SUCCESS-6G: VERIFY

WP5 Deliverable E12

Over-the-air software updates for seamless operation of vehicles

Project Title:	SUCCESS-6G: VERIFY
Title of Deliverable:	Over-the-air software updates for seamless operation of vehicles
Status-Version:	v1.0
Delivery Date:	31/01/2024
Contributors:	Ricard Vilalta, Raul Muñoz (CTTC), Maria A. Serrano (NBC), Miguel Fornell, Francisco Paredes (Idneo)
Lead editor:	CTTC
Reviewers:	Ricard Vilalta (CTTC)
Keywords:	Service establishment; service management; bandwidth management

Document revision history

Version	Date	Description of change
v0.1	30/11/23	Table of Contents (ToC) and initial content added
v0.2	20/12/23	Main content added
v0.3	15/01/24	Final additions, proofreading
v1.0	31/01/24	Final version uploaded to the website

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

This deliverable provides the description of the architectural framework adopted in SUCCESS-6G-VERIFY for the computationally efficient process of vehicular software updates in an over-the-air manner. Besides detailing the innovations applicable to use case 2, the document elaborates on the synergistic relationship between the bandwidth management service and the overall network architecture in optimizing resource allocation for over-the-air software updates. In addition, the workflow defining the sequence of commands that facilitate bandwidth management and integration with the software-defined-network controller is explained.

Table of Contents

List of Figures

1 Introduction

This deliverable provides the description of the architectural framework adopted in SUCCESS-6G-VERIFY for the computationally efficient process of vehicular software updates in an over-the-air manner. Besides detailing the innovations applicable to use case 2, the document elaborates on the synergistic relationship between the bandwidth management service and the overall network architecture in optimizing resource allocation for over-the-air software updates. In addition, the workflow defining the sequence of commands that facilitate bandwidth management and integration with the software-defined-network controller is explained.

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 for 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.

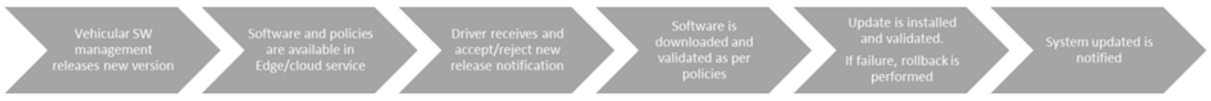

Vehicular SW management releases new version
Software and policies are available in Edge/cloud service
Driver receives and accept/reject new release notification
Software is downloaded and validated as per policies
Update is installed and validated.
If failure, rollback is performed
System updated is notified

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 2.3: Over-the-air vehicular software updates with efficient computation

Over-the-air software updates would substantially benefit from the realization of computationally efficient mechanisms to dramatically reduce data processing times. By leveraging edge-processing capabilities in conjunction with distributed learning techniques, the associated computational overhead is expected to be relieved. Additionally, zero-touch orchestration of container-based solutions will ensure the automatic reconfiguration of vehicular on-board units with an efficient usage of computational resources, while adapting to V2X network topology changes. This would require agile management of edge-cloud continuum to guarantee seamless service delivery.

2.3 Overall UC2 architecture

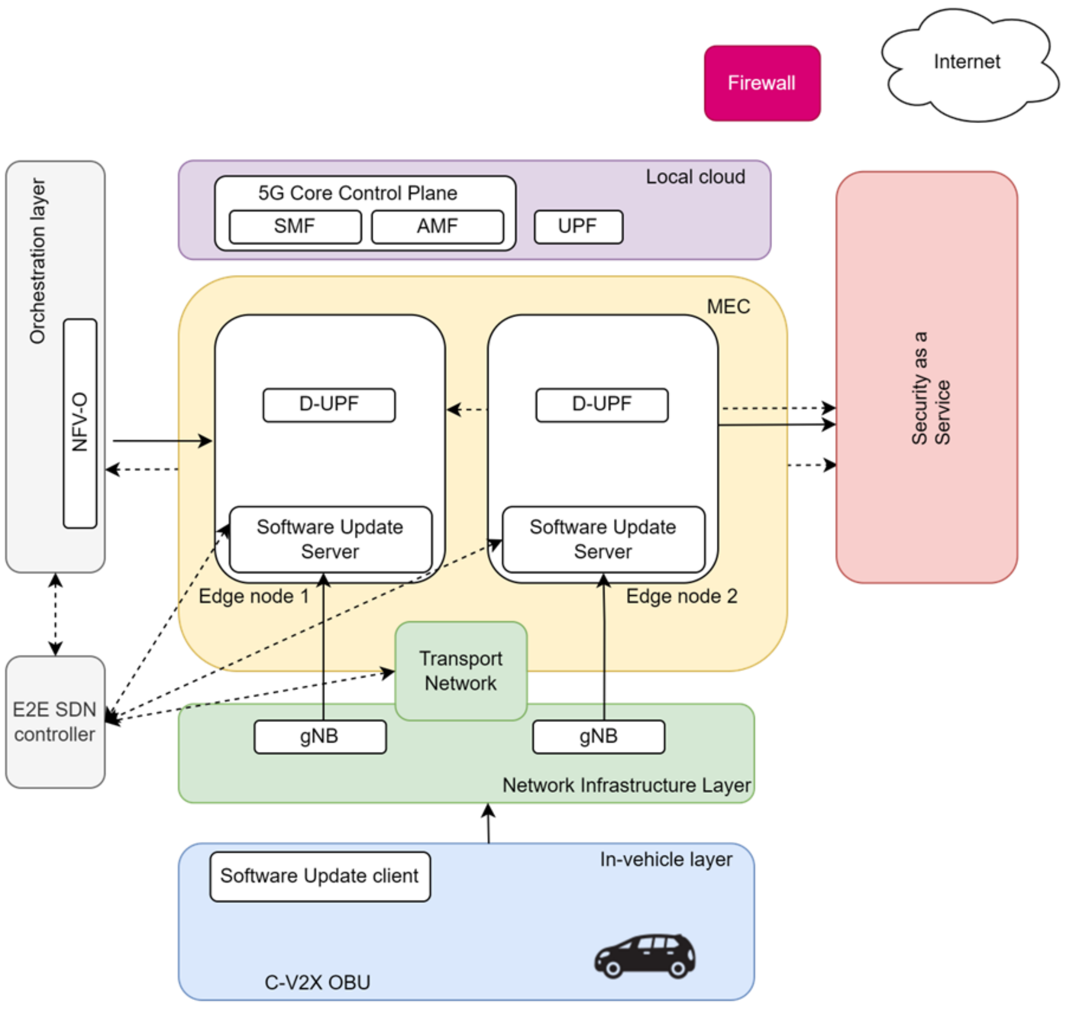


Figure 2 Proposed overall UC2 system architecture

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.

Central to this system is the Security as a Service component, a crucial pillar ensuring secure OTA updates. It acts as a protective conduit between the server, situated within the Multi-Access Edge Computing (MEC) environment, and the client applications. This setup is bolstered by strategically placed probes on the Transport Network. These probes are not mere passive elements; they actively monitor and analyse traffic between the client and server, effectively identifying and mitigating potential security threats. This proactive approach is pivotal in maintaining the integrity and security of the system.

The architecture's robustness is further enhanced through its innovative use of location-awareness, particularly in the End-to-End (E2E) Software Defined Networking (SDN) controller. The system cleverly integrates GPS data, enriching the network with spatial intelligence. This information plays a crucial role in managing Connectivity Service Requests, enabling the E2E SDN controller to utilize an advanced algorithm for optimal location matching. This feature not only improves the efficiency of the network but also bolsters the reliability and responsiveness of the system.

