UNIVERSITY OF SURREY

SIGNetS Project, Work Package 3

Efficient resource management for target tracking in UAV-assisted mobile edge computing

Shidrokh Goudarzi, **Pei Xiao**, **Wenwu Wang**

Centre for Vision Speech and Signal Processing
Centre 5GIC & 6GIC, Institute for Communication Systems (ICS)
Department of Electronic Engineering
University of Surrey

Outline

- Introduction
- Research objectives
- Network model
- UAV-aided target tracking
- Efficient target tracking process
- Preliminary results in MATLAB
- Implementation of target tracking scenario in OMNeT++
- Next steps of project implementation

Introduction

Connected IoT devices such as automated cars in target tracking tasks generate massive data, resulting in two main challenges.

- Large amount of raw data and computing tasks need to be processed, while the **computing capacity** of each automated car is limited.
- A huge volume of data needs to be transmitted through the network with a **low latency** to fulfill the requirements of **the real-time tasks**, while both the wireless and the wired transmission resources are inadequate in the networks.

So, we need to propose a cooperative target tracking approach for a multi-agent system.

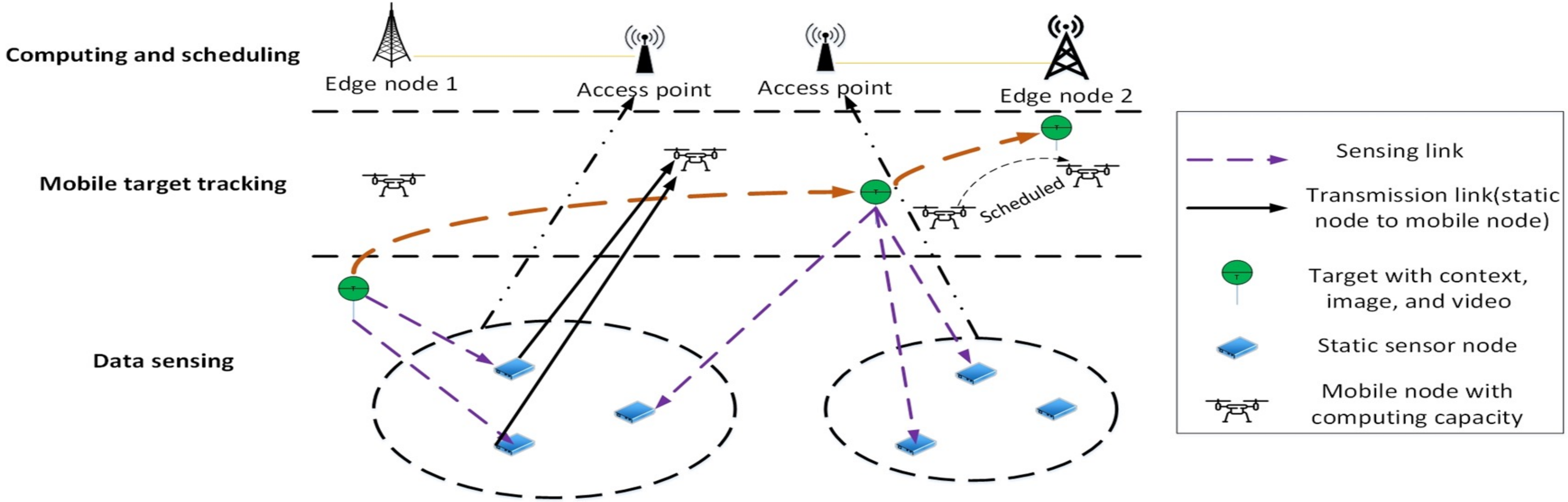
We plan to design a multi-agent learning model in **a cooperative target tracking system** to decrease sensing **latency** and the **sensing cost** (i.e., the energy consumption).

Objectives

The main objectives are summarized as follows:

- Designing a reinforcement learning **(RL) resource allocation** algorithm
- Designing **trajectory prediction** algorithm
- Designing **unmanned aerial vehicle (UAV) mobility model** for target tracking in OMNeT++

