



SUCCESS-6G: EXTEND

WP5 Deliverable E13

Project Title:	SUCCESS-6G: EXTEND
Title of Deliverable:	Initial testing and preliminary validation of service KPIs
Status-Version:	v1.0
Delivery Date:	18/02/2025
Contributors:	Allen Abishek, Ricard Vilalta, Raul Muñoz (CTTC), Maria A. Serrano (NBC), Miguel Fornell, Francisco Paredes (IDNEO)
Lead editor:	CTTC
Reviewers:	Charalampos Kalalas (CTTC)
Keywords:	Real-time location awareness; end-to-end latency; closest node selection

Document revision history

Version	Date	Description of change
v0.1	30/09/24	Table of Contents (ToC)
v0.2	15/10/24	Content added
v0.3	30/11/24	Additional inputs
v0.4	31/01/25	Final inputs and revision
v1.0	18/02/25	Final version uploaded to the website

Disclaimer

This report contains material which is the copyright of certain SUCCESS-6G Consortium Parties and may not be reproduced or copied without permission. All SUCCESS-6G Consortium Parties have agreed to publication of this report, the content of which is licensed under a Creative Commons Attribution-NonCommercial-NoDerivs 3.0 Unported¹.



CC BY-NC-ND 3.0 License – 2022-2025 SUCCESS-6G Consortium Parties

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

Executive Summary	3
Table of Contents	4
List of Figures	5
1 Introduction	6
2 Use case 2: Automated software updates for vehicles	7
2.1 General description and overall objectives	7
2.2 User story 2.1: Over-the-air vehicular software updates with robust V2X connectivity	8
2.3 Overall UC2 architecture and network deployments	8
2.4 Facilities for use case 2	9
2.4.1 ADRENALINE Testbed	9
2.4.2 Nextcloud platform	10
3 Over-the-air vehicular software updates with robust V2X connectivity: Implementation at the ADRENALINE testbed	12
3.1 Location-awareness	12
3.2 Requirements and KPIs	12
3.2.1 Requirements	13
3.2.2 Key Performance Indicators (KPIs)	14
3.3 UC2 Architecture and network deployment.....	15
3.4 Exposed interfaces.....	16
3.5 Workflow	18
3.6 Preliminary experimental validation of the functionalities	20
4 Use case 2 proof-of-concept (PoC)	21
4.1 Phase 1: PoC at Nextcloud	21
4.1.1 On-board unit test setup	21
4.1.2 TLS Reverse Tunnel.....	22
4.1.3 Firewall on the on-board unit.....	23
4.2 Phase 2: PoC at Castellolí.....	24
5 Conclusions	26
6 References	27

List of Figures

Figure 1: Implementation phases for the automated software updates.....	7
Figure 2 Proposed overall UC2 system architecture	9
Figure 3 ADRENALINE Testbed to be used for Use Case 2	10
Figure 4 Instantiation of SUCCESS-6G-EXTEND architecture for OTA vehicular software updates with robust V2X connectivity	15
Figure 5 Sequence diagram for OTA vehicular software updates with robust V2X connectivity	19
Figure 6 Wireshark of location-aware service establishment.....	20
Figure 7: UC2 phase 1: development architecture	21
Figure 8: Test setup	22
Figure 9: UC2 phase 2: real architecture (Castelloli).....	25

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. SUCCESS-6G-EXTEND 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 primary 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 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 on-board vehicular devices. Wireless channel impairments may, however, adversely impact the 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. The 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. In particular, Figure 2 Proposed overall UC2 system architecture provides a high-level system architecture for OTA vehicular software updates within a robust Vehicle-to-Everything (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.

