Online-Journals.org

iJOE International Journal of Online and Biomedical Engineering

iJOE | eISSN: 2626-8493 | Vol. 22 No. 2 (2026) | OPEN ACCESS

https://doi.org/10.3991/ijoe.v22i02.58823

PAPER

iFogSUMO: An Integrated Platform of iFogSim and SUMO to Enhance Simulation Capabilities of Adaptive Traffic Control for Smart City Applications

Ashwini Matange[1,2] (✉), **Jibi Abraham**[1]

[1]COEP Technological University, Pune, India

[2]Pimpri Chinchawad College of Engineering, Pune, India

matangea21.comp@coeptech.ac.in

ABSTRACT

The increasing demand for smart city applications and Internet of Things (IoT) solutions has led to a growing need for simulation tools that can accurately model and analyze complex systems. iFogSim and Simulator of Urban Mobility (SUMO) are two popular simulation tools that cater to different aspects of smart city and IoT applications. iFogSim focuses on fog computing and IoT simulations, while SUMO specializes in traffic and mobility simulations. The integration of iFogSim and SUMO simulators can enable researchers and developers to design and optimize smart city applications more effectively by providing a robust testbed. This paper proposes integrating iFogSim and SUMO to create a comprehensive simulation framework, iFogSUMO, that accurately models and analyzes adaptive traffic control systems. The framework was evaluated for ambulance routing by simulating traffic with various vehicle types and executing the adaptive traffic control algorithm to assess its performance in terms of trip time, waiting time, and time loss during ambulance routing, while maintaining scalability across heterogeneous vehicle fleets.

KEYWORDS

iFogSUMO, Simulator of Urban Mobility (SUMO), iFogSim, co-simulation, smart city, adaptive traffic control, ambulance routing

1 INTRODUCTION

Road traffic problems are increasing due to rapid urbanization, causing congestion, long waiting times, increased fuel consumption, pollution, accidents, and health issues [1]. Adaptive and intelligent traffic control mechanisms proposed by the researchers have yielded promising results in addressing these issues, compared with static traffic signal timing systems [2], [3]. These systems deploy optimization

Matange, A., Abraham, J. (2026). iFogSUMO: An Integrated Platform of iFogSim and SUMO to Enhance Simulation Capabilities of Adaptive Traffic Control for Smart City Applications. *International Journal of Online and Biomedical Engineering (iJOE)*, 22(2), pp. 38–54. https://doi.org/10.3991/ijoe.v22i02.58823

Article submitted 2025-09-27. Revision uploaded 2025-12-18. Final acceptance 2025-12-18.

algorithms to fine-tune traffic light durations, increasing traffic flow, reducing congestion, and improving overall traffic efficiency. However, implementing such systems has to be first tested on robust simulation tools that enable modeling of real-time traffic and produce logs to evaluate the performance under various scalable scenarios [4].

Simulator of Urban Mobility (SUMO) is a widely used open-source traffic simulator that provides detailed modeling of road networks, vehicle generation, traffic simulation, and traffic signal operations [2]. Its flexibility and extensibility make it an ideal platform for testing and validating adaptive traffic control strategies. However, SUMO has several limitations: it primarily uses static traffic signal plans unless combined with external control logic, real-time computation of complex adaptive algorithms is not natively supported, and latency issues arise when simulating large-scale networks [4]. Moreover, SUMO alone cannot model distributed computing environments for low-latency decision-making.

To overcome these limitations, Fog Computing provides a decentralized architecture in which computation and storage are moved closer to the traffic sources (e.g., traffic lights and roadside units). This enables low-latency processing of real-time traffic data and rapid execution of adaptive control algorithms [4], [5]. iFogSim is a simulation toolkit that models Fog and Edge computing infrastructures, allowing analysis of latency, resource allocation, and task offloading strategies in distributed environments [5], [6].

Integrating SUMO with iFogSim through co-simulation allows the development and testing of Fog-based adaptive traffic control systems. Fog nodes can process real-time vehicle data from SUMO, execute adaptive signal control algorithms, and update traffic signals with minimal delay. This integration not only overcomes SUMO's limitations but also enables the evaluation of adaptive strategies in realistic, latency-sensitive scenarios, bridging the gap between traffic modeling and Fog-based computation [4], [6].

Despite advances in isolated traffic and fog simulators, no unified platform exists to co-simulate fine-grained vehicular dynamics with distributed edge computing latency and resource constraints—critical for deploying next-generation Adaptive Traffic Control Simulation (ATCS) in smart cities.

This study has two main objectives. First, to develop iFogSUMO, a unified co-simulation middleware that seamlessly integrates SUMO (microscopic traffic simulation) with iFogSim (fog computing emulation), providing a flexible platform for researchers to design, prototype, and benchmark adaptive traffic signal control algorithms under realistic mobility and computational constraints. Second, to validate the framework's functionality using a heterogeneous vehicle fleet (cars, buses, bikes, and ambulances) on a representative urban road segment from a real city network, across varying congestion levels (1000–2500 vehicles), ensuring robust performance in both normal and emergency scenarios.

The paper is organized as follows: Section 2 elaborates on the related work in this area, it provides the background information about the simulators and various combinations of these simulators with other simulators proposed in the past, whereas Section 3 discusses the proposed framework iFogSUMO, Section 4 gives the experimental setup and results for a scenario where a green channel is to be created for the ambulance, and Section 5 concludes the paper and also highlights the direction for future research.

2 RELATED WORK

The integration of IoT with fog computing has been recognized as a key enabler for low-latency, context-aware, and real-time services in smart city applications. SUMO Simulator has been extensively used for modeling traffic and network data. These simulators have been integrated with other simulators to enhance simulation power.

This section discusses the usage of Fog Computing in various areas, the iFogSim simulator, the use of SUMO, and the integration of these simulators with other simulators.

Bonomi et al. [6] first introduced the fog computing paradigm as a middle layer between cloud and edge, highlighting its ability to reduce latency and network load. Chiang and Zhang [7] further outlined the research opportunities in fog–IoT systems, emphasizing applications in transportation, healthcare, and industry. The edge-fog-cloud continuum has enabled latency-critical applications in domains like eHealth, where localized data filtering reduces transmission overhead by up to 70% [8]. This study extends similar principles to urban traffic control by co-simulating fog-enabled ATCS with realistic vehicular dynamics.

Recent studies have applied IoT-fog architectures specifically to traffic management. Zanella et al. [9] proposed an IoT-driven smart city framework capable of handling massive data flows from urban infrastructures. Similarly, Micko et al. [10] provided a review of IoT sensors used for road infrastructure monitoring, highlighting their potential in predictive maintenance and congestion detection. Alzyoud et al. [11] proposed an Adaptive Smart Traffic Accidents Management System that utilizes IoT sensors and dynamic data exchange to detect accidents, reroute vehicles, and coordinate emergency responses. These works illustrate the increasing adoption of IoT-Fog technologies for mobility management but do not provide a comprehensive simulation-based evaluation.

iFogSim has become a commonly used simulator for evaluating fog and IoT systems. Gupta et al. [12] introduced iFogSim, a Java-based toolkit to model resource management, latency, and energy trade-offs in fog architectures. Yi et al. [13] extended its applicability to platform design and applications in IoT domains.

