj}.
- Computational resources are provided by RoI task vehicles (f_{v_i}) and corresponding service nodes (f_{s_j}).
- Offloading Levels (x_i): Defines whether tasks are processed fully locally, partially, or fully offloaded.
- Offloading Indicator (o_{ij}): Indicates if a task is offloaded to a specific server node.
- Resource Allocation (μ_{ij}): Indicates the percentage of server resources allocated to tasks.

Cooperative Perception Task Offloading Scheme

Computation Offloading Model

- The overall delay for different x_i:

$$D_{v_i}(x_i=0)=\begin{cases}\frac{I_{raw}\lambda_{v_i}^{raw}+I_{fea}\lambda_{v_i}^{fea}}{f_{v_i}}, & v_i=v_{ego}\\ \frac{I_{raw}\lambda_{v_i}^{raw}+I_{fea}\lambda_{v_i}^{fea}}{f_{v_i}}+\frac{I_{obj}}{R_{v_i\to ego}}, & v_i\in\Omega_v^{RoI}, v_i\neq v_{ego}\end{cases}$$

$$D_{v_i}(x_i=1,o_{ij},\mu_{ij})=\frac{I_{raw}\lambda_{v_i}^{raw}}{f_{v_i}}+\sum_{j=1}^{|\Omega_{server}^{RoI}|}o_{i,j}\left(\frac{I_{fea}}{R_{v_i\to s_j}}+\frac{I_{fea}\lambda_{v_i}^{fea}}{\mu_{ij}f_{s_j}}+\frac{I_{obj}}{R_{s_j\to ego}}\right),$$

$$D_{v_i}(x_i=2,o_{ij},\mu_{ij})=\frac{I_{raw}}{R_{v_i\to s_j}}+\sum_{j=1}^{|\Omega_{server}^{RoI}|}o_{i,j}\left(\frac{I_{raw}\lambda_{v_i}^{raw}+I_{fea}\lambda_{v_i}^{fea}}{\mu_{ij}f_{s_j}}+\frac{I_{obj}}{R_{s_j\to ego}}\right),$$

- The transmission rate between vehicle v_i and node s_j:

$$R_{v_i\to s_j}=Blog_2\left(1+\frac{p|h_{ij}|^2\left(\sqrt{d_{ij}^2+(H_j-h)^2}\right)^{-\alpha}}{\sigma_j^2+\sum_{v_k\in C_{s_j}^{V2I/V2V}\setminus v_i}ph_{kj}}\right),$$

- The completion latency of RoI tasks for the ego vehicle:

$$D_{ego}^{RoI}=max\{D_{v_i}\}, v_i\in\Omega_v^{RoI}, s_j\in\Omega_{server}^{RoI}.$$

Cooperative Perception Task Offloading Scheme

Problem Formulation

- Our optimization problem aims to minimize the delay of RoI tasks while considering low-delay and resource constraints.
- This is a mixed-integer nonlinear programming (MINLP) problem.

$$\underset{\{x_i, o_{ij}, \mu_{ij}\}}{\text{minimize}} \; D_{ego}^{RoI} \tag{9}$$

$$\text{subject to } D_{v_i}(x_i, o_{ij}, \mu_{ij}) \leq min\{\tau_i^{tole}, \tau_{j,ego}^{hold}\}, \tag{10}$$

$$\sum_{j=1}^{|\Omega_{server}^{RoI}|} o_{i,j} = 1, v_i \in \Omega_v^{RoI}, s_j \in \Omega_{server}^{RoI}, \tag{11}$$

$$\sum_{i=1}^{|\Omega_v^{RoI}|} \mu_{ij} \leq 1, v_i \in \Omega_v^{RoI}, s_j \in \Omega_{server}^{RoI}, \tag{12}$$

$$x_i \in \{0, 1, 2\}, \tag{13}$$

$$o_{ij} \in \{0, 1\}, \tag{14}$$

$$\mu_{ij} \in [0, 1]. \tag{15}$$

Cooperative Perception Task Offloading Scheme

Algorithm 1 Optimal Task Offloading and Resource Allocation

Input: The initial parameters: Ω_v^{RoI}, Ω_{server}^{RoI}, f_{v_i}, f_{r_j}, R_{V2V},R_{V2I}, τ_i^{tole}.

Output: Task offloading decision $\mathbf{X}^*$, $\mathbf{O}^*$ and resource allocation $\mathbf{U}^*$.