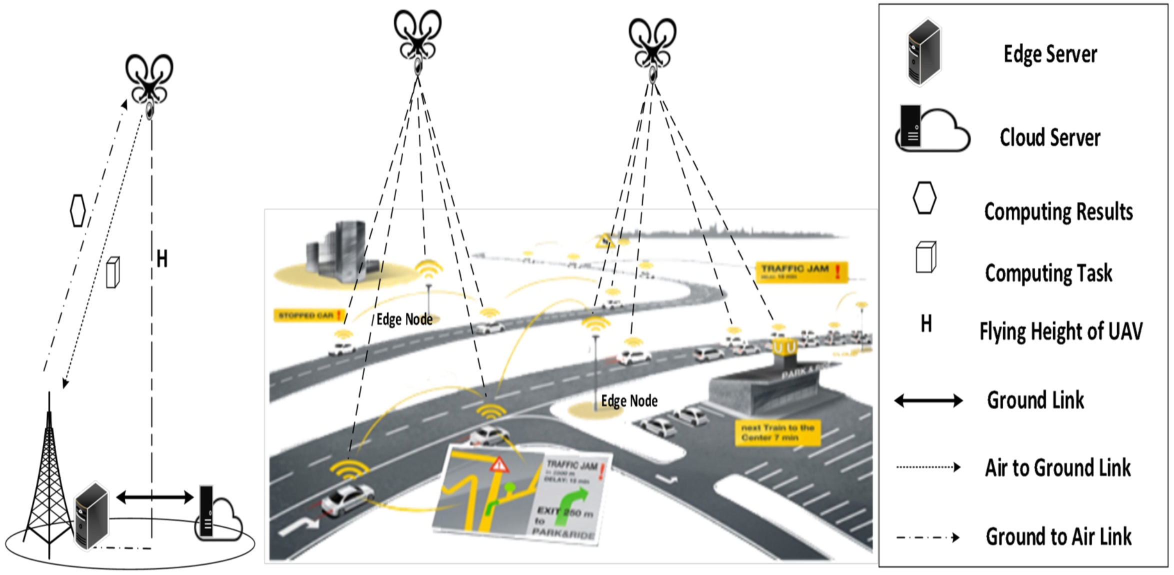
Network Model



- The Figure shows the target tracking structure.
- *M* heterogeneous sensor nodes (**SNs**) are deployed randomly to detect targets on the bottom level.
- *N* mobile nodes (**MNs**) can **process** and **compute** the sensing tasks on the middle level, so that blind sensing zones can be reduced based on their flexible mobility.
- **Intelligent scheduling decisions** are made by edge servers from a global view.

UAV-aided Target Tracking

Target tracking by UAV	
Case 1	The entire tracking process can be completed by the **UAV alone**. High energy consumption of the UAV during flight.
Case 2	Computing tasks are offloaded from the UAV to **ground nodes/servers**[1]. High energy consumption and wireless bandwidth consumption caused by the transmission of a large amount of data. High computational time.
Case 3	Computing tasks are offloaded from the UAV to **edge node**, and then the computing results are sent back to the UAV [2].



[1]. Y. Du, K. Wang, K. Yang, and G. Zhang, "Energy-efficient resource allocation in UAV based MEC system for IoT devices," in 2018 IEEE Global Communications Conference (GLOBECOM). IEEE, 2018, pp. 1–6.
[2]. Y. Zeng and R. Zhang, "Energy-efficient UAV communication with trajectory optimization," IEEE Transactions on Wireless Communications, vol. 16, no. 6, pp. 3747–3760, 2017.

UAV-aided Target Tracking

We consider a scenario with K targets.

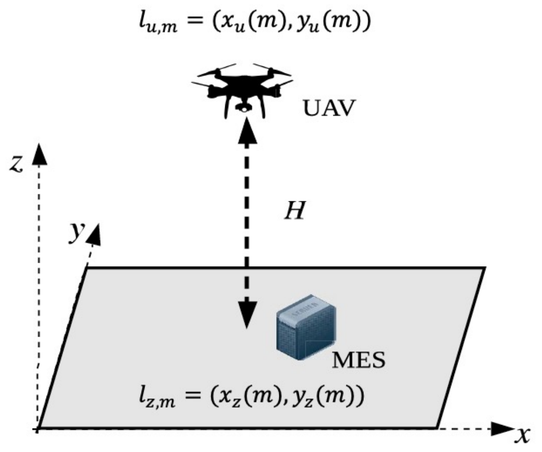
- The set of **targets** is $\mathcal{K} = \{1, 2, \ldots, K\}$.
- We assume that S **sensor nodes**, the set of SNs $\mathcal{S} = \{1, 2, \ldots, S\}$ are deployed to detect targets with diverse on-board sensors in a monitoring area and n UAVs that may offload the tasks to **Edge Nodes** (ENs) through the cellular network and set of **UAVs** is defined as $\mathcal{N} = \{1, 2, \ldots, N\}$ are used to detect the targets.
- We assume that there are Z (ENs) denoted by $\mathcal{Z} = \{1, 2, \ldots, \mathrm{Z}\}$.
- A UAV offloads the tasks to an edge node $x_{z,m}$ for a duration of T.

Target tracking by UAV	
UAV altitude	H
Task offloading duration	T
Time slot length τ is discretized into M time slots.	$T = M\tau$
UAV's location at time slot m; $\{1 \leq m \leq M\}$	u_m
UAV generates data of size S_m bits at time slot m	S_m
$x_{z,m}$ is the EN selection variable. Edge node $\{x_{z,m}, 1 \leq z \leq Z, 1 \leq m \leq M\}$	$x_{z,m}$

UAV-aided Target Tracking

- If edge node $x_{z,m}$ is selected as the service node, then the **distance** between edge node and UAV is defined as[3]:

$$d_{u,z}^{m} = \sqrt{H^2 + \left\| l_{u,m} - l_{z,m} \right\|^2}$$



where $l_{z,m} \in R^{2\times 1}$ is the location of $x_{z,m}$ and $l_{u,m} \in R^{2\times 1}$ is the UAV's trajectory projected on the ground at time slot m.