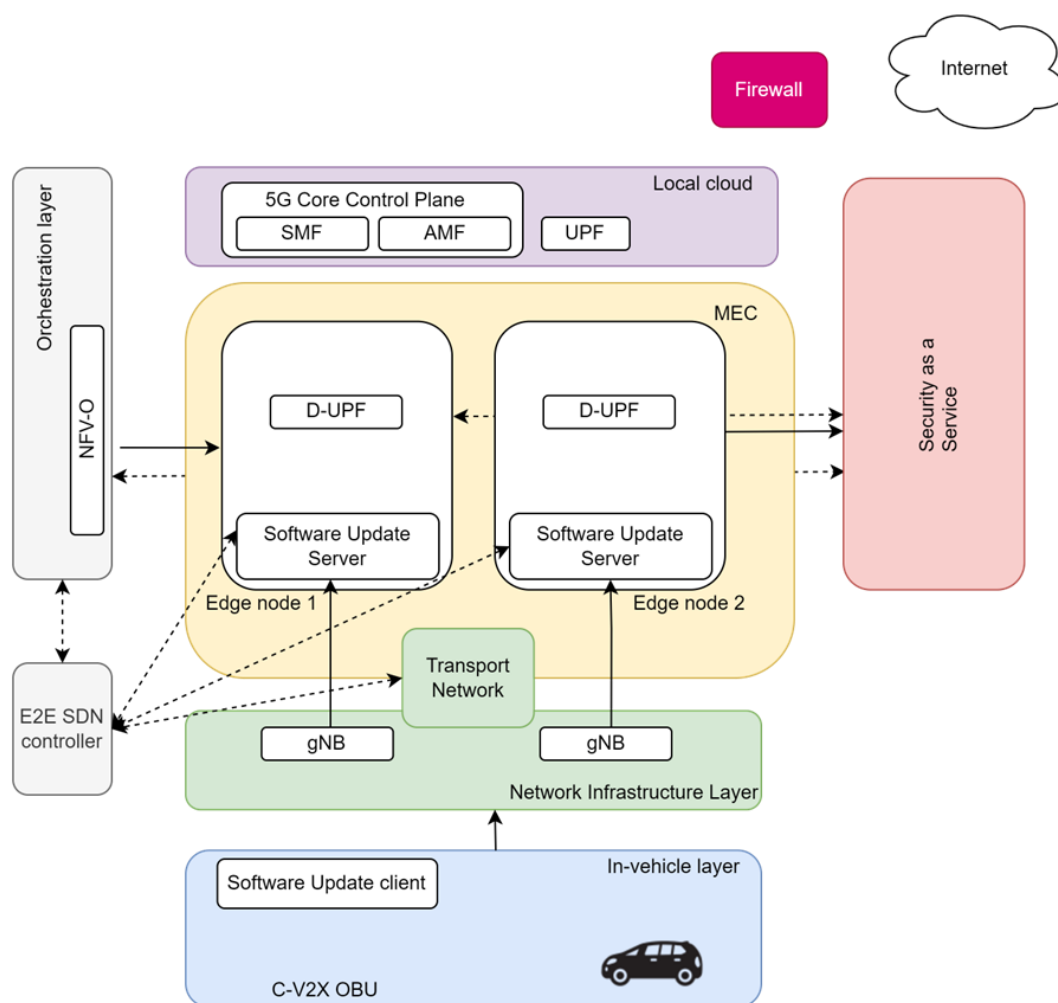


Figure 2 Proposed overall UC2 system architecture

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

2.4.1 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 the ADRENALINE testbed, to be used for the execution of SUCCESS-6G-EXTEND.