A standout feature of this architecture is the integration of the European Telecommunications Standards Institute (ETSI) MEC 015 API within the E2E SDN controller. This API is a game-changer in bandwidth management at the Transport Network level. It ensures that computational resources are distributed effectively, catering specifically to the unique bandwidth demands of various applications. This level of resource management is key to maintaining a high-performance, efficient system.

To further detail the figure describing the system architecture for Over-the-air (OTA) Software Updates within the SUCCESS-6G framework, let's envision the components and their interactions more vividly:

- Security as a Service Component:
    - Positioned at the heart of the architecture, it ensures secure communication channels between the MEC server and client applications.
    - Implements advanced encryption and authentication protocols to safeguard data integrity and privacy.
- Multi-Access Edge Computing (MEC) Server:
    - Hosts the server-side components of OTA updates.
    - Employs high-speed computing capabilities to process requests efficiently.
    - Connected to various network interfaces to handle diverse client requests.
- Probes on the Transport Network:
    - Strategically placed to monitor data flow between the server and clients.
    - Equipped with anomaly detection algorithms to identify and alert any unusual patterns or potential security threats.
- End-to-End (E2E) Software Defined Networking (SDN) Controller:
    - Serves as the central command center for network management.
    - Integrates GPS data for location-aware routing and optimal path selection.
    - Employs a sophisticated algorithm for dynamic allocation of network resources based on real-time data.
- European Telecommunications Standards Institute (ETSI) MEC 015 API Integration:
    - Facilitates the smooth interaction between the client application and the MEC server.
    - Optimizes bandwidth management, ensuring that resources are allocated based on the specific needs of each application.
- Client Applications:
    - Various endpoints that receive OTA updates.

- Could range from individual devices to larger systems within vehicles or other connected entities.
- Designed to request and seamlessly integrate updates without disrupting their primary functions.

- Transport Network:
  - Acts as the backbone, connecting all components.
  - Designed for high throughput and low latency to support real-time data exchange.
- Connectivity Service Requests Handling:
  - Managed through the E2E SDN controller.
  - Prioritizes requests based on urgency, bandwidth requirements, and geographical location.

In summary, this system architecture is a comprehensive solution that not only efficiently manages OTA Software Updates but also establishes a robust Vehicle-to-Everything (V2X) connectivity framework. It achieves this through a combination of advanced security measures, intelligent use of spatial data, and efficient resource allocation. This architecture is not just a framework for today's needs but a forward-thinking approach, ready to meet the evolving demands of connected systems in the future.

2.4 Innovations applicable to use case 2

In this section we detail the different innovations that are being proposed in UC2: Over-the-air software updates, Orchestration Layer, Transport Network and Security as a Service.

2.4.1 Firmware Over-The-Air Software update

In this section, the architecture for FOTA is firstly provided (as in Figure 3), then the focus is given on Lightweight Machine-to-Machine (LwM2M) protocol for the FOTA process in the two sides of the communication, and finally, the lifecycle of the FOTA process is described.

2.4.1.1 Software innovation

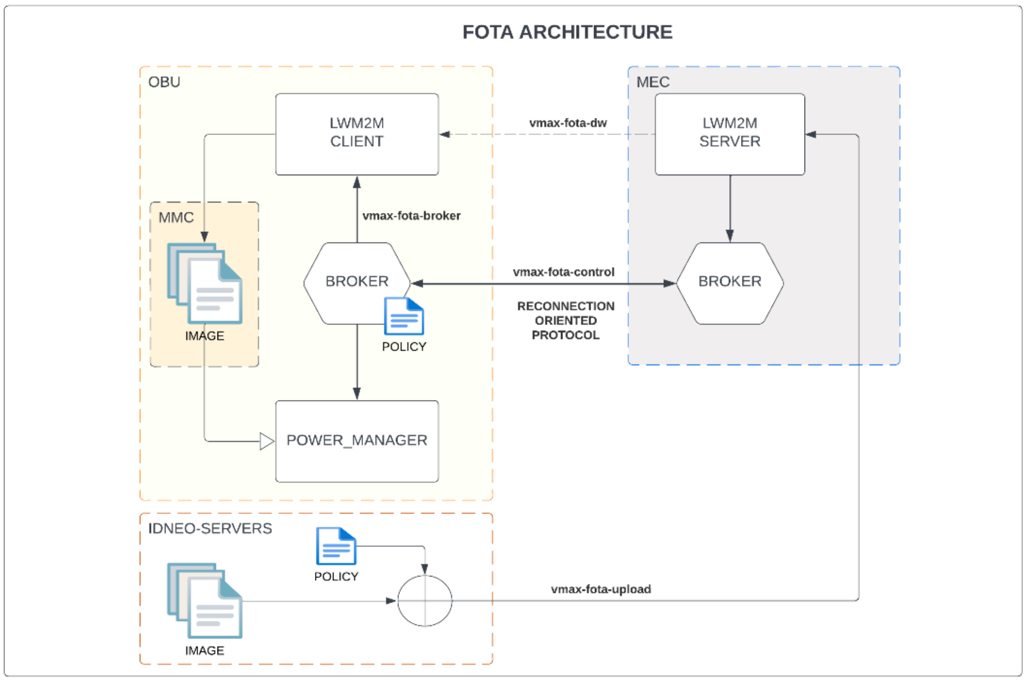
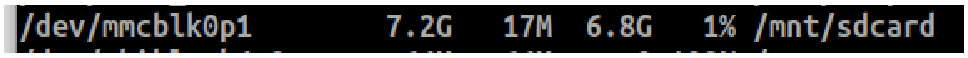


Figure 3 Architecture of the innovation associated to UC2

A series of innovations take place in the software side in the UC2:

- Client-Server. The image download mechanism will be carried out by the client and server, based on LWM2M.
- Broker. The control plane for policy transfer will be carried out by a bridge connection between MQTT brokers.
- Power Manager. The Power manager application subscribes to the broker to receive the policies of the new Image. Also, it is responsible for **executing deployment tasks** to load in the OBU when the car is turned off **(upgrade process)**.
- eMMC. It's a large space of memory (7.2G) integrated to the filesystem in the OBU to host the last version of the Image.



```
/dev/mmcblk0p1          7.2G    17M   6.8G    1% /mnt/sdcard
```

Description of the network interfaces:

- Vmax-fota-upload: Interface dedicated to upload the new image and policies to the server in MEC (LWM2M protocols)
- Vmax-fota-dw: Interface dedicated to downloading the image and saving it in eMMC.
- Vmax-fota-broker: Interface that permits notifications between heterogenous applications and a broker in the MEC.

2.4.1.2 LwM2M (Lightweight Machine-to-Machine)

The primary innovation related to use case 2 is the use of the Lightweight Machine-to-Machine (LwM2M) protocol for the FOTA process in the two sides of the communication, the OBU and the MEC. The OMA Lightweight Machine-to-Machine (LwM2M) protocol is a standard driven by the Open Mobile Alliance for M2M communication and device management in IoT environments. It is a specialized protocol designed for the management of IoT devices, particularly those operating within constrained environments and networks. Originally developed to cater to the needs of constrained devices in the IoT ecosystem, LwM2M has demonstrated its efficacy in deployment across a spectrum of IoT devices, including high-end ones. Its applications span diverse verticals such as smart energy, building automation, precision farming, and logistics, among others. LwM2M is built on the Constrained Application Protocol (CoAP) and supports the User Datagram Protocol (UDP) at the transport layer, with additional support for Short Message Service (SMS). LwM2M standard defines the application layer communication protocol between an LwM2M server and an L2M2M client, as shown in Figure 4.