Researchers have applied iFogSim to diverse contexts. For example, Puliafito et al. [14] examined fog resource orchestration for healthcare applications, while Sarkar and Misra [15] focused on task offloading strategies in fog-based smart city systems.

SUMO is a microscopic, open-source traffic simulator widely adopted for modeling transportation networks and evaluating traffic control strategies [16]. Behrisch et al. [17] provided an overview of SUMO, detailing its capabilities for route assignment, traffic signal optimization, and emissions modeling. Researchers have used SUMO to simulate a variety of scenarios. Lopez et al. [16] demonstrated their ability to model large-scale city traffic, including multimodal systems. In another study, SUMO was used to evaluate dynamic traffic-light control policies, demonstrating measurable improvements in congestion reduction [18].

Integration of multiple simulators has emerged as a promising approach to emulate the multifactor complexity of smart city systems. iFogSim has been integrated with CloudSim, Sniper Multi-Core Simulator, and AirFogSim. These simulators build on iFogSim to broaden the scope of what can be modeled and evaluated. iFogSim provides the base discrete-event framework for IoT/edge/fog resource management. Similarly, SUMO has been coupled with OMNeT++, NS-3, Veins, CommonRoad, and Webots. Such frameworks demonstrate the feasibility of cross-simulator interoperability through middleware layers. Table 1 describes the purpose of these integrations.

Table 1. Integration/co-simulation of SUMO and iFogSim with other simulators

Simulator	Ref. No.	Integrated/Co-Simulated with	Purpose
iFogSim	[12]	CloudSim	iFogSim is built on top of CloudSim and extends it to support modeling of fog and edge computing scenarios.
	[19]	Sniper Multi-Core Simulator	Integrated with iFogSim for accurate multi-core CPU modeling and resource management in IoT and fog systems.
	[20]	AirFogSim	Extends fog simulation with UAVs and integrates mobility using SUMO for simulating vehicular traffic along with iFogSim-based fog computing workflows.
SUMO	[21], [22]	OMNeT++	Network communication/VANET/V2X.
	[23]	NS-3	Discrete event network simulation.
	[24]	Veins	Provides coupling with SUMO for mobility + OMNeT++ for communication.
	[25]	CommonRoad	Automated driving framework.
	[26]	Webots	3D vehicle/robot simulator/visualization + dynamics.

Other open-source simulators have also contributed to traffic research. MATSim has been widely applied for large-scale, agent-based transport planning, while CityFlow provides a high-performance alternative tailored for AI and multi-agent learning experiments in signal control [27]. However, SUMO's modularity and compatibility with external frameworks such as OMNeT++ and NS-3 have made it the preferred choice for micro-level traffic analysis with a primary focus on network-layer or IoT data simulation.

Despite extensive work with SUMO and iFogSim independently, few studies have attempted to integrate these platforms into a unified co-simulation framework. iFogSUMO, an integration of SUMO with iFogSim, can enable the modelling of real-time vehicle mobility data captured by IoT devices and processed in fog/edge nodes, supporting adaptive traffic control and low-latency decision-making. This will also enable extensibility toward real-world deployment through mapping simulated inputs (e.g., vehicle trajectories, traffic density) to physical sensor data and linking emulated fog nodes to actual edge infrastructure, thereby bridging the simulation-to-reality gap.

3 PROPOSED FRAMEWORK

The proposed framework, iFogSUMO, integrates iFogSim and SUMO to enable co-simulation of traffic mobility and fog computing infrastructures in a smart city environment. The framework enables fog nodes to process real-time traffic data, providing a realistic simulation of fog-based traffic management. Although SUMO offers a powerful platform for mobility modeling, it does not account for computation or communication infrastructure, which limits its use in IoT-fog environments.

In real life, the SUMO-generated data would be replaced by data from devices such as microphones, RFID sensors, smart cameras, smart traffic lights, induction loops, connected vehicles (V2X), air-quality monitors, and other devices on roads. The iFogSim-modeled fog nodes would correspond to real edge servers or traffic management units installed at road intersections, capable of processing sensor streams in real time. The closed-loop system depicted in the right half of Figure 1, where data is collected, analyzed locally, and adaptive control signals are sent back to traffic lights or vehicles, can be directly implemented using real hardware and communication networks, as illustrated in the left half of Figure 1. This integration surely offers researchers a unified testing platform.

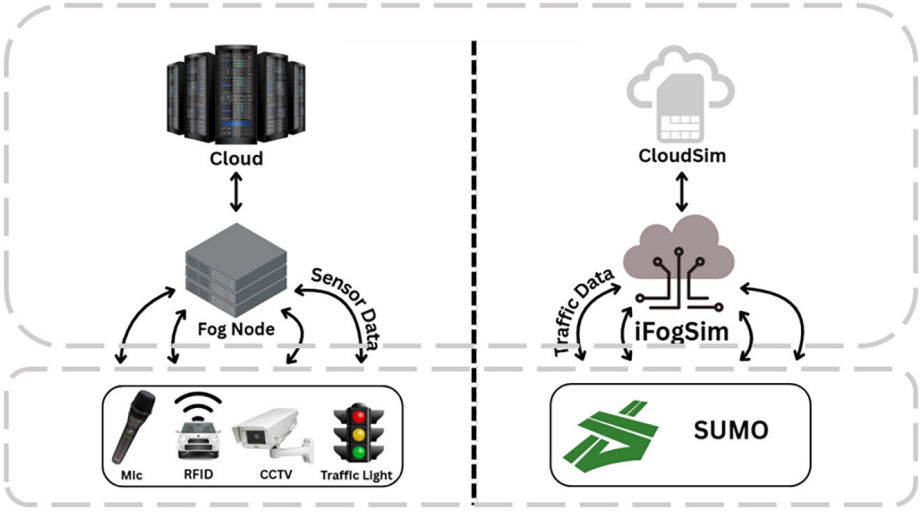


Fig. 1. Real world to simulators mapping

The proposed iFogSUMO framework consists of the following three major components, as shown in Figure 2:

1. **SUMO:** Simulates road networks, traffic lights, various types of vehicles, road routes, trips for each vehicle, and various traffic scenarios. It provides all this information to the middleware component via TracCI on port 8813.
2. **Middleware:** It acts as a bridge between SUMO and iFogSim. It is responsible for data conversion and translation of SUMO vehicle states into IoT events understandable by iFogSim. It handles bidirectional communication between vehicle data sent from SUMO to iFogSim and Fog-based decisions sent from iFogSim to SUMO for adaptive traffic control.
3. **iFogSim:** Simulates fog nodes, cloud nodes, IoT devices, and applications. It evaluates latency, energy consumption, and resource utilization for traffic management applications. It receives traffic-related IoT events from middleware through Socket port 9999 and generates the corresponding control actions.