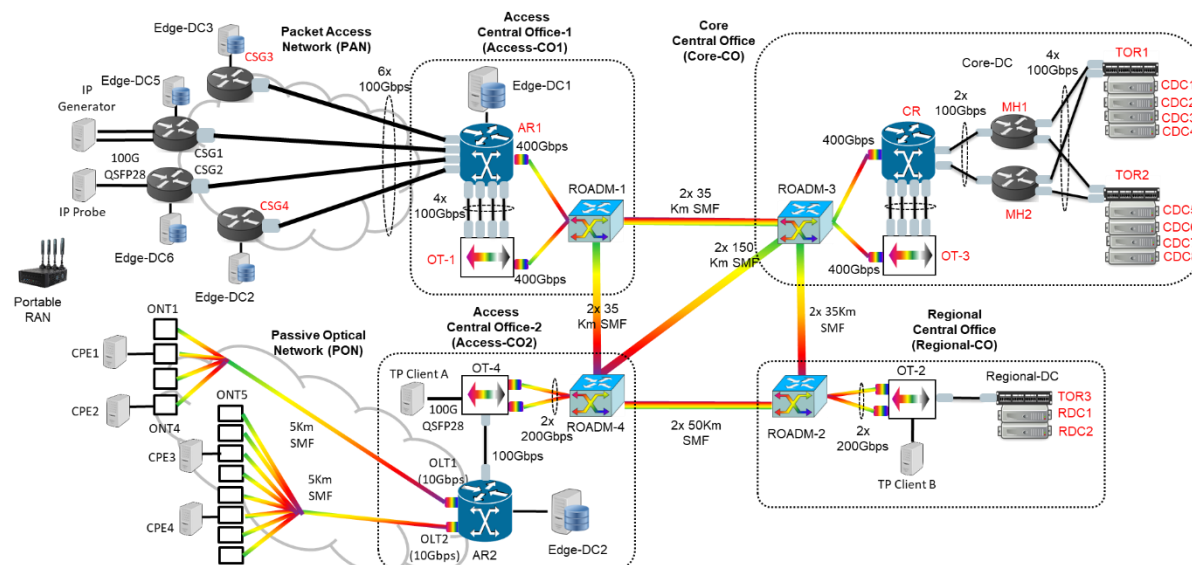


Figure 3 ADRENALINE Testbed to be used for Use Case 2

ADRENALINE spans the access, aggregation-metro, and core segments, and includes distributed Data Centres (DCs) geographically dispersed and located at the edge or in central locations. As depicted in Figure 3, 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 endpoints. 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; ii) a network slice is composed of one or several concatenated network services that deploy a set of NFs.

2.4.2 Nextcloud platform

NextCloud is an open-source private cloud platform based on a dedicated server with extensive resources, orchestrated by a Debian-based distribution called Proxmox to manage virtualization

environments, offering a simple web interface to manage both virtual machines (VMs) and containers. These machines can be of various types, but we ideally use two of them:

- KVM (Kernel-based Virtual Machine): It is used for full virtualization, allowing to run complete operating systems as VMs.
- LXC (Linux Containers): It also supports LXC containers, which are lighter and more efficient environments to run applications or services.

These virtual machines can be scaled on demand, whether in storage, memory, or network resources. NextCloud bundles core technologies such as PHP, JavaScript, HTML, CSS, MySQL, SQLite, Redis and Nginx, and provides full control over the data and privacy, offering self-host option to deploy custom applications. The following capabilities ensure that NextCloud is ideal for delivering SUCCESS-6G-EXTEND catalogue services such as vehicle data storage and Firmware Over-The-Air (FOTA). NextCloud will be fully customized to fit the needs of the project.

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 behavior 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 behavior, 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 **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 connected vehicles.