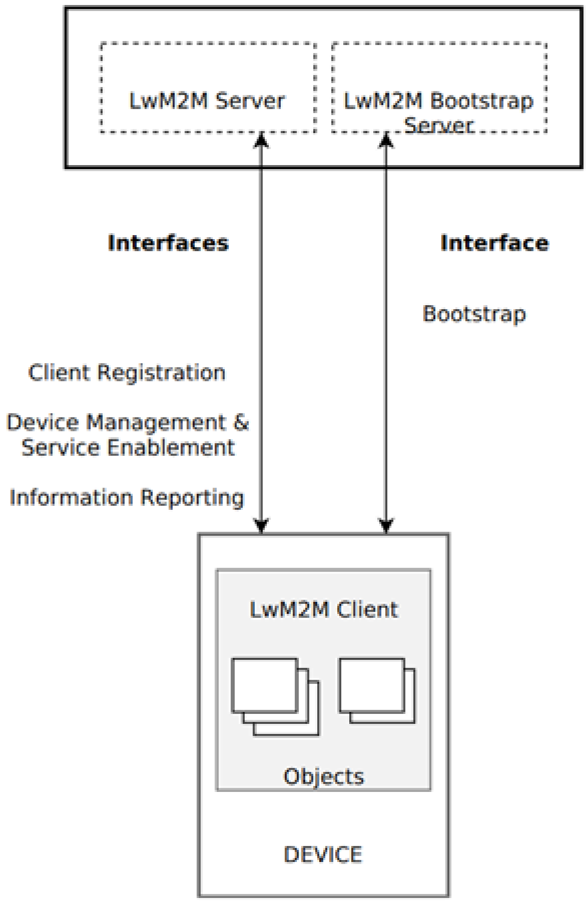


Figure 4 LwM2M overall architecture of the LwM2M enabler

Among all the LwM2M objects defined in the protocol, special mention will be made of the Firmware Update Object. This object can be used to perform a download, verification or update a remote device.

LwM2M Object enables management of firmware to be updated. This Object includes installing a firmware package, updating firmware, and performing actions after updating firmware. The firmware update *may* require rebooting the device, for this reason the automotive general procedure is to flash the device once detected that the car ignition signal is off and before shutdown the OBU. The envisioned functionality is to allow a LwM2M Client to connect to any LwM2M Server to obtain a firmware image using the object and resource structure defined. A LwM2M Server may also instruct a LwM2M Client to fetch a firmware image from a dedicated server (instead of pushing firmware images to the LwM2M Client). The Package URI resource is contained in the Firmware object and can be used for this purpose. A LwM2M Client MUST support block-wise transfer [CoAP_Blockwise] if it implements the Firmware Update object. A LwM2M Server will support block-wise transfer. The protocol layers are clearly defined and allow certain versatility in terms of their integration. For example, it is common for data transfer at the network layer to be done via UDP, which makes DTLS protocol necessary for the security layer. But the stack of LWM2M is configurable and TCP with TLS could be used. Other protocols, such as HTTP/HTTPs, *may* also be used for downloading firmware updates (via the Package URI resource). One of the innovations in SUCCESS6G is to use LwM2M for very large firmware sizes as the used in high ends OBUs.

2.4.1.3 Update process

The typical procedure for FOTA processes in vehicles involves waiting for the vehicle to shut down before proceeding with system updates. To ensure this, it is crucial to have precise control over the shutdown process of the OBU. This control allows for the interruption of the shutdown sequence to execute any pending updates. This process is carried out by the Power Manager, which is governed by a finite state machine. A complete sequence of the entire state machine running in the OBU is presented in Figure 5.

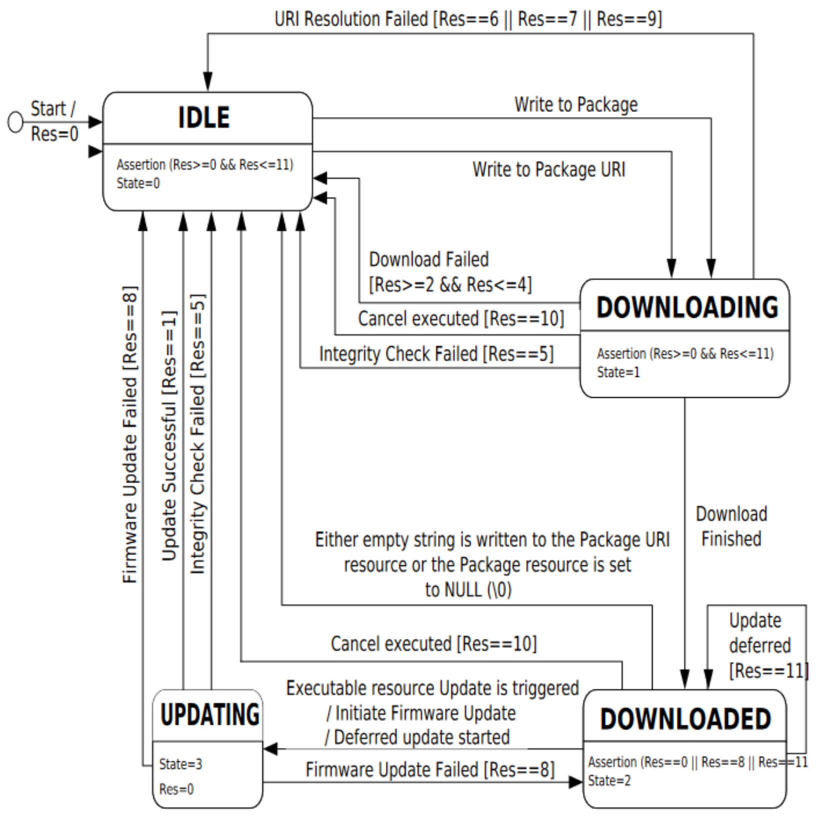

URI Resolution Failed [Res==6 || Res==7 || Res==9]
Start / Res=0
IDLE
Assertion (Res>=0 && Res<=11)
State=0
Write to Package
Write to Package URI
Download Failed [Res>=2 && Res<=4]
Cancel executed [Res==10]
Integrity Check Failed [Res==5]
DOWNLOADING
Assertion (Res>=0 && Res<=11)
State=1
Firmware Update Failed [Res==8]
Update Successful [Res==1]
Integrity Check Failed [Res==5]
Download Finished
Either empty string is written to the Package URI resource or the Package resource is set to NULL (\0)
Update deferred [Res==11]
Cancel executed [Res==10]
Executable resource Update is triggered / Initiate Firmware Update / Deferred update started
UPDATING
State=3
Res=0
Firmware Update Failed [Res==8]
DOWNLOADED
Assertion (Res==0 || Res==8 || Res==11
State=2

