Collaborative Vehicular Edge Computing Design for Delay-Sensitive Applications

Jing-Yang Voon, Yao Chiang, Cheng-Rui Jia, Hung-Yu Wei

Wireless Mobile Network Laboratory of National Taiwan University

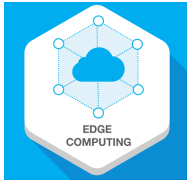
Department of Electrical Engineering

國立臺灣大學 National Taiwan University

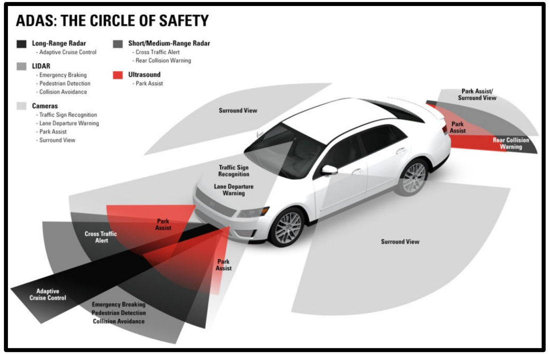
Background

Autonomous Driving

Traffic Management

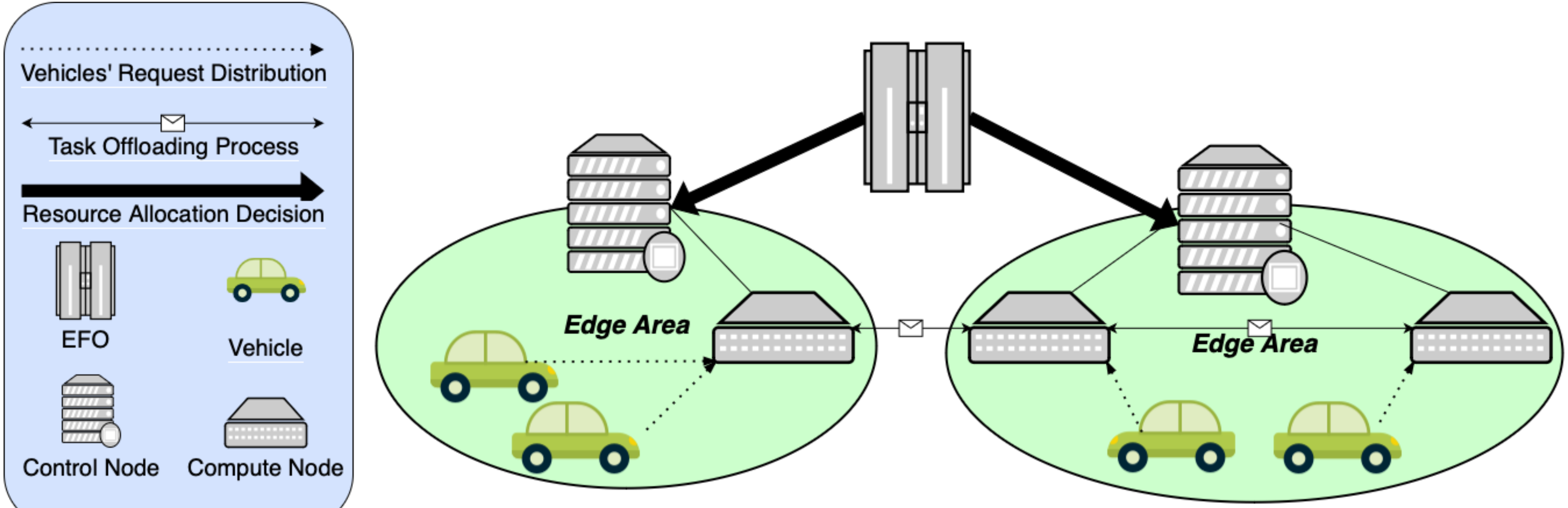


Entertainment



Enhanced Driver Assistance Systems

Background (scenario)



High workload area

Low workload area

Background (related work)