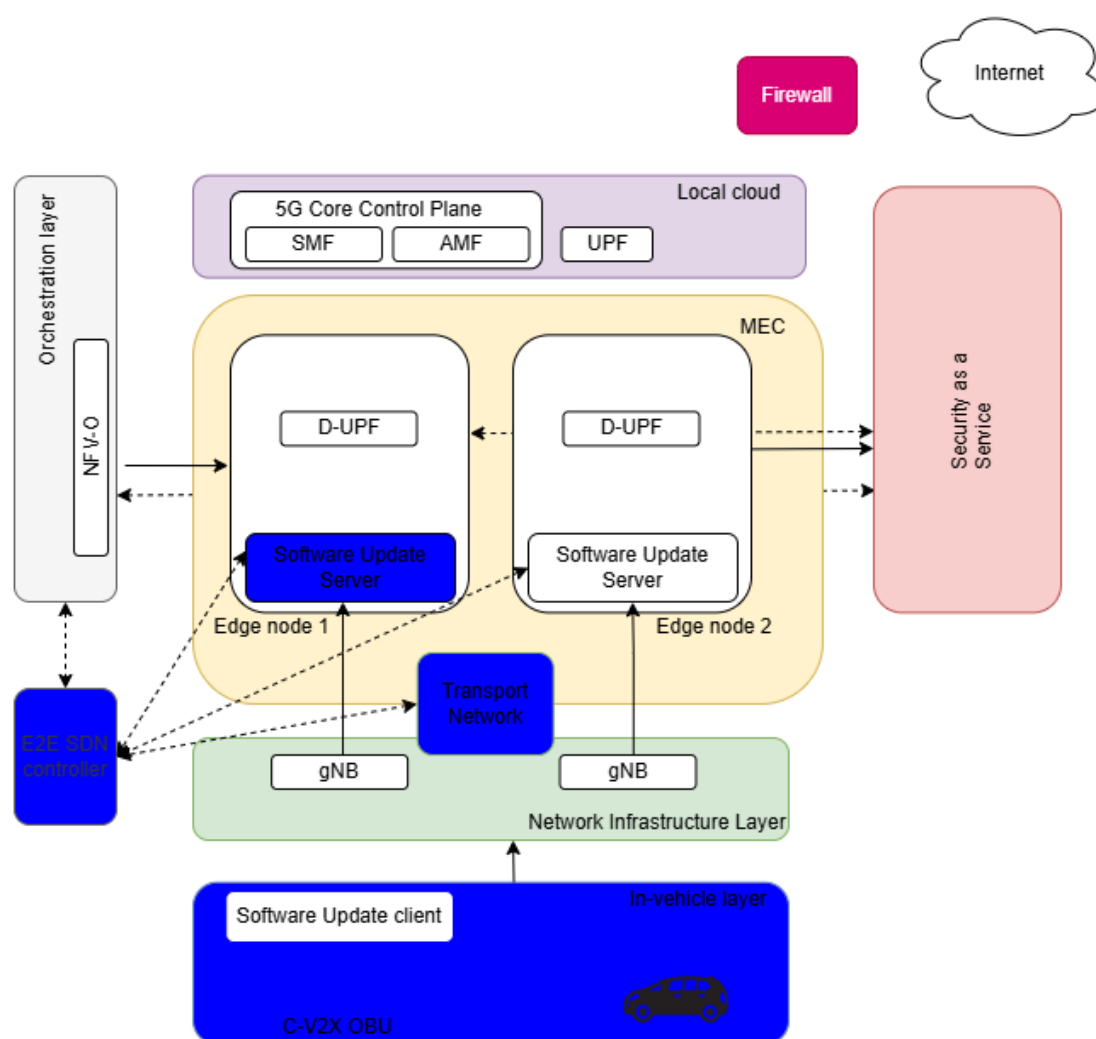


Figure 4 Instantiation of SUCCESS-6G-EXTEND architecture for OTA vehicular software updates with robust V2X connectivity

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 OTA 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 OTA 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.