Figure 5 LwM2M software update mechanisms

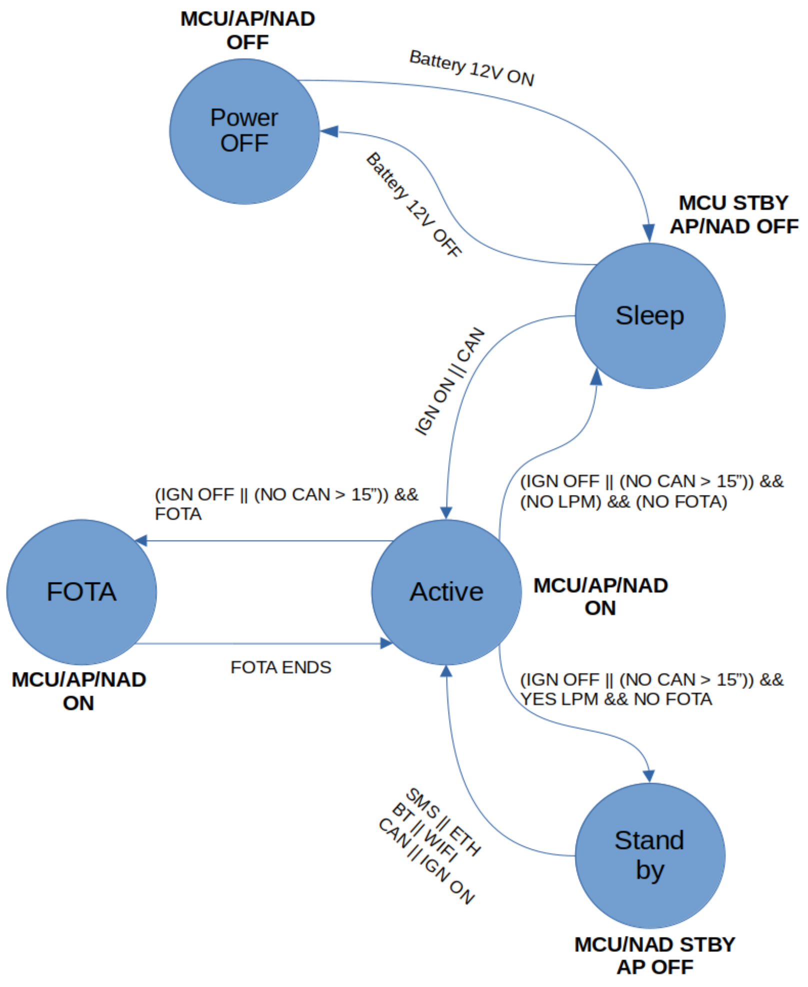

MCU/AP/NAD OFF
Power OFF
Battery 12V ON
Battery 12V OFF
MCU STBY AP/NAD OFF
Sleep
IGN ON || CAN
(IGN OFF || (NO CAN > 15")) && FOTA
(IGN OFF || (NO CAN > 15")) && (NO LPM) && (NO FOTA)
FOTA
Active
MCU/AP/NAD ON
FOTA ENDS
MCU/AP/NAD ON
(IGN OFF || (NO CAN > 15")) && YES LPM && NO FOTA
SMS || ETH BT || WIFI CAN || IGN ON
Stand by
MCU/NAD STBY AP OFF

Figure 6 OBU Power Manager flow diagram

The FOTA machine state occurs when a new image becomes available in the eMMC, previously downloaded by the Client LwM2M. Therefore, during the shutdown process, the presence of this file will be checked, and deployment policies, if any, will also be considered.

Once FOTA machine state finish its procedure, the power manager will continue its normal sequence of shutdown process as shown in Figure 6.

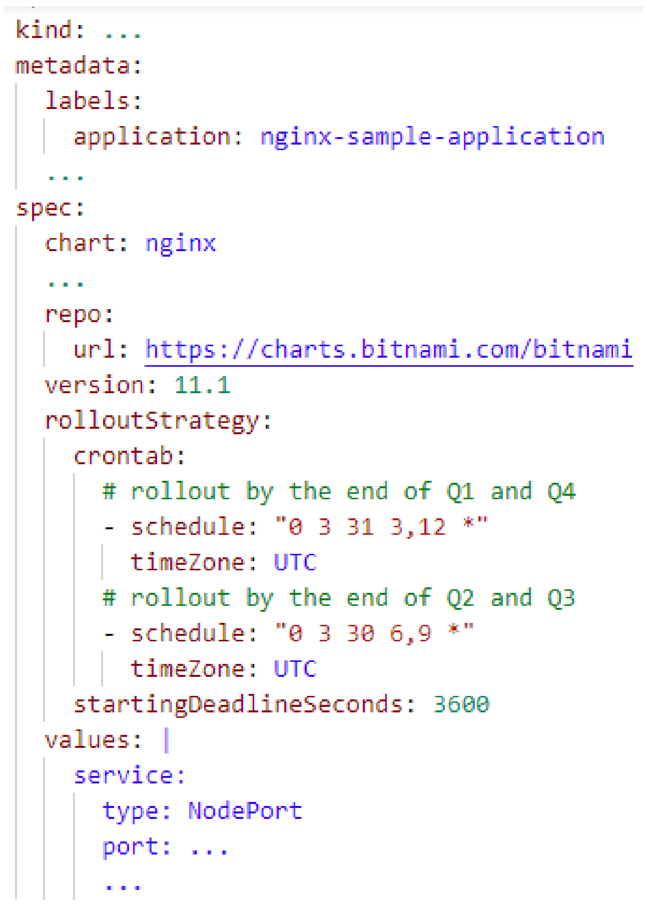
2.4.2 Orchestration layer

The E2E orchestrator NearbyOne is responsible for the service onboarding and lifecycle management (LCM) of cloud-native applications and infrastructure at a global scale, across the edge and cloud sites of the SUCCESS-6G architecture in general, and the UC2 architecture in particular.

The orchestration layer is described in SUCCESS-6G deliverable E6. “Diseño de la arquitectura del Sistema”. This section presents a relevant NearbyOne feature which is of interest to the use case 2. This feature allows to automate the software updates of the orchestrated services, implementing dynamic upgrades with rollout and rollback strategies. In particular, this feature defines how to upgrade different versions of applications, minimizing the downtime and disruption of the service, while ensuring a smooth deployment process.

The automated software update feature is configured in a YAML file as part of the Nearby Block orchestration resources (see Deliverable E6). A rollout strategy replaces, when configured, the old version of a running service with a new one, if it exists. Moreover, a rollback strategy is also supported to undo a rollout and revert to a previous version of the service if the new version is not in the desired state.

Figure 7 shows an example of the configuration of an automated software upgrade with rollout and rollback strategies. In this case, the default deployment is version 11.1 of a nginx service by Bitnami2. The rolloutStrategy specifies when to upgrade to a new version if it exists in the repository specified by the URL, using schedule statements based on Cron expressions3 and timeZone definition. A Cron Expressions is in the form <Minute> <Hour> <Day_of_the_Month> <Month_of_the_Year> <Day_of_the_Week>. In this particular example, the automated upgrade is scheduled to take place every quarter of the year (the 31st of March and December at 03:00 am, and the 30th of June and September at 03:00 am, UTC times). The startingDeadlineSeconds rule specifies the rollback strategy: in this example, if the new version is not ready after 3600 seconds, the previous version is reverted.



```
kind: ...
metadata:
  labels:
    application: nginx-sample-application
  ...
spec:
  chart: nginx
  ...
  repo:
    url: https://charts.bitnami.com/bitnami
  version: 11.1
  rolloutStrategy:
    crontab:
      # rollout by the end of Q1 and Q4
      - schedule: "0 3 31 3,12 *"
        timeZone: UTC
      # rollout by the end of Q2 and Q3
      - schedule: "0 3 30 6,9 *"
        timeZone: UTC
    startingDeadlineSeconds: 3600
  values: |
    service:
      type: NodePort
      port: ...
      ...
```

Figure 7. Automated upgrade with rollout strategy in a Nearby Block orchestrated service definition.

2.4.3 Transport network

TeraFlowSDN is an innovative, open-source, cloud-native, and carrier-grade SDN controller, a result of the collective efforts under the European Union-funded TeraFlow 5G PPP research project. It stands out as one of the first in its league to offer a micro-service-based architecture designed to integrate with current NFV (Network Functions Virtualization) and MEC (Multi-access Edge Computing) frameworks. With its release, TeraFlowSDN brings revolutionary features to flow management and network equipment integration, enhancing the service layer and infrastructure layer of network management.

[2] https://bitnami.com/stack/nginx

[3] https://www.nncron.ru/help/EN/working/cron-format.htm

TeraFlowSDN's ambition is clear: to introduce its deployments into operators' networks, providing a path to a more advanced and secure network management. It promises to push the boundaries of the current state-of-the-art in software-defined networks, particularly in the context of beyond 5G networks. The technology is centered around a cloud-native architecture with stateless microservices that interact to fulfill network management tasks, promoting open competition and innovation in transport network devices and network applications.

