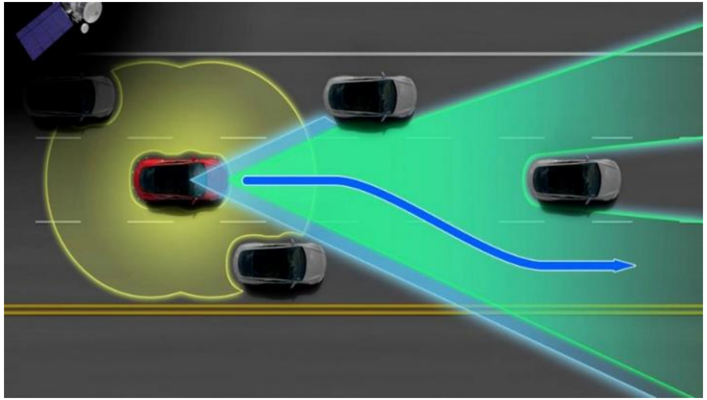
Crowdsourcing Real Time Dynamic Map in Automotive Edge Computing

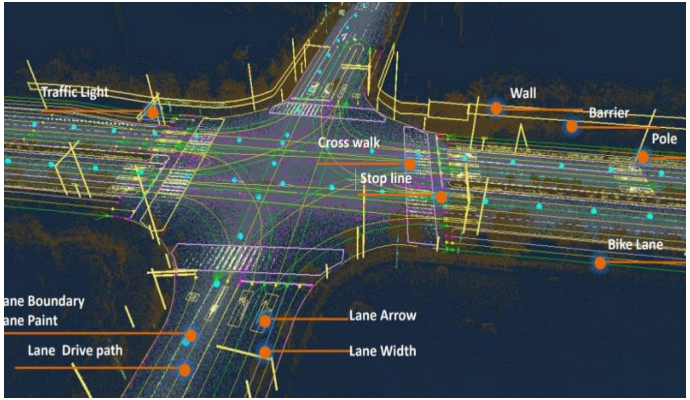
Qiang Liu, Assistant Professor, UNL

IN OUR GRIT, OUR GLORY™

High-Definition Map

- ❖ Autonomous cars are isolated
  - Line-of-sight sensors
  - Blind spots
- ❖ HD map is static
  - Massive dataset
  - Non-real time

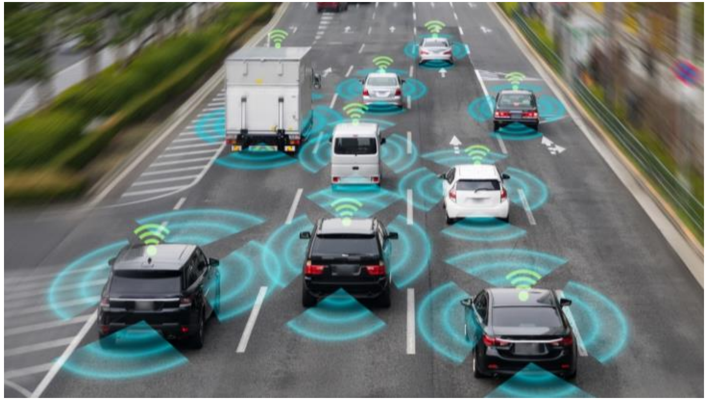




* Tesla autopilot, HTML
* HD map, HTML

Connected Vehicles

- Connecting vehicles together helps
  - Wirelessly vehicle-to-everything
  - Allows sharing & collaboration
- Sharing information is challenging
  - Massive real time perception data
  - Dynamic networks



SLAM
>100Mbps

Speed
>30kph

* Connected vehicle, [HTML]
* Ahmad, F., et. al., Carmap: Fast 3d feature map updates for automobiles. NSDI 2020