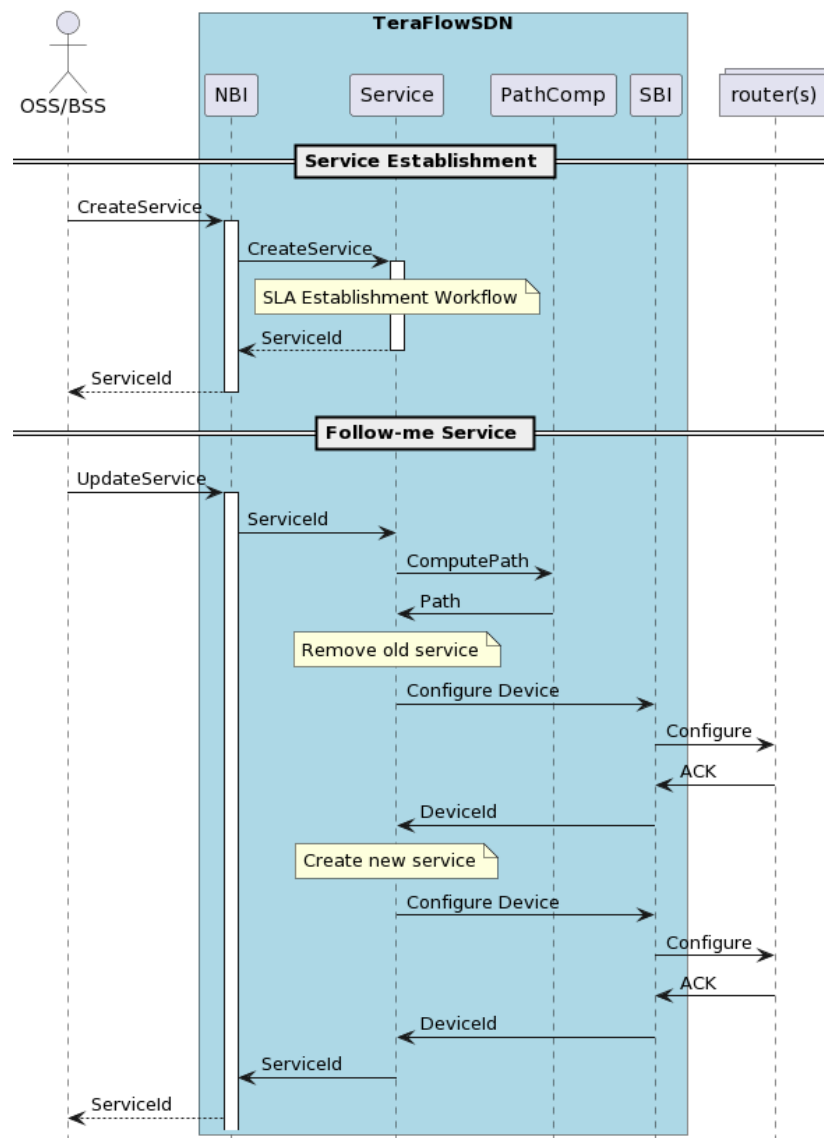


Figure 5 Sequence diagram for OTA vehicular software updates with robust V2X connectivity

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 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 it is acknowledged. The 3rd packet corresponds to the update and actual provisioning of the connectivity service, while the 4th packet corresponds to it is acknowledged.

```
*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

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

4.1 Phase 1: PoC at Nextcloud

To carry out phase 1 testing, a test setup has been provided where the entire team can work on the same devices. This has the following advantages.

The acquisition of vehicular data requires tools to facilitate development work. For this, logs were created from the data provided by the sensors, which we called Datasets. These corresponded to the acquired analog values along with their respective timestamps. Figure 7 shows the distinction between the actual implemented architecture and the development architecture.

Remote Collaboration: Developers can access hardware from anywhere in the world, allowing them to work on the project regardless of their physical location.

Reduced operational costs: Only one set of hardware is needed, which can be shared by all developers. This reduces the costs of purchasing multiple devices for each team member.

More efficient testing: Hardware can be accessed 24/7, allowing for continuous testing and debugging, even when developers are in different time zones.

Ease of maintenance and updating: Updates to the operating system or applications built for the project can be performed centrally and instantly available to all developers.

Scalability: If more developers need to work on a project, access to hardware can be easily scaled without the need to purchase more physical equipment.

Security and Access Control: Security controls can be implemented to restrict access to hardware to only those who need it, ensuring that sensitive or confidential data is protected.

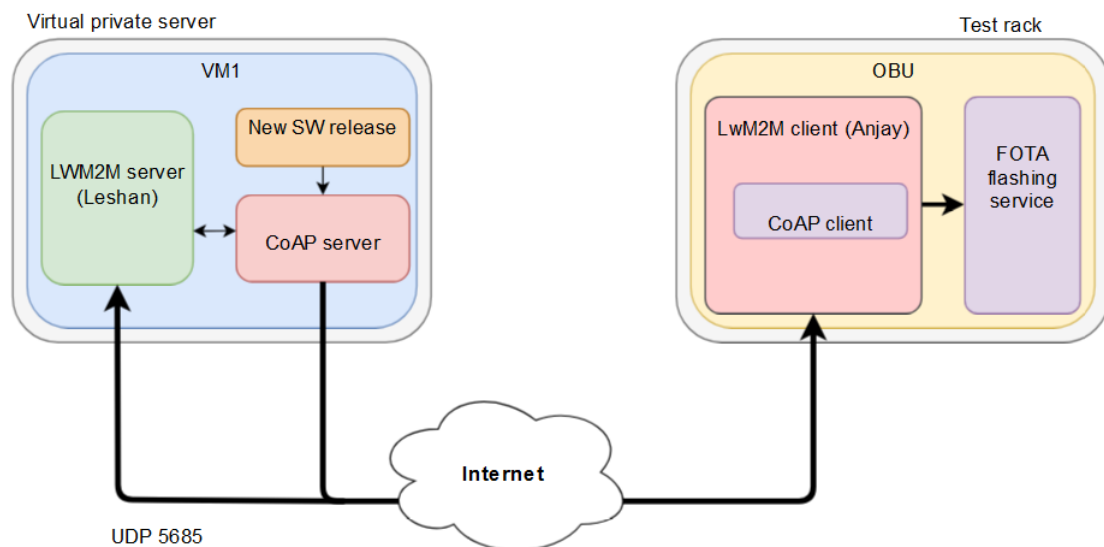


Figure 7: UC2 phase 1: development architecture

4.1.1 On-board unit test setup

The on-board unit (OBU) developed, called VMAX, is based on an aarch64 architecture, as opposed to the PCs used for development, which is based on the X86-64 architecture.

