SUCCESS-6G: EXTEND

WP5 Deliverable E14

Project Title:	SUCCESS-6G: EXTEND
Title of Deliverable:	Final testing and validation of service KPIs
Status-Version:	v1.0
Delivery Date:	30/04/2025
Contributors:	Allen Abishek, Ricard Vilalta, Raul Muñoz (CTTC), Miguel Fornell, Francisco Paredes (Idneo), Angelos Antonopoulos (Nearby Computing)
Lead editor:	CTTC
Reviewers:	Charalampos Kalalas (CTTC), Francisco Paredes (Idneo)
Keywords:	Real-time location awareness; end-to-end latency; closest node selection

Document revision history

Version	Date	Description of change
v0.1	20/02/25	Table of Contents (ToC)
v0.2	28/02/25	Content added
v0.3	24/03/25	Additional inputs
v0.4	16/04/25	Final inputs and revision
v1.0	30/04/25	Final version uploaded to the website

Disclaimer

Acknowledgment

The research conducted by SUCCESS-6G - TSI-063000-2021-39/40/41 receives funding from the Ministerio de Asuntos Económicos y Transformación Digital and the European Union-NextGenerationEU under the framework of the “Plan de Recuperación, Transformación y Resiliencia” and the “Mecanismo de Recuperación y Resiliencia”.

[1] http://creativecommons.org/licenses/by-nc-nd/3.0/deed.en_US

Executive Summary

Ensuring robust and seamless Vehicle-to-Everything (V2X) connectivity is crucial for delivering reliable over-the-air (OTA) software updates to connected vehicles. SUCCESS-6G-EXTEND integrates advanced networking solutions such as Software-Defined Networking (SDN) and Multi-access Edge Computing (MEC) to optimize network performance and reduce latency. By leveraging real-time location-awareness, dynamic resource allocation, and AI-driven orchestration, the system ensures resilient update delivery even under varying network conditions. This deliverable highlights significant improvements in update success rates, transmission reliability, and real-time network adaptation, demonstrating the potential of SUCCESS-6G-EXTEND in revolutionizing OTA update methodologies for connected and autonomous vehicles.

Table of Contents

List of Figures

1 Introduction

The increasing reliance on software-driven functionalities in modern vehicles necessitates a robust and efficient method for delivering over-the-air (OTA) software updates. As connected and autonomous vehicle ecosystems continue to evolve, manufacturers and service providers must ensure timely software deployment without requiring physical intervention. Vehicle-to-Everything (V2X) communication is a key enabler of this transformation, facilitating seamless and reliable OTA updates by leveraging advanced wireless networking technologies. However, ensuring uninterrupted connectivity, mitigating security risks, and optimizing computational resources remain significant challenges. The SUCCESS-6G-EXTEND project aims to address these challenges through the integration of Software-Defined Networking (SDN), and Multi-access Edge Computing (MEC) network orchestration.

A fundamental requirement for OTA software updates is robust connectivity, which is often hindered by varying network conditions, congestion, and latency. Cellular V2X (C-V2X) technology, enabled by 5G and edge computing, presents an efficient solution for overcoming these challenges by dynamically optimizing update delivery paths and enhancing service resilience. The SUCCESS-6G-EXTEND framework incorporates the ETSI TeraFlowSDN controller to manage and optimize transport networks, ensuring efficient routing and resource allocation. Additionally, the introduction of location-awareness within SDN and MEC environments allows for geographically optimized routing, reducing latency and improving update success rates.

This deliverable presents results on the implementation and validation of OTA software updates within a robust V2X connectivity framework. Experimental evaluations conducted on the ADRENALINE testbed demonstrate significant improvements in update efficiency, network resilience, and security enforcement. The integration of SDN, MEC, and security mechanisms has resulted in a scalable and adaptive solution capable of addressing the evolving demands of connected vehicle ecosystems. The findings from this research highlight the potential of SUCCESS-6G-EXTEND in revolutionizing vehicular software update methodologies. By leveraging cutting-edge networking and security technologies, the proposed framework ensures that vehicles remain updated with the latest software versions while minimizing cybersecurity risks and operational disruptions. As the automotive industry continues to transition towards fully connected and autonomous systems, the implementation of robust, secure, and efficient OTA update mechanisms will be instrumental in enhancing vehicle performance, safety, and regulatory compliance.

The subsequent sections focus on the specific methodologies employed, experimental setup, and detailed preliminary evaluations of the proposed OTA update system, providing a comprehensive analysis of its benefits and potential industry applications.

2 Use case 2: Automated software updates for vehicles

2.1 General description and overall objectives

Over-the-air software updates are delivered remotely from a cloud-based server, through a cellular connection, to the connected vehicle with the aim of providing new features and updates to the vehicle's software systems. Such software updates may include changes to any software that controls the vehicle's physical parts or electronic signal processing system. In practice, the updates often tend to apply more to user interfaces like infotainment screens and navigation (i.e., vehicle maps). The update procedure, when performed over the air, enables a vehicle's performance and features to be continuously up-to-date and improved. The integration of advanced data analytics, automated and remote service delivery eliminates the need for visiting repair/service centres, while technological advancements in these updates give vehicle manufacturers the freedom to constantly "freshen up" finished products remotely. C-V2X technology plays a crucial role in the update process, enabling efficient, scalable, and seamless wireless communication between vehicles and software management platforms. Figure 1 illustrates the implementation phases for this use case.

Figure 1 Implementation phases for the automated software updates

The overall **objectives** of this use case can be summarized as follows:

- Safer and more entertaining driving experience.
- Hardware and software components maintained and updated regularly during a vehicle's lifespan, implying a slower rate of depreciation.
- Prevention of cyberattacks targeting outdated software.
- Compliance to new rules and standards.
- Lower repair costs and elimination of labour charges.
- Lower warranty costs for manufacturers and lower downtime for customers

The key **stakeholders** involved in the use case are:

- The ***Mobile Network Operator (MNO)***, providing wireless connectivity between the vehicle, the edge computing infrastructure, and the vehicular software management system. The MNO is interested in optimizing the network operation by enhancing its energy efficiency and coverage, while offering novel services to accommodate more users.
- The ***edge infrastructure provider***, offering and managing computational resources at the edge and supporting real-time services as well as virtualized network functions and AI-empowered algorithms for advanced computational tasks.
- The ***equipment provider,*** providing in-vehicle embedded devices, e.g., hardware components and sensor devices, that can be remotely reconfigured and updated.
- The ***vehicular software management system,*** operated by the equipment provider or vehicle manufacturer, is responsible for issuing periodically new software updates.
- The ***software developers***, devising and applying data-processing modules for automated update of vehicular components' software.
- The ***cloud providers*** can optionally be involved, offering additional computational resources to host the service.

Note that, without loss of generality, some stakeholders may assume multiple roles or, equally, some roles may be assumed by multiple stakeholders. For instance, the MNO could also be the owner of the

edge infrastructure, or an equipment provider may also be responsible for the operation of the vehicular software management system or outsource it to a third party.

2.2 User story 2.1: Over-the-air vehicular software updates with robust V2X connectivity

Over-the-air software updates deliver critical information to onboard vehicular devices. Wireless channel impairments may, however, adversely impact access to up-to-date content, issues' remediation, and availability of new vehicular features. Therefore, a robust V2X connectivity needs to be established among the vehicular software management system, the cloud/edge infrastructure, and the vehicle. To this end, edge-specific orchestration of dynamic vehicular software upgrades is necessary to address potential connectivity failures and re-provision *on-the-fly* the automation of updates. Service continuity can be thus guaranteed by properly balancing the load across the edge infrastructure to ensure lifecycle management. Robustness of the software update service can be further enhanced via a zero-touch closed loop.

2.3 Overall UC2 architecture and network deployments

The elaboration of Figure 2 details a system architecture specifically designed for Over-the-air (OTA) Software Updates, integral to the SUCCESS-6G framework. This architecture addresses the complex requirements of Use Case 2. Figure 2 Proposed overall UC2 system architectureprovides a high-level system architecture for OTA vehicular software updates within a robust V2X connectivity framework, leveraging ETSI TeraFlowSDN for network automation and control. The figure illustrates the key components enabling software update dissemination to connected vehicles via 5G mobile edge computing (MEC) nodes.

At the core of this system is the ETSI TeraFlowSDN Controller, which manages the network infrastructure, including the gNBs (5G base stations) and Transport Network. The NFV Orchestrator (NFV-O) enables dynamic deployment and scaling of virtualized network functions, such as Distributed User Plane Functions (D-UPF) within MEC nodes.

Each edge node (Edge Node 1 & Edge Node 2) hosts a Software Update Server, responsible for caching and distributing updates to C-V2X On-Board Units (OBU) in connected vehicles. These updates are delivered via the 5G network, passing through the transport network, controlled by the TeraFlowSDN controller.

To ensure security and integrity, the system integrates a Security-as-a-Service module, providing firewall protection and secure communications for software updates. The updates originate from local cloud infrastructure, which includes 5G Core Control Plane components such as SMF (Session Management Function), AMF (Access and Mobility Management Function), and UPF (User Plane Function).

The software update client within the vehicle's C-V2X OBU interacts with the Software Update Servers over the network, ensuring efficient and timely delivery of critical updates for vehicle applications.