The second release of TeraFlowSDN, known as TeraFlowSDN 2.1, marks a significant leap in the field of software-defined networking. The update includes a host of new features and enhancements that emphasize high performance, seamless hardware integration, advanced SDN automation, and improved cybersecurity measures. It also brings forth transport network slicing, multi-tenancy, cyber threat analysis and protection, and extended support for distributed ledger technology and smart contracts.

TeraFlowSDN 2.1's focus on scalability allows multiple TFS instances to be deployed within the same MicroK8s environment, enhancing the system's flexibility. The introduction of a metrics collection framework and the optimization of the deployment process, which leverages the Grafana instance provided by the monitoring MicroK8s add-on, underscores the commitment to performance and user experience. Additionally, the controller ensures seamless integration with hardware and multi-layer networks, providing a more intuitive management of XR services and OpenConfig-based routers.

The release further fortifies cybersecurity mechanisms with updated components for attack detection, inference, and mitigation. The SDN framework's capabilities are expanded to include horizontal scalability for centralized attack detection and mitigation, and the addition of a Cybersecurity Grafana dashboard for monitoring.

Looking ahead, TeraFlowSDN is poised to collaborate with OpenSlice, emphasizing the development of Network Slice as a Service and reinforcing its position as a reference implementation for the networking community. This advancement showcases the project's commitment to continuous improvement and innovation, setting a new standard for SDN controllers and orchestrators in the rapidly evolving network landscape.

To explore more about TeraFlowSDN and its developments, the source code for TeraFlowSDN is available for public download under an Apache2 license, encouraging further experimentation and innovation within the community.

TeraFlowSDN can control a transport network by leveraging its cloud-native SDN controller capabilities, designed to enhance the performance, scalability, and security of network services. Specifically, TeraFlowSDN offers several features that enable effective control over transport networks:

- Micro-Service Based Architecture: TeraFlowSDN utilizes a micro-service architecture, which allows for modular deployment and management of network functions. This approach enables network operators to control various aspects of the transport network independently and dynamically, ensuring more agile and flexible network management.
- Integration with NFV and MEC Frameworks: By integrating with current NFV and MEC frameworks, TeraFlowSDN can control the virtualization of network functions and edge computing services. This integration allows the transport network to efficiently handle the increased data traffic and the need for low-latency services that are characteristic of modern networking environments.
- Advanced Flow Management: TeraFlowSDN provides advanced flow management capabilities at the service layer, which is crucial for controlling the transport network. By managing the flow of data across the network, TeraFlowSDN can optimize the routing of traffic, reduce congestion, and improve overall network efficiency.
- Network Equipment Integration: At the infrastructure layer, TeraFlowSDN can control and integrate various network equipment, including traditional routers and switches as well as

more modern whiteboxes and P4 programmable devices. This level of control allows the transport network to be more responsive to changing network conditions and demands.

- Transport Network Slicing: The concept of network slicing is integral to 5G and beyond, and TeraFlowSDN can facilitate the creation and management of transport network slices. These slices can be used to segregate network traffic based on service requirements, ensuring that critical services receive the bandwidth and latency they need.
- Multi-Tenancy: With TeraFlowSDN, transport networks can support multiple tenants, each with their unique requirements and service level agreements. This multi-tenancy capability is essential for service providers looking to offer differentiated services to various customer segments.
- Cybersecurity Features: TeraFlowSDN includes ML-based security and PDL-based forensic evidence features for multi-tenancy, providing a secure environment for transport networks. This means that the network can not only control traffic but also monitor, detect, and respond to cyber threats in real-time.
- Interworking Across Beyond 5G Networks: TeraFlowSDN supports interworking capabilities across Beyond 5G networks, which means it can control and manage a transport network that is part of a broader, interconnected, and heterogeneous networking ecosystem, paving the way for future-proof scalability and adaptability.

In controlling a transport network, TeraFlowSDN acts as a robust orchestrator and controller, utilizing these features to ensure that the network can meet the evolving demands of users and applications while maintaining high levels of performance and security.

2.4.4 Security as a Service (SECaaS)

This section introduces the general framework adopted for providing Security as a Service (SECaaS).

The proposed architecture is based on the use of ETSI ZSM principles (ETSI GS ZSM 002), which aim to enable zero-touch automation of network and service management. The main principles adopted are:

- Multilevel Domain: This is the approach that defines the scope and granularity of the automation processes. It allows for different levels of abstraction and decomposition of the network and service functions, from end-to-end to domain specific.
- Integration fabric: This is the layer that connects the different domains and provides a unified view of the network and service resources. It enables interoperability, data exchange, and orchestration across domains.
- Closed loop: This is the mechanism that enables continuous monitoring, analysis, and optimization of the network and service performance. It allows for proactive and reactive actions to ensure service quality and efficiency.

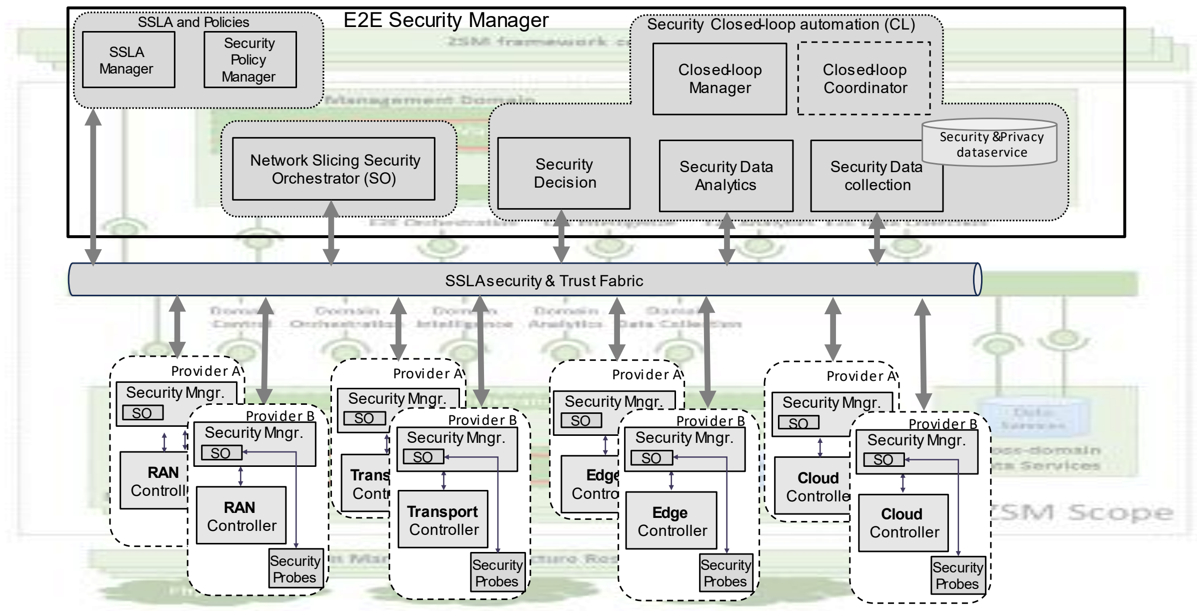


Figure 8 General framework for security as a service

The adopted framework is reflected in Figure 8, where there is natural mapping between the components identified and the general ZSM framework.

First the E2E Security Management will have an end-to-end vision to create the security network slices in a complex NoN, as well as the use of the use of closed-loop automation to address security requirements. Also, in each specific domain it will be deployed a Security Manager (SM) that will cover equivalent functions but at its own administrative domain. Only providers with this SM component will be considered to be used in the deployment of the SSLA.