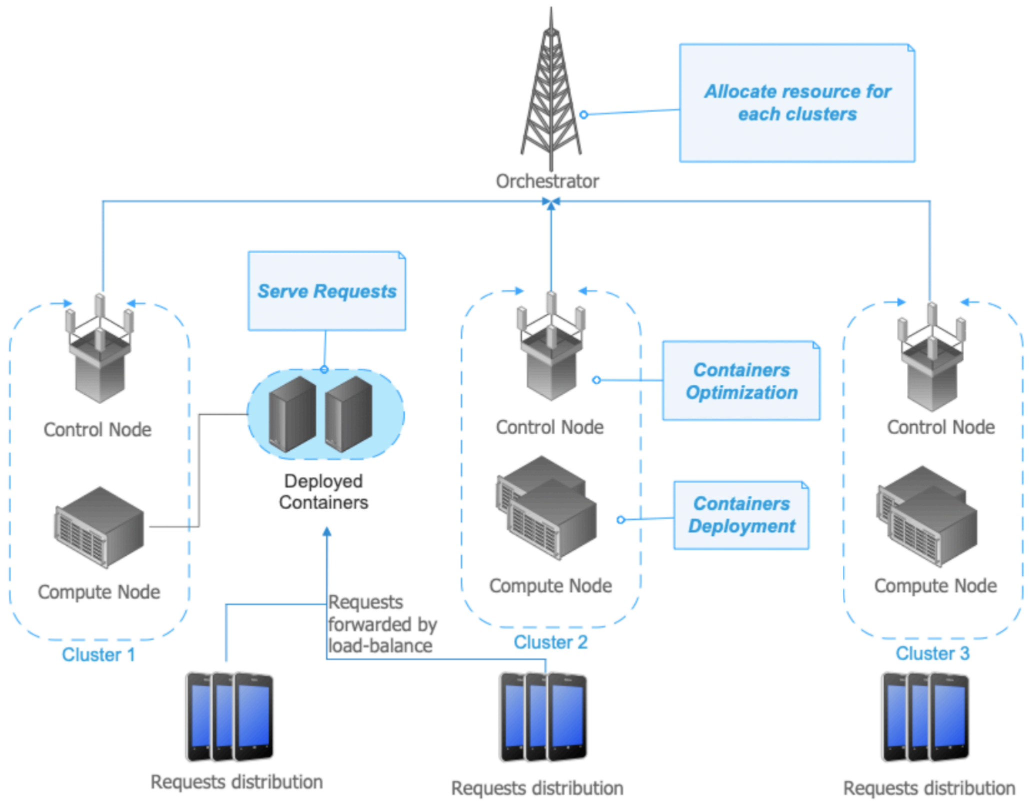
Past researches

- Resource allocation(RA)/task scheduling optimizing processing delay.
- Task offloading(TO) between servers optimizing propagation delay.
- Game-based edge-edge offloading with joint RA & TO.

Novelty of this paper

Collaborative edge-edge cooperation with joint RA & TO.

System Model



System Model(Delay Model)

Average delay of the overall system:

$$D = avg([D_i]) = \frac{\sum_{i \in C}(D_i \cdot \sum_{k \in A} Req_{ik})}{\sum_{i \in C} \sum_{k \in A} Req_{ik}}$$

Averaging cluster delay

Average delay of a single edge cluster:

$$D_i = avg([D_{P_{ij}} + D_{T_{ij}}]) = \frac{\sum_{j \in C}(T_{ij} + P_{ij})}{\sum_{k \in A} Req_{ik}}, i \in C$$

Averaging processing delay

Averaging propagation delay

$$P_{ij} = \sum_{k \in A}(\eta_{jk} \cdot N_{ijk}), i \in C, j \in C$$

Cumulative processing delay of requests forwarded from cluster i to j

$$T_{ij} = e_{ij} \cdot \sum_{k \in A}(N_{ijk} \cdot d_k), i \in C, j \in C$$

Cumulative propagation delay of requests forwarded from cluster i to j

System Model (Propagation Delay)

Cumulative propagation delay of requests forwarded from cluster i to j:

$$T_{ij} = e_{ij} \cdot \sum_{k \in A} (N_{ijk} \cdot d_k), i \in C, j \in C$$

Aggregation of propagation delay

The amount of application k's requests forwarded from cluster i to j

Propagation delay for a single request forwarded from cluster i to j:

$$t_{ij} = e_{ij} \cdot d_k, i \in C, j \in C, k \in A$$

Propagation speed from cluster i to j

Average data size of a single request for application k

System Model (Processing Delay)