1: **/*Exhaustive algorithm*/**
2: Exhaust all possible solutions for $\mathbf{X}, \mathbf{O}$ with complexity $M = (1 + 2\left|\Omega_{server}^{RoI}\right|)^{\left|\Omega_v^{RoI}\right|}$.
3: **for** $m = 1, 2, ...M$ **do**
4: **/*Feasibility check*/**
5: **for** $i = 1$ to $\left|\Omega_v^{RoI}\right|$ **do**
6: **for** $j = 1$ to $\left|\Omega_{server}^{RoI}\right|$ **do**
7: **if** $x_i = 0$ and $D_{v_i}(0) \leq \min\{\tau_0^{tole}, \tau_{j,ego}^{hold}\}$ **then**
8: continue
9: **else if** $x_i = 1, o_{ij} = 1$ and $\frac{I_{raw}\lambda_{v_i}^{raw}}{f_{v_i}} + \frac{I_{fea}}{R_{v_i \to s_j}} + \frac{I_{obj}}{R_{s_j \to ego}} < \min\{\tau_0^{tole}, \tau_{j,ego}^{hold}\}$ **then**
10: $\mu_{ij} = \frac{I_{fea}\lambda_{v_i}^{fea}}{f_{s_j}} / (\min\{\tau_0^{tole}, \tau_{j,ego}^{hold}\} - \frac{I_{raw}\lambda_{v_i}^{raw}}{f_{v_i}} - \frac{I_{fea}}{R_{v_i \to s_j}} - \frac{I_{obj}}{R_{s_j \to ego}})$
11: **else if** $x_i = 2, o_{ij} = 1$ and $\frac{I_{raw}}{R_{v_i \to s_j}} + \frac{I_{obj}}{R_{s_j \to ego}} < \min\{\tau_0^{tole}, \tau_{j,ego}^{hold}\}$ **then**
12: $\mu_{ij} = \frac{I_{raw}\lambda_{v_i}^{raw} + I_{fea}\lambda_{v_i}^{fea}}{f_{s_j}\left(\min\{\tau_0^{tole}, \tau_{j,ego}^{hold}\} - \frac{I_{raw}}{R_{v_i \to s_j}} - \frac{I_{obj}}{R_{s_j \to ego}}\right)}$
13: **else**
14: go to line 3 and proceed to the next iteration of the loop for m.
15: **end if**
16: **end for**
17: **end for**
18: **for** $j = 1$ to $\left|\Omega_{server}^{RoI}\right|$ **do**
19: **if** $\sum_{i=1}^{\left|\Omega_v^{RoI}\right|} \mu_{ij} \leq 1$ **then**
20: continue
21: **else**
22: go to line 3 and proceed to the next iteration of the loop for m.
23: **end if**
24: **end for**
25: **/*NLP solver*/**
26: Generate the initial matrix $\mathbf{U}_0$ based on $\mathbf{X}_m, \mathbf{O}_m$ that satisfies the feasibility check.
27: Define the inequality and equality constraints;
28: Find the optimal resource allocation $\mathbf{U}_m$ via NLP solver.
29: **end for**
30: Obtain the optimal decision $\mathbf{X}^*$, $\mathbf{O}^*$ and resource allocation $\mathbf{U}^*$.

Figure 15: Optimal task offloading and resource allocation

OTORA Scheme

- Task offloading decision X^*, O^* and resource allocation U^*.
- Exhaustive exploration of all combinations of discrete variables along with feasibility checks.
- Once variables are set, converts the MINLP problem into simpler NLP problems, leveraging known gradients to tackle non-convexity.

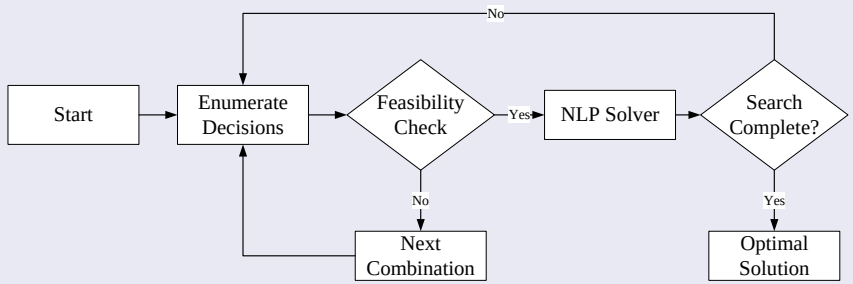
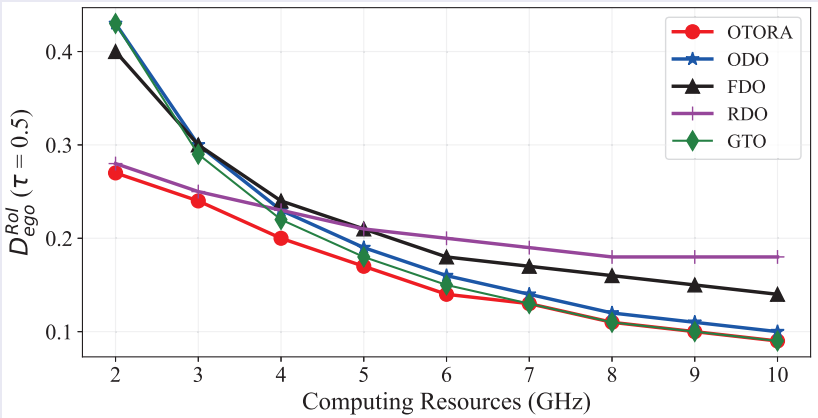


Figure 16: Optimal task offloading and resource allocation

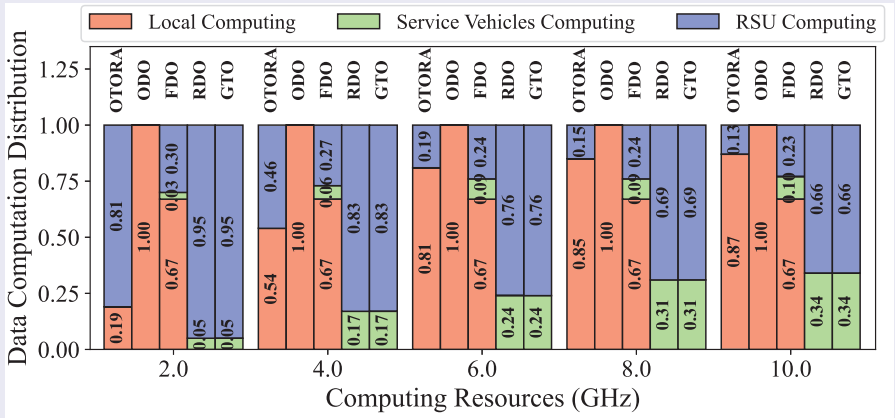
Cooperative Perception Task Offloading Scheme

Simulation Results

- As computational resources increase, latency decreases for all methods.
- When resources are scarce, the RDO scheme excels by avoiding local computations, while the ODO scheme performs best in resource-rich environments by processing all tasks locally.
- As computational resources increase, the OTORA scheme optimally maintains task completion latency by balancing data computation between local resources and RSUs.