- We assume quasi-static block fading channels for the **communication link from the UAV to a ground EN**, where the channel remains unchanged within each time slot and may change over time slots.
- The quasi-static block fading channel follows the free-space path loss model[1], which can be expressed as:

$$h_{u,z}^{m} = \beta_0 d_{m,z}^{-2} = \frac{\beta_0}{H^2 + \left\| l_{u,m} - l_{z,m} \right\|^2}$$

where β_0 refers to channel power gain at the distance $d_0 = 1m$.

[3] Yang, B., Cao, X., Yuen, C., & Qian, L. (2020). Offloading Optimization in Edge Computing for Deep-Learning-Enabled Target Tracking by Internet of UAVs. IEEE Internet of Things Journal, 8(12), 9878-9893.

UAV-aided Target Tracking

- Then the **channel capacity in bps** can be expressed as[4]:

$$R_{m,z} = \frac{B}{n}\log_2\left(1 + \frac{p_{m,z}\left|h_{m,z}\right|^2}{\sigma^2}\right)$$

where B shows the the channel bandwidth between UAV and EN that can be divided into n subbands for the offloading communication, σ^2 is the white Gaussian noise power at the EN, and $\frac{p_{m,z}\left|h_{m,z}\right|^2}{\sigma^2}$ is the signal-to-noise ratio (SNR) at $d_0 = 1m$. The **execution time** $t_{m,z}^{total}$ of each task is considered as the sum of transmission time and computational time of the EN that serves for the UAV.

$$t_{m,z}^{total} = t_{m,z}^{transmission} + t_{m,z}^{computation} = \frac{S_m}{B\log_2(1 + \frac{\beta_0 p_{m,z}}{\sigma^2 d_{m,z}^2})} + \frac{S_m}{r_{m,z}}$$

where $r_{m,z}$ be the data processing capability (in bps) of the z-th available EN of each task S_m at time slot m.

- The $E_{m,z}$is the transmission **energy consumption** in Joule which can be expressed as

$$E_{m,z} = t_{m,z}^{transmission} p_{m,z} = \frac{S_m}{B\log_2(1 + \frac{\beta_0 p_{m,z}}{\sigma^2 d_{m,z}^2})} p_{m,z}$$

where $p_{m,z}$ is the transmit power allocated by the UAV to EN.

[4] Gu, X., Zhang, G., Wang, M., Duan, W., Wen, M., & Ho, P. H. (2021). UAV-aided Energy Efficient Edge Computing Networks: Security Offloading Optimization. *IEEE Internet of Things Journal*.

Efficient Target Tracking Process

UNIVERSITY OF SURREY

- The goal is to minimize the energy cost and time cost of executing a task, which is defined as **total cost metric**:

$$Cost_{m,z} = \alpha E_{m,z} + \beta t_{m,z}^{total}$$

where α and β represent the relative weight on transmission energy consumption and task execution time, which can be set and tuned to meet different situations.

- Main objective is to jointly optimize the UAV's transmission power P and edge node selection schedule X so as to **minimize** the tradeoff between the UAV's transmission **energy consumption and the execution time**, while ensuring normal tracking.
- The problem formulated as:

$$\min_{P,X} \sum_{m=1}^{M} \sum_{z=1}^{Z} x_{m,z} Cost_{m,z}$$

$$\text{s.t.} \quad C_1 \quad t_{m,z}^{total} \leq \tau, \ \forall z, m$$

$$C_2 \quad \sum_{z=1}^{Z} x_{m,z} = 1, \ \forall z, m$$

- A computing task has a delay tolerance τ, and $t_{m,z}^{total} \leq \tau$ ensures that the UAV can receive the results and makes adjustment in time to track target normally.
- If $x_{m,z}$ is selected as the service EN at time slot m, $x_{m,z} = 1$. We assume that only one EN can be selected to serve for the UAV at each time slot.

Local Computing

- **Local computation delay**:

$$\tau_l^i = \left(\frac{c_i}{f_l^i} + \rho_i((1-\eta)\epsilon_L + \eta)\right)$$

where c_i is the total number of CPU cycles required to accomplish the computation of data for task i; f_l^i is the allocated CPU computation resource to U_i per second. ρ_i is defined as all the DL tasks failing and dropping penalty of delay and it is no smaller than the tasks processing delay. ϵ_L is inference error rate given by UAVs. η is the percentage of data with low quality [5].

- **Local energy consumption**:

$$\varepsilon_l^i = k(f_l^i)^2 c_i + \xi_i((1-\eta)\epsilon_L + \eta)$$

where k is the energy efficiency parameter that is mainly depends on the chip architecture [6], f_l^i is the CPU clock speed and ξ_i is defined as all the DL tasks failing and dropping penalty of energy consumption.