Cumulative processing delay of requests forwarded from cluster i to j:

$$P_{ij} = \sum_{k \in A} (\eta_{jk} \cdot N_{ijk}), i \in C, j \in C$$

Aggregation of processing delay

Processing delay of a single request forwarded from cluster i to j:

$$\eta_{jk} = \eta_k(Req'_{jk}, [R_{jk}]), j \in C, k \in A$$

Requests amount being handled by cluster j

Resources allocated to cluster j for the deployment of application k

System Model (Requests Forwarding)

Amount of application k's requests being handled locally:

$$N_{iik} = \begin{cases} Req'_{ik}, Req_{ik} \geq Req'_{ik} \\ Req_{ik}, Req_{ik} < Req'_{ik} \end{cases}$$

Determination for each conditional cases.

Amount of application k's requests being forwarded from cluster i to j:

$$N_{ijk} = \begin{cases} 0, Condition_{invalid} \\ min(x, y), Condition_{valid} \end{cases}$$

$$condition_{invalid} : Req'_{ik} + \sum_{\beta \in C - prevC_j} N_{i\beta k} \geq Req_{ik}$$

$$|| Req'_{jk} \leq Req_{jk} + \sum_{\alpha \in C - prevC_i} N_{\alpha jk};$$

$$condition_{valid} : else;$$

$$x = Req'_{jk} - Req_{jk} - \sum_{\alpha \in C - prevC_i} N_{\alpha jk};$$

$$y = Req_{ik} - Req'_{ik} - \sum_{\beta \in C - prevC_j} N_{i\beta k};$$

$$Req'_{jk} = \frac{|R_{jk}|}{\sum_{n \in C} |R_{nk}|} \sum_{n \in C} Req_{nk}$$

$$|R_{jk}| = \sqrt{\sum_{l \in R} R^2_{jkl}}$$

Original amount of application k's requests in cluster n

Amount of resources with type l allocated in cluster j for application k

Re-distributed application k's requests on cluster j.

System Model (Requests Forwarding)

Amount of application k's requests being handled locally:

$$N_{iik} = \begin{cases} Req'_{ik}, & Req_{ik} \geq Req'_{ik} \\ Req_{ik}, & Req_{ik} < Req'_{ik} \end{cases}$$

Indicates that cluster i has more requests that it should handle.

Thus it should handle only partial of its received requests locally. The amount is exactly Req'ik as we computed.

Indicates that cluster i has less requests that it should handle.

Thus it should handle all of its received requests locally.

System Model (Requests Forwarding)

Indicates that cluster i has no exceed request or cluster j is not available to handle any exceed request.

Thus no request forwarding between cluster i and j.

Indicates that cluster i has exceed request(s) and cluster j is available to handle exceed request(s).

Cluster j can handle all the exceed request(s) from cluster i.

Cluster j can handle only partial of the exceed request(s) from cluster i.

Amount of application k's requests being forwarded from cluster i to j:

$$N_{ijk} = \begin{cases} 0, Condition_{invalid} \\ min(x,y), Condition_{valid} \end{cases}$$

$$condition_{invalid} : Req'_{ik} + \sum_{\beta \in C - prevC_j} N_{i\beta k} \geq Req_{ik}$$

$$|| Req'_{jk} \leq Req_{jk} + \sum_{\alpha \in C - prevC_i} N_{\alpha jk};$$

$$condition_{valid} : else;$$

$$x = Req'_{jk} - Req_{jk} - \sum_{\alpha \in C - prevC_i} N_{\alpha jk};$$

$$y = Req_{ik} - Req'_{ik} - \sum_{\beta \in C - prevC_j} N_{i\beta k};$$

Availability of cluster j

Remaining exceed request(s) in cluster i.

System Model (Constraint)

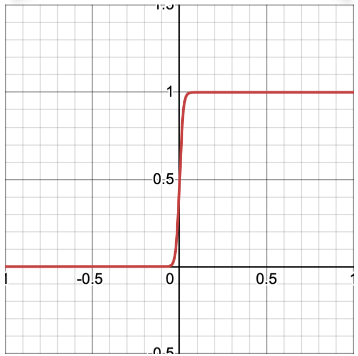
$$D^* = D + \boxed{\Omega \sum_{\gamma_i \in Consts} \sigma(a\gamma_i)}$$

$$\sigma(ax) = \frac{1}{1+e^{-ax}}$$

Filter that helps in avoiding resource allocation exceed the resource constraints in each cluster.