(a)



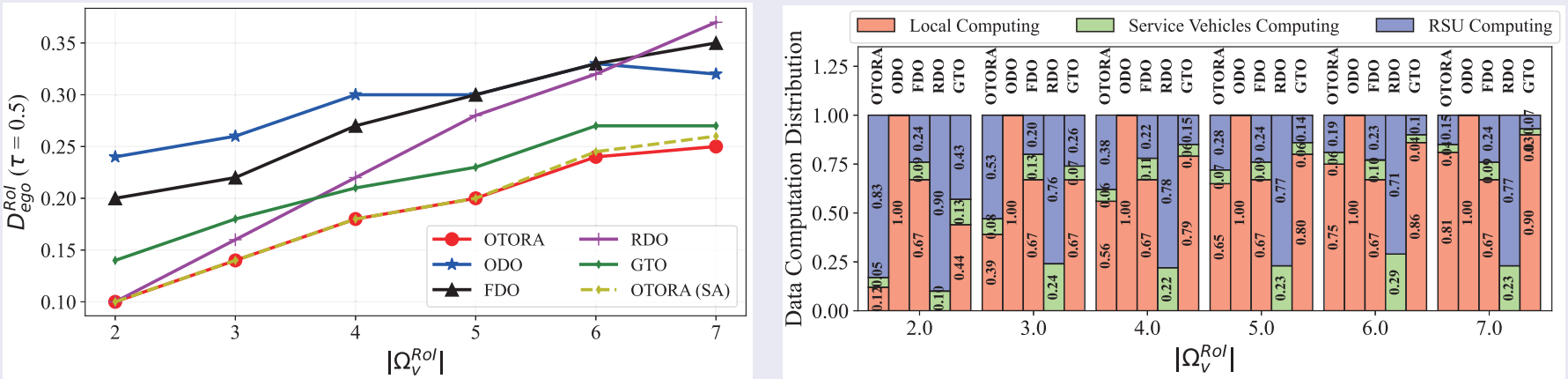
(b)

Figure 17: Impact of increased vehicle computing resources on (a) RoI task processing delay and (b) data computation distribution across nodes ($|\Omega_v^{RoI}| = 4, |\Omega_{server}^{RoI}| = 5$).

Cooperative Perception Task Offloading Scheme

Simulation Results

- As the number of task vehicles increases, resulting in higher computational demands, the resources of the service nodes remain unchanged, leading to an increase in processing delay.
- The flexibility of the OTORA scheme in dynamically arranging offloading based on task stages enables a balanced utilization of both demand and service node resources, thereby maintaining optimal performance even as demand increases.



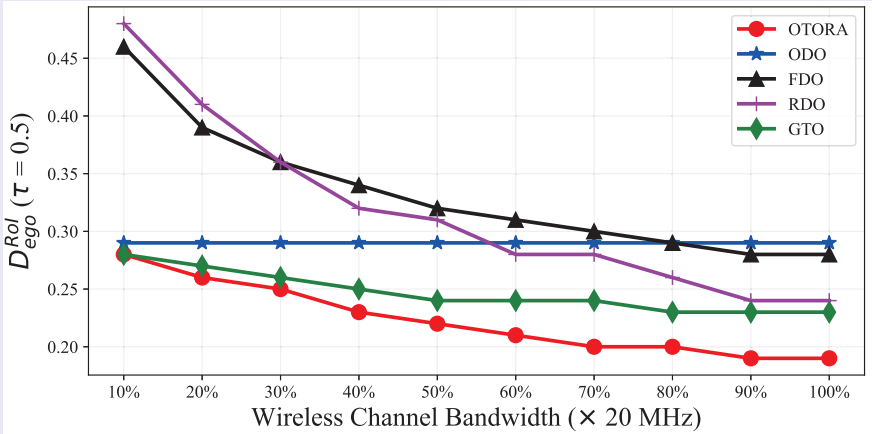
(a) (b)

Figure 18: Effect of increasing number of task vehicles on (a) RoI task processing delay and (b) data computation proportions among various nodes ($\left|\Omega_{server}^{RoI}\right| = 5$).