- The weighted cost for local computing is defined as

$$\mathcal{O}_l^i = \theta\tau_l^i + (1-\theta)\varepsilon_l^i$$

where θ and $(1-\theta)$, $0 \leq \theta \leq 1$, specify the UAV's preference on processing delay and energy consumption, respectively.

[5] Zhang, Tiankui, et al. "Joint computation and communication design for UAV-assisted mobile edge computing in IoT." IEEE Transactions on Industrial Informatics 16.8 (2019): 5505-5516.
[6] Gu, X., Zhang, G., Wang, M., Duan, W., Wen, M., & Ho, P. H. (2021). UAV-aided Energy Efficient Edge Computing Networks: Security Offloading Optimization. *IEEE Internet of Things Journal*.

Offloading Computing

U_i offloads tasks to edge node, the delay and energy consumption comprise two items:

- Delay and energy consumption to EN via the wireless link
- Delay and energy consumption at EN

- The **delay for offloading** the task to the EN is given by:

$$\tau_o^i = \left(\frac{c_i}{f_l^i} + \gamma_i(\frac{s_i}{R_i} + \frac{c_i}{f_i})\rho_i(\eta\epsilon_H)\right)$$

- where s_i[bits] shows the size of computation input data, γ_i shows the scale coefficient of data size output from U_i and $\gamma_i = \frac{s_{out}^i}{s_i}$; s_{out}^i is the data size output from U_i. R_i is the available transmission rate between U_i and EN. ϵ_H is inference error rate given by UAVs.

- The **energy consumption** of U_i using **offloading computing** is calculated as[7]

$$\varepsilon_o^i = k(f_l^i)^2\, c_i + \gamma_i(P_t^i \frac{s_i}{R_i} + P_t^i \frac{c_i}{f_i}) + \xi_i\, \eta\epsilon_H$$

where P_t^i is the power consumption of U_i, when U_i sending task to EN and staying idle while waiting for the execution results from EN.

- The weighted cost for offloading computing is defined as

$$\mathcal{O}_o^i = \theta\tau_o^i + (1-\theta)\varepsilon_o^i$$

where θ and $(1-\theta)$, $0 \le \theta \le 1$, specify the UAV's preference on processing delay and energy consumption, respectively.

[7] Alsenwi, Madyan, et al. "UAV-assisted multi-access edge computing system: An energy-efficient resource management framework." 2020 International Conference on Information Networking (ICOIN). IEEE, 2020.

Tradeoff Between Delay and Energy Consumption

UNIVERSITY OF SURREY

- The **total system cost** is considered, taking tracking delay and energy consumption [4].

$$\mathcal{O}_{total} = \sum_{i=1}^{n}\left(\theta\left(\frac{c_i}{f_l^i} + max\{\delta_p^i, \delta_{op}^i\}\right) + (1-\theta)(\varepsilon_l^i + \varepsilon_o^i)\right)$$

$$= \begin{cases} \sum_{i=1}^{n}\left(\theta\left(\frac{c_i}{f_l^i} + \delta_p^i\right) + (1-\theta)(\varepsilon_l^i + \varepsilon_o^i)\right), if\, \delta_p^i \geq \delta_{op}^i \\ \sum_{i=1}^{n}\left(\theta\left(\frac{c_i}{f_l^i} + \delta_{op}^i\right) + (1-\theta)(\varepsilon_l^i + \varepsilon_o^i)\right), if\, \delta_p^i \leq \delta_{op}^i \end{cases}$$

where $\delta_p^i = (1-\beta_i)\rho_i\widetilde{\epsilon_L}$ is the delay penalty for processing $(1-\beta_i)$ of tasks at UAV,

and $\delta_{op}^i = \beta_i(\gamma_i\left(\left(\frac{s_i}{R_i}\right) + \left(\frac{c_i}{f_i}\right)\right) + \rho_i\widetilde{\epsilon_L})$ including the transmission delay of intermediate data, processing delay at EN, and the delay penalty for processing β_i of task i at EN. (β_i is the **offloaded ratio of $\boldsymbol{U_i}$**).

- In this case, we formulate the **cost minimization** as an **optimization problem**:

$$\min_{\beta_i} \mathcal{O}_{total}$$

$$\text{s.t. } C_1, C_2$$

$$\text{s.t.} \quad C_1 \quad t_{m,z}^{total} \leq \tau, \ \forall z, m$$

$$C_2 \quad \sum_{z=1}^{Z} x_{m,z} = 1, \ \forall z, m$$

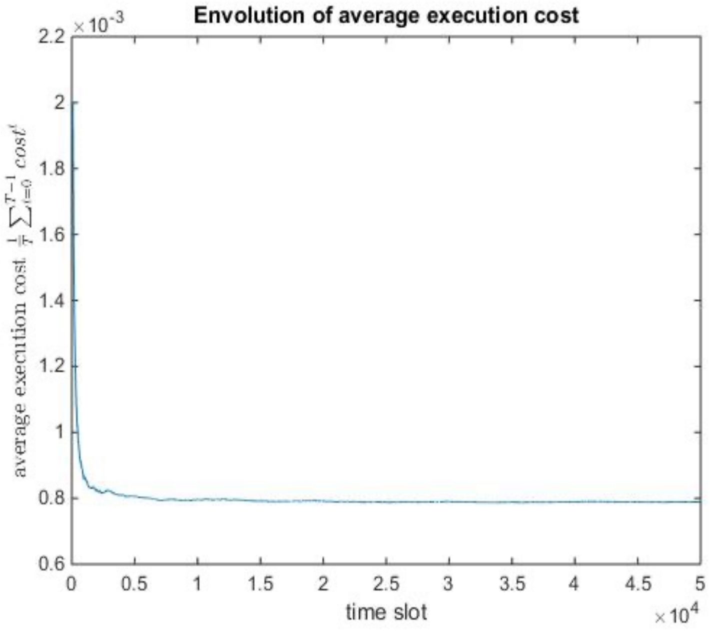
Simulation Setup in MATLAB

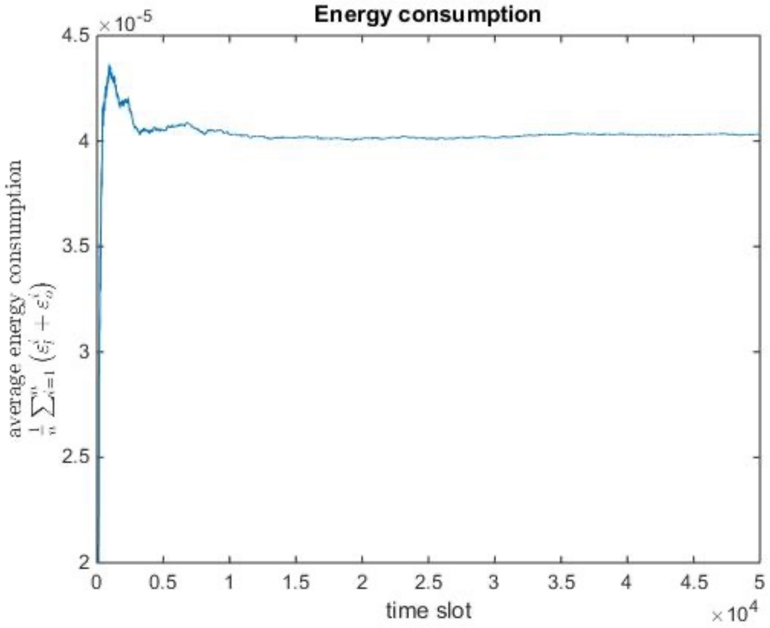
UNIVERSITY OF SURREY

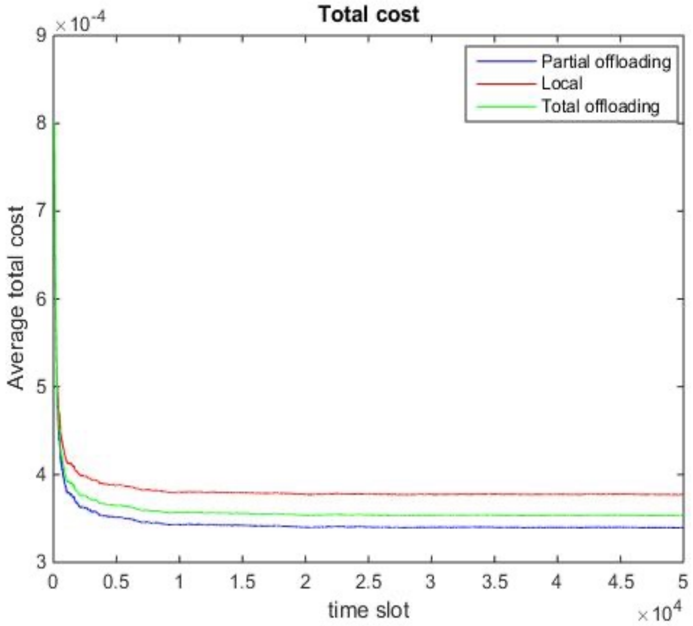
- The system parameters that used in the MATLAB simulation are as follows:

MATLAB Simulation Parameters			
Communication bandwidth	20 MHz	Maximum CPU-cycle frequency	0.5 GHz
Channel power gain (β_0)	20 dBm	Input size of the computation task	1000 bits
Upper bound of the energy at the UAV	48×10^{-6} J	Effective switched capacitance	10^{-28}
Length of time slot	0.002 s	Maximum transmit power of UAV	20 dBm
Maximum distance between UAV and EN	50 m	Number of time slot	50000
Task offloading duration	30 s	Delay tolerance τ	0.002 s
UAV altitude	100 m	Noise power at the EN (σ^2)	−110 dBm