```

vmaxsetup@vmax:~$ ssh -oHostKeyAlgorithms=+ssh-rsa root@192.168.2.4
root@192.168.2.4's password:
~# uname -a
Linux ag215scnaa 4.14.206-perf #1 SMP PREEMPT Tue Dec 12 09:58:25 UTC 2023 aarch64 GNU/Linux
  
```

This means our code must undergo a process known as cross-compilation, which involves compiling a software project on a machine with one architecture, in this case, X86-64, so that it can run on a machine with a different architecture. For this reason, a test setup is necessary for the cross-compiled code. This setup consists of a Raspberry Pi connected to two device units equipped with a C-V2X PC5 interface for radio communications, linked via an Ethernet switch (see Figure 8). The Raspberry Pi is accessible over the Internet through a TLS reverse tunnel to a dedicated server managed by IDNEO, and from there, access to the VMAX units is provided via serial and/or local network interfaces.

Alternatively, the setup also allows access to the VMAX units through a Uu cellular connection, either via access permissions granted to the public IP of the IDNEO dedicated server or through other permanent TLS reverse tunnels.

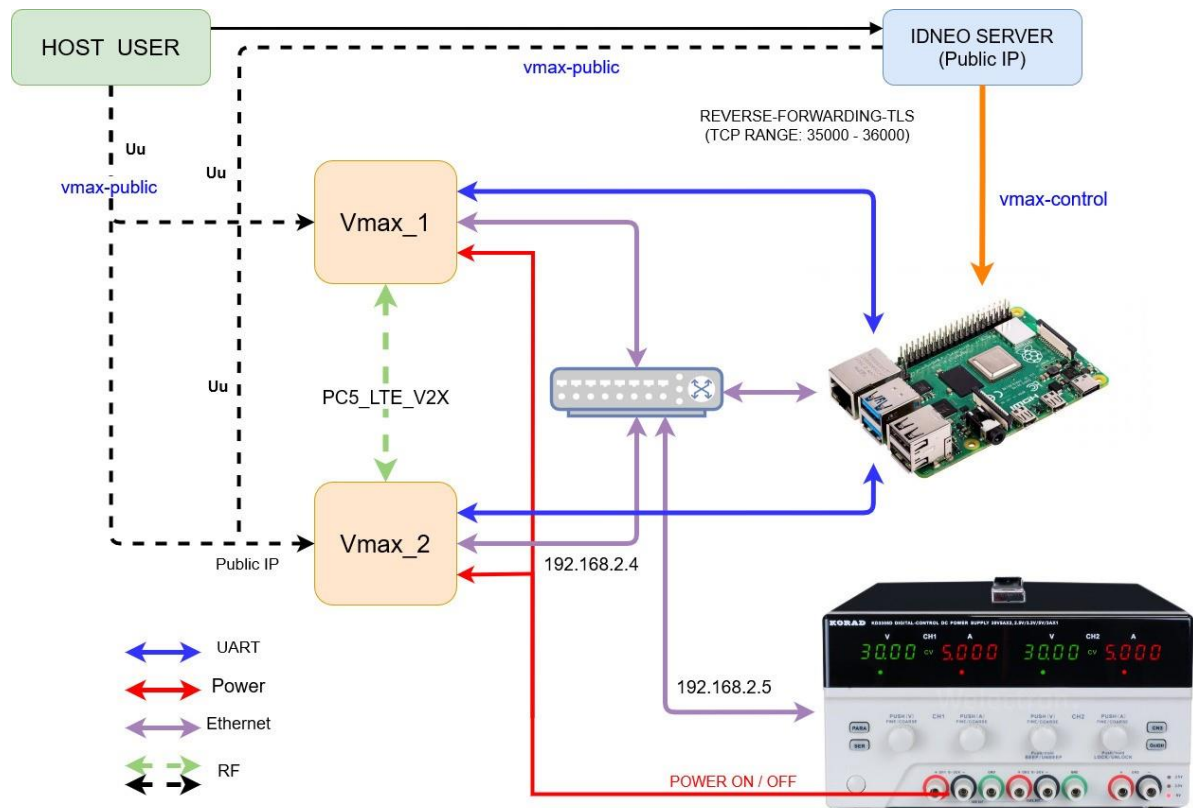


Figure 8: Test setup

In Figure 8, it is possible to identify the access interfaces that allow remote control of the Vmax. The control that this setup allows with respect to each Vmax, goes from turning it on or off, through a network-managed power source or serial interfaces that allow debugging from the Vmax boot start-up from the Raspberry Pi.

4.1.2 TLS Reverse Tunnel

This tunnel allows to create a permanent connection between the Raspberry Pi, which is located behind a local network, and the IDNEO server. To run this tunnel, the openssl TLS client is used in reverse forwarding mode, which establishes a port on the remote host and redirects it locally to the Raspberry Pi TLS server. To do this, a systemd service is created that executes the corresponding command and manages the recovering process.