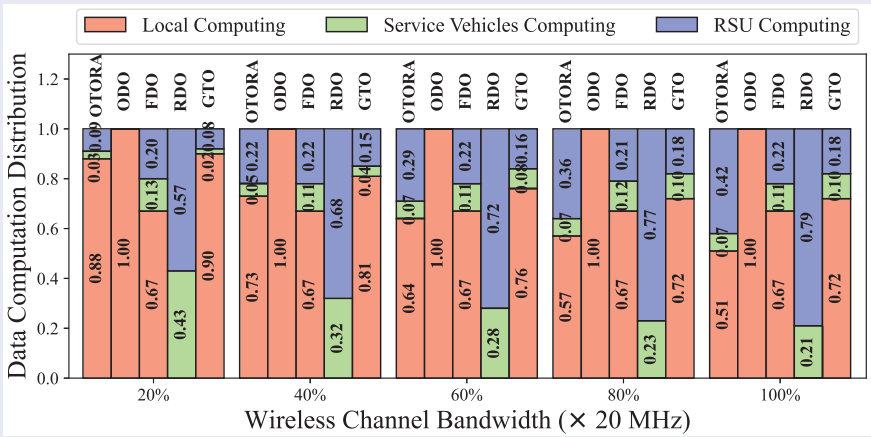
Cooperative Perception Task Offloading Scheme

Simulation Results

- As wireless channel bandwidth increases, processing delays decrease for all schemes except ODO, which does not utilize channel bandwidth.
- The GTO scheme takes into account the computational resources of the offloading nodes but does not comprehensively consider the data types involved.
- The OTORA scheme optimizes performance by judiciously distributing computational and communication resources between demand and service sides.



(a)



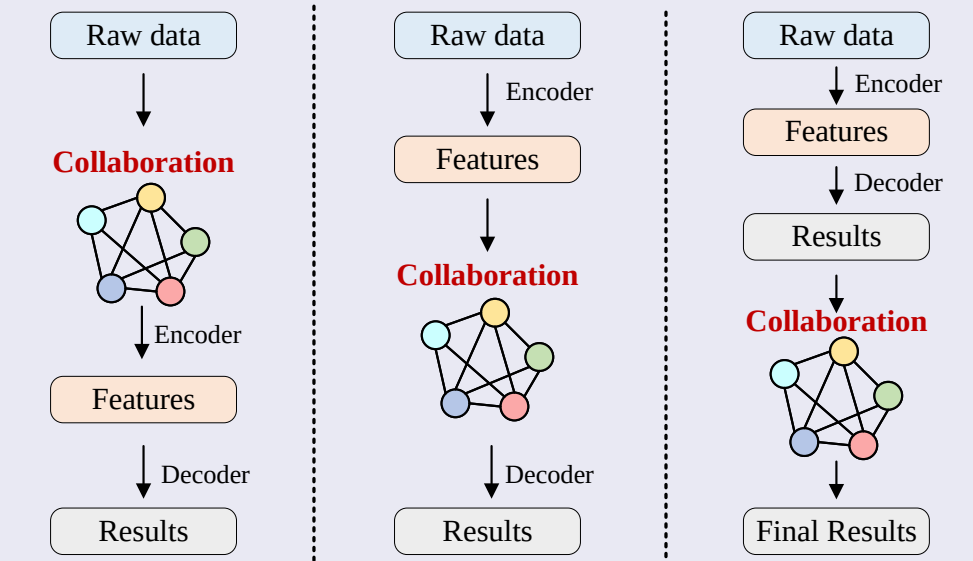
(b)

Figure 19: Impact of increased wireless bandwidth on (a) RoI task delay and (b) computation distribution across nodes ($|\Omega_v^{RoI}| = 4, |\Omega_{server}^{RoI}| = 5$).

Object-Level Cooperative Perception System

Cooperative Perception

- Raw data contains the most comprehensive information and substantial description of agents.
- Feature extraction often causes information loss and unnecessary information redundancy.
- Late collaboration (object-level) is more bandwidth-economic and simpler than early and intermediate collaboration.



(a) Early collaboration (b) Intermediate collaboration (c) Late collaboration

Figure 20: The collaboration scheme in the collaborative perception system.

Object-level Cooperative Perception System

Hardware-in-Loop Platform Framework

- LTE-V2X OBU device supports 3GPP Release 14 PC5 communication, operates between 5905-5925 MHz with 10 MHz/20 MHz bandwidth, and has a maximum transmission power of 23± 2 dBm. It includes a GNSS antenna for receiving satellite signals.
- NVIDIA's Jetson AGX Orin 32 GB has a computing power of up to 200 TOPS, with a maximum CPU frequency of 2.2 GHz and a maximum GPU frequency of 930 MHz.