The communication between components at E2E level or at Domain level is defined following the concept of a communication Fabric. This is done following the modern principles in 3GPP with SBA (Service Based Architecture) for 5G and the ETSI ZSM approach. Each component will expose their services capabilities for invocation, also each component will take care of request the use of the needed services of other components to accomplish their functions.

The main components and their main features are:

- SSLA and Policies: This component has a main role of processing the Operator Security SLAs, for example a template document with a list of SLAs related to security requirements. E2E SSLA & Policies will integrate a set of deterministic or advanced rules to translate the SSLA requirements into some high-level security policies, that usually are expressed in a High-level Security Policy Language (HSPL) or intent policies. These policies will be delivered to be applied later.
- Network Slicing Security Orchestrator: This component is the core of the service, interacting with all elements and the state of the processes and workflows, in the Security Manager. Its main role at E2E level is the coordination with the other components, the identification, selection, and coordination of the NoN domains to use in the security slice. At Domain level the Security Orchestrator will take care of the translation at domain level the high-level policies or capabilities defined in the policies in specific configuration that will be enforced by the domain network controller.
- Security Closed-loop Automation is responsible for managing and governing the lifecycle of closed-loop (CL) processes, which involve data collection, analytics and decision making based on predefined rules. Also, it coordinates the execution of CL actions and resolves any conflicts that may arise from them. In case of need security probes may be needed to collect security data.

All these components may exist at E2E or Domain level.

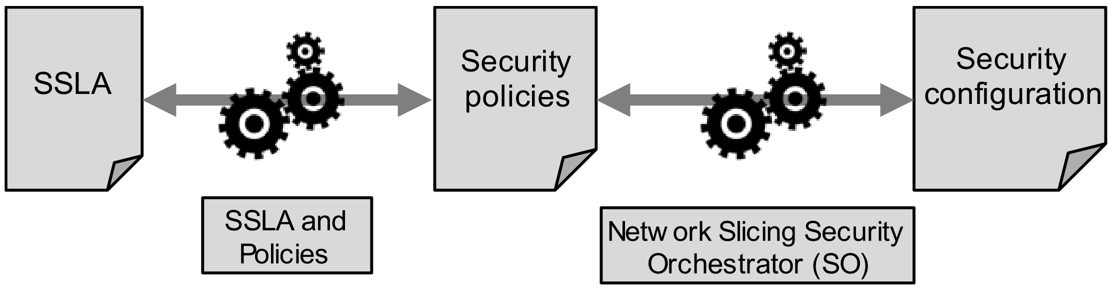


Figure 9 Information processing pipe

Figure 9 shows how the information provided in the SSLA is processed by the components. The SSLA is expanded in a set of high-level security policies, by the SSLA & Policies component. Later on, the Network Slicing Security Orchestrator, will split the policies into domain's specific ones in each network segment where the local SO through interactions with the network controller at each domain should translate these policies into specific configurations to enforce. Next are some examples:

Example of a SSLA list:

- SSLA1: "DDoS protection requested",
- SSLA2: "Traffic must be motorized to detect malware and blocked automatically".

Example of Trust policies translation of previous TSLA list:

- SSLA1 -> HSPL1: "Activate DDoS detection in the slice traffic " AND HSPL2: "Activate automatic mitigation of an DDoS attack".
- SSLA2 -> HSPL1: "Deploy an Intrusion Detection System (IDS) with rules related to malware" AND HSPL2 "Block traffic related to malware" AND "Isolate from the network compromised devices".

Example of the Trust Metrics translation for the first HSPL1:

- SSLA1-HSPL1: The Provider in Edge Domain activate an Anti-DDoS software, e.g., Netscout, PaloAlto, etc.
- SSLA1-HSPL2: The provider configures the service to automatically block suspicious traffic.

The security management process detailed in the sequence diagrams unfolds in three stages, delineating a complex workflow. The first stage centers on deploying security service slices, integrating E2E SO and SSLAP components, and incorporating trust elements. The second stage focuses on defining and enforcing closed-loop configurations for security within network slicing, involving a Closed-Loop Manager and a series of coordinated actions to maintain security integrity. The third stage reacts to security incidents, engaging closed-loop automation to address and mitigate attacks, and ensuring ongoing communication with the Operator for transparency and oversight. This structured approach ensures a comprehensive and responsive security management system.

In the first stage of service deployment, the Operator initiates the creation of a security service slice according to the specified SSLAs. This involves intricate interactions with the End-to-End Network Slicing Security Orchestrator (E2E SO) and the SSLA & Policies (SSLAP) component to translate the requirements into actionable policies across multiple providers' domains.

The second stage involves closed-loop configurations where the Closed-Loop Manager (CLM) works with the Closed-Loop Coordinator (CLC) to resolve any conflicts in the security policies and configurations. This ensures that the security conditions for the network service are defined, enforced, and any necessary security measures are ready to be activated automatically.

In the third stage, the system responds to security incidents by activating the closed-loop automation. The Security Data Collection (SDC) component gathers data on security threats, which the Security Data Analytics (SDA) component analyzes to identify potential attacks. Appropriate mitigation actions are then determined and executed, keeping the Operator informed throughout the process to maintain transparency and control.

Based on the architecture and the workflows previously described the following interface for the Network Slicing Security orchestrator was defined with the basic operations that this component must take care based on the capabilities defined and the interactions with the other components.

Service name	Network Slicing Security Orchestrator			
Service capabilities Exposed	**Description**	**Priority**	**External visibility**	**Mechanism (API REST)**
Secured Network Slice Service Deployment	Generates the requests to deploy the Secured Network Slice based on the incoming policies.	Mandatory	Optional	POST /opensecSecurity/v1/nsso/slice
Enforce security on network slice	Generates requests to enforce security updates on a Security Network Slice based on the decisions received.	Mandatory	Optional	PUT /opensecSecurity/v1/nsso/slice/<id_service>
Secured Network Slice Service Termination	Generates the requests to terminate the Secured Network Slice.	Mandatory	Optional	DELETE /opensecSecurity/v1/nsso/slice/<id_service>
Obtains instantiated Secured Network Slices	Retrieves the information of all Secured Network Slices.	Mandatory	Optional	GET /opensecSecurity/v1/nsso/slice
Obtains instantiated Secured Network Slices	Retrieves the information of a specific Secured Network Slice.	Mandatory	Optional	GET /opensecSecurity/v1/nsso/slice/<id_service>

2.5 Facilities for use case 2

2.5.1 ADRENALINE Testbed

The ADRENALINE testbed® is an open and disaggregated SDN/NFV-enabled packet/optical transport network and edge/cloud computing infrastructure for Beyond 5G, 6G and IoT/V2X services. It embraces several network segments such as access, metro, and core. The key elements include (1) SDN-enabled partially disaggregated optical network. It is composed of: i) 1 photonic (flexi-grid DWDM) mesh network (PMN) with 4 nodes (2 ROADMs & 2 OXCs) and 5 bidirectional DWDM amplified optical links up to 150 km (overall 600 km of optical fibre); ii) a Spatial Division Multiplexing (SDM) domain formed by 2 Spatial Cross Connect devices (SXCs) connected by a 19-core 25Km multi core fibre (MCF); iii) a pair of packet optical nodes with optical pluggable transponders providing aggregated 400G data rates for transporting traffic flows between the access and the core network segments; iv) heterogeneous access network technologies are connected to the metro infrastructure such as IP Cell Site Gateways (CSGs) equipped with Edge DC capacities, a Passive Optical Network (PON) tree formed by disaggregated Optical Network Terminals (ONTs) offering connectivity to several Customer Premises Equipment (CPEs), and a pool of (OpenFlow-based) packet switches domain deployed on COTS and using Open vSwitch (OvS) for the network connectivity needed by several Edge/Core DC nodes; v) optical metro nodes are also connected to programmable SDN-enabled Sliceable Bandwidth Variable Transponders (S-BVTs) to transmit multiple flows at variable data rate/reach up to 1 Tb/s.