Generate large virtual delay if doesn't obey the resource constraints.

Have no effect if obey the resource constraints



Example when a=100

$$C_1 : \sum_k R_{jkr} \leq Res_{jr}, j \in C, k \in A, r \in R;$$

System Model (Problem Formulation)

$$min\{D^* = D^*([Req_{ik}], [Res_{il}], [R_{ikl}])\}, i \in C, k \in A, l \in R;$$

Algorithm (Dynamical Requests Forwarding)

- This algorithm helps compute requests forwarding once the resources allocation and propagation speed between clusters are fixed.

$$Req'_{jk} = \frac{|R_{jk}|}{\sum_{n \in C} |R_{nk}|} \sum_{n \in C} Req_{nk}$$

$$|R_{jk}| = \sqrt{\sum_{l \in R} R_{jkl}^2}$$

Memorize the previously forwarded requests from cluster n

Compute local requests first as they have no propagation

Algorithm 1 Dynamical Requests Forwarding

Input: $[Req_{jk}], [Req'_{jk}], [e_{ij}]$
Output: $[N_{ijk}]$

```
Initialize [N_ijk] as a 3D-array filled with zeros.
Initialize prevC_n as a 2D-array filled with zeros.
for i loop through each cluster do
    for k loop through each application do


Sort [e_ij] in ascending order
for i, j loop through [e_ij] do
    if i! = j then
        for k loop through each application do
            Compute N_ijk using Eq. 9
            Append j, i into prevC_i, prevC_j respectively
return [N_ijk]
```

$$N_{iik} = \begin{cases} Req'_{ik}, Req_{ik} \geq Req'_{ik} \\ Req_{ik}, Req_{ik} < Req'_{ik} \end{cases}$$

Append i into $prevC_i$

Algorithm (Dynamical Requests Forwarding)

Algorithm 1 Dynamical Requests Forwarding

Input: $[Req_{jk}], [Req'_{jk}], [e_{ij}]$
Output: $[N_{ijk}]$

```
Initialize [N_ijk] as a 3D-array filled with zeros.
Initialize prevC_n as a 2D-array filled with zeros.
for i loop through each cluster do
    for k loop through each application do
        N_iik = Req'_ik
        Append i into prevC_i
Sort [e_ij] in ascending order
for i, j loop through [e_ij] do
    if i! = j then
        for k loop through each application do


return [N_ijk]
```

Ensure requests can be offloaded as near as possible.

$$N_{ijk} = \begin{cases} 0, Condition_{invalid} \\ min(x, y), Condition_{valid} \end{cases}$$

$$condition_{invalid} : Req'_{ik} + \sum_{\beta \in C - prevC_j} N_{i\beta k} \geq Req_{ik}$$

$$|| Req'_{jk} \leq Req_{jk} + \sum_{\alpha \in C - prevC_i} N_{\alpha jk};$$

$$condition_{valid} : else;$$

$$x = Req'_{jk} - Req_{jk} - \sum_{\alpha \in C - prevC_i} N_{\alpha jk}$$

$$y = Req_{ik} - Req'_{ik} - \sum_{\beta \in C - prevC_j} N_{i\beta k};$$

Append j, i into $prevC_i$, $prevC_j$ respectively

Simulation (Settings)

- **System Settings:** 3 clusters.
- **Resource Settings:** CPUs random in [5, 15], Memory random in [4GB, 10GB] in each cluster.
- **Parameter Settings:** Propagation speed between clusters random in [1MB/s, 20MB/s].
- **Processing Efficiencies:** $\eta_{ik} = \dfrac{Req'_{ik}}{k_1 \cdot CPU_{ik}} + \dfrac{Req'_{ik}}{k_2 \cdot Memory_{ik}}$