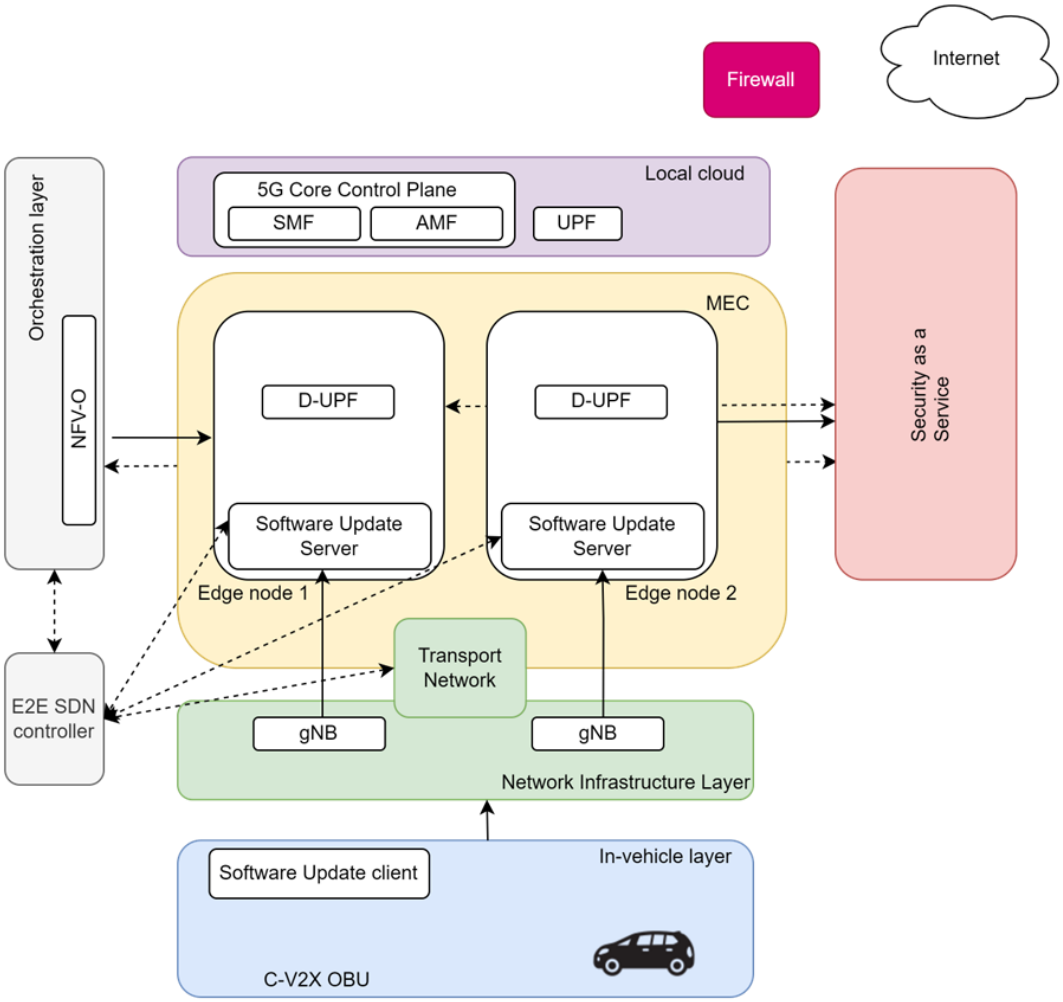

Figure 2 Proposed overall UC2 system architecture

This architecture highlights the interplay between 5G, MEC, SDN, and V2X technologies to facilitate secure and efficient OTA software updates, enabling reliable vehicle connectivity and automation.

2.4 Facilities for Use Case 2: ADRENALINE Testbed

The ADRENALINE testbed® is an open and disaggregated SDN/NFV-enabled packet/optical transport network and edge/core cloud infrastructure for 6G, IoT/V2X and AI/ML services, constantly evolving since its creation in 2002, and reproducing operators' networks from an End to End (E2E) perspective and Data Centre Interconnect (DCI). The figure below summarizes the networking scenario of ADRENALINE testbed, to be used for the execution of SUCCESS-6G.

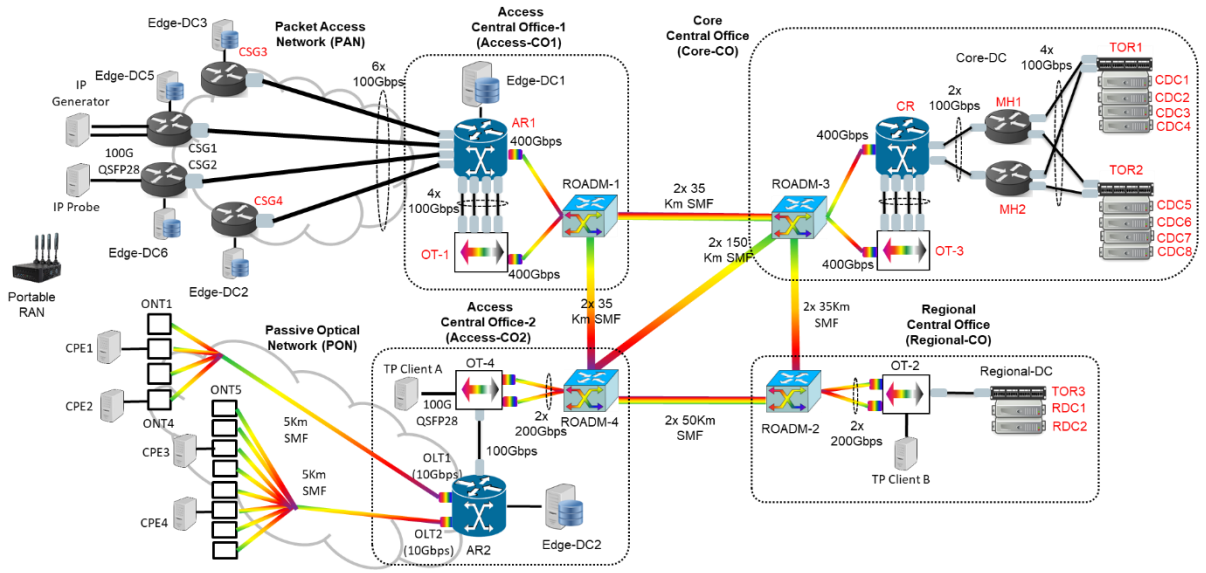

Figure 3 ADRENALINE testbed to be used for Use Case 2

ADRENALINE spans the access, aggregation-metro and core segments, and includes distributed Data Centres (DCs) geographically disperse and located at the edge or in central locations. As depicted in the figure, the key elements are: (1) an SDN-controlled optical network (flexi-grid DWDM photonic mesh), with 4 ROADM nodes and over 600km of amplified DWDM links. Currently, all the links of the mesh are based on amplified C-band transmission, but one of them also supports amplified flexible L-band transmission; (2) packet-optical nodes with optical pluggable transceivers, providing aggregated 400G data rates (muxponders) for transporting traffic flows between the access networks and the core central offices or data centers; (3) programmable SDN-enabled S-BVTs able to transmit multiple flows at variable data rate/reach up to 1 Tb/s; (4) a Packet Access Network (PAN) connected to the metro infrastructure with IP Cell Site Gateways (CSGs); (5) a PON tree formed by disaggregated Optical Network Terminals (ONTs), offering connectivity to several Customer Premises Equipment (CPEs). ADRENALINE also includes a Portable 5G RAN platform for testing and validation of 5G and beyond use cases. The different access networks (i.e., PON) and the photonic mesh are managed by dedicated orchestrators and controllers (e.g., CTTC FlexOpt Optical Controller) to automatically handle the connectivity services entailing the de-/allocation of heterogeneous network resources (i.e., packet and optical devices). The *domain-specific* controllers and orchestrators are coordinated hierarchically by the ETSI TeraFlowSDN controller, which exposes a North Bound Interface to allow interaction of resources to request network connectivity services. This service platform orchestrates the transport (optical/packet) and computing:

i) Multi-VIM (virtualized infrastructure managers) combining OpenStack and K8s controllers for virtual machines and containers;

ii) TeraFlowSDN controller for E2E connectivity among virtual machines, containers, and end-points. The service platform is also in charge of managing the life-cycle of network services and network slices: i) a network service is composed of chained NFs;

iii) a network slice is composed of one or several concatenated network services that deploy a set of NFs.

3 Over-the-air vehicular software updates with robust V2X connectivity: Implementation at the ADRENALINE testbed

3.1 Location-awareness

The Location-Aware Software-Defined Networking (SDN) Controller and Service Orchestrator is an innovative solution designed to revolutionize network management and service provisioning in modern data centers and communication infrastructures. This cutting-edge technology seamlessly integrates software-defined networking with intelligent service orchestration capabilities, creating a highly efficient and dynamic network environment.