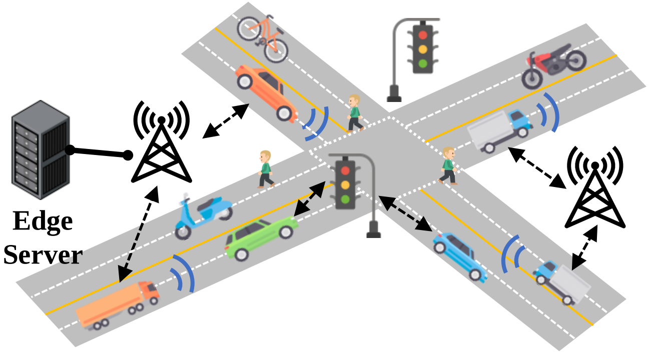
Agenda

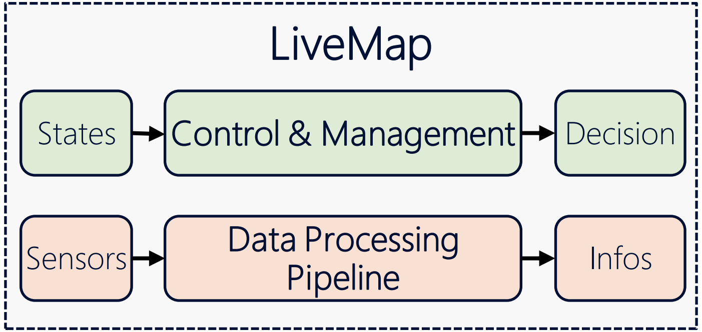
- System Overview
- Data Plane Design
- Control Plane Design
- System Implementation
- Results and Analysis

System Overview

- How to design a system to:
  - Connect vehicles wirelessly?
  - Detect, match, track objects on the road?
  - Sharing info among vehicles in sub-seconds?

- LiveMap: real-time dynamic map
  - Crowdsourcing from connected vehicles
  - Efficient info sharing platform, e.g., ped., bike
  - Intelligent management, e.g., sched., offload

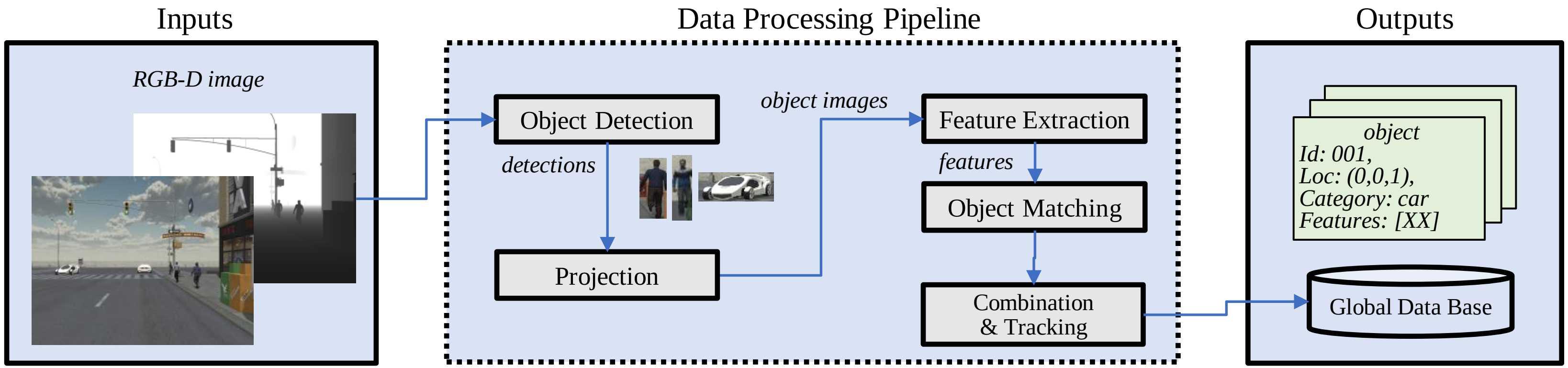




Data Plane Design

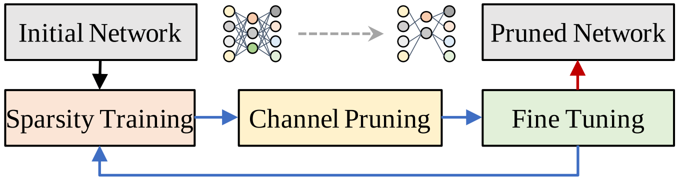
- How to process vehicle data efficiently?
  - Inputs: raw sensor data (RGB-D images)
  - Outputs: object info (id, 3d loc., category)
  - Objective: low process time, high accuracy

- Components
  - Object detection
  - Projection
  - Feature extraction
  - Object matching
  - Combination & tracking