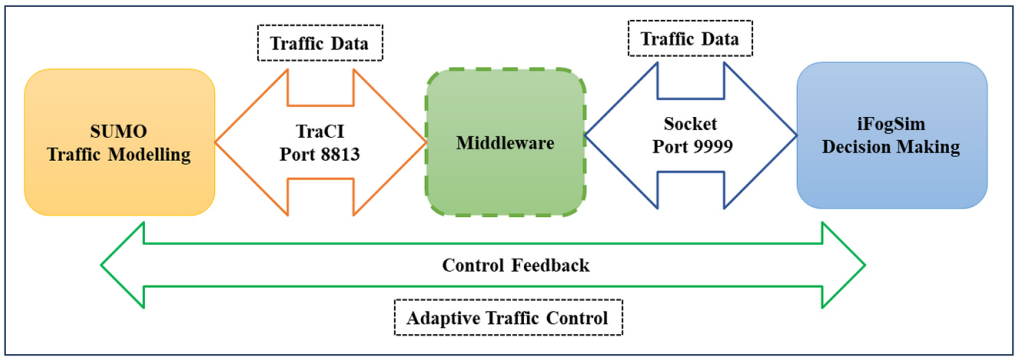


Fig. 2. Proposed framework for iFogSUMO

Figure 2 illustrates the integration workflow between SUMO and iFogSim, emphasizing the file conversions and communication mechanism. SUMO is implemented in C++ with Python-based control. It generates mobility outputs, including vehicle positions, speeds, and queue lengths, in XML/CSV formats. These raw outputs are not directly compatible with iFogSim, which is Java-based and expects IoT

event streams. To bridge this gap, a middleware layer (developed in Python or Java) is introduced. This middleware performs two key tasks:

i) Reading SUMO outputs via the TraCI API and converting them into IoT events (e.g., JSON objects), which are then forwarded to iFogSim.
ii) Receiving fog application decisions from iFogSim (e.g., adaptive traffic light updates and rerouting commands), converting them into TraCI-compatible commands, and sending them back to SUMO through the TCP/IP socket on port 9999.

Figure 2 illustrates the integration workflow between SUMO and iFogSim, emphasizing the file conversions and communication mechanism. SUMO is implemented in C++ with Python-based control. It generates mobility outputs, including vehicle positions, speeds, and queue lengths, in XML/CSV formats. These raw outputs are not directly compatible with iFogSim, which is Java-based and expects IoT event streams. To bridge this gap, a middleware layer (developed in Python or Java) is introduced. This middleware performs two key tasks:

i) Reading SUMO outputs via the TraCI API and converting them into IoT events (e.g., JSON objects), which are then forwarded to iFogSim.
ii) Receiving fog application decisions from iFogSim (e.g., adaptive traffic light updates and rerouting commands), converting them into TraCI-compatible commands, and sending them back to SUMO through the TCP/IP socket on port 9999.

This closed-loop interaction enables co-simulation, where SUMO provides real-time mobility data and iFogSim optimizes fog-based decision-making for smart traffic management. It requires careful handling of file formats, middleware-based conversions, and socket-based communication to enable seamless co-simulation. Table 2 provides details on the files used by each simulator and their purposes.

Table 2. File formats in SUMO and iFogSim with conversion requirements

File Type/Extension	Role in Simulation	Conversion Requirement
SUMO		
.net.xml	Road network description (roads, lanes, intersections, traffic lights).	Used directly by SUMO No conversion needed.
.sumocfg	Master configuration file linking networks,routes, and additional files.	Internal to SUMO; not transferred.
.rou.xml	Vehicle flows, routes, departure times, vehicle types.	Converted to IoT “vehicle sensor” events for iFogSim depending on the scenario.
.add.xml/.tls.xml	Traffic signal logic, additional simulation elements (e.g., detectors).	Detector data may be converted into IoT events for iFogSim.
.xml/.csv (outputs)	Trip details, travel time, emissions, waiting time, queue lengths.	Converted into IoT event streams (e.g., JSON objects) for iFogSim.
iFogSim		
.java source files	Core simulator code; models IoT devices, fog nodes, cloud resources.	No conversion; written directly by researchers.
Java class definitions (App topology)	Define sensors, actuators, application modules,and data flows.	Middleware maps SUMO outputs to these modules.
.csv/log outputs	Simulation results: latency, energy, network usage, QoS metrics.	Optionally converted into performance reports comparable with SUMO metrics.

The proposed co-simulation framework, iFogSUMO, was set up and implemented in four phases as illustrated in the high-level view in Figure 3. Fine-grained algorithms for each phase are defined subsequently.

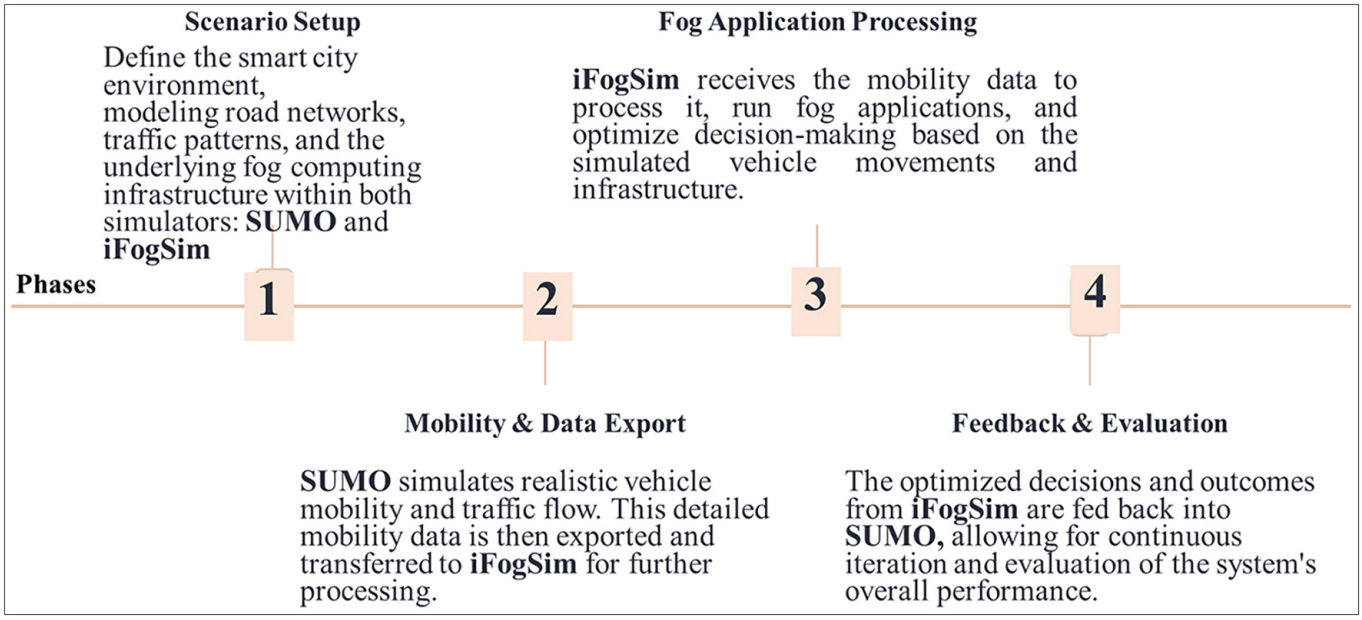


Fig. 3. Co-simulation phases of iFogSUMO

In **Phase 1**—Scenario Setup, the road network is designed in SUMO, including traffic light placement, while the fog infrastructure is modeled in iFogSim by defining fog nodes with specific computational and networking capabilities and mapping them to the traffic lights. The algorithms for Phase 1 to set up SUMO and iFogSim are illustrated in detail in Algorithms 1 and 2, respectively.

