Grant Agreement No.: 101015857
Research and Innovation action
Call Topic: ICT-52-2020: 5G PPP - Smart Connectivity beyond 5G

Secured autonomic traffic management for a Tera of SDN flows

D2.1 - Preliminary requirements, architecture design, techno-economic studies and data models

Deliverable type	R (Document, report)
Dissemination level	PU (Public)
Due date	31/12/2021
Submission date	31/12/2021
Lead editor	Ricard Vilalta (CTTC)
Authors	Victor Lopez, Cristian Pilco, Oscar González, Antonio Pastor, Juan-Pedro Fernández-Palacios (TID), Ricard Vilalta, Ricardo Martínez, Lluís Gifre, Michela Svaluto-Moreolo, Francisco Vázquez, Javier Vilchez (CTTC), Carlos Natalino (CHAL), Hanne Kristine Hallingby, Min Xie, Håkon Lønsethagen (TNOR), Esther Garrido, Sergio González, Javier Moreno (ATOS), Alberto Mozo, Stanislav Vakaruk, Luis de Marcos (UPM), Luis Mata (UPM), Rahul Bobba (NEC), Georgios P. Katsikas (UBITECH), Sébastien Merle (STR), Stanislav Lange (NTNU), Sami Petteri Valiviita (INF), Nicola Carapellese (SIAE), Daniel King (ODC)
Reviewers	Sébastien Merle (STR), Ramon Casellas (CTTC)
Quality Check team	Adrian Farrel (ODC) and Ricard Vilalta (CTTC)
Work package	WP2

Abstract

This document provides the first release of the proposed use cases, requirements, architecture design, business models and data models for the TeraFlow OS. It is aligned with the features and data models that will be present in TeraFlow OS release 1.

[End of abstract]

Revision History

Revision	**Date**	**Responsible**	**Comment**
0.1	01.09.2021	Editor	Initial structure of document
0.2	31.09.2021	Editor	Input content from MS2.1
0.3	31.10.2021	All	Updated content on use cases and requirements
0.4	31.11.2021	All	Updated Architecture and Data Models
0.5	31.11.2021	TNOR, ATOS	Updated Section 4
0.9	22.12.2021	Reviewers	Reviewed version
1.0	31.12.2021	Quality check	Version to be submitted

Disclaimer

Acknowledgment

The research conducted by TeraFlow receives funding from the European Commission H2020 programme under Grant Agreement No 101015857. The European Commission has no responsibility for the content of this document.

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

EXECUTIVE SUMMARY

This deliverable includes the methodology to be applied in TeraFlow, initial use case definitions, requirement elicitation, the draft architecture to be used in WP3 and WP4 for the release of TeraFlow OS v1. Furthermore, data models for the different components and interfaces are also detailed, use case-specific workflows are described, and next steps and conclusions are provided.

In the first section, the methodology is presented. It includes the agile approach proposed for the development of TeraFlow, which focuses on use case definition and requirement elicitation. TeraFlow will produce three releases of the software and an uptake of Continuous Integration/Continuous Deployment (CI/CD) principles.

Then we present the proposed use cases, which will help identify the specific requirements. They are classified based on the following topics: inventory, topology, service, transport network slicing, monitoring, traffic engineering, automation, policy enforcement, Machine Learning (ML)-based security, distributed ledger and smart contracts, compute integration and inter-domain. These use cases provide a starting point for drafting the behaviour of the TeraFlow Software Defined Networking (SDN) controller. Not all use cases will be demonstrated, but specific ones will be provided within each of the different proposed scenarios.

Business model analysis for the TeraFlow SDN controller is also essential, as we need to design a valid business model for the controller. An initial draft of the value proposition is being explored, and we present the proposed methodology.

After that, the TeraFlow functional and non-functional requirements are presented. They serve as the basis for the TeraFlow SDN controller Year 1 release. The functional requirements are classified into components, and the non-functional requirements include performance, usability, scalability, reliability, and portability.