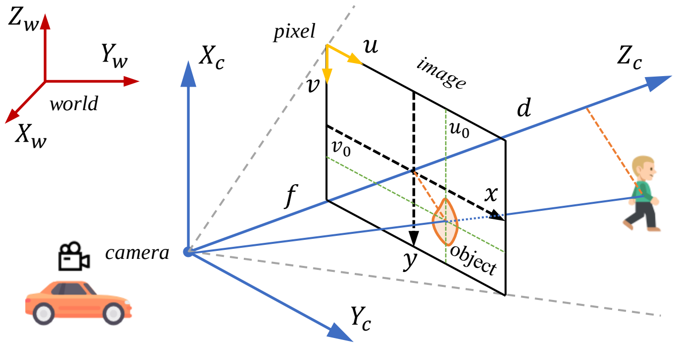
Detection & Projection

- Object detection
  - **Objective**: detect objects in RGB image
  - **Insight**: existing detectors are designed for generic categories in high-end server
  - **Contribution**: specific category for transportation, neural network pruning
  - **Results**: 93.7% network size reduction w/ 0.01 mAP loss



Detection Networks	mAP@0.5 640x	Num. of parameters	time(Nano) w/o TensorRT
YOLOv3 tiny	0.534	8.69e+06	191.9/37.4 ms
Pruned Network	0.524	0.54e+06	154.7/30.4 ms

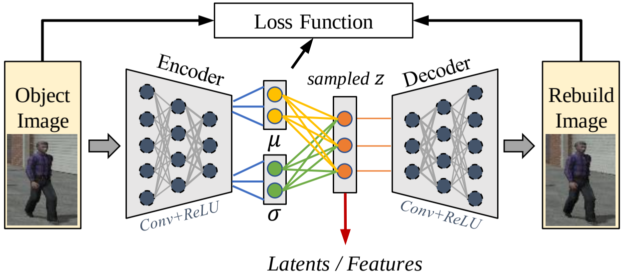
- Object projection
  - **Objective**: project 2d loc. of objects to 3d with depth info
  - **Insight**: inaccurate depth estimation w/ bounding box
  - **Contribution**: sample-based avg. depth calculation



Extraction & Matching

❖ Feature extraction

- Objective: extract features from cropped objects
- Insight: existing extractors (ORB) generate large feature size, perform not well for small objects
- Contribution: adopt variational autoencoder (VAE) as extractor



❖ Object matching

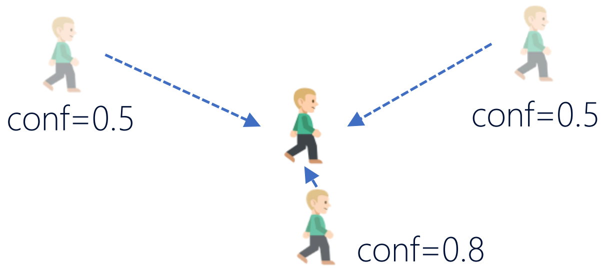
- Objective: match objects within global database
- Insight: feature distance neglects phy. distance among objects
- Contribution: new location-aware distance function w/ predicted object loc

$$D_{i,j} = \min([||z_{i,m} - z_{j,m}||^2, \forall m \in \mathcal{M}]) + w||g_i - g_j||^2$$

Combination & Tracking

❖ Object combination

- **Objective**: combine objects loc seen from diff. angles
- **Insight**: inaccurate loc combination w/ same id
- **Contribution**: new confidence weighted method



❖ Global dataset

- **Objective**: record objects w/ multi-attributes
- Share new obj. updates for all vehicles
- **Entry**: unique id, category, location, confidence, direction, speed, features, etc.

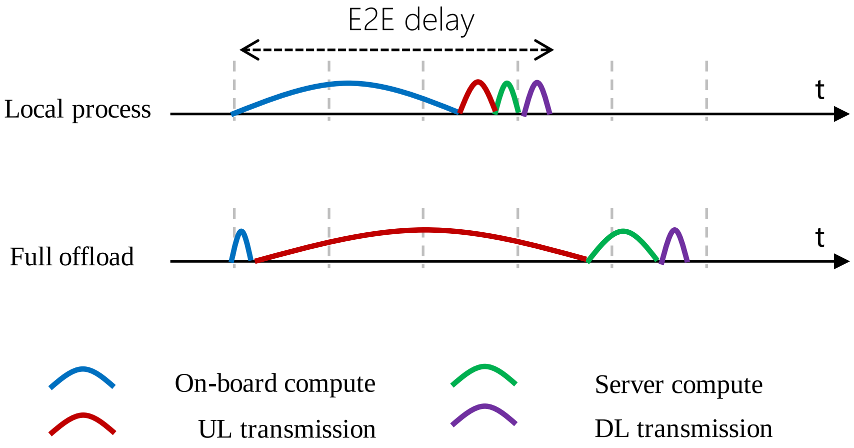
object id	class	geo-location	confidence
speed	direction	update time	multi-view latents