Algorithm 1: Initialize SUMO – Network and Random Traffic Generation

Input: Nodes_file, Edges_file, Types_file, Simulation_time, Trip_parameters
Output: Network_file (*.net.xml*), Random_trips_file (*.rou.xml*)

I. **Start Initialization**
II. **Step 1: Load Network Definition Files**
  a. Read Nodes_file containing intersections and junctions
  c. Read Edges_file defining roads and connectivity
  d. If Types_file exists, load default road type parameters (e.g., speed, lanes)
III. **Step 2: Generate Final Network**
  a. Run Netconvert command: *netconvert --node-files Nodes_file --edge-files Edges_file --type-files Types_file -o Network_file*
  b. Verify that Network_file (*.net.xml*) is generated successfully
  c. Configure Simulation Parameters
  d. Set total simulation duration ← Simulation_time
  e. Define simulation step length (e.g., 1.0 sec)
  f. Set vehicle generation parameters: number of vehicles, departure interval, vehicle types
IV. **Step 3: Generate Random Traffic Demand**
  a. Execute randomTrips.py script: *python randomTrips.py -n Network_file -o Random_trips_file --begin 0 --end Simulation_time --period <departure_rate> --random*
  b. Validate that Random_trips_file (*.rou.xml*) is created successfully
V. **End Initialization**

Algorithm 2: Initialize iFogSim with SUMO Junctions and Emergency Routing

Input: Sumo_network_file (*.net.xml*), Hospital_nodes, Traffic_light_nodes
Output: Fog nodes mapped from junctions, shortest paths with stored junction sequences

I. **Start Initialization**
II. **Step 1: Import SUMO Junctions**
    a. Parse SUMO_network_file to extract all junctions
    b. For each junction *J_i*, store ID, geographic coordinates (x, y), and connected edges
    c. Step 2: Map Junctions to Fog Nodes
    d. For cluster of: Create fog node *F_i* in iFogSim
    e. Assign fog node attributes: Processing power (MIPS), Memory, Bandwidth, Location = (x, y)
III. **Step 2: Build Graph**
    a. Represent SUMO network as graph *G(V, E)*
    b. Nodes *V* = set of all junctions
    c. Edges *E* = road segments between junctions with weights (distance or travel time)
IV. **Step 3: Define Dynamic Traffic Control Algorithm(s)** //as in Algorithm 4
V. **End Initialization**

In **Phase 2**—The Mobility and Data Export: Vehicle movements are simulated in SUMO, and traffic-related data such as speed, density, waiting time, vehicle type, and position are exported in real-time to iFogSim.

In **Phase 3**—Fog Application Processing: The fog nodes ingest this data, process it locally, and make intelligent decisions such as dynamic traffic signal adjustments, congestion mitigation strategies, or ambulance route prioritization.

Phases 2 and 3 execute in parallel at runtime, integrating the two simulators in the iFogSUMO setup. Algorithm 3 illustrates the same. Algorithm 4 is the Dynamic Traffic Control algorithm that turns the traffic lights green upon detecting an ambulance along its path.

Algorithm 3: SUMO and iFogSim Co-simulation Process

Input: Sumo_network_file (*.net.xml*), Routes_file (*.rou.xml*), Fog_nodes, Path_database (junction-to-hospital paths)
Output: Simulation metrics including trip information, waiting times, and emergency routing statistics

I. **Start Co-simulation**
II. **Step 1: Initialize SUMO and iFogSim**
    a. Establish Communication
    b. Connect SUMO and iFogSim via TraCI interface
    c. Define data exchange protocol:
        i. SUMO sends vehicle states: position, speed, route ID
        ii. iFogSim sends traffic control decisions to SUMO
III. **Step 2**: **Main Simulation Loop**
    a. While Simulation_Time < Simulation_end do
        i. Advance SUMO simulation by one step
        ii. Extract vehicle states from SUMO
        iii. Send data to iFogSim for processing
        iv. Update fog nodes with traffic data
        v. Run decision-making applications on fog nodes
        vi. Compute optimal control action:
            - Adjust traffic lights dynamically
        vii. Send control actions back to SUMO via TraCI
        viii. Update vehicle routing and signal states
IV. **Step 3: Collect performance metrics**
    a. Extract trip time, waiting time
V. **Step 4: Finalize and Save Outputs**
    a. Save trip statistics to *tripinfo.xml*
    b. Save fog metrics (CPU usage, latency, network load)
    c. Export final emergency routing performance results
VI. **End Co-simulation**

Algorithm 4: Ambulance Routing //Dynamic Traffic Control Algorithm

Input: Sumo_network_file (*.net.xml*), Hospital_nodes, Traffic_light_nodes
Output: Fog nodes mapped from junctions, shortest paths with stored junction sequences

1. **Start Ambulance Routing**
2. **Step 1: Build Graph for Shortest Path Computation**
    a. Represent SUMO network as graph *G(V, E)*
    b. Nodes *V* = set of all junctions
    c. Edges *E* = road segments between junctions with weights (distance or travel time)
3. **Step 2: Compute Shortest Path for Emergency Routing**
    a. For each traffic light node *T_j* in Traffic_light_nodes
        i. For each hospital node *H_k* in Hospital_nodes
        ii. Calculate shortest path *P_jk* using Dijkstra's algorithm
        iii. Select hospital *H_min* with minimum path cost
        iv. Store final path *P_jH*_min for traffic light *T_j*
        v. Save ordered list of junctions in *P_jH_min* to database
4. **Step 3:** Store Paths for Real-Time Routing
    a. Link fog nodes corresponding to junctions in each path
    b. Mark as priority route for emergency vehicles
5. **End Ambulance Routing**

Finally, in **Phase 4**—Feedback and Evaluation: The processed outputs and control decisions are sent back to SUMO to alter traffic signal timings, reroute vehicles, or update mobility patterns. Additionally, performance metrics from SUMO (e.g., average waiting time, throughput) are derived from *tripinfo.xml* log files. These files store the information in the format below for each vehicle:

<tripinfo id="13" depart="26.00" departLane="10_down3_0" departPos="5.10" departSpeed="0.00" departDelay="0.00" arrival="37.20" arrivalLane="10_down3_0" arrivalPos="119.83" arrivalSpeed="15.18" duration="11.20" routeLength="114.73" waitingTime="0.00" waitingCount="0" stopTime="0.00" timeLoss="3.71" rerouteNo="0" devices="tripinfo_13" vType="DEFAULT_VEHTYPE" speedFactor="1.12" vaporized=""/>

These performance parameters are evaluated from this file using Equations 1 to 5 for SUMO.

$$Trip\ Duration = Arrival\ Time - Departure\ Time \tag{1}$$

where *Trip Duration* is the difference between the *Arrival Time* at the destination and the *Departure Time* from the source for a vehicle during the trip.

$$Waiting\ Time = \sum_{t=0}^{T} \Delta t \ \ here\, v(t) < v_{threshold} \tag{2}$$