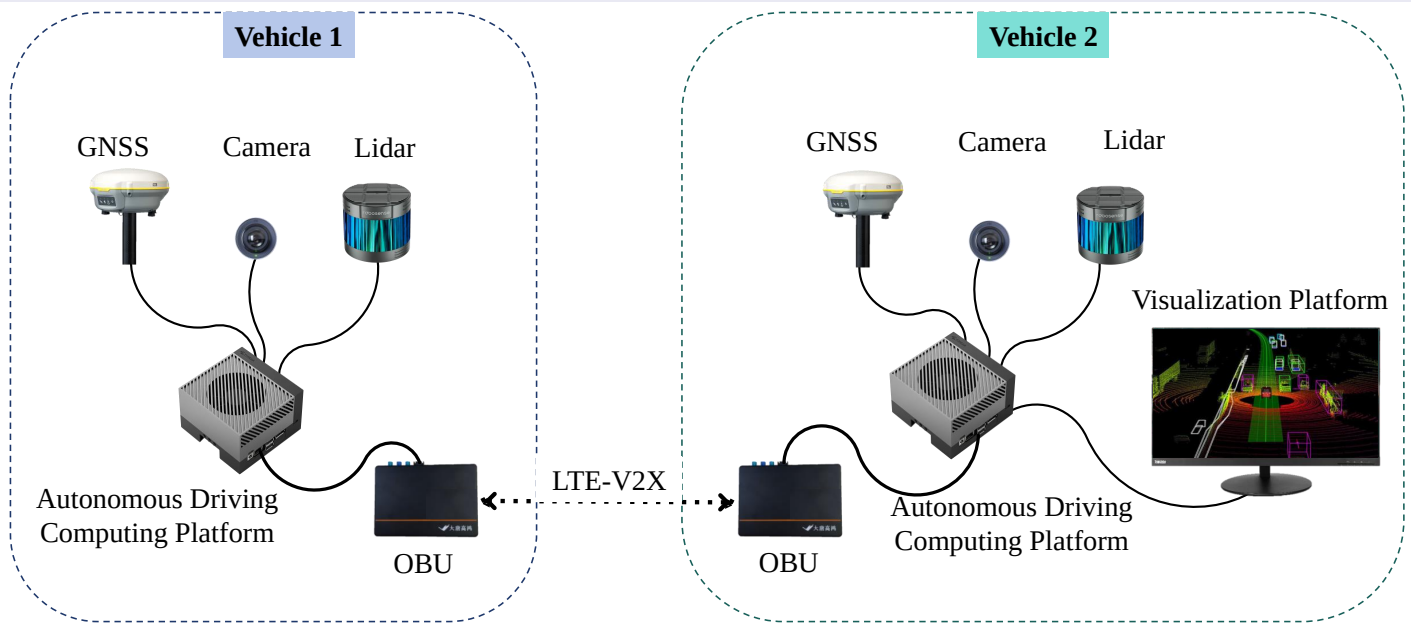


Figure 21: Hardware-in-loop testing platform.

ROS2 Sender Node Graph and Receiver Node Graph

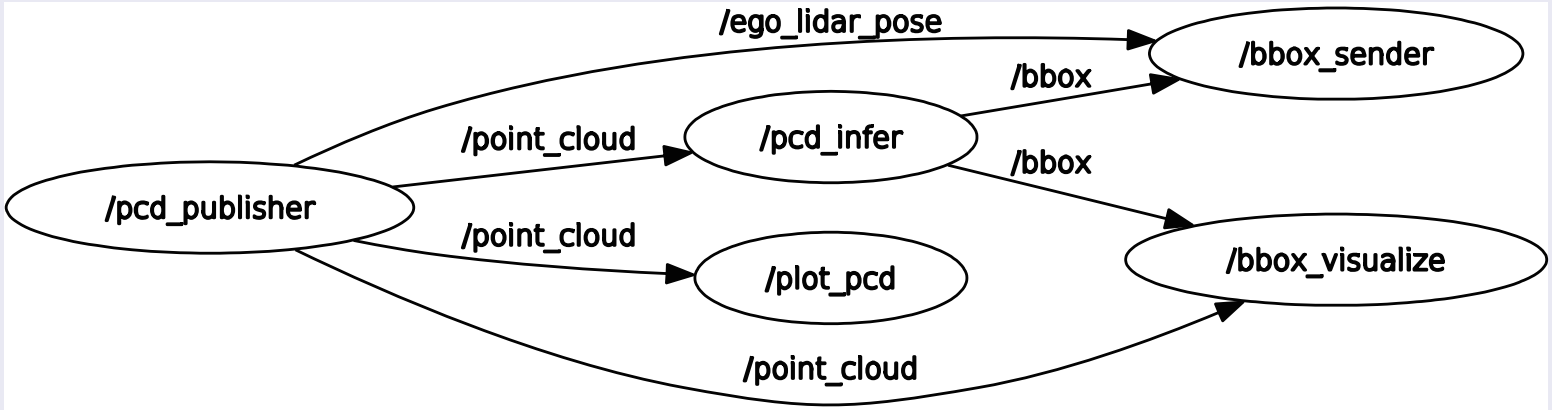


Figure 22: ROS2 sender node graph.