```
* backdoor.service - Backdoor Service
   Loaded: loaded (/etc/systemd/system/backdoor.service; enabled; preset: enabled)
   Active: active (running) since Thu 2024-10-10 15:34:38 CEST; 6min ago
     Until: Thu 2024-10-10 18:34:38 CEST; 2h 53min left
   Main PID: 28106 (bash)
     Tasks: 2 (limit: 8734)
        CPU: 136ms
   CGroup: /system.slice/backdoor.service
           |-28106 /bin/bash /usr/bin/backdoor.sh
           `--28108 /usr/bin/ssh -N -R 35990:localhost:22 ficosa-root@193.70.33.60 -p 27022

oct 10 15:34:38 vmax systemd[1]: Started backdoor.service - Backdoor Service.
oct 10 15:34:38 vmax bash[28106]: New port: 35990
```

On the IDNEO server side, this TLS client has automatic access because the public key of the client host, Raspberry Pi, is in the list of authorized keys on the TLS server, so no user and password authentication is needed.

4.1.3 Firewall on the on-board unit

The iptables architecture relies on these three components working together:

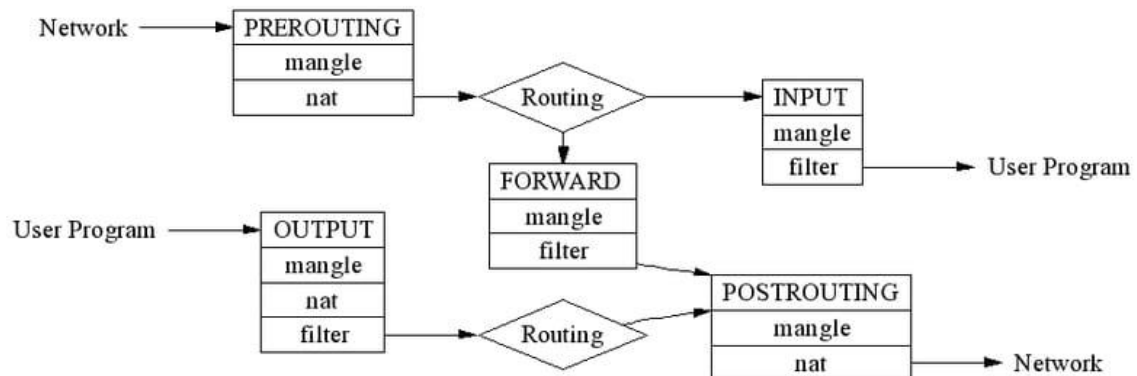
- **Tables:** Organize rules by function (e.g., filtering, mangling).
- **Chains:** Define the path packets take and the order in which rules are applied.
- **Rules:** Specify the criteria for matching packets and the actions to take (e.g., ACCEPT, DROP, REJECT).

The filter table is the default table used to filter packets. Its rules determine whether a packet should be accepted or rejected. This table contains the core chains used for traffic management: INPUT, OUTPUT, and FORWARD.

INPUT: This chain handles incoming traffic destined for the local system itself. When a packet arrives at a network interface and is intended for the device, it is processed by the rules within the INPUT chain.

OUTPUT: This chain handles outgoing traffic originating from the local system. Any packets generated by the device and sent out through a network interface are processed by the rules in the OUTPUT chain.

FORWARD: This chain handles traffic that is neither destined for nor originating from the local system. It is used for packets that are being routed or forwarded to other networks or machines. This chain is essential for devices acting as routers or gateways.



In the OBU, restrictive input rules have been configured, which are automatically configured at system startup. OBUs have access to the public network through the Quectel AG550 modem. In this modem, incoming and outgoing traffic rules are configured for network management. By configuring the INPUT and OUTPUT chains to discard unauthorized traffic and the FORWARDING chain to only allow access to port 22 (TLS server of the AG215 AP) to certain configured public IP addresses, a filter is achieved for all traffic that is not allowed access.