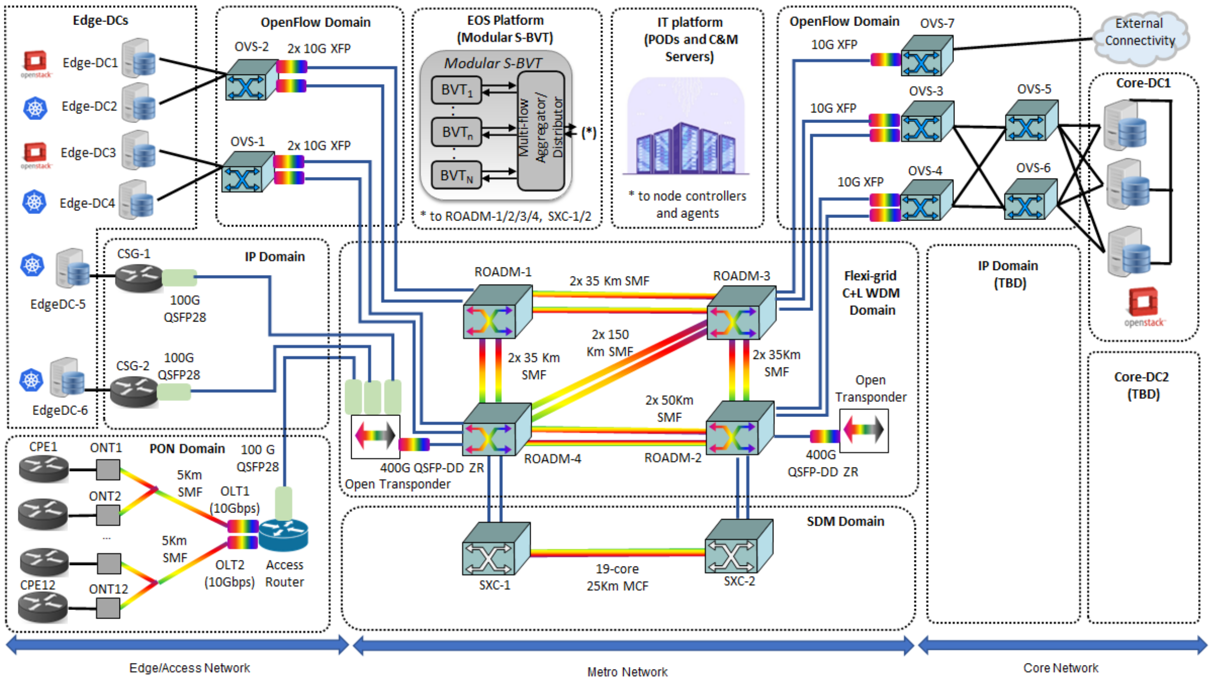


Figure 10: CTTC Adrenaline Testbed

The different domains access (i.e., PON, IP CSGs, OpenFlow) and metro (i.e., Flexi-grid DWDM and SDM) are managed by dedicated orchestrators/controllers (e.g., Optical Line System SDN controller or OpenDayLight) to automatically handle the connectivity services entailing the de-/allocation of heterogeneous network resources (i.e., packet and optical devices). Those domain-specific controllers and orchestrators are coordinated hierarchically by implementing the ETSI TeraFlowSDN controller. The TeraFlowSDN controller exposes a NorthBound Interface to allow an external system (e.g., NFV service platform) to request network connectivity services. This NFV service platform orchestrates the transport (optical/packet) and computing (edge/cloud) resources: i) Multi-VIM (virtualized infrastructure managers) combining OpenStack and K8s controllers for virtual machines and containers; ii) TeraFlowSDN controller for end-to-end connectivity among virtual machines, containers, and end-points. The NFV service platform is also in charge of managing the life-cycle of NFV network services and network slices: i) an NFV network service is composed of chained VNFs or cloud-native network functions (CNFs) deployed on VM and/or containers; ii) a network slice is composed of one or several concatenated NFV network services that deploy a set of VNFs and/or CNFs.

3 Over-the-air vehicular software updates with efficient computation

3.1 MEC Bandwidth Management

In the field of vehicular technology, the integration of Multi-Access Edge Computing (MEC) with advanced network control and management, such as TeraFlowSDN (TFS), presents a formidable solution for Over-the-air (OTA) Software Updates. MEC offers ultra-low latency and high bandwidth, along with real-time access to data and radio network information, which are crucial in swiftly and efficiently managing OTA updates.

The MEC BandWidth Management (BWM) service plays a pivotal role in allocating and adjusting bandwidth resources, including bandwidth size and priority, specifically tailored for MEC applications. This feature enables MEC applications to specify bandwidth requirements, essential for the smooth transmission of OTA updates.

This demonstration examines the synergistic relationship between the BWM service and TeraFlowSDN (TFS) in optimizing resource allocation for OTA software updates in vehicular networks. The BWM service empowers applications to designate specific bandwidth allocations and other quality of service constraints, such as latency, crucial for the timely and effective delivery of OTA updates. Concurrently, TFS provides sophisticated orchestration of traffic flow management and control, ensuring that OTA update traffic is given priority.

Exploring the benefits of MEC BWM and TFS within the context of OTA updates reveals several advantages. Improved update efficiency, reduced network congestion, and enhanced scalability are prominent among these. The prioritization of OTA update traffic, facilitated by BWM and TFS, is instrumental in reducing latency, thereby streamlining the update process. Allocating dedicated bandwidth to OTA updates also alleviates network congestion, ensuring a more efficient update process for all connected vehicles.

Moreover, the scalability offered by MEC BWM and TFS is critical in adapting to the rapidly evolving field of vehicular technology and the increasing volume of OTA updates. As vehicles become more connected and reliant on software, the need for robust solutions that can adapt to growing demands becomes paramount.

In summary, the integration of MEC BWM and TFS emerges as a significant enabler for network operators aiming to deliver seamless and efficient OTA software updates to vehicles. This demonstration unveils the dynamics and potential of this integration, highlighting its capacity to transform the landscape of vehicular network management.

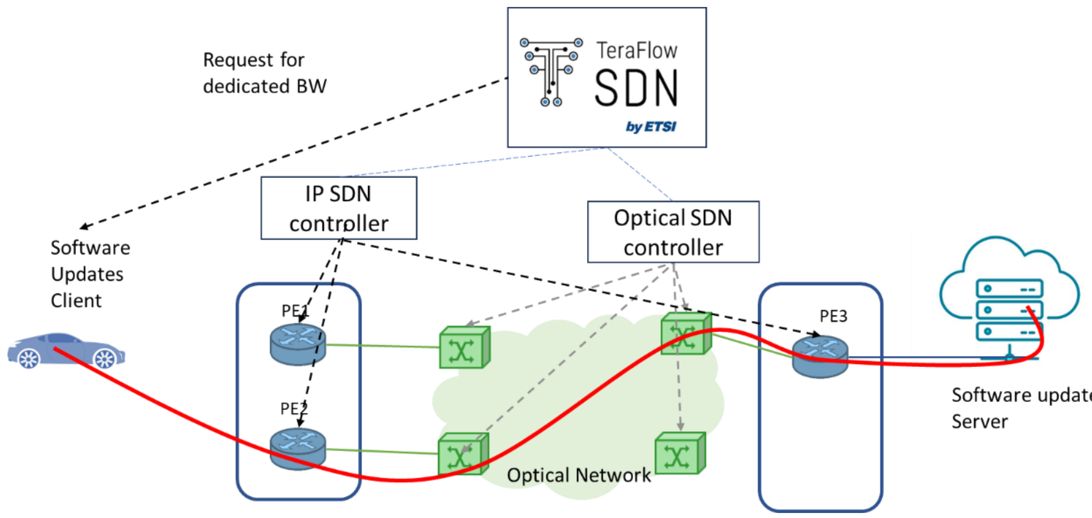


Figure 11 Proposed architecture for over-the-air vehicular software updates with efficient computation