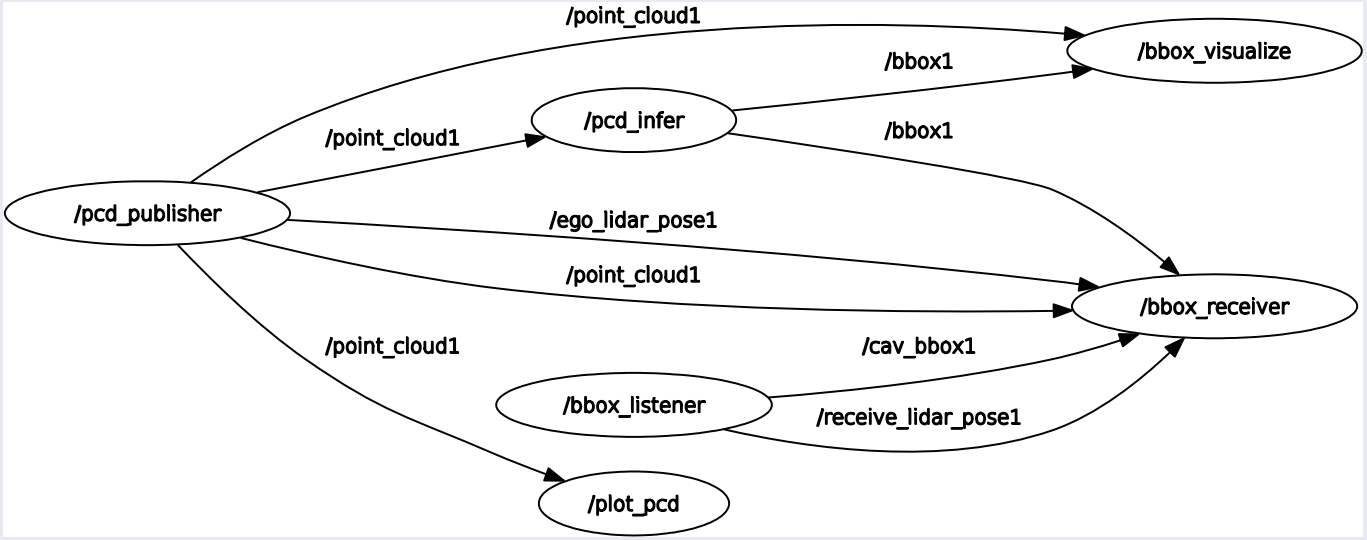


Figure 23: ROS2 receiver node graph.

Deployment Process for ROS2 Inference Nodes

- Train the PointPillars model using NVIDIA TAO Toolkit, generate an inference module, and encapsulate it as a ROS2 node.

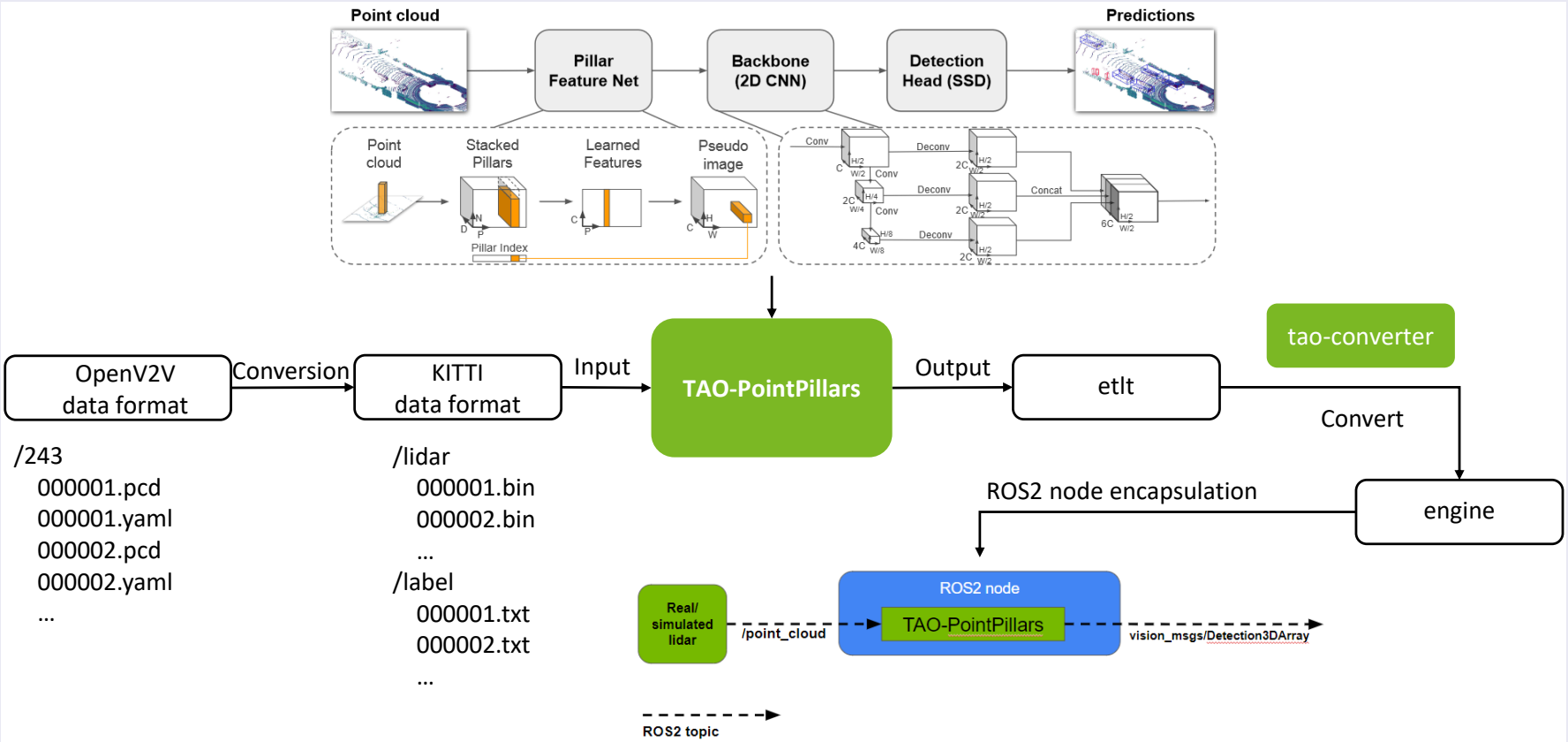


Figure 24: Deployment process for ROS2 inference nodes.

Object-Level Cooperative Perception System

Visual Interface for Object-Level Cooperative Perception

Figure 25: Visualization interface of the sender.