Some of its key features to be exploited in SUCCESS-6G-EXTEND are:

- Software-Defined Networking (SDN) Integration: The solution incorporates an advanced SDN controller that centralizes the network control plane, enabling administrators to dynamically manage and configure network resources, policies, and services through a unified, programmable interface.
- Real-Time Location Awareness: One of the defining features of this innovation is its ability to harness real-time location data. By integrating with location-aware devices and sensors, the SDN controller can identify the physical location of connected network devices, end-users, or IoT devices with remarkable precision.
- Geographically Optimized Routing: Leveraging the location information, the SDN controller and service orchestrator can make intelligent routing decisions based on the physical proximity of network elements. This allows for efficient data transmission, reduced latency, and improved overall network performance.
- Dynamic Service Orchestration: The service orchestrator component of this innovation can dynamically provision and manage services based on location and network conditions. It enables automatic scaling of resources, load balancing, and failover mechanisms to ensure optimal service delivery.
- Context-Aware Service Deployment: With location-awareness and real-time data, the SDN controller can intelligently deploy services or allocate resources according to the specific needs of users or devices in different locations. This level of context-awareness enhances the overall user experience and resource utilization.
- Enhanced Network Security: Location-based access control and security policies can be enforced, allowing the SDN controller to detect and respond to potential security threats in a timely manner. Unauthorized access attempts or anomalies in device behaviour can trigger immediate actions to safeguard the network.
- Analytics and Insights: The location-aware SDN controller and service orchestrator also come equipped with powerful analytics tools that provide valuable insights into network performance, user behaviour, and resource utilization. These insights can be leveraged to optimize network design and resource allocation over time.

3.2 Requirements and KPIs

OTA vehicular software updates have become an essential component of modern transportation ecosystems, especially with the emergence of connected and autonomous vehicles. Robust V2X connectivity underpins this capability, enabling vehicles to communicate with infrastructure, other vehicles, pedestrians, and the cloud. By leveraging high-speed and reliable networks, OEMs and service providers can seamlessly deliver security patches, firmware upgrades, and feature enhancements to vehicles without requiring a physical visit to a service center. This section outlines the key requirements and Key Performance Indicators (KPIs) that characterize an effective OTA update system with strong V2X capabilities, focusing particularly on the role of Location-Aware Software-Defined Networking (SDN) Controller and Service Orchestrator in achieving these objectives.

Location-awareness introduces a powerful dimension to OTA vehicular updates, as it allows services to adapt their behavior based on the precise geographical context of vehicles. The Location-Aware SDN Controller and Service Orchestrator functions by integrating real-time location data from vehicles, network devices, and IoT sensors. This enables geofencing, localized policy enforcement, and context-specific resource allocation. By incorporating location-awareness, network providers and service orchestrators can optimize data routing, ensure localized scaling of resources during peak demand, and provide tailored services to connected vehicles. The following sections detail the requirements and KPIs essential for successfully deploying location-aware OTA systems over robust V2X networks.

3.2.1 Requirements

3.2.1.1 Software-Defined Networking (SDN) Integration

Requirement Description: The system must integrate an advanced SDN controller capable of centralizing the network control plane. This controller should offer programmable interfaces to manage and configure V2X resources dynamically. By abstracting the underlying network infrastructure, the SDN controller provides a single pane of glass to orchestrate vehicular communications, ensuring seamless handovers, load balancing, and policy enforcement in real time.

Rationale: Traditional static networking approaches are insufficient for handling the dynamic, high-mobility environment of connected vehicles. An SDN-based approach facilitates on-the-fly adjustments to routing and bandwidth allocations, ensuring minimal service disruption and optimized paths.

3.2.1.2 Real-Time Location Awareness

Requirement Description: The system must support the ingestion and processing of real-time location data from connected vehicles, base stations, roadside units (RSUs), and other IoT devices. Accuracy and responsiveness in location tracking are crucial for timely decision-making.

Rationale: Granular location data enables features like geofenced updates, route-optimized software distribution, and localized analytics. With precise positioning information, the SDN controller can segment the network in ways that directly benefit vehicles in high-density or rural areas, ensuring that resources are allocated appropriately.

3.2.1.3 Geographically Optimized Routing

Requirement Description: Based on the real-time geographic location of vehicles, the SDN controller must dynamically compute and select the most suitable routes for OTA data delivery. The system should utilize multi-path strategies and consider latency, congestion, and geographical proximity to improve performance.

Rationale: By exploiting location data, the network can reduce latency and packet loss, especially in areas prone to high mobility and interference. This is critical for maintaining reliable OTA updates without interrupting other in-vehicle or roadside services.

3.2.1.4 Dynamic Service Orchestration

Requirement Description: The service orchestrator must manage vehicular services with an awareness of network conditions and location demands, automatically scaling compute and bandwidth resources as vehicles move. This includes adaptive load balancing, fault tolerance, and continuous monitoring of service performance.

Rationale: Connected vehicles exhibit fluctuating demands based on traffic density, driving patterns, and location-specific events (e.g., congestion around urban centers). Dynamic orchestration ensures that the system remains robust and cost-effective in the face of changing conditions.

3.2.1.5 Context-Aware Service Deployment

Requirement Description: The platform must enable context-specific resource allocation for vehicular services, including OTA updates, based on location, vehicle type, and real-time network conditions. Policies should be adaptable to handle factors such as rush-hour peaks or localized emergencies.

Rationale: Not all vehicles or locations require the same level of service quality at all times. For instance, an emergency response vehicle might need prioritized bandwidth, while personal cars during off-peak hours might tolerate slower update rates. Context-aware deployments maximize efficiency and user experience.

3.2.2 Key Performance Indicators (KPIs)

3.2.2.1 End-to-End Latency for OTA Updates

Definition: The time elapsed from initiating a software update in the cloud to successful reception and installation in the vehicle's onboard system.

Importance: Minimizing latency is vital in scenarios where urgent patches or critical updates must be deployed. Low latency also indicates effective routing and resource allocation by the SDN controller.

Target Values/Thresholds: In most urban and suburban contexts, an OTA update latency of under a few seconds for the data transfer portion (excluding the install time on the vehicle's system) is often the target. In rural or remote areas, slightly higher thresholds may be permissible, but should remain within an acceptable range (e.g., under 10 seconds).

3.2.2.2 Geographical Routing Efficiency

Definition: A measure of how effectively the system uses real-time location data to make routing decisions that minimize total hop count, latency, or congestion. Often expressed as a ratio between the optimal route metrics and actual route metrics taken by network packets.

Importance: This KPI directly illustrates the advantage of having a location-aware SDN solution. Higher efficiency means fewer packet drops, lower latency, and better resource utilization.

Target Values/Thresholds: A well-optimized system might achieve above 90% routing efficiency under normal conditions. Sudden changes in topology or unexpected interferences might cause temporary deviations, but the system should quickly revert to near-optimal routing.

3.2.2.3 Closest Node Selection Latency (CNSL)

Definition: Closest Node Selection Latency (CNSL) measures the time it takes for the system's algorithm to determine which edge or network node is geographically and/or topologically closest to a requesting vehicle or device. Specifically, CNSL is the duration between (1) when a request for node selection is received and (2) when the algorithm completes its computations and provides the optimal node for that request. This KPI captures how swiftly the system reacts to dynamic network conditions and user mobility, thereby influencing overall service efficiency and user experience.

Importance: In a highly dynamic environment—such as an autonomous vehicle network or a location-aware edge-computing scenario—decisions about which node should handle a request must be made in near real-time to prevent service degradation. High CNSL values may result in avoidable handoff delays, missed opportunities for load balancing, and slower response times for services hosted at the edge. A low CNSL, on the other hand, ensures that requests are processed by the most appropriate nodes with minimal delay, improving throughput, latency, and overall system responsiveness. This KPI thus provides direct insight into the effectiveness and efficiency of the decision-making algorithms underpinning the network's orchestration layer.

Target Values/Thresholds: Ideal CNSL targets can vary based on the complexity of the algorithm, the size of the network, and the frequency of incoming node-selection requests. For example, in a dense

urban environment with high mobility and large volumes of real-time traffic, a CNSL of under 50 milliseconds is often desirable to maintain seamless service. In less latency-sensitive or lower-traffic scenarios, thresholds can be relaxed slightly to under 100–200 milliseconds. Regardless of the exact target, any consistent rise in CNSL beyond the established threshold should be treated as a performance regression, triggering investigations into algorithmic inefficiencies, data structure optimization, or hardware constraints.

3.3 UC2 Architecture and network deployment

Figure 4 provides a high-level system architecture for OTA vehicular software updates within a Vehicle-to-Everything (V2X) connectivity framework, leveraging ETSI TeraFlowSDN (E2E SDN controller) to orchestrate and automate the end-to-end update process. The figure highlights the key network components enabling the robust dissemination of software updates to the connected vehicles.