Figure 11 depicts an advanced vehicular network architecture within a TeraFlow SDN framework. A vehicle is shown initiating a network connection, which is routed through a series of networking switches, symbolized by the 'X' icons. These switches are connected to optical network components—marked by green boxes—that handle the light-wave-based data transmission. The network's optical backbone is essential for high-speed data transfer and is indicative of an underlying high-capacity communication infrastructure. The data flow, represented by the red line, traces the vehicle's connection through the optical network to a cloud computing environment, where data is processed, stored, or further acted upon. This cloud is depicted as a cluster of servers within a cloud icon, denoting the end-point of data transfer and the location of service provision. The solid lines typically represent tangible connections, while dashed lines may represent wireless links or other forms of indirect connectivity. The entire setup is indicative of a modern, software-defined approach to network management, optimizing data paths dynamically for efficiency and performance.

3.2 Workflow

Figure 12 offers a sophisticated and scholarly representation of the orchestrated network operations within an End-to-End (E2E) Software-Defined Network (SDN) environment. This diagram serves not merely as a visual aid but as an academic illustration of the intricate interplay among various controllers, infrastructure components, and end-user interactions. The processes depicted in this figure are emblematic of the advanced capabilities and integrations within modern network architectures.

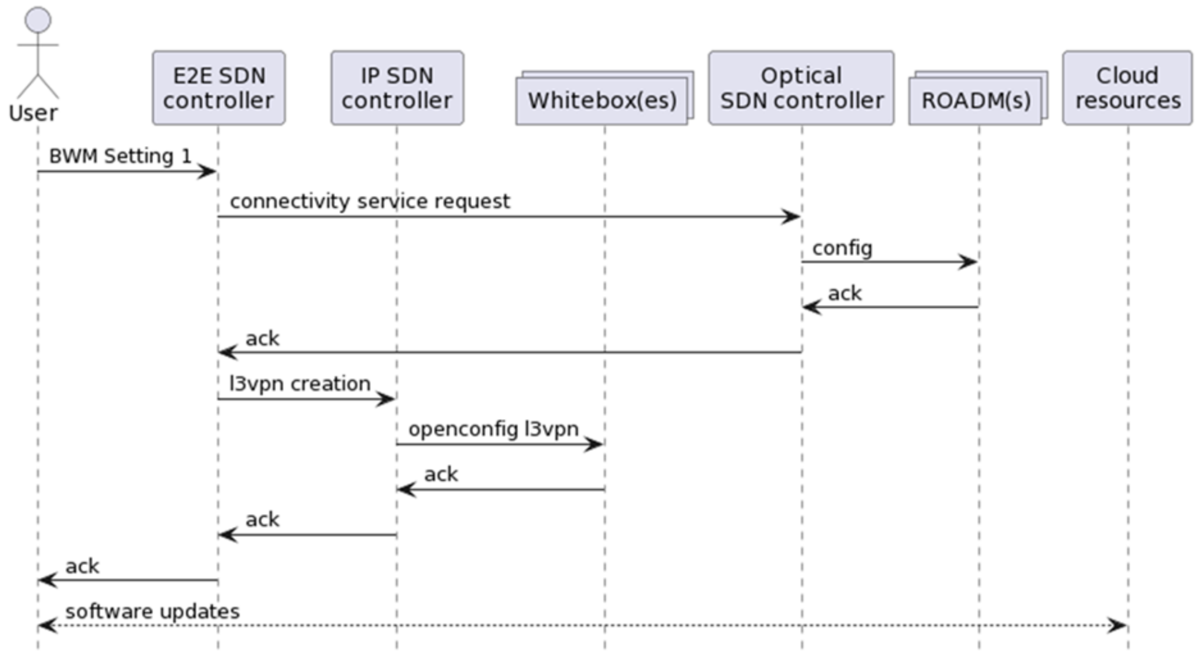


Figure 12 Sequence diagram for over-the-air vehicular software updates with efficient computation

3.2.1 User Interaction and Bandwidth Management

The diagram's narrative commences with the user, denoted as "edge1", initiating a bandwidth setting, referred to as BWM Setting 1. This action triggers a critical communication sequence with the E2E SDN controller, a pivotal entity within the network framework. The E2E SDN controller, embodying advanced computational capabilities, processes this request, engaging in a nuanced dialogue with the Optical SDN controller. This interaction is vital for the fulfillment of a connectivity service request, an operation that is foundational to network performance and efficiency.

The request encompasses the configuration of pivotal optical network infrastructure components, namely Whiteboxes and Reconfigurable Optical Add-Drop Multiplexers (ROADMs). These components play an instrumental role in the optical networking realm, facilitating efficient data transmission and network flexibility. The sequence diagram meticulously illustrates the exchange of acknowledgments between the Optical SDN controller and the ROADMs, a process that guarantees the successful configuration and establishment of connectivity. Subsequently, the Optical SDN controller communicates the acknowledgment back to the E2E SDN controller, completing this segment of the orchestration.

3.2.2 Integration with IP SDN Controller

In parallel, a sophisticated integration process unfolds involving the E2E SDN controller and the IP SDN controller. The E2E SDN controller, in this context, initiates the creation of a Layer 3 Virtual Private Network (L3VPN). This is achieved through a communicative interface with the IP SDN controller. The IP SDN controller, operating under the OpenConfig standard, interacts with Whiteboxes to configure the L3VPN. This interaction highlights the seamless integration of different network layers and the versatility of the controllers in managing diverse network functions.

The diagram further delineates the exchange of acknowledgments between the Whiteboxes and the IP SDN controller, an essential step in confirming the successful execution of the L3VPN creation. This successful interaction is communicated back to the E2E SDN controller, which, in an emblematic demonstration of the integrated network architecture, acknowledges the User's initial request.

3.2.3 Cloud Resource Interaction and Software Updates

The concluding segment of the sequence diagram emphasizes the dynamic interaction between the User and Cloud resources. This interaction is particularly significant in the context of software maintenance and updates. The User engages in a bidirectional dialogue with the Cloud, reflecting the ongoing need for software updates and the dynamic nature of cloud-based resource management. This interaction underscores the importance of cloud resources in contemporary network environments, particularly in the realm of software deployment and updates.

3.2.4 Summary

In summation, Figure 12 presents a detailed and academically rigorous overview of the orchestrated interactions and processes within an E2E SDN environment. It intricately outlines the mechanisms of bandwidth management, connectivity service provisioning, L3VPN creation, and software updates. The figure stands as a testament to the complexities and integrations inherent in modern network architectures, seamlessly weaving together diverse controllers, infrastructure components, and end-user interactions in a harmonious and systematic fashion.

4 Conclusions

This deliverable provides the description of the architectural framework adopted in SUCCESS-6G-VERIFY for the computationally efficient process of vehicular software updates in an over-the-air manner. Besides detailing the innovations applicable to use case 2, the document elaborates on the synergistic relationship between the bandwidth management service and the overall network architecture in optimizing resource allocation for over-the-air software updates. In addition, the workflow defining the sequence of commands that facilitate bandwidth management and integration with the software-defined-network controller is explained.

[end of document]