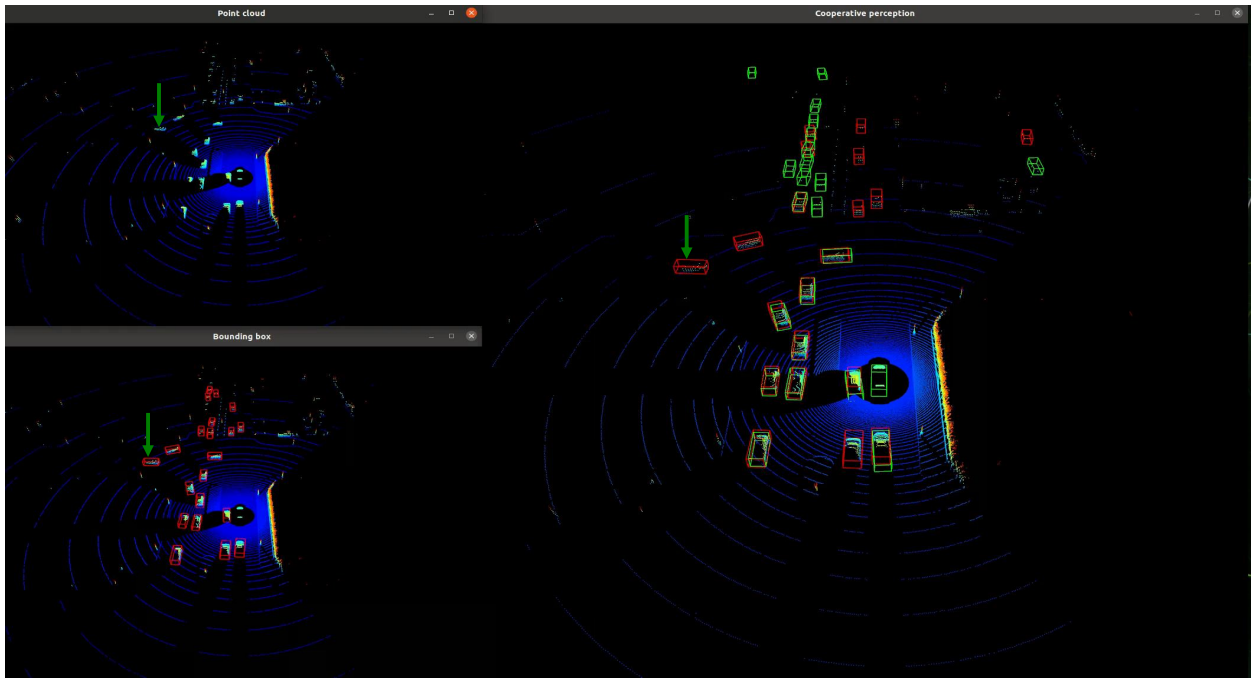


Figure 26: Visualization interface of the sender.

Object-Level Cooperative Perception System

Performance Evaluation

- The link latency of the cooperative perception system is crucial for ensuring real-time data exchange and decision-making among multiple vehicles.
- Test the latency of each module in the cooperative link.

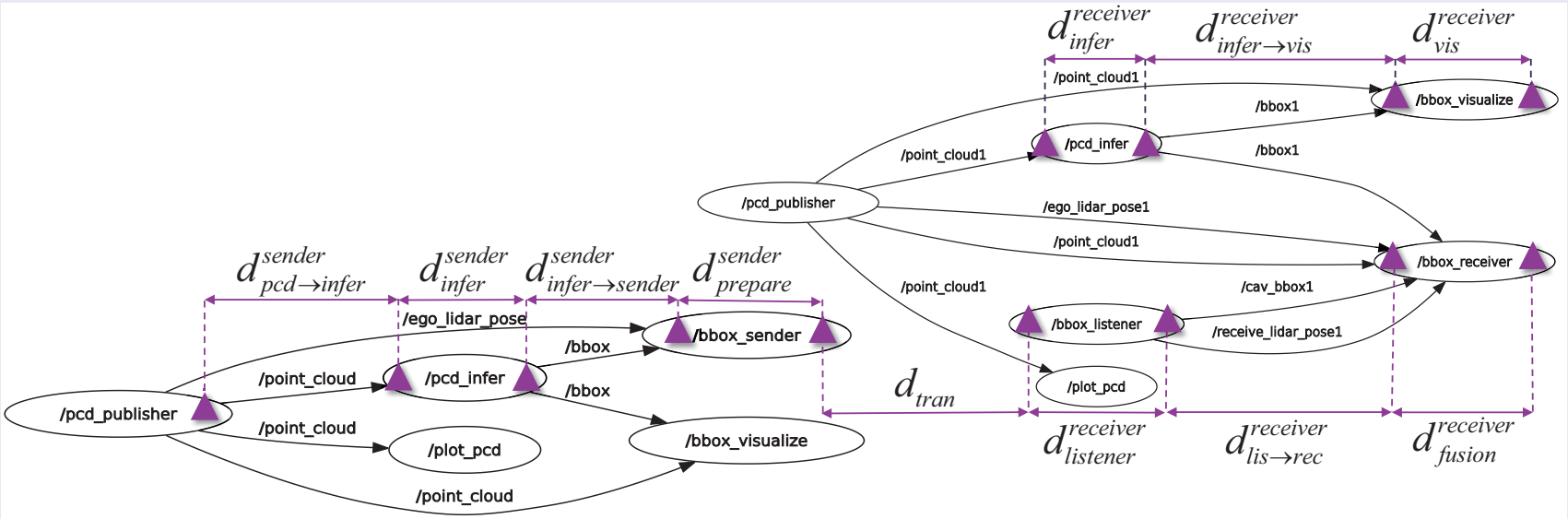


Figure 27: Illustration of link latency.