Where Δt is the time interval, which is 1 sec by default in SUMO, $v(t)$ is the vehicle speed at time t, and $v_{threshold}$ is the minimum expected speed below which it is considered to be waiting or not moving. The sum of this waiting time during the interval $t = 0$ to T (the total simulation time) is the waiting time of the vehicle.

$$Time\ Loss = Trip\ Duration - Ideal\ Travel\ Time \tag{3}$$

$$Ideal\ Travel\ Time = \frac{Route\ Length}{Max\ Allowed\ Speed} \tag{4}$$

$$Route\ Length = \sum_{e=1}^{n} Edge\ Length(e) \tag{5}$$

where, *Trip Duration* is the total time taken by a vehicle to travel from its departure point to its destination. It includes driving time, waiting at the signals, congestion, and delays. *Ideal Travel Time* is the minimum possible travel time assuming free-flow traffic conditions, i.e., the vehicle moves at the maximum allowed speed without stopping, which is calculated as in Equation 4, and *Route Length* is the sum total of edges' length in the route. *Time Loss* is the extra time spent due to congestion, traffic lights, and other delays.

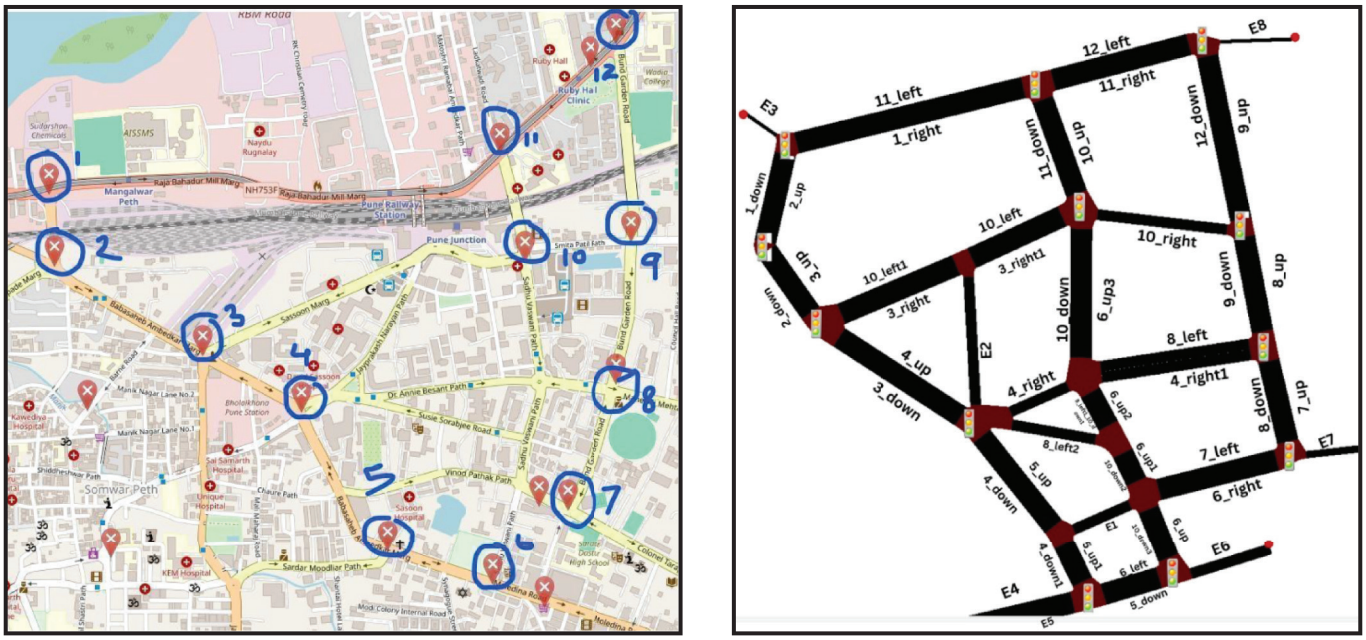
In the proposed iFogSUMO framework, synchronization between SUMO and iFogSim is maintained through a TCP/IP socket interface, which provides reliable, ordered, and loss-free message delivery. Consequently, no packet losses occur during data exchange. The primary overhead arises from data serialization and format conversion (e.g., from SUMO's XML or JSON output to the Java object representation used in iFogSim). This overhead remains constant and predictable, as the structure and size of exchanged messages are fixed per simulation step. Therefore, the synchronization delay introduced by the communication layer is negligible compared to the overall fog processing and network transmission delays modeled within iFogSim.

4 EXPERIMENTAL SETUP AND RESULTS

For testing the co-simulation iFogSUMO, the scenario considered was routing an ambulance from any location to a hospital by turning the traffic lights along its path green immediately after detection. The objective was to minimize the ambulance's travel time, waiting time, and time loss. The traffic lights for the cluster of junctions were mapped to one Fog node. This Fog node stores the sequence of junctions from the source to the destination, which are to be turned green for the passage of an ambulance.

Implementation was carried out in the phases described in Figure 3. In this section, we elaborate on each phase. To validate the integration of the two simulators, the experiment was also conducted using only SUMO, with static signal timing managed solely by SUMO, and the results were compared with dynamic signal timing achieved through the integration of SUMO and iFogSim in iFogSUMO.

A small section of the city map from OpenStreetMap with multiple traffic lights was captured (see Figure 4a). This map was replicated using the *netedit* utility in SUMO, keeping the number of traffic junctions the same and scaling the road segments. The edges were numbered with reference to the junction number (see Figure 4b). For junction number 4, the connected edges were numbered 4_up, 4_right, and 4_down.



a) Section Captured from OpenStreetMap

b) Equivalent image in SUMO

Fig. 4. Road network

As per Algorithms 1 and 2, all required files *.net.xml*, *.rou.xml*, and *.sumocfg* were created for this road network in SUMO. Vehicle trips were generated using SUMO's *randomTrips.py* tool for vehicles of type 'car.' It was ensured that one trip was generated for the vehicle type 'ambulance.' During simulation, traffic lights were grouped and mapped to four Fog Nodes as shown in Figure 5. The entire road graph was replicated in iFogSim. The shortest path from each edge-to-edge E8 using Dijkstra's algorithm [28] in iFogSim was determined and saved in the Fog Nodes as in Algorithm 4.