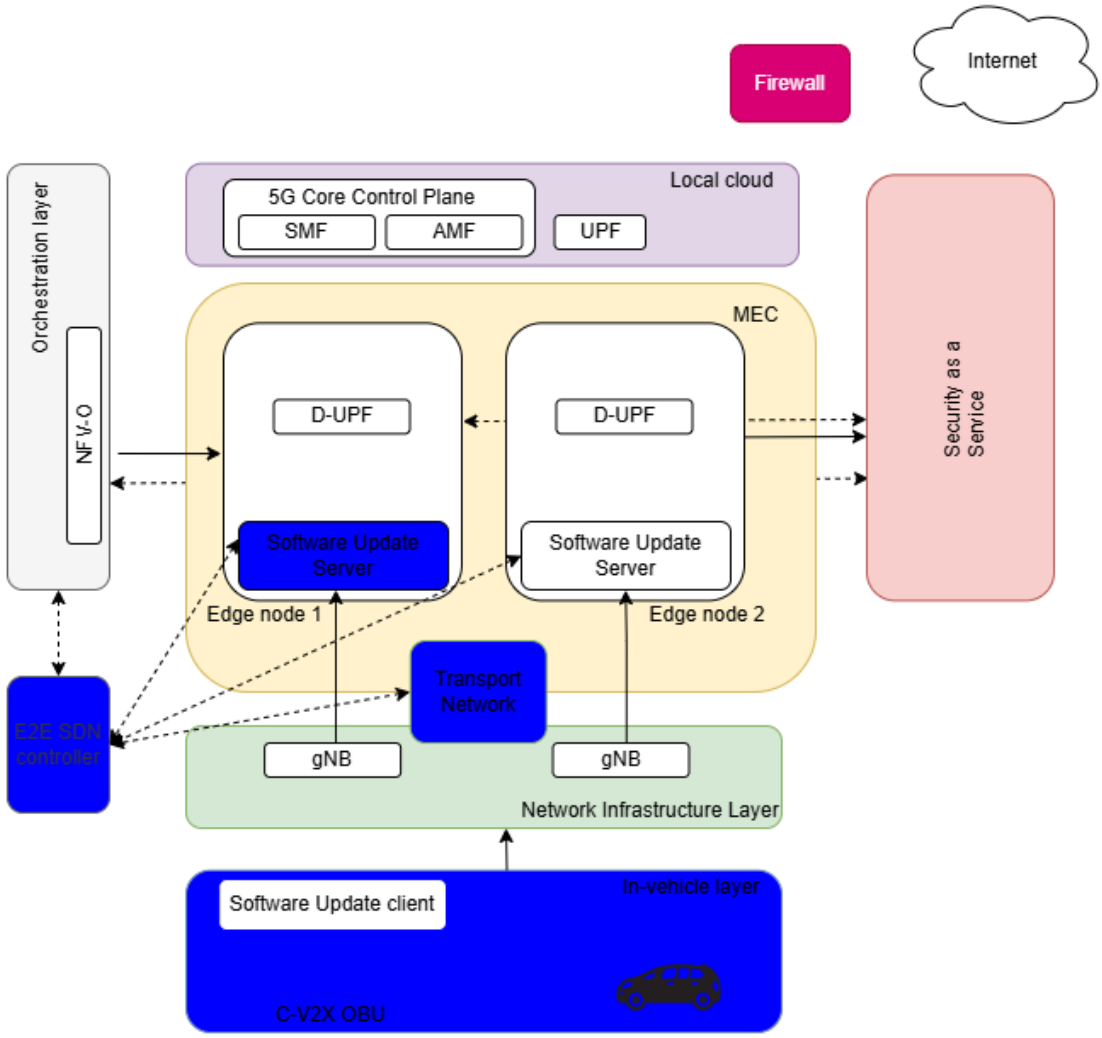

Figure 4 Instantiation of SUCCESS-6G architecture for OTA vehicular software updates with robust V2X connectivity

At the heart of the system is the **ETSI TeraFlowSDN Controller**, a component responsible for dynamically managing the **Transport Network** and optimizing update delivery. Acting as a centralized SDN-based controller, it ensures intelligent routing of update traffic across the **5G-enabled transport infrastructure**, including **gNB (5G base stations)** and **network links to edge nodes**.

The **Software Update Server**, another element in the architecture, plays a crucial role in securely distributing updates to vehicles. Each **Multi-access Edge Computing (MEC) node** (Edge Node 1 & Edge Node 2) hosts a **Software Update Server**, which caches and delivers software patches and updates to

vehicles over the **Transport Network**. This ensures low-latency distribution by leveraging **edge computing**, reducing reliance on centralized cloud infrastructure.

The **Transport Network** forms the backbone of the OTA update system, interconnecting the **gNBs, Software Update Servers, and edge nodes** to maintain a seamless flow of software updates. The **TeraFlowSDN Controller** plays a pivotal role in managing this transport layer, optimizing network paths for reliability and efficiency.

On the vehicle side, the **Software Update Client**, embedded in the **C-V2X On-Board Unit (OBU)**, is responsible for receiving and applying updates. This component ensures the successful deployment of software updates to critical vehicle systems, enabling **real-time enhancements** in connectivity, security, and performance.

Together, these elements form a robust software update framework with location-awareness, being able to modify edge locations based on client position. They enable **secure, low-latency, and scalable OTA updates** for connected vehicles, ensuring seamless integration with 5G and MEC infrastructure.

3.4 Exposed interfaces

The defined Protocol Buffer interface defines a set of messages that together provide a structured way to describe geographical positioning, network location information, constraints, services, and network slices within a system. The GPS_Position message is straightforward, capturing geographic coordinates with float latitude and float longitude fields, ensuring that location data can be precisely specified in terms of global latitude-longitude positioning. The Location message builds on this by offering a oneof construct that allows for either a named region (region) or a more detailed GPS position (gps_position) to describe where an entity resides. Additionally, it includes two string fields, interface and circuit_pack, to store hardware or network-specific location references, thus making the location specification flexible enough to handle both physical and logical contexts.

Next, the Constraint_EndPointLocation message pairs an endpoint identifier (endpoint_id) with a Location object to provide a method of specifying location-based constraints or requirements for that particular endpoint. This structure can be particularly useful in networking scenarios where certain endpoints must be placed at or moved to specific geographic or network coordinates. The Constraint message itself uses a ConstraintActionEnum to define the type of action being considered (e.g., allow, deny, reserve), and it leverages a oneof block to encapsulate a broad range of possible constraint types, such as custom constraints (Constraint_Custom), scheduling requirements (Constraint_Schedule), endpoint location constraints (Constraint_EndPointLocation), priority constraints (Constraint_EndPointPriority), and several SLA-related constraints like capacity, latency, availability, and isolation, among others. By supporting this variety of constraint messages, the system ensures that different policy or resource requirements can be flexibly added, updated, or validated according to the needs of a network or service configuration.

The Service message captures details about a logical service within the network, identified uniquely by a ServiceId alongside descriptive information like name and service_type. It references one or more endpoints (via repeated EndPointId service_endpoint_ids) and applies a list of constraints (repeated Constraint service_constraints) that must be honored for proper service operation. The Service message also keeps track of its overall status (service_status), user-specified configurations (service_config), and an associated timestamp for versioning or logging purposes. Finally, the Slice message functions similarly to Service but focuses on describing a larger logical partition or “slice” of the network, identified by a SliceId. It can contain multiple endpoints, constraints, services, and even subslices (repeated SliceId slice_subslice_ids), thus enabling hierarchical structuring of network segments. Like Service, the Slice message includes a status field (slice_status), configuration data (slice_config), ownership information (slice_owner), and a timestamp to accurately manage lifecycle events. Together, these messages form a cohesive schema for defining, configuring, and monitoring sophisticated network entities, from basic GPS-based endpoints to higher-level constructs such as services and network slices, while maintaining a rich set of constraints that drive policy-based

orchestration.

The complete protocol buffer is available at:

https://labs.etsi.org/rep/tfs/controller/-/blob/master/proto/context.proto?ref_type=heads

We hereby provide a selection of the required extensions.

```
message GPS_Position {
  float latitude = 1;
  float longitude = 2;
}
message Location {
  oneof location {
    string region = 1;
    GPS_Position gps_position = 2;
    string interface=3;
    string circuit_pack=4;
  }
}
message Constraint_EndPointLocation {
  EndPointId endpoint_id = 1;
  Location location = 2;
}
message Constraint {
  ConstraintActionEnum action = 1;
  oneof constraint {
    Constraint_Custom custom = 2;
    Constraint_Schedule schedule = 3;
    Constraint_EndPointLocation endpoint_location = 4;
    Constraint_EndPointPriority endpoint_priority = 5;
    Constraint_SLA_Capacity sla_capacity = 6;
    Constraint_SLA_Latency sla_latency = 7;
    Constraint_SLA_Availability sla_availability = 8;
    Constraint_SLA_Isolation_level sla_isolation = 9;
    Constraint_Exclusions exclusions = 10;
    Constraint_QoSProfile qos_profile = 11;
  }
}
message Service {
```

```
  ServiceId service_id = 1;
  string name = 2;
  ServiceTypeEnum service_type = 3;
  repeated EndPointId service_endpoint_ids = 4;
  repeated Constraint service_constraints = 5;
  ServiceStatus service_status = 6;
  ServiceConfig service_config = 7;
  Timestamp timestamp = 8;
}
message Slice {
  SliceId slice_id = 1;
  string name = 2;
  repeated EndPointId slice_endpoint_ids = 3;
  repeated Constraint slice_constraints = 4;
  repeated ServiceId slice_service_ids = 5;
  repeated SliceId slice_subslice_ids = 6;
  SliceStatus slice_status = 7;
  SliceConfig slice_config = 8;
  SliceOwner slice_owner = 9;
  Timestamp timestamp = 10;
}
```

3.5 Workflow

Figure 5 provides an in-depth view of the sequence diagram for Over-the-air vehicular software updates within a robust V2X connectivity framework, specifically within the context of the TeraFlowSDN system. This figure meticulously illustrates the sequence of interactions and the flow of commands that facilitate the seamless operation of software updates in a vehicular network environment.