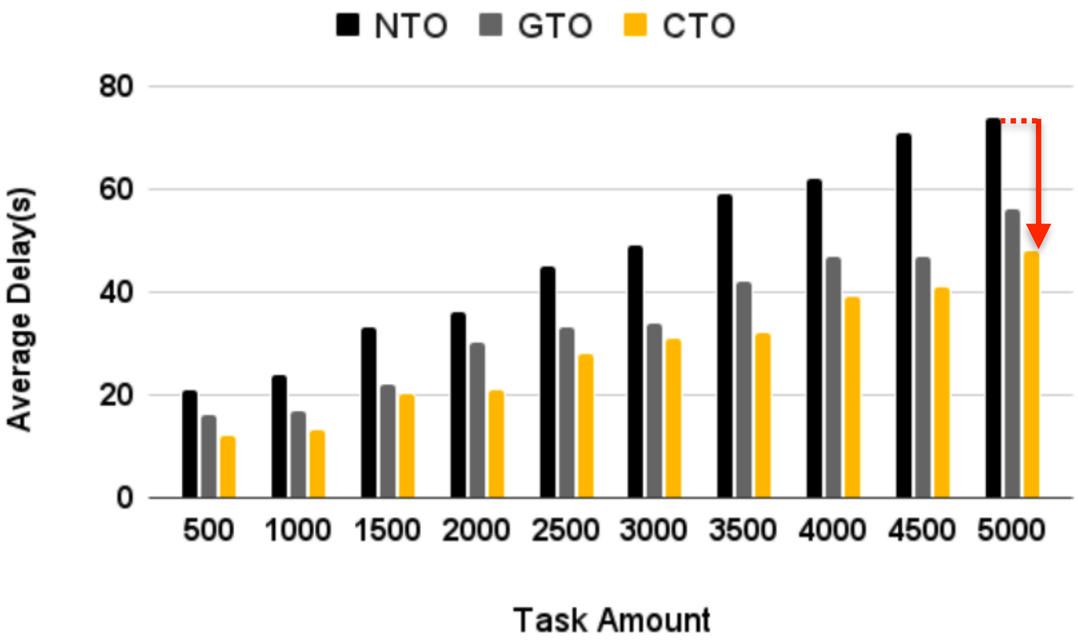
Simulation (Algorithms compared)

CTO: Clusters make RA & TO decision collaboratively.

- **NTO**: No task offloading between clusters.
- **GTO**: Each cluster make rational resource allocation and task offloading decision without considering the others.

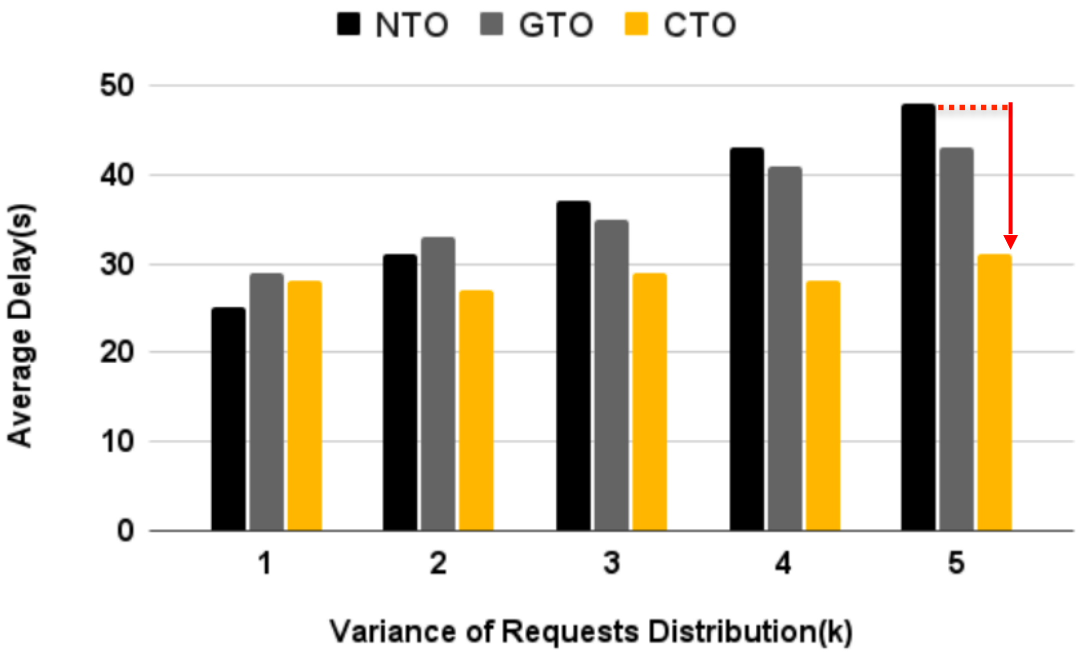
Simulation (Results Evaluation)