TABLE II: Attributes of an object in the database

Control Plane Design

❖ Data Offloading Procedures

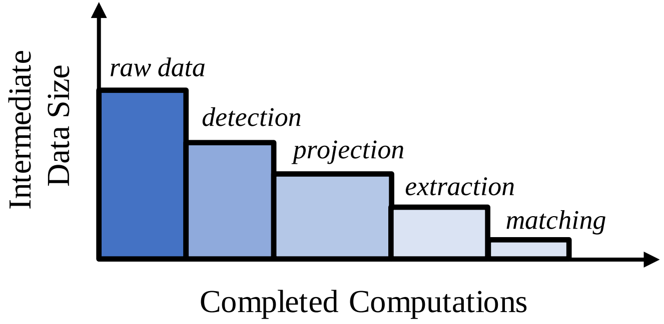
- **Local process**: low trans. traffic, long delay as limited on-board comp.
- **Full offload**: heavy trans. traffic, long delay as sharing trans., and comp.



Control Plane Design

❖ Observed Insights

- **Partial DP offload:** more on-board computation, less data size to transmit
- **Schedule vehicles:** vehicle coverage overlapping, esp. urban area



The Problem

- How to manage vehicles in system?
  - Inputs: network state (traffic, network condition)
  - Outputs: vehicle schedule, partial offload of DP
  - Objective: low info sharing delay
  - Difficulty: E2E delay is too complicated to be mathe. modeled

E2E delay

$$\min_{\{\mathcal{X},\mathcal{Y}\}} \boxed{\sum_{t\in\mathcal{T}}\sum_{i\in\mathcal{I}} L_i^{(t)}}$$

coverage

$$s.t. \quad \boxed{\bigcup_{i\in\mathcal{I},x_i\neq 0} C_i^{(t)} \geq \beta \bigcup_{i\in\mathcal{I}} C_i^{(t)}, \forall t\in\mathcal{T},}$$

schedule

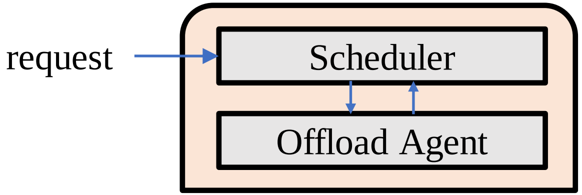
$$\boxed{x_i^{(t)} \in \{0,1\}, \forall i\in\mathcal{I}, t\in\mathcal{T},}$$

$$\boxed{y_i^{(t)} \in \{0,1,...,N\}, \forall i\in\mathcal{I}, t\in\mathcal{T},}$$

partial offload

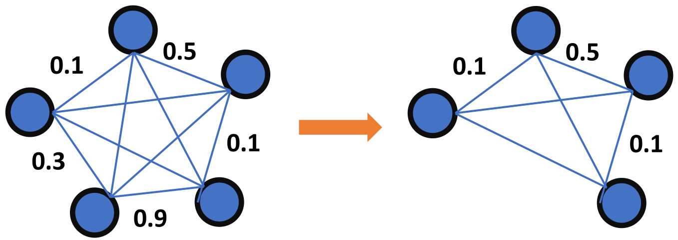
High-Level Solution

- Overview
  - **Insight:** DP offload (subsec.) has smaller time scale than vehicle sched (sec.)
  - **Scheduler:** pre-sched. vehicle based on coverage to meet requirement
  - **Offload agent:** decide partial DP offloading w/ DRL technique



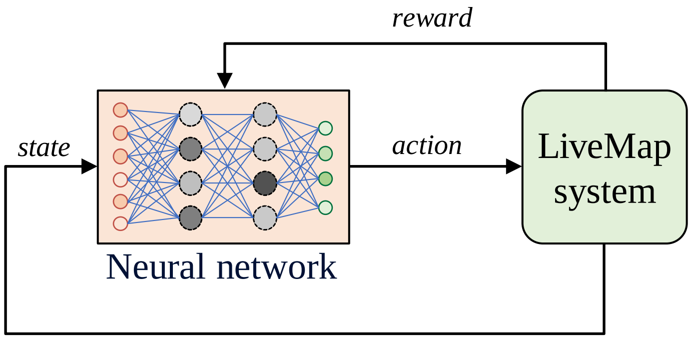
Vehicle Scheduling

- Vehicle scheduling
  - Inputs: vehicle coverage, requirement
  - Outputs: scheduled vehicle list
  - Objective: minimum vehicles, meet requirement
  - Method: graph trimming, vehicles as vertices, coverage overlap ratio as edges