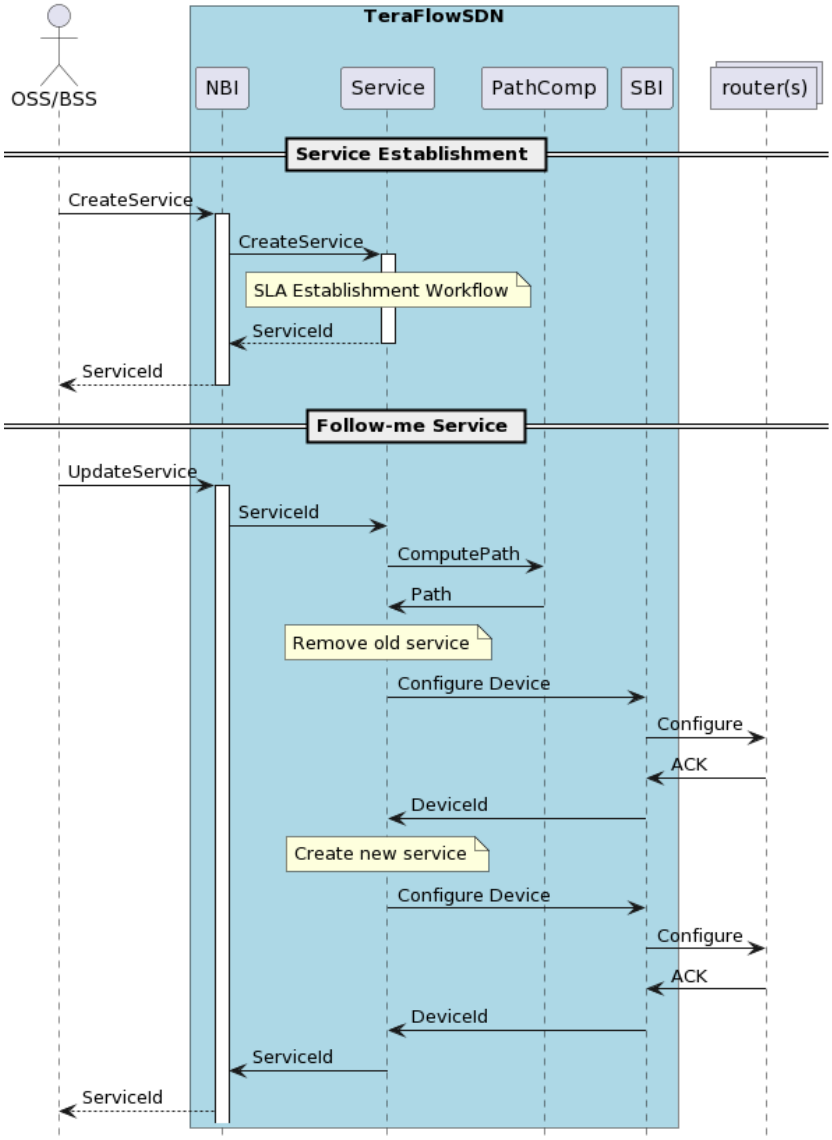

Figure 5 Sequence diagram for OTA vehicular software updates with robust V2X connectivity

The diagram begins with the Operations Support Systems/Business Support Systems (OSS/BSS) actor, a critical entity in the TeraFlowSDN framework. The OSS/BSS actor initiates the process by triggering the CreateService directive. This directive is aimed at the Northbound Interface (NBI), a key interface in the system that serves as a conduit between the higher-level operations support and the underlying network management components.

Upon activation by the OSS/BSS actor, the NBI communicates with the Service component of the TeraFlowSDN system. This interaction sets off a detailed and structured workflow, central to which is the establishment of a Service Level Agreement (SLA). The SLA outlines the terms, conditions, and expectations associated with the service, ensuring alignment with the network's capabilities and the customer's requirements. Following the agreement, a unique ServiceId is generated, marking the successful creation of the new service. This ServiceId is a vital element, serving as a distinct identifier for the service throughout its lifecycle. It is then communicated back to the OSS/BSS actor, signaling the successful completion of the service creation phase.

The sequence diagram further delves into the modification of an existing service, labeled as the "Follow-me Service." In this phase, the OSS/BSS actor issues an UpdateService command. This command is relayed to the NBI, which in turn interacts with the existing Service component using the ServiceId as a reference. The diagram vividly illustrates the involvement of the PathComputation (PathComp) component at this stage. PathComp is responsible for calculating a new, optimized path for the service, a critical step in ensuring efficient and reliable service delivery.

An essential aspect of this sequence is the removal of the outdated service configuration. This process is crucial for making way for the reconfiguration of network devices, such as routers, along the newly computed path. The Southbound Interface (SBI) emerges as a fundamental player in this process. It facilitates the reconfiguration of these devices, with a series of acknowledgments exchanged to confirm the successful reconfiguration.

Once the devices are reconfigured, the Service component proceeds with the creation of the updated service. This involves configuring the network devices along the new path and subsequently communicating the updated ServiceId back to the OSS/BSS actor. This step marks the completion of the service modification process.

In essence, Figure 5 encapsulates the systematic and intricate processes of service creation, modification, and management within the TeraFlowSDN framework. It highlights the critical role played by various actors, interfaces, and components in orchestrating the complex operations necessary for efficient and secure Over-the-air vehicular software updates in a V2X environment. The diagram serves as a comprehensive guide, employing a scientific and methodical language to elucidate the dynamic interplay of various elements within the system.

3.6 Preliminary experimental validation of the functionalities

Figure 6 shows the Wireshark capture of the creation of a location-aware connectivity service, and it is updated with a new location. The 1st packet corresponds to the creation of the connectivity service, while the 2nd packet corresponds to its acknowledgment. The 3rd packet corresponds to the update and actual provisioning of the connectivity service, while the 4th packet corresponds to its acknowledgment.

```
*REF*     SETTINGS[0], HEADERS[1]: POST /service.ServiceService/CreateService, WINDOW_UPDATE[1], DATA[1] (GRPC) (PROTOBUF), WINDOW_UPDATE[0]
0.037609  HEADERS[1]: 200 OK, WINDOW_UPDATE[1], DATA[1] (GRPC) (PROTOBUF), HEADERS[1], WINDOW_UPDATE[0]
0.041850  HEADERS[3]: POST /service.ServiceService/UpdateService, WINDOW_UPDATE[3], DATA[3] (GRPC) (PROTOBUF), WINDOW_UPDATE[0]
1.258795  HEADERS[3]: 200 OK, WINDOW_UPDATE[3], DATA[3] (GRPC) (PROTOBUF), HEADERS[3], WINDOW_UPDATE[0]
```

Figure 6 Wireshark of location-aware service establishment

3.7 Final testing and validation

In this section, we present the comprehensive testing and validation process carried out to confirm that the TeraFlowSDN (TFS) meets all its functional and performance requirements. Our evaluation framework addresses four key aspects of requirement validation—Software-Defined Networking (SDN) Integration, Real-Time Location Awareness, Geographically Optimized Routing, and Dynamic Service Orchestration—followed by a detailed Key Performance Indicator (KPI) assessment.

Firstly, SDN Integration is validated through packet capture analysis, showcasing the creation and update of connectivity services in a dynamic environment. This step ensures that the system's communication flows, commands, and acknowledgments function correctly when integrating with SDN controllers. Next, Real-Time Location Awareness is tested by incorporating location-based constraints and verifying that the new data models effectively support location-specific service requests across both packet and optical domains.

Furthermore, Geographically Optimized Routing is validated by implementing an algorithm designed to select the nearest access node to a requester's location, thereby minimizing latency. The correctness of the underlying logic and the performance benefits of using geospatial data in routing decisions are confirmed through controlled scenarios and real-world testing. Finally, Dynamic Service Orchestration is tested by simulating user mobility; the system's ability to update, remove, and recreate connectivity services in real-time is validated through packet tracing and orchestration logs.

To quantify how effectively these functionalities work in practice, we also introduce KPI Validation (Section 3.7.2). In particular, we focus on the Closest Node Selection Latency (CNSL) metric, which

measures the responsiveness of the system in selecting the optimal access node upon receiving a new or updated request. This KPI reflects how quickly TFS adapts to dynamic conditions, highlighting its efficiency and robustness in handling location-aware connectivity services.

Through this structured testing and validation, we demonstrate not only the correctness of the proposed TFS enhancements but also their readiness for deployment in scenarios that demand high reliability, low latency, and real-time adaptation.

3.7.1 Software-Defined Networking (SDN) Integration

Figure 7 illustrates a Wireshark capture of the creation and subsequent update of a location-aware connectivity service. The communication follows a gRPC-based exchange where multiple messages are exchanged between the client and the server.

The first packet represents the initiation of the connectivity service creation request, using a gRPC POST request to the CreateService API endpoint. The second packet corresponds to the server acknowledging the creation request, confirming receipt, and possibly providing initial metadata about the service.

The third packet captures an update request to the connectivity service, where a new location is provided, triggering an update in the backend. This update is executed through another gRPC POST request, this time directed at the UpdateService API endpoint. Finally, the fourth packet represents the acknowledgment of this update by the server, signaling that the new location has been successfully registered and the connectivity service has been provisioned accordingly.