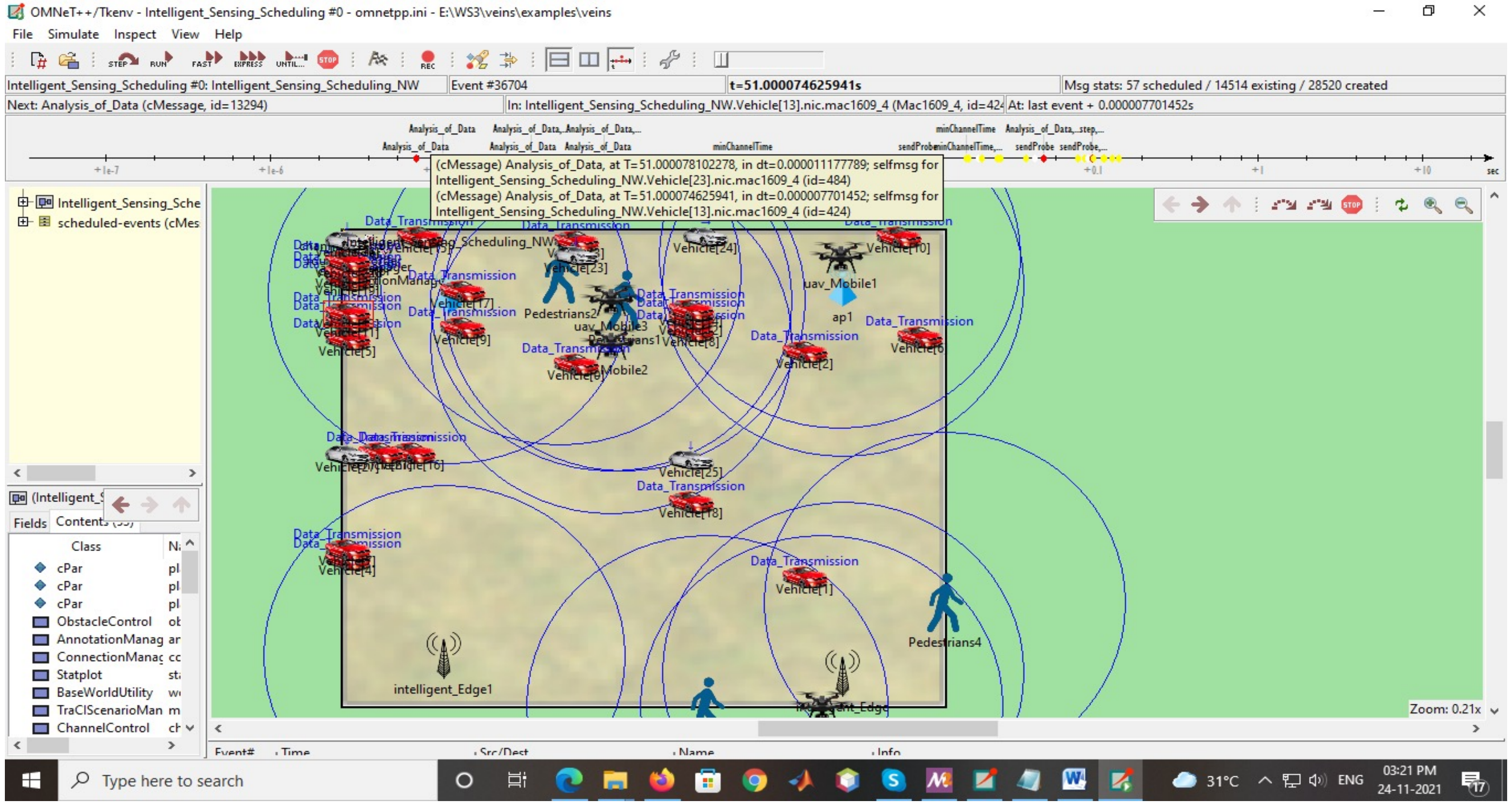
MATLAB Results







Implementation of Target Tracking Scenario in OMNeT++

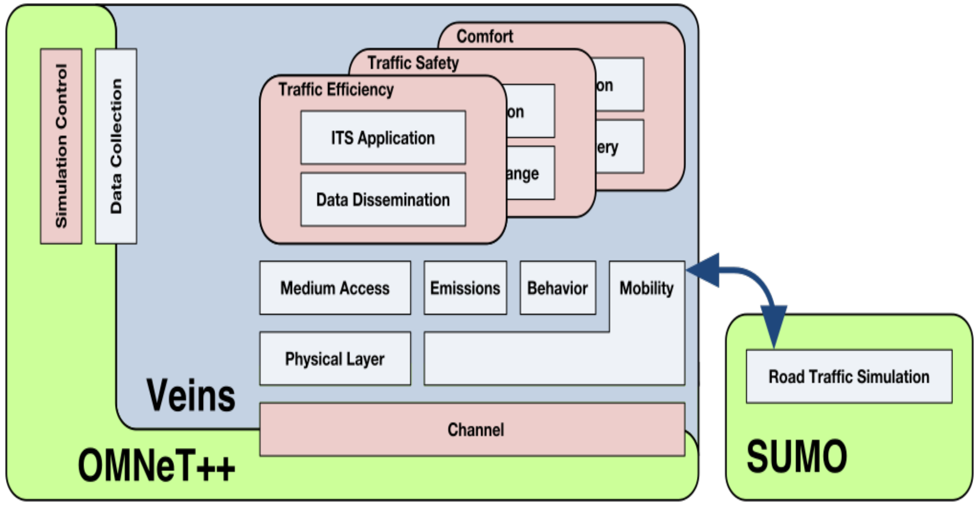


Target tracking scenario in OMNeT++

Veins is an open-source framework for running vehicular network simulations. It is based on two well-established simulators:

- **OMNeT++,** an event-based network simulator, and
- **SUMO**, a road traffic simulator.

It extends these to offer a comprehensive suite of models for vehicular network simulation.



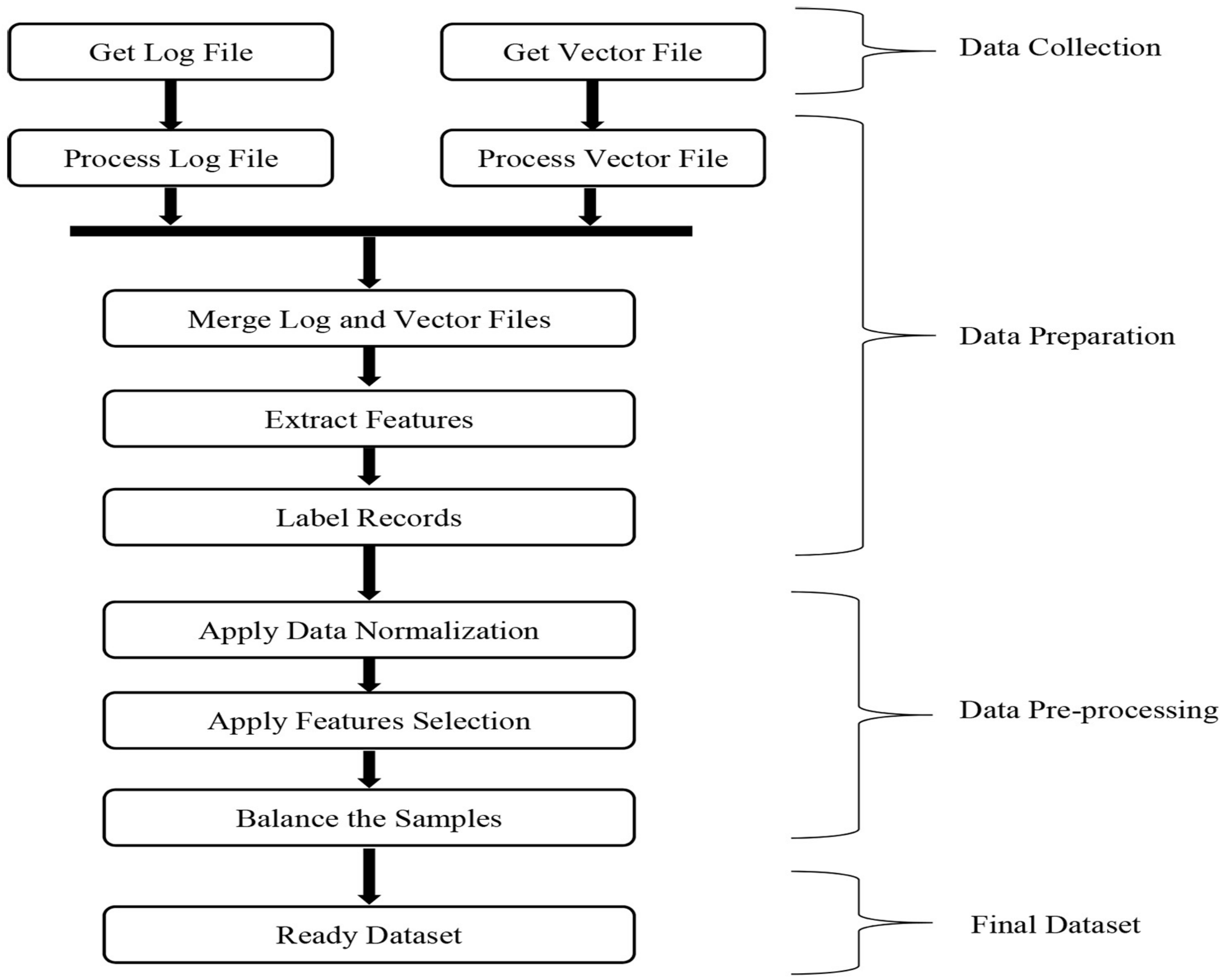
Target Tracking Scenario in OMNeT++

- Implementation of target tracking scenario in OMNeT++.