35% lower delay with large amount of task



Simulation (Results Evaluation)

30% lower delay at large variance of requests distribution



Conclusions

- The collaboration task offloading between edge clusters is investigated through a joint RA & TO mechanism
- A centralized and greedy algorithm is proposed, yielding better results than existing algorithms
  - 30% lower delay than a non-cooperative one
  - 20% lower delay than one proposed in a previous work

Future Works

- Consider scenarios involving micro-services and diverse network topologies.
- Consider container scaling to improve processing efficiency.
- Conduct experiment on IEEE1935 testbed.

References

[1] Y. Chiang et al., ”Management and Orchestration of Edge Computing for IoT: A Comprehensive Survey,” in IEEE Internet of Things Journal, vol. 10, no. 16, pp. 14307-14331, 15 Aug.15, 2023.

[2] M. Najm, M. Patra, and V. Tamarapalli, “Cost-and-delay aware dynamic resource allocation in federated vehicular clouds,” IEEE Trans. Veh. Tech- nol., vol. 70, no. 6, pp. 6159–6171, Jun. 2021.

[3] Y. Chiang, C. -H. Hsu, G. -H. Chen and H. -Y. Wei, ”Deep Q-Learning-Based Dynamic Network Slicing and Task Offloading in Edge Network,” in IEEE Transactions on Network and Service Management, vol. 20, no. 1, pp. 369-384, March 2023.

[4] L. H. Phuc, M. Kundroo, D. -H. Park, S. Kim and T. Kim, ”Node-Based Horizontal Pod Autoscaler in KubeEdge-Based Edge Computing Infrastructure,” in IEEE Access, vol. 10, pp. 134417-134426, 2022.

[5] L. Liu, J. Feng, X. Mu, Q. Pei, D. Lan and M. Xiao, ”Asynchronous Deep Reinforcement Learning for Collaborative Task Computing and On-Demand Resource Allocation in Vehicular Edge Computing,” in IEEE Transactions on Intelligent Transportation Systems, vol. 24, no. 12, pp. 15513-15526, Dec. 2023.Magnetics Japan, p. 301, 1982].

[6] T. X. Tran and D. Pompili, ”Joint Task Offloading and Resource Allocation for Multi-Server Mobile-Edge Computing Networks,” in IEEE Transactions on Vehicular Technology, vol. 68, no. 1, pp. 856-868, Jan. 2019.

[7] “IEEE standard for edge/fog manageability and orchestration,” IEEE standard 1935-2023, 2023.

[8] ETSI, “Mobile edge computing (mec); framework and reference architecture,” ETSI, DGS MEC, standard 3, 2016.

[9] K. Ye, Y. Kou, C. Lu, Y. Wang and C. -Z. Xu, ”Modeling Application Performance in Docker Containers Using Machine Learning Techniques,” 2018 IEEE 24th International Conference on Parallel and Distributed Systems (ICPADS), Singapore, 2018.

[10] W. Fan et al., ”Game-Based Task Offloading and Resource Allocation for Vehicular Edge Computing With Edge-Edge Cooperation,” in IEEE Transactions on Vehicular Technology, vol. 72, no. 6, pp. 7857-7870, June 2023.

[11] Y. -J. Ku, P. -H. Chiang and S. Dey, ”Real-Time QoS Optimization for Vehicular Edge Computing With Off-Grid Roadside Units,” in IEEE Transactions on Vehicular Technology, vol. 69, no. 10, pp. 11975-11991, Oct. 2020.

[12] W. Fan et al., ”Joint Task Offloading and Resource Allocation for Vehicular Edge Computing Based on V2I and V2V Modes,” in IEEE Transactions on Intelligent Transportation Systems, vol. 24, no. 4, pp. 4277-4292, April 2023.

Q&A

Thanks