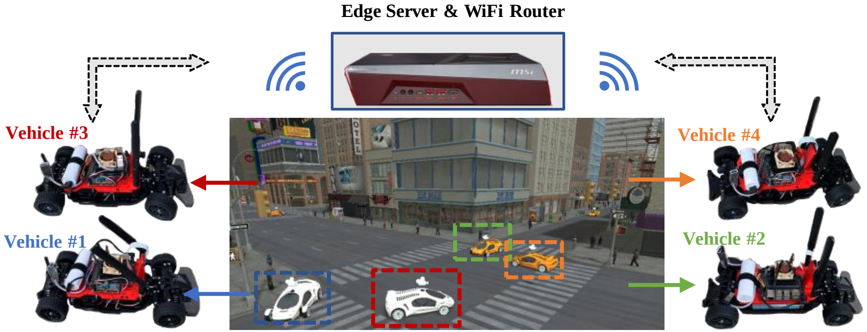
Offloading Decision

- DP offload agent
  - State: local state (wireless quality, loc., comp.); global state (traffic, server workload, etc.)
  - Action: index of partial offloading
  - Reward: negative E2E delay



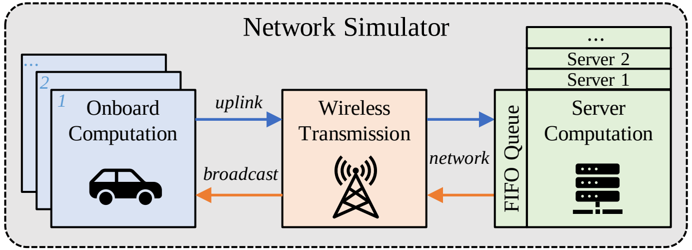
System Prototype

- **Vehicles**: 4x JetRacers w/ ARM CPU, Nvidia Nano
- **Server**: Intel i7 w/ Nvidia GTX 1070
- **Access point**: 5GHz WiFi router w/ 20MHz BW
- **Dynamics**: Linux "iw" adjust WiFi TX power
- **Data set**: Unity3D, record 1000 RGB-D images/ location per vehicle



Network Simulator

- **Wireless:** system-level 5G simulator, UMi channel, 1MHz BW
- **Computing:** FIFO service model, real comp delay derived from experiments
- **Time driven:** time scale 1 ms
- **Trace:** real world measurement on system prototype



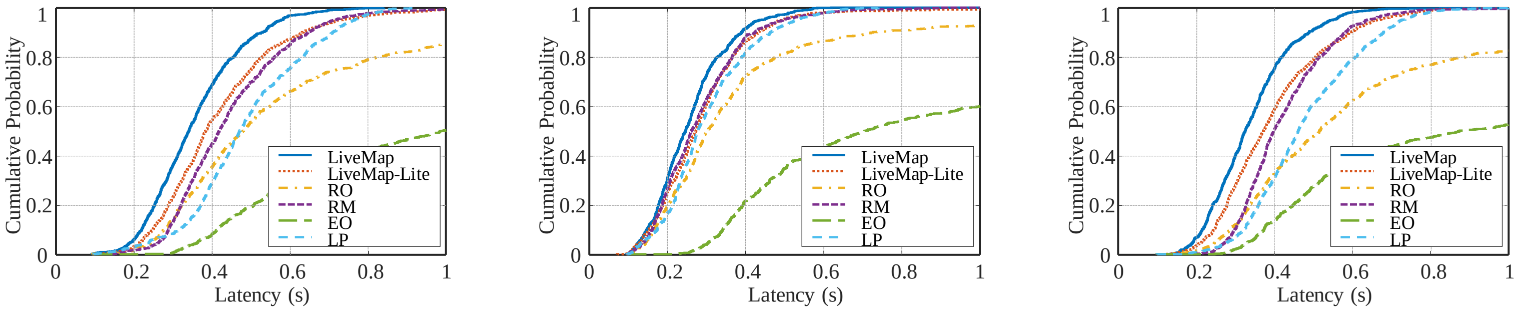
Experimental Results

❖ Comparison Alg.