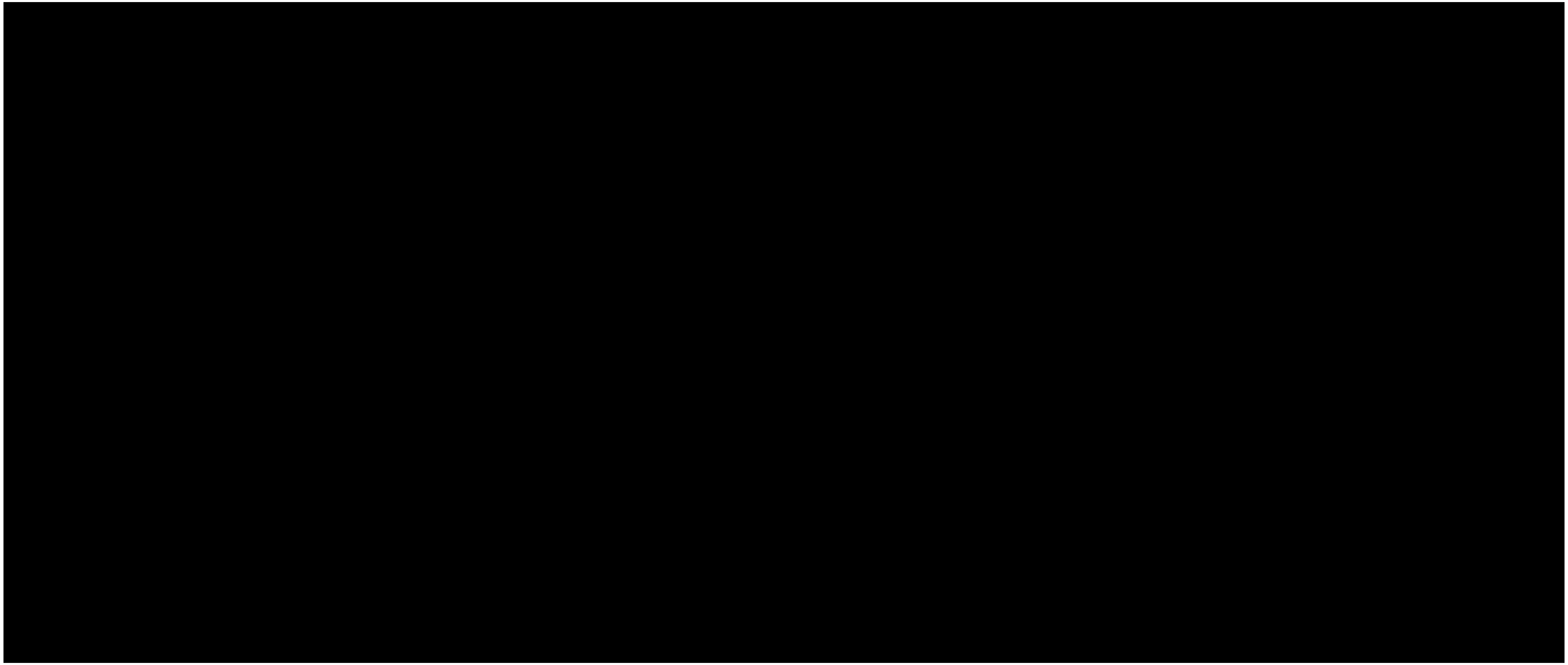
Important Parameters and value			
Parameters	Value	Parameters	Value
Network Simulator	OMNeT++ 4.6	Pedestrians	4
Traffic Simulator	SUMO 0.19.0	MAC Protocol	IEEE 802.11p
V2X Simulator	VEINS 4.4	Header size	256 bits
Simulation area	2750×2250×5	Data size	1024 bits
Simulation time	500s	UAVs Mobility types	RectangleMobility
Intelligent edge nodes	2	Pedestrians Mobility types	RectangleMobility
Number of Access points	2	Vehicle Mobility types	TraCIMobility
Number of Vehicles	100	Velocity of mobile target	Range (0m/s; 1m/s)
Radio Propagation Model	FreeSpacePathLoss	Initial location of mobile target	Range (0m; 1500m)

Generating Dataset Using OMNeT++

- The following Figure shows the workflow for generating an informative dataset.



Target tracking scenario in OMNeT++



Conclusions and Future work

We have evaluated the UAV-aided target tracking in **local computing**, **offloading computing and partial offloading** in MATLAB.

In current step, we have used random selection scheme to schedule the sensors in OMNeT++.

We plan to implement **multi target tracking** algorithm for trajectory prediction in OMNeT++.

We plan to extend the implementation of **multi-agent reinforcement learning (RL)** network for sensor scheduling during target tracking in OMNeT++.

Distributed/federated learning for decentralized target tracking.

Proper sensor **distribution** model, **target mobility** model and **blind sensing zone**/block effect.