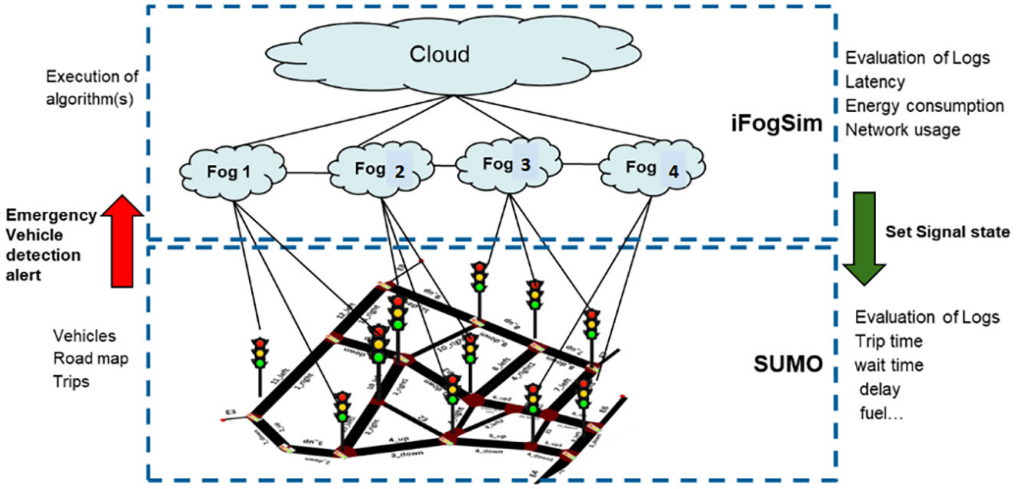
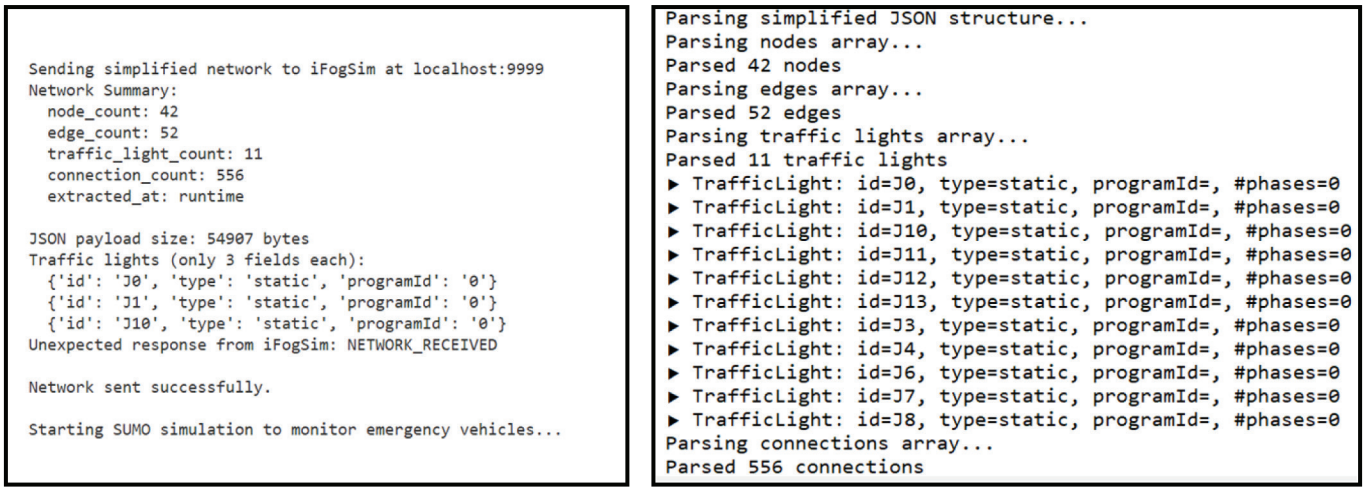


Fig. 5. System architecture

During simulation, vehicle parameters, such as *type*, *speed*, *position*, *lane occupancy*, *acceleration*, and *waiting time,* were continuously extracted using *TraCI* and sent to the middleware. A socket-based communication channel was employed to

transmit traffic state information from the middleware layer; mainly, the detection of the ambulance, its edge, and traffic density at the edges was communicated to iFogSim.

A fog node, upon receiving an ambulance detection message, retrieves the prestored list of junctions along the shortest path. Through the middle layer, a message was sent to SUMO to turn the traffic lights green for these junctions. Sumo handles synchronizing the non-conflicting phases of traffic lights. The road network details, such as *node_count*, *edge_count*, *traffic_light_count*, and *connection_count*, were successfully extracted and shared by SUMO with iFogSim and are validated by Figures 6a and 6b, respectively.



a) SUMO Sending Road Network to iFogSim **b) iFogSim Receiving Road Network from SUMO**

Fig. 6. Communication messages between SUMO and iFogSim

At the end of the simulation, SUMO generated *tripinfo* log files (one with iFogSim co-simulation and one without) that provided detailed performance metrics for every vehicle in the network. This XML file records essential parameters, such as the *vehicle ID*, *departure time*, and *arrival time*, used to calculate the *total trip duration*. It also includes *waiting time*, representing the cumulative time a vehicle spends waiting. The time-loss parameter quantifies the extra time a vehicle takes compared to its ideal travel time under free-flow conditions, providing a measure of congestion levels. These parameters collectively provide a comprehensive dataset for assessing traffic flow performance, signal control strategies, and the effectiveness of dynamic traffic control for the scenarios.

The integration setup was evaluated using varying traffic volumes of 1000, 1500, 2000, and 2500 vehicles, encompassing multiple modes of transport such as cars, bikes, buses, and an ambulance. The simulations were conducted on the road network depicted in Figure 4b.

Table 3 consolidates the exact values extracted from the *tripinfo.xml* log files. These log files were generated through repeated simulations conducted in two scenarios: (i) using SUMO alone with *static signal control* and (ii) using the proposed iFogSUMO architecture implementing *Adaptive Traffic Control*. In the table, C represents the *Vehicle Count*, ATT denotes the *Average Trip Time*, AWT stands for the *Average Waiting Time*, and S indicates the *Number of Stops*.

Table 3. Comparative results of SUMO and iFogSUMO for multiple simulations

Total No. of Vehicle	1000				1500				2000				2500			
Type	C	ATT	AWT	S	C	ATT	AWT	S	C	ATT	AWT	S	C	ATT	AWT	S
SUMO																
Ambulance	1	465	350	4	1	305	160	3	1	455	345	3	1	465	320	9
Bike	300	144	30	2	450	142	27	2	600	140	26	2	750	140	26	2
Bus	100	125	29	2	150	128	32	2	200	123	29	2	250	124	29	2
Car	599	141	32	2	899	141	32	2	1199	140	32	2	1499	140	31	2
iFogSUMO																
Ambulance	1	105	23	1	1	105	23	1	1	105	23	1	1	105	23	1
Bike	300	317	165	6	450	476	306	10	600	625	447	10	750	890	687	14
Bus	100	313	179	6	150	440	289	9	200	547	386	12	250	788	607	14
Car	599	273	143	5	899	394	248	8	1199	526	373	9	1499	835	656	14

These comparative results indicate that iFogSUMO consistently outperforms SUMO in reducing ambulance travel time, wait time and reducing the number of stops to 1. This indicates that after an ambulance is detected, iFogSUMO turns all the signals green along the path, so it doesn't have to wait later. There is an average travel time reduction of almost 75%. It is notable that the count of vehicles remains consistent in both the simulations, indicating no message loss.

As desired, the ambulance's waiting time in the adaptive traffic control scenario, when simulators are integrated, is negligible and has dropped drastically as the signals along the path turn green immediately upon detection, which eliminates any time loss, as shown in the plot of Figure 7.