Throughout this exchange, WINDOW_UPDATE, HEADERS, and DATA frames are observed, indicating the flow control mechanisms of HTTP/2 over gRPC. The PROTOBUF encoding confirms that the payloads are serialized in the Protocol Buffers format, a common choice for gRPC communications. The sequence of interactions showcases the efficient and structured approach used to manage the lifecycle of location-aware connectivity services.

```
*REF*     SETTINGS[0], HEADERS[1]: POST /service.ServiceService/CreateService, WINDOW_UPDATE[1], DATA[1] (GRPC) (PROTOBUF), WINDOW_UPDATE[0]
0.037609  HEADERS[1]: 200 OK, WINDOW_UPDATE[1], DATA[1] (GRPC) (PROTOBUF), HEADERS[1], WINDOW_UPDATE[0]
0.041850  HEADERS[3]: POST /service.ServiceService/UpdateService, WINDOW_UPDATE[3], DATA[3] (GRPC) (PROTOBUF), WINDOW_UPDATE[0]
1.258795  HEADERS[3]: 200 OK, WINDOW_UPDATE[3], DATA[3] (GRPC) (PROTOBUF), HEADERS[3], WINDOW_UPDATE[0]
```

Figure 7 Wireshark of location-aware service establishment

3.7.2 Real-Time Location Awareness

The introduction of location awareness in transport networks, encompassing both packet and optical domains, necessitates the integration of novel concepts into several SDN-based data models. These enhancements aim to enable precise and dynamic service provisioning by incorporating geographical and topological awareness.

Among the most affected data models are those related to service endpoints, which define the network termination points where connectivity services can be requested. Traditional models primarily focus on logical and physical attributes such as interfaces, circuit packs, or identifiers, but location-aware transport networking requires additional attributes to support geospatial constraints. This adaptation is crucial for optimizing service delivery, particularly in scenarios involving mobile edge computing (MEC), low-latency services, and adaptive network slicing.

Furthermore, connectivity services must accommodate new constraints based on geographic location, either as specific GPS coordinates or predefined regions. This capability ensures that services can be dynamically adjusted based on real-world locations, facilitating more efficient routing, failover mechanisms, and service assurance.

To support this functionality, the modified data models for internal TeraFlowSDN (TFS) usage introduce a Location message structure that allows specifying an endpoint's location in multiple ways, such as by

region, GPS coordinates, interface name, or circuit pack association. These attributes enable enhanced path computation algorithms that take into account geographic constraints and network topology.

The following protocol buffer messages define the proposed data model extensions:

```
message Location {
 oneof location {
  string region = 1;
  GPS_Position gps_position = 2;
  string interface=3;
  string circuit_pack=4;
 }
}
message Constraint_EndPointLocation {
 EndPointId endpoint_id = 1;
 Location location = 2;
}
```

3.7.3 Geographically Optimized Routing

To provide location-aware services within TeraFlowSDN (TFS), three key components have been modified to support geospatial constraints and dynamic service provisioning:

1. Context Module: The Context module's database has been extended to accommodate the new data models introduced in the previous section. This includes storing and managing location attributes such as GPS coordinates, regional identifiers, and network topology-related location constraints. The enhancements enable TFS to efficiently index and query location-aware endpoints and their associated constraints, allowing for more accurate and dynamic service deployment.
2. Device Module: The Device module has been upgraded to retrieve real-time location data from network devices. This functionality is crucial for dynamically adapting connectivity services based on device mobility, network topology changes, and geospatial conditions. The location data can be obtained through network telemetry, device metadata, or SDN controller interactions, ensuring an up-to-date view of the network's geographic structure.
3. Service Module: Within the Service module, a novel algorithm has been implemented to determine the closest access edge node for a given user's location. This algorithm enables the automatic selection of the most optimal network entry point for service requests, improving performance and minimizing latency.

The proposed algorithm establishes a service between a data center (DC) and a user by dynamically identifying the closest access edge node relative to the user's location. The steps of the algorithm are as follows:

1. User’s Location Retrieval:

The user's location is extracted from the connectivity service request. This can be in the form of GPS coordinates, a region identifier, or a network interface linked to a geographical area.

2. Access Edge Node Discovery:

- The system retrieves the list of available access edge nodes from the Context module.
- These nodes represent network devices at the network edge, such as routers, switches, or base stations, that serve as entry points for end-user connectivity.

3. Geographical Distance Computation:

- The geographical distance between each access edge node and the user's location is computed.
- Various methods can be employed for distance calculation, but one of the most accurate and widely used approaches is the Haversine formula.
- The Haversine formula calculates the great-circle distance between two points on the Earth's surface based on their latitude and longitude. This method is particularly useful in networking applications where precise geolocation-based routing is required.

The Haversine formula is given as:

$$d = 2r \arcsin\left(\sqrt{\sin^2\left(\frac{\Delta\phi}{2}\right) + \cos(\phi_1)\cos(\phi_2)\sin^2\left(\frac{\Delta\lambda}{2}\right)}\right)$$

where:

- d is the distance between the two points,
- r is the Earth's radius (~6,371 km),
- ϕ_1, ϕ_2 are the latitudes of the two points in radians,
- λ_1, λ_2 are the longitudes of the two points in radians,
- $\Delta\phi = \phi_2 - \phi_1$ and $\Delta\lambda = \lambda_2 - \lambda_1$.

4. Nearest Node Selection:
- The access edge node with the shortest calculated distance to the user is selected.
- If multiple nodes have similar distances, additional parameters such as network congestion, bandwidth availability, or link latency can be used as tie-breakers.
5. Service Establishment:
- Once the optimal access edge node is identified, the service connection is established between the data center (DC) and the user through the selected node.
- The service request is provisioned dynamically, ensuring an efficient and low-latency connection.

3.7.4 Dynamic Service Orchestration

After the user node changes its position, the TeraFlowSDN (TFS) Northbound Interface (NBI) is requested to update the existing connectivity service. This process involves a sequence of gRPC-based exchanges, as illustrated in the captured network traces (Figure 8).

```
*REF*     HEADERS[3]: POST /service.ServiceService/UpdateService, WINDOW_UPDATE[3], DATA[3]
0.601359  HEADERS[15]: POST /context.ContextService/RemoveConnection, WINDOW_UPDATE[15], DATA[15] (GRPC) (PROTOBUF), WINDOW_UPDATE[0]
0.648959  HEADERS[17]: POST /context.ContextService/RemoveService, WINDOW_UPDATE[17], DATA[17] (GRPC) (PROTOBUF), WINDOW_UPDATE[0]
0.736901  HEADERS[21]: POST /context.ContextService/SetService, WINDOW_UPDATE[21], DATA[21] (GRPC) (PROTOBUF)
1.898406  HEADERS[13]: POST /context.ContextService/SetConnection, WINDOW_UPDATE[13], DATA[13] (GRPC) (PROTOBUF), WINDOW_UPDATE[0]
2.030466  HEADERS[3]: 200 OK, WINDOW_UPDATE[3], DATA[3] (GRPC) (PROTOBUF), HEADERS[3], WINDOW_UPDATE[0]
```

Figure 8 Wireshark of the location-aware update service process

The 1st packet initiates the connectivity service update request, where the user's new location is provided. This is executed via a POST request to the UpdateService API, signaling that the system needs to adjust the existing service based on the user's updated geographical position.

The 2nd packet corresponds to the removal of the existing connection that was previously established for the user. This step ensures that the outdated connection is properly decommissioned, avoiding any inconsistencies in network resource allocation.

The 3rd packet represents the removal of the entire connectivity service, effectively deregistering the previous user-to-network association before establishing a new one. This step ensures that no legacy configurations interfere with the upcoming provisioning process.

The 4th packet captures the setup of a new connection, where the system provisions a fresh network path for the user. This is crucial to ensure that the updated service request is fulfilled efficiently, considering the user's new geographical location.

Finally, the 5th packet acts as an acknowledgment (ACK) for the service update, confirming that the changes have been successfully applied and that the new connectivity service is now active.

This sequence of operations demonstrates the dynamic reconfiguration capabilities of TFS, ensuring that network services can adapt to mobility changes in real-time. The process follows a structured tear-down and re-establishment approach, which prevents conflicts and optimizes network resource utilization. Key benefits include:

- Seamless Mobility Handling: The network automatically adapts to user location changes by modifying service endpoints.
- Efficient Resource Management: Old connections and services are removed before establishing new ones, preventing redundant resource allocation.
- Low-Latency Reconfiguration: The update process is executed efficiently using gRPC transactions, ensuring rapid service adjustments.

This mechanism is essential for edge computing, mobile backhaul, and next-generation transport networks, where dynamic connectivity adjustments are critical for maintaining service quality.

3.7.5 Closest Node Selection Latency

The Closest Node Selection Latency (CNSL) measures the time required for the system’s algorithm to determine the geographically and/or topologically closest edge or network node to a requesting vehicle or device. This metric captures the time interval between:

1. The moment a request for node selection is received.
2. The moment the algorithm computes and returns the optimal node for that request.

The histogram in Figure 9 provides an empirical distribution of CNSL values based on collected request samples. Key observations include:

- Latency Range: The selection process takes between 0.10s and 0.17s, indicating a tightly controlled processing time with minimal variations.
- Most Frequent Latency: The highest percentage of requests (~40%) are processed within 0.15s, making it the dominant CNSL value.
- Distribution Trends: A notable proportion of requests (~30%) experience CNSL values in the 0.13s–0.14s range, while a smaller fraction (~10%) falls around 0.10s or 0.17s.