The proposed architecture based on a micro-service architecture is presented. Each component is detailed through a template that describes the main functionality, the proposed operations for the component, and the suggested internal data models. A protocol buffer is described for each component to model the services.

Later, data models are described, including external interfaces (northbound and southbound) and internal data models. Finally, sequence diagrams are presented to depict micro-service interactions.

In conclusion, we present the planned next steps for TeraFlow.

Table of contents

List of Figures

List of Tables

Abbreviations

5G-PPP	5G Infrastructure Public Private Partnership
API	Application Programming Interface
AI	Artificial Intelligence
B5G	Beyond 5G
BGP	Border Gateway Protocol
CCAM	Cooperative, Connected, and Automated Mobility
CD	Continuous Delivery
CE	Customer Edge
CI	Continuous Integration
DC	Data Centre
DDoS	Distributed Denial of Service
DLT	Distributed Ledger Technologies
DWDM	Dense Wavelength Division Multiplexing
E2E	End-to-End
E-NNI	External Network-to-Network Interface
GMPLS	Generalized Multi-Protocol Label Switching
GUI	Graphical User Interface
IETF	Internet Engineering Task Force
IP	Internet Protocol
KPI	Key Performance Indicators
L2	Layer 2
L2VPN	Layer 2 Virtual Private Network
L3	Layer 3
L3NM	Layer 3 Network Model
L3VPN	Layer 3 Virtual Private Networks
LSP	Label-Switched Path
MANO	Management and Orchestration
MEC	Multi-access Edge Computing
ML	Machine Learning

MPLS	Multi-Protocol Label Switching
MQP	Managed Quality Path
NBI	North-Bound Interface
NFV	Network Functions Virtualization
NFVI	NFV Infrastructure
NFVI-PoP	NFVI Point of Presence
NOS	Network Operating System
NSP	Network Services Platform
OAP	Online Application service Provider
ONF	Open Networking Foundation
OPM	Optical Performance Monitoring
OS	Operating System
OSM	Open Source MANO
OSS/BSS	Operation Support System/Business Support System
OTT	Over The Top
PE	Provider Edge
POI	Point of Interconnection
PCE	Path Computation Element
PCEP	Path Computation Element Protocol
PoP	Point of Presence
RSVP	Resource Reservation Protocol
SBI	South-Bound Interface
SCS	Specialized Connectivity Services
SDN	Software-Defined Networking
SDO	Standards Defining Organization
SLA	Service Level Agreement
SLO	Service Level Objective
SME	Small and Medium Enterprises
SR	Segment Routing
TDM	Time Division Multiplexing

TE	Traffic Engineering
TED	Traffic Engineering Database
TRL	Technology-Readiness Level
TSN	Time Sensitive Networking
UNI	User-to-Network Interface
VLAN	Virtual Local Area Network
VM	Virtual Machine
VRF	Virtual Routing and Forwarding
VSI	Virtual Server Instance
WAN	Wide Area Network
WDM	Wavelength Division Multiplexing
WIM	WAN Infrastructure Manager
ZTP	Zero-Touch Provisioning

1. Introduction

The TeraFlow SDN controller is a new type of secure cloud-native SDN controller that will radically advance the state-of-the-art in B5G networks. This new SDN controller will be able to integrate with the current NFV and MEC frameworks, provide revolutionary features for flow management (service layer), and provide optical/microwave network equipment integration (infrastructure layer). These capabilities will also incorporate enhanced security using ML (Machine Learning) and forensic evidence for multi-tenancy based on DLT (Distributed Ledger Technology).

The focus of the TeraFlow project is to develop a Carrier Grade SDN controller for B5G networks. "Carrier Grade" refers to well-tested networks or infrastructures with extremely high levels of reliability, redundancy, and security. This objective will be demonstrated in relevant environments with solutions at TRL 5.

Use cases for IP, optical, and microwave networks will be studied and demonstrated in commercial and open-source solutions based on standard interfaces. The covered network domains for the proposed solution are transport scenarios integrated with (edge) computing and storage resources. TeraFlow will demonstrate its dynamic adaptation based on flows and applications requirements. TeraFlow covers a wide variety of network types, ranging from distributed edge-computing, through a transport backhaul (including optical and microwave solutions), to the network core. In addition, TeraFlow provides carrier-grade connectivity services for B5G networks.