Object-Level Cooperative Perception System

Performance Evaluation

- The average detection latency of the receiver's inference module is about 110 ms, and the visualization latency is about 50 ms. The total average latency of the cooperative link is about 300 ms, with the inference module's average detection latency being about 110 ms.
- Object-level cooperative perception is shown as alerts on the human-machine interface, overcoming the perception limitations of human drivers and single vehicles.

Test data	Total packets	$S_{pcd}^{av}\ (MB)$	$S_{bbox}^{av}\ (B)$	$d_{infer}^{receiver}\ (ms)$	$d_{infer\rightarrow vis}^{receiver}(ms)$	$d_{vis}^{receiver}(ms)$
Test 1	2151	0.85	1053.37	112.56	34.70	50.01
Test 2	1981	0.85	1051.77	99.33	48.32	49.33
Test 3	4986	0.85	1052.34	122.74	23.11	49.60
Test 4	4242	0.85	1052.93	99.38	47.11	49.62
Test 5	4211	0.85	1052.35	105.30	40.43	49.25

Figure 28: Inference link latency of the receiver.

Test data	Packet loss rate	$d_{pcd\rightarrow infer}^{sender}$	d_{infer}^{sender}	$d_{infer\rightarrow sender}^{sender}$	$d_{prepare}^{sender}$	d_{tran}	$d_{listener}^{receiver}$	$d_{lis\rightarrow rec}^{receiver}$	$d_{fusion}^{receiver}$
Test 1	0.092%	5.66	108.39	18.97	20.29	18.02	4.06	8.23	99.72
Test 2	0	5.63	109.23	18.82	20.30	32.28	4.16	8.47	100.55
Test 3	0	5.75	109.02	18.75	20.28	25.87	3.95	18.48	101.84
Test 4	0.047%	5.83	102.71	18.28	20.25	26.19	4.18	8.65	99.85
Test 5	0.023%	5.75	108.79	18.91	20.29	35.71	3.99	8.31	101.73

Figure 29: Latency of cooperative perception links.

Concluding Remarks

Conclusions

In this talk, we presented resource scheduling strategies for cooperative perception and computing in the IoV.

- Proposed a dynamic service caching and computation offloading scheme to minimize task latency and improve resource utilization.
- Developed an optimal task offloading and resource allocation (OTORA) algorithm for multi-source cooperative perception tasks.
- Implemented a prototype system based on NVIDIA Orin and OBU platforms to validate the late-fusion-based cooperative perception framework.

This work demonstrates that efficient resource scheduling enables real-time cooperative perception and computation in IoV, providing a promising pathway for intelligent connected vehicle applications.

廣東工業大學
Guangdong University of Technology

References

[1] Z. Xue, F. Wen, C. Liu and G. Han, "Joint Optimization of Task Offloading and Resource Allocation for Cooperative Perception in Vehicular Edge Computing Systems" accepted, *IEEE Trans. Veh. Technol.* 2025.

[2] Z. Xue, C. Liu, C. Liao, G. Han and Z. Sheng, "Joint Service Caching and Computation Offloading Scheme Based on Deep Reinforcement Learning in Vehicular Edge Computing Systems," *IEEE Trans. Veh. Technol.*, vol. 72, no. 5, pp. 6709-6722, May 2023.

[3] Z. Song, T. Xie, H. Zhang, J. Liu, F. Wen and J. Li, "A Spatial Calibration Method for Robust Cooperative Perception," *IEEE Rob. Autom. Lett.*, vol. 9, no. 5, pp. 4011-4018, May 2024.

[4] C. Liu, Z. Xue, C. Liao, J. Kang and G. Han, "DRL-Enhanced Vehicular Edge Caching Addressing Content Dynamics and Complex Intersections," *IEEE Internet Things J.*, vol. 12, no. 2, pp. 1732-1745, Jan 2025.

[5] Z. Xue, Y. Liu, G. Han, F. Ayaz, Z. Sheng and Y. Wang, "Two-Layer Distributed Content Caching for Infotainment Applications in VANETs," *IEEE Internet Things J.*, vol. 9, no. 3, pp. 1696-1711, Feb 2022.

[6] S. Chen, J. Hu, Y. Shi, L. Zhao and W. Li, "A Vision of C-V2X: Technologies, Field Testing, and Challenges With Chinese Development," *IEEE Internet Things J.*, vol. 7, no. 5, pp. 3872-3881, May 2020.

[7] W. Fan et al., "Joint Task Offloading and Resource Allocation for Vehicular Edge Computing Based on V2I and V2V Modes," *IEEE Trans. Intell. Transp. Syst.*, vol. 24, no. 4, pp. 4277-4292, April 2023.

[8] Z. Xiao, J. Shu, H. Jiang, G. Min, H. Chen and Z. Han, "Perception Task Offloading With Collaborative Computation for Autonomous Driving," *IEEE J. Sel. Areas Commun.*, vol. 41, no. 2, pp. 457-473, Feb. 2023.

[9] P. Lv et al., "Edge Computing Task Offloading for Environmental Perception of Autonomous Vehicles in 6G Networks," *IEEE Trans. Network Sci. Eng.*, vol. 10, no. 3, pp. 1228-1245, 1 May-June 2023.

[10] R. Xu, H. Xiang, X. Xia, X. Han, J. Li and J. Ma, "OPV2V: An Open Benchmark Dataset and Fusion Pipeline for Perception with Vehicle-to-Vehicle Communication," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 2022, pp. 2583-2589.

Ending

Thanks for Your Attention!

Q & A

廣東工業大學
Guangdong University of Technology