- LP: local process for all vehicles
- EO: offload raw data for all vehicles
- RO: random offload decision
- RM: regress a model to make offload decision
- LiveMap-lite: schedule all vehicles

❖ LiveMap outperforms others

- Avg. 20.3% latency reduction than RM
- Avg. 16.4% latency reduction than LiveMap-lite
- Better performance under crowd scenarios



(a) intersection (b) highway (c) circle

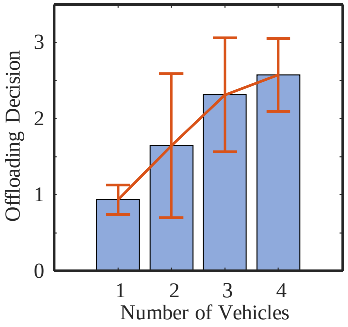
Experimental Results

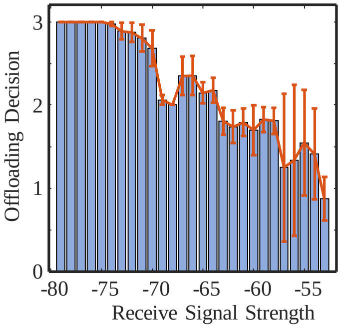
❖ LiveMap is intelligent

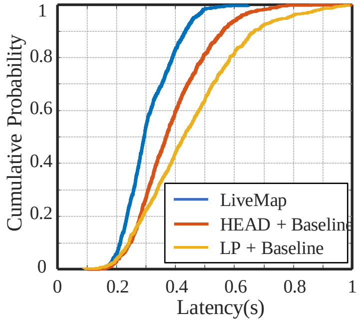
- make larger offloading decisions if more vehicles
- Lower offloading decision if better wireless quality

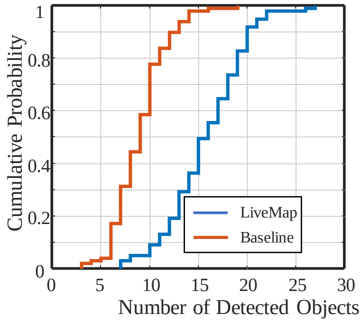
❖ LiveMap is more efficient

- Avg. 34.1% latency reduction than LP + Baseline
- Avg. 74.9% improvement of #objects* than Baseline







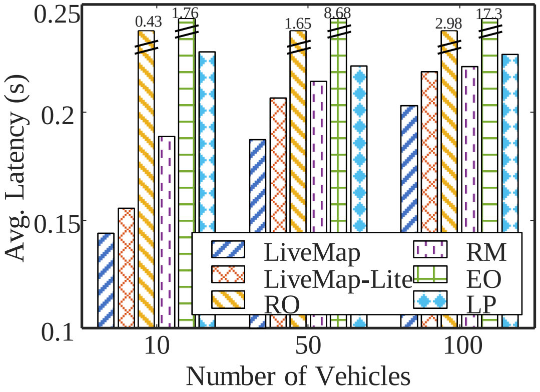


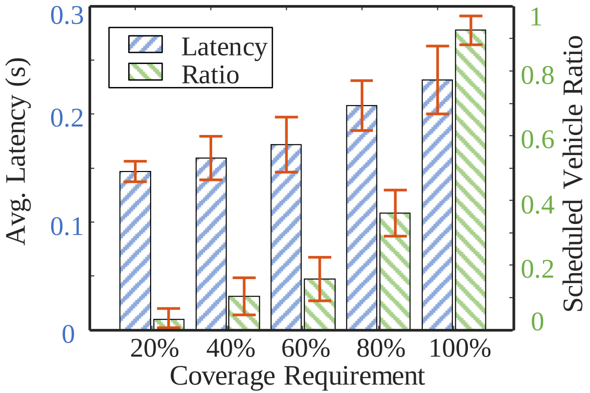
* We consider an object is successfully detected only if its object ID is matched correctly and the estimated geo-location error is less than 1 meter.

Simulation Results

- LiveMap is more scalable
  - Avg. 10.2% latency reduction than RM if 50 vehicles
  - RO and EO are not scalable if more vehicles

- LiveMap exploits overlapping coverages
  - Avg. 31.1% latency reduction if 60% requirement
  - Only. 15.6% vehicles are scheduled meanwhile





Conclusion

- We propose LiveMap, a real-time dynamic map, allows efficient info. sharing among connected vehicles
- We design data plane to efficiently detect, match, track objects on the road based on crowdsourcing data from connected vehicles in sub-second
- We design control plane to intelligently schedule vehicles and determine DP offload under network dynamics
- Future directions: cooperative perception, communication-efficient offloading

Qiang Liu
Assistant Professor
School of Computing
University of Nebraska–Lincoln
qiang.liu@unl.edu
https://cse.unl.edu/~qliu/