This deliverable, based on MS2.1, is a crucial document for the TeraFlow architecture definition. Additionally, it includes the detailed design specifications for the TeraFlow SDN controller.

2. Methodology

Figure 1 shows the proposed use of case-based methodology. It is an iterative approach to deliver a release of TeraFlow. First, the value of the proposed use cases is analysed from an operations perspective. This results in requirements that will be included in open technical specifications introduced to SDOs or as contributions to Open-Source Software. Then, an implementation is provided, which can be adopted by network operators for immediate use.

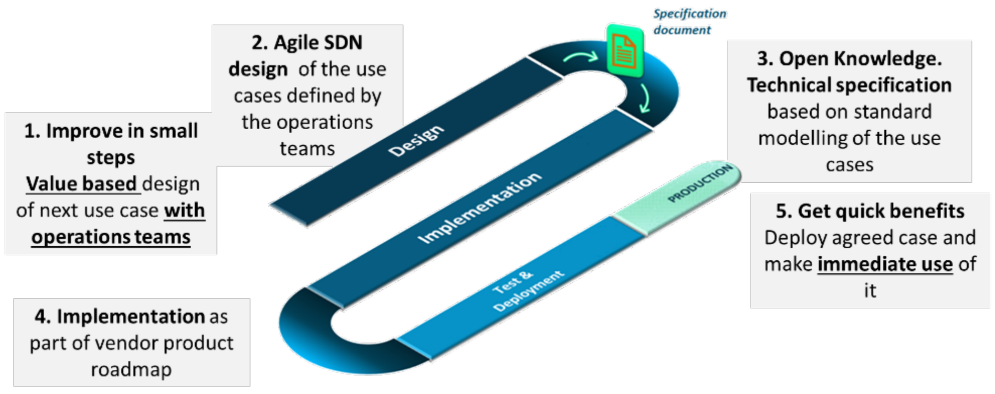

Specification document
2. Agile SDN design of the use cases defined by the operations teams
1. Improve in small steps Value based design of next use case with operations teams
3. Open Knowledge. Technical specification based on standard modelling of the use cases
Design
Implementation
PRODUCTION
Test & Deployment
5. Get quick benefits Deploy agreed case and make immediate use of it
4. Implementation as part of vendor product roadmap

Figure 1 Proposed methodology

The main drivers and strengths for such a methodology are:

a) It accelerates implementation and industrial production of standard interfaces for network applications and devices;
b) It opens the market at both the network application and device layers. Currently, the lack of standard interfaces and the complexity of the software developments in multivendor networks is creating a considerable barrier of entrance for new companies and SMEs;
c) It simplifies the network and service modelling process, making it affordable for a network operator; and
d) It focuses the development on the key functionalities for network operators.

TeraFlow will use this agile methodology to produce three major software releases: v1 will be released at M12, v2 will be released at M24, and v2.1 will be released at M30. These releases will cover the length of the technical work packages (WP3 and WP4). The primary software releases will deliver the main functionality, while the minor release will provide bug fixes and possibly additional features required by the operating experiments. To support software development and the ability to make releases with minimal impact on the running services, TeraFlow will adopt a CI/ CD strategy.

3. Use Cases

This section presents the use cases proposed for the TeraFlow SDN controller. They are classified into the following topics: inventory, topology, service, transport network slicing, monitoring, traffic engineering, automation, policy enforcement, ML-based security, distributed ledger and smart contracts, compute integration, and inter-domain.

These use cases refer to multiple network technologies covering IP, Optical, and Microwave. Optical and Microwave are deployed as transport networks, while IP is intended to provision L3VPN services.

3.1. Inventory

This set of uses cases provides means to recover information and state about hardware components of network elements and the logical configuration of the devices intended to be performed by any NBI client controller, module, or