```

/ # iptables -L -t filter
Chain INPUT (policy DROP)
target    prot opt source                destination
ACCEPT    icmp -- anywhere              anywhere
ACCEPT    all  -- anywhere              anywhere
ACCEPT    all  -- 192.168.225.0/30      192.168.225.0/30

Chain FORWARD (policy ACCEPT)
target    prot opt source                destination
ACCEPT    tcp  -- 193.70.33.60          anywhere          tcp dpt:ssh
ACCEPT    tcp  -- 88.98.97.234          anywhere          tcp dpt:ssh
ACCEPT    tcp  -- 46.136.172.101        anywhere          tcp dpt:ssh
DROP      tcp  -- anywhere              anywhere          tcp dpt:ssh

Chain OUTPUT (policy DROP)
target    prot opt source                destination
ACCEPT    icmp -- anywhere              anywhere
ACCEPT    all  -- anywhere              anywhere
ACCEPT    all  -- 192.168.225.0/30      192.168.225.0/30
  
```

This configuration allows to avoid brute force attacks since the TLS server will only serve the clients that the iptables rules allow.

4.2 Phase 2: PoC at Castellolí

To deploy phase 2 of the tests, all software, both server-side and OBU software, has been migrated to their final locations: the Castellolí server and the IDNEO-developed OBU installed in the test vehicle. Figure 9 details the phase 2 architecture.

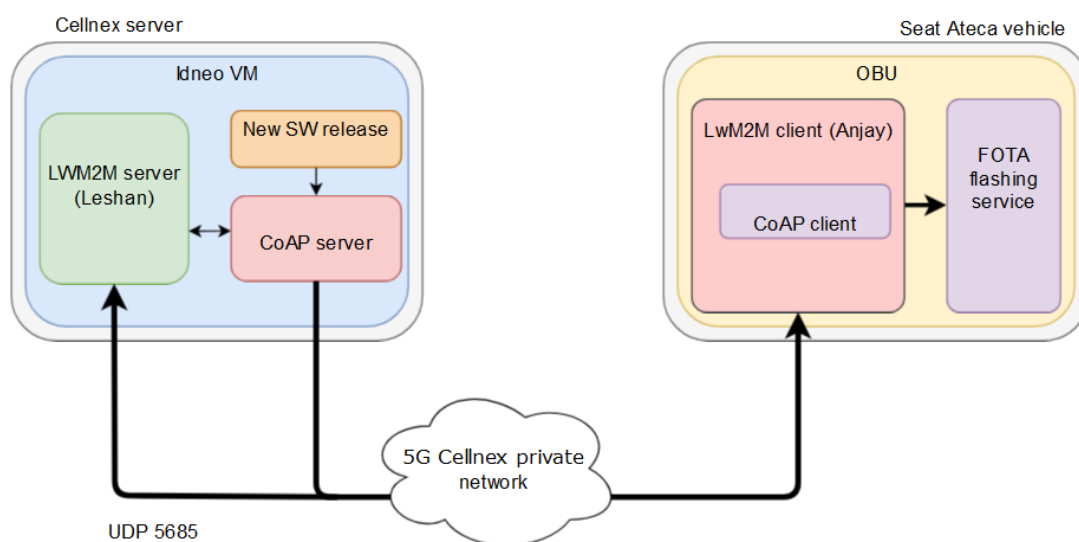


Figure 9: UC2 phase 2: real architecture (Castelloli)

This tool simplifies development to the extent that accessing the vehicle for algorithm testing was no longer necessary, as real values were used for testing, albeit with different timestamps.

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

- [1] 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. In 2024 24th International Conference on Transparent Optical Networks (ICTON) 2024 Jul 14 (pp. 1-4). IEEE.
- [2] 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. In 2018 4th IEEE Conference on Network Softwarization and Workshops (NetSoft) 2018 Jun 25 (pp. 378-383). IEEE.
- [3] 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.
- [4] 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-everything communications. *IEEE vehicular technology magazine*. 2018 Jul 17;13(3):28-38.
- [5] 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.
- [6] Varma IM, Kumar N. A comprehensive survey on SDN and blockchain-based secure vehicular networks. *Vehicular Communications*. 2023 Aug 22:100663.
- [7] 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. In 2020 IEEE Vehicular Networking Conference (VNC) 2020 Dec 16 (pp. 1-2). IEEE.