The CNSL values demonstrate low-latency decision-making, which is essential for applications requiring real-time mobility management, such as V2X communications, edge computing, and dynamic service orchestration. The clustering of values around 0.15s suggests a predictable and efficient algorithmic execution, minimizing uncertainty in response times. The absence of outliers beyond 0.17s indicates robust system performance with no significant delays or computational bottlenecks.

To further enhance CNSL, potential optimizations may include:

- Parallelized distance computations to accelerate decision-making.
- Caching mechanisms for frequently accessed node selections.
- Machine learning-based predictive node selection to anticipate user movements and precompute optimal routes.

By maintaining a consistently low CNSL, the system ensures fast service reconfiguration, contributing to reduced handover delays, lower network latency, and an improved user experience in dynamic networking environments.

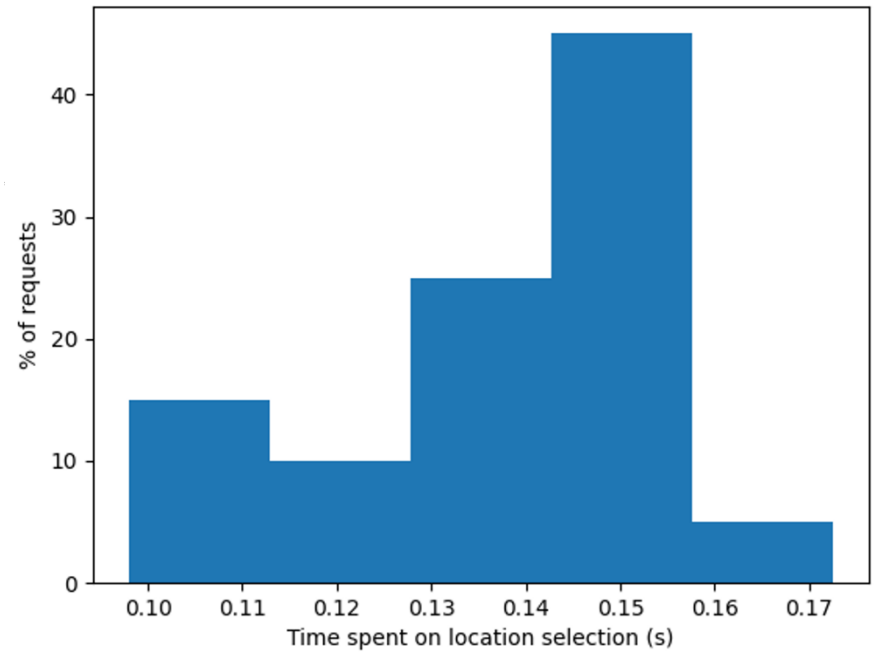

Figure 9 Location algorithm latency

4 Use case 2 proof-of-concept (PoC)

For the proof-of-concept of OTA vehicular software updates at the Circuit Parcmotor Castellolí, all software, both server-side and on-board unit (OBU) software, has been migrated to their final locations: the Castellolí server and the IDNEO-developed OBU installed in the test vehicle (Figure 10).

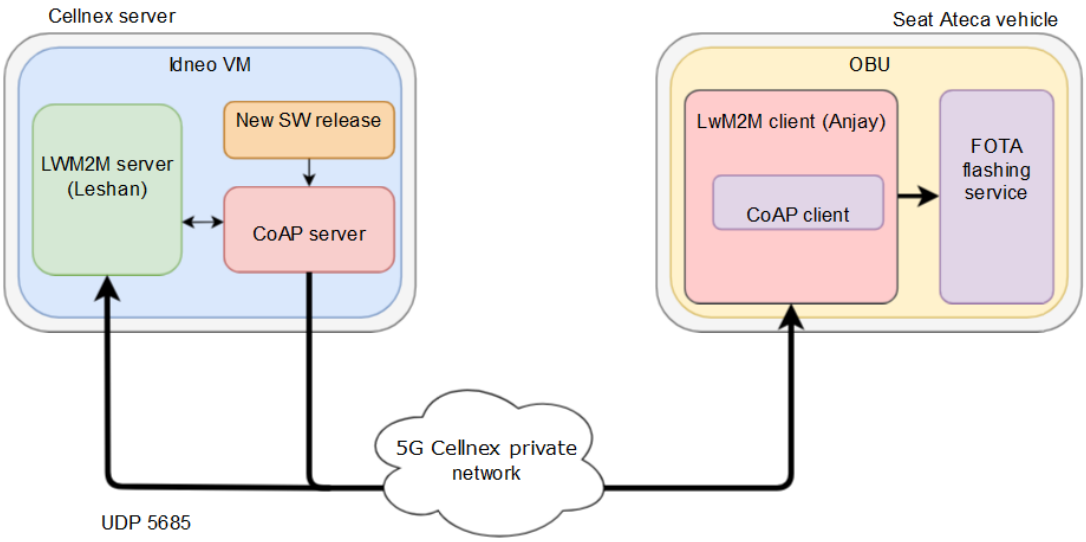

Figure 10 PoC architecture for OTA vehicular software updates at Castelloli

During the third round of tests, the focus was on evaluating network connectivity and performance in communications between the OBU and the Firmware Over-the-Air (FOTA) server. The assessment included analyzing persistent connections, reconnection processes, failure scenarios, download times, and network events that could potentially impact communication between the OBU application and the FOTA server. Additionally, the use case involving software updates from the management application was tested to ensure its functionality and reliability.

Below is the procedure followed to evaluate the use case. Figure 11 shows the dashboard used to manage the devices. There were no devices registered on the dashboard initially. This dashboard is generated by an application that, in addition to having a communications port, also has a web management port. This application was containerized and deployed at the edge of the network.

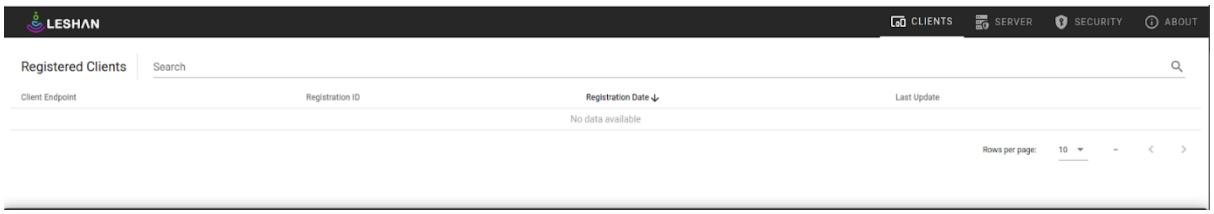

Figure 11 LWM2M server deployed on the virtual machine waiting for connections

Once the OBU was powered on, the device registration process with the server was observed, displaying the assigned endpoint name on the dashboard. Additionally, real-time logs from the client application running on the device were monitored during the test. Simultaneously, network communication packets were captured to facilitate further analysis of system performance and connectivity.

Figure 12 Top: Client connected to Server, Bottom: TCU console showing server logs

Within the management platform, relevant data can be obtained from the OBU. This information is communicated between the OBU and the server via COAP communication encrypted with DTLS. A relevant piece of information, which we can see in Figure 13, is the version of the release deployed on the OBU.

Figure 13 TCU and Server interaction, device data observation

Next, the server is instructed on the location of the file containing the new software version to be deployed. For the server, this file is considered a resource that must be transmitted as a configurable item. Prior to this step, the file was uploaded to a repository accessible via HTTPS, ensuring secure and efficient retrieval during the update process.

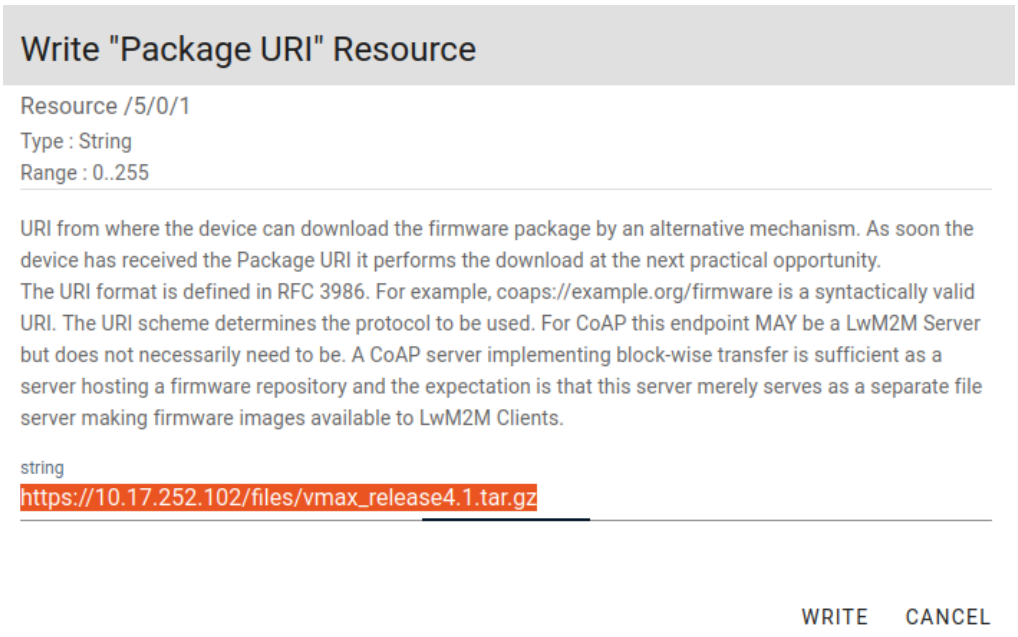

Figure 14 Top: Software Release configuration for update, Bottom: Software Releases in HTTPS repository

The file is deployed to the OBU, which downloads it once it has been instructed where to retrieve it. Figure 15 shows the *"firmware.fota"* file, which contains the updated software and is stored on the OBU's internal SD card. At this point, the Update button will be used to begin deploying the new release, since in a real-world environment, this new release will be deployed when the system is rebooted during an engine shutdown.