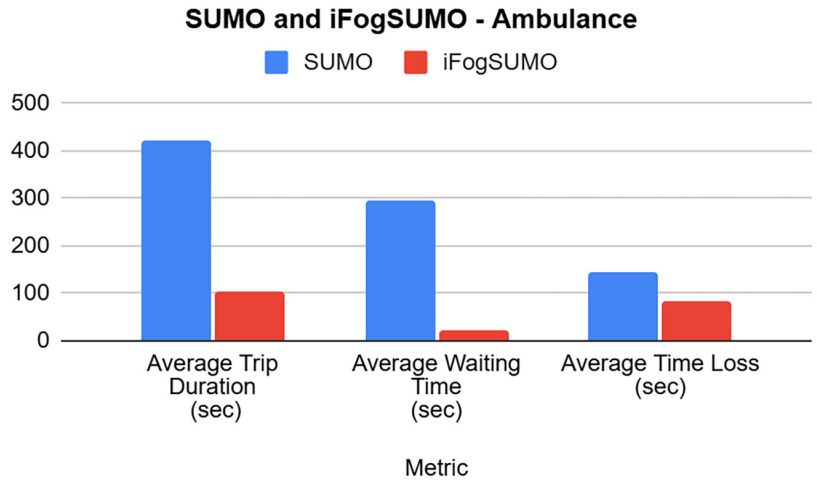


Fig. 7. Ambulance performance metrics

Figure 8 shows that the drastic increase in trip duration, waiting time, and time loss for non-emergency vehicles (cars, buses, and bikes) is a direct consequence of traffic signal preemption, which grants uninterrupted progression to the ambulance. This phenomenon is not merely a simulation artifact but a well-documented

real-world outcome of emergency vehicle priority systems: when traffic signals are held in green along the ambulance's path, cross-traffic flows are starved, leading to queue buildup, signal cycle imbalance, and cascading congestion—effects empirically observed in field studies using V2X-enabled preemption in urban environments [29] and adaptive preemption strategies in high-density corridors [30].

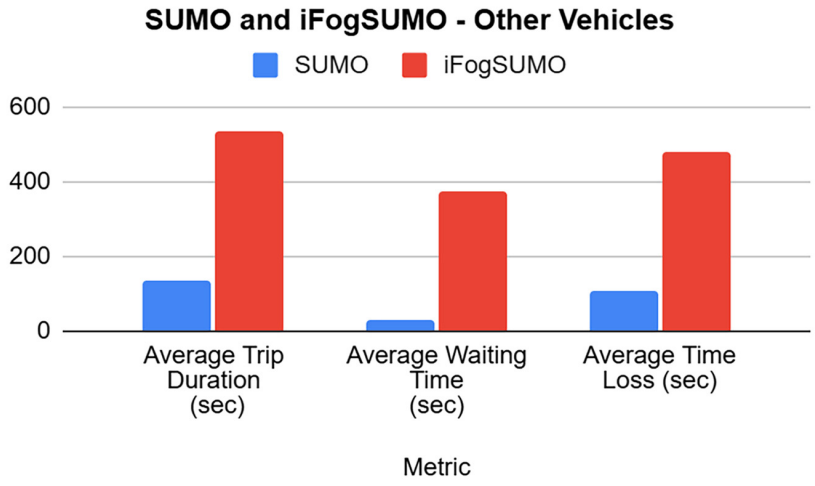


Fig. 8. Other vehicles performance metrics

The operational efficacy of iFogSUMO is validated through three key performance indicators observed across all simulation scenarios (1000–2500 vehicles). First, ambulance preemption is conclusively demonstrated: the number of stops is consistently reduced to 1, with trip time and waiting time decreased by ~50% and ~95%, respectively, confirming real-time, fog-orchestrated signal coordination. Second, consistent vehicle counts across runs ensure reproducibility and controlled experimental conditions with no message loss. Third, while average trip time, waiting time, and stops increase for non-emergency vehicles (cars, buses, and bikes), these rises also validate that the traffic lights along the other routes were adaptively adjusted to allow passage of the ambulance, which is the expected trade-off of prioritization of the emergency vehicle. Collectively, these outcomes—priority enforcement, experimental consistency, and controlled collateral impact—robustly validate the correctness, reliability, and practical utility of the iFogSUMO co-simulation framework for adaptive, emergency-aware traffic management.

5 CONCLUSION AND FUTURE WORK

This study presents iFogSUMO, an integrated co-simulation framework that enhances SUMO's native capabilities through real-time fog-based computation. By seamlessly coupling SUMO's granular traffic modeling with iFogSim's distributed resource management, iFogSUMO enables adaptive traffic signal control with substantially reduced decision-making latency. The framework efficiently supports heterogeneous vehicular environments—including cars, buses, bikes, and an ambulance—while maintaining minimal computational overhead and ensuring zero message loss even under high communication load. Extensive evaluations conducted across simulation scales ranging from 1000 to 2500 vehicles demonstrate that iFogSUMO eliminates ambulance waiting times, reduces overall time loss by

approximately 95%, and decreases average trip duration by 50% for a small road network, reducing stops to 1, with consistent vehicle counts across runs validating working of iFogSUMO.

To enhance scalability and general applicability in large urban environments, future work will evaluate iFogSUMO on city-wide networks with thousands of intersections and diverse layouts, using real traffic datasets. We will conduct detailed fog resource profiling (CPU, memory, latency, network overhead) under resource-constrained edge deployments and task allocation scenarios. Validation will include multimodal integration (pedestrians, cyclists, and public transit). Incorporating OMNeT++/Veins will enable V2X communication testing for cooperative ITS. Reinforcement learning at fog nodes will support predictive, adaptive control and anomaly detection. These extensions will empirically confirm performance and real-world feasibility across complex, heterogeneous smart city scenarios.

6 ACKNOWLEDGMENTS

The authors thank Simran Veer and Yash Hodlur, students at COEP Technological University, Pune, for their assistance in experimental testing.

7 REFERENCES

[1] S. Veer, S. Ravichandran, P. Trivedi, J. Abraham, and A. Matange, "A review of traffic-induced stress mitigation through adaptive traffic signaling and fog computing," in *Proc. 5th Int. Conf. Pervasive Comput. Social Netw. (ICPCSN)*, Salem, India, 2025, pp. 1215–1221. https://doi.org/10.1109/ICPCSN65854.2025.11035795

[2] X. Ma *et al.*, "Evaluation of accuracy of traffic flow generation in SUMO," *Appl. Sci.*, vol. 11, no. 6, p. 2584, 2021. https://doi.org/10.3390/app11062584

[3] N. Rida, M. Ouadoud, and A. Hasbi, "Coordinated signal control system in urban road network," *Int. J. Online Biomed. Eng. (iJOE)*, vol. 16, no. 10, pp. 4–22, 2020. https://doi.org/10.3991/ijoe.v16i10.15473

[4] T. Azfar, "Traffic co-simulation framework empowered by CARLA and SUMO," *arXiv preprint arXiv:2412.03925*, 2024. https://doi.org/10.48550/arXiv.2412.03925

[5] M. Fahimullah *et al.*, "A review of fog computing and its simulators," *J. Cloud Comput. Adv. Syst. Appl.*, vol. 12, no. 1, pp. 1–18, 2023.

[6] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the Internet of Things," in *Proc. 1st MCC Workshop Mobile Cloud Comput.*, 2012, pp. 13–16. https://doi.org/10.1145/2342509.2342513

[7] M. Chiang and T. Zhang, "Fog and IoT: An overview of research opportunities," *IEEE Internet Things J.*, vol. 3, no. 6, pp. 854–864, 2016. https://doi.org/10.1109/JIOT.2016.2584538

[8] C. Zaoui, F. Benabbou, and A. Ettaoufik, "Edge–Fog–Cloud data analysis for eHealth-IoT," *Int. J. Online Eng. (iJOE)*, vol. 19, no. 7, pp. 184–199, 2023. https://doi.org/10.3991/ijoe.v19i07.38903

[9] A. Zanella, N. Bui, A. Castellani, L. Vangelista, and M. Zorzi, "Internet of Things for smart cities," *IEEE Internet Things J.*, vol. 1, no. 1, pp. 22–32, 2014. https://doi.org/10.1109/JIOT.2014.2306328

[10] K. Micko, P. Papcun, and I. Zolotova, "Review of IoT sensor systems used for monitoring the road infrastructure," *Sensors*, vol. 23, no. 9, p. 4469, 2023. https://doi.org/10.3390/s23094469

[11] F. Y. Alzyoud, A. A. Alnuaimi, and F. Al Shrouf, "Adaptive smart traffic accidents management system," *Int. J. Interact. Mobile Technol. (iJIM)*, vol. 15, no. 14, pp. 72–89, 2021. https://doi.org/10.3991/ijim.v15i14.19099

[12] H. Gupta, A. V. Dastjerdi, S. K. Ghosh, and R. Buyya, "iFogSim: A toolkit for modeling and simulation of resource management techniques in Internet of Things, edge and fog computing environments," *Software: Practice and Experience*, vol. 47, no. 9, pp. 1275–1296, 2017. https://doi.org/10.1002/spe.2509

[13] S. Yi, Z. Hao, Z. Qin, and Q. Li, "Fog computing: Platform and applications," in *Proc. 3rd HotWeb Workshop*, 2015. https://doi.org/10.1109/HotWeb.2015.22

[14] C. Puliafito *et al.*, "Fog computing for the Internet of Things: A survey," *ACM Comput. Surveys*, vol. 52, no. 3, pp. 1–36, 2019. https://doi.org/10.1145/3204947

[15] S. Sarkar and S. Misra, "Theoretical modelling of fog computing: A green computing paradigm to support IoT applications," *IET Netw.*, vol. 5, no. 2, pp. 23–29, 2016. https://doi.org/10.1049/iet-net.2015.0034

[16] P. Lopez *et al.*, "Microscopic traffic simulation using SUMO," in *Proc. IEEE Intell. Transp. Syst. Conf. (ITSC)*, 2018, pp. 2575–2582. https://doi.org/10.1109/ITSC.2018.8569938

[17] M. Behrisch, L. Bieker, J. Erdmann, and D. Krajzewicz, "SUMO—Simulation of Urban Mobility: An overview," in *Proc. 3rd Int. Conf. Adv. Syst. Simulation (SIMUL)*, 2011, pp. 55–60.

[18] R. M. Rodriguez *et al.*, "Smart traffic light control with reinforcement learning," *Appl. Sci.*, vol. 10, no. 1, p. 233, 2020.

[19] D. Seo *et al.*, "Dynamic iFogSim: A framework for full-stack simulation of dynamic resource management in IoT systems," in *Proc. Int. Conf. Omni-layer Intell. Syst. (COINS)*, 2020, pp. 1–7. https://doi.org/10.1109/COINS49042.2020.9191663

[20] Z. Wei *et al.*, "AirFogSim: A light-weight and modular simulator for UAV-integrated vehicular fog computing," *IEEE Transactions on Mobile Computing*, 2024. https://doi.org/10.1109/TMC.2025.3641373

[21] F. Dressler and C. Sommer, "Using the right simulator for the job: A case study on vehicular network evaluation," *IEEE Commun. Mag.*, vol. 52, no. 11, pp. 132–139, 2014.

[22] S. Bajpai, G. K. Sahoo, S. K. Das, and P. Singh, "An efficient inter-vehicle communication framework on road traffic accident detection using OMNET++ and SUMO," in *Proc. IEEE Int. Symp. Sustain. Energy, Signal Process. Cyber Secur. (iSSSC)*, Gunupur, India, 2022, pp. 1–6. https://doi.org/10.1109/iSSSC56467.2022.10051410

[23] R. A. Pratama *et al.*, "Performance evaluation on VANET routing protocols in the way of central Jakarta using ns-3 and SUMO," in *Proc. Int. Seminar Appl. Technol. Inf. Commun. (iSemantic)*, 2020. https://doi.org/10.1109/iSemantic50169.2020.9234202

[24] K. H. M. Gularte *et al.*, "Integrating cybersecurity in V2X: A review of simulation environments," *IEEE Access*, vol. 12, pp. 177946–177985, 2024. https://doi.org/10.1109/ACCESS.2024.3504404

[25] M. Klischat, O. Dragoi, M. Eissa, and M. Althoff, "Coupling SUMO with a motion planning framework for automated vehicles," *EPiC Ser. Comput.*, vol. 62, pp. 1–9, 2019. https://doi.org/10.29007/1p2d

[26] M. Franchi *et al.*, "Webots.HPC: A parallel simulation pipeline for autonomous vehicles on high performance computing," in *Proc. ACM/IEEE Conf.*, 2022, pp. 1–4. https://doi.org/10.1145/3491418.3535133

[27] H. Zhang *et al.*, "CityFlow: A multi-agent reinforcement learning environment for large-scale city traffic scenario," *arXiv preprint arXiv:1905.05217*, 2019.

[28] Z. Grujic and B. Grujic. "Optimal routing in urban road networks: A graph-based approach using Dijkstra's algorithm," *Appl. Sci.*, vol. 15, no. 8, p. 4162, 2025. https://doi.org/10.3390/app15084162

[29] M. Maile, F. Ahmed-Zaid, and S. Bai, "Influence of emergency vehicle preemption on travelling time and traffic safety in urban environments enabled by innovative behavioral models and V2X communication—Simulation and case study," in *Proc. IEEE Intell. Transp. Syst. Conf. (ITSC)*, Auckland, New Zealand, 2019, pp. 2705–2710. https://doi.org/10.1109/ITSC.2019.8916890

[30] S. Kleitsch, M. Budzynski, and M. Ferreira, "Adaptive preemption of traffic for emergency vehicles," in *Proc. IEEE Intell. Transp. Syst. Conf. (ITSC)*, Maui, HI, USA, 2018, pp. 1–6. https://doi.org/10.1109/ITSC.2018.8359042

8 AUTHORS

Ashwini Matange is a Research Scholar in the Department of Computer Science and Engineering, COEP Technological University, Pune, India, and an Assistant Professor in Computer Engineering Department, Pimpri Chinchawad College of Engineering, Pune, India (E-mail: matangea21.comp@coeptech.ac.in).

Jibi Abraham is a Professor in the Department of Computer Science and Engineering, COEP Technological University, Pune, India (E-mail: ja.comp@coeptech.ac.in).