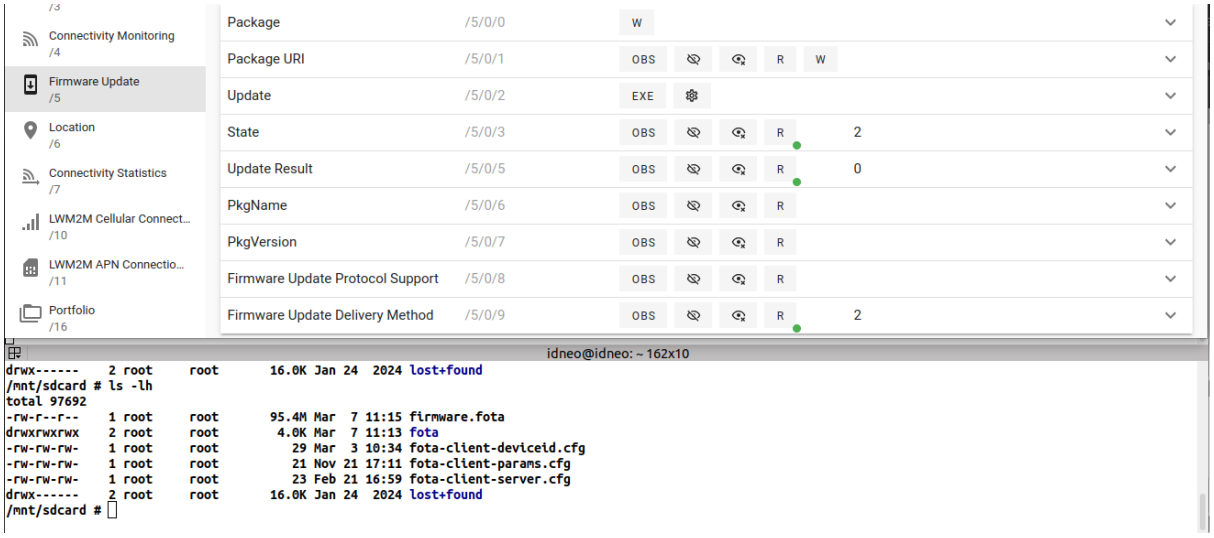

Figure 15 Top: Software Update User Interface, Bottom: Release Download to the TCU File System

Upon system reboot, the OBU executes a startup script that verifies whether a firmware update is required. If an update is necessary, the script initiates the update process and ensures its completion. Once the update is successfully applied, the OBU is expected to automatically reconnect to the edge server, as illustrated in Figure 16.

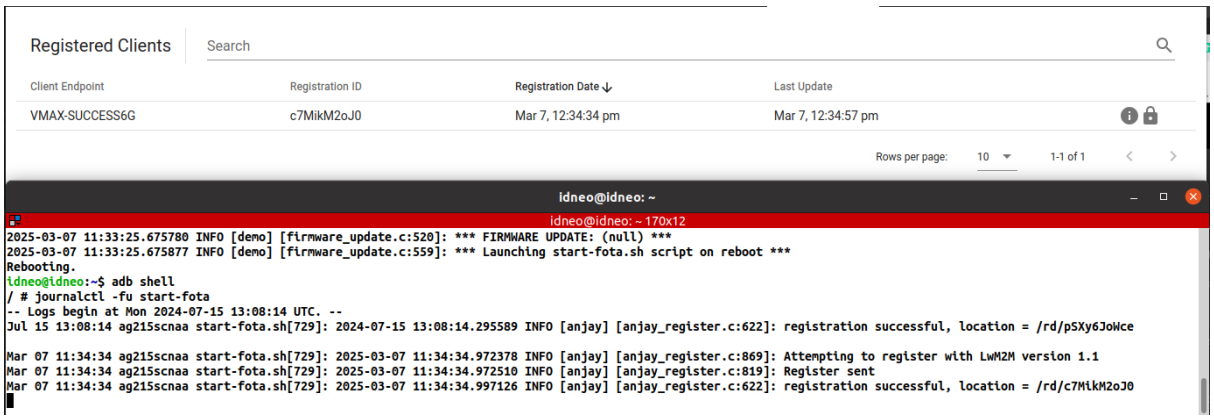

Figure 16 Reboot and automatic reconnection to the LWM2M server

A further analysis is performed based on the captured network traffic. As illustrated in Figure 17, the OBU, assigned the IP address 10.17.201.240, receives a reset command from the server at IP address 10.17.252.102. Following this, the OBU initiates a DTLS handshake to re-register with the network. This event marks the completion of the update process.

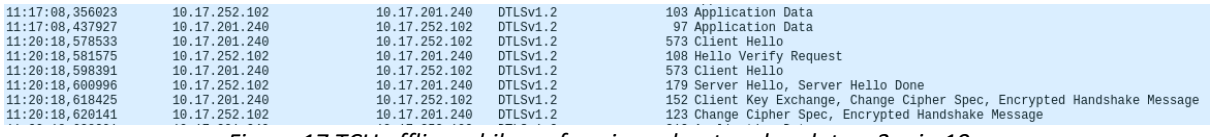

Figure 17 TCU offline while performing reboot and update = 3 min 10 s

5 Conclusions

The implementation of OTA vehicular software updates with robust V2X connectivity is crucial for enhancing the reliability and efficiency of modern connected vehicles. By leveraging advanced networking technologies such as SDN, MEC, and AI-driven resource management, the SUCCESS-6G-EXTEND framework ensures seamless update delivery, even in challenging network conditions. The results from this study validate the effectiveness of real-time location-awareness and dynamic resource allocation in improving update success rates and transmission stability. Moving forward, further advancements in network resilience and adaptive data routing will be essential to maintaining high-quality OTA update services as vehicle connectivity continues to evolve.

A major challenge in implementing robust V2X connectivity is the need for continuous optimization of network resources to accommodate varying vehicular mobility patterns and traffic conditions. The integration of AI-driven network orchestration techniques ensures that the system can dynamically adapt to network congestion and disruptions. Moreover, the use of SDN controllers enables intelligent routing and efficient bandwidth allocation, reducing transmission delays and improving service continuity.

Another critical aspect is the seamless coordination between different network elements, including edge nodes, roadside units, and cloud infrastructure. By establishing a cooperative and distributed update management approach, the system enhances scalability and improves the reliability of software dissemination across geographically diverse environments. The ability to maintain high-speed, low-latency connectivity ensures that vehicles receive timely updates, reducing the risks associated with outdated software and enhancing overall vehicular performance.

6 References

Abishek A, Vilalta R, Gifre L, Alemany P, Manso C, Casellas R, Martínez R, Muñoz R. Network Extensions to Support Robust Secured and Efficient Connectivity Services for V2X Scenario. In2024 24th International Conference on Transparent Optical Networks (ICTON) 2024 Jul 14 (pp. 1-4). IEEE.

Vilalta R, Vía S, Mira F, Casellas R, Muñoz R, Alonso-Zarate J, Kousaridas A, Dillinger M. Control and management of a connected car using sdn/nfv, fog computing and yang data models. In2018 4th IEEE Conference on Network Softwarization and Workshops (NetSoft) 2018 Jun 25 (pp. 378-383). IEEE.

Muñoz R, Vilalta R, Yoshikane N, Casellas R, Martínez R, Tsuritani T, Morita I. Integration of IoT, transport SDN, and edge/cloud computing for dynamic distribution of IoT analytics and efficient use of network resources. Journal of Lightwave Technology. 2018 Apr 1;36(7):1420-8.

Fallgren M, Dillinger M, Alonso-Zarate J, Boban M, Abbas T, Manolakis K, Mahmoodi T, Svensson T, Laya A, Vilalta R. Fifth-generation technologies for the connected car: Capable systems for vehicle-to-anything communications. IEEE vehicular technology magazine. 2018 Jul 17;13(3):28-38.

Garg S, Kaur K, Kaddoum G, Ahmed SH, Jayakody DN. SDN-based secure and privacy-preserving scheme for vehicular networks: A 5G perspective. IEEE Transactions on Vehicular Technology. 2019 May 20;68(9):8421-34.

Varma IM, Kumar N. A comprehensive survey on SDN and blockchain-based secure vehicular networks. Vehicular Communications. 2023 Aug 22:100663.

Meyer P, Hackel T, Langer F, Stahlbock L, Decker J, Eckhardt SA, Korf F, Schmidt TC, Schüppel F. A security infrastructure for vehicular information using sdn, intrusion detection, and a defense center in the cloud. In2020 IEEE Vehicular Networking Conference (VNC) 2020 Dec 16 (pp. 1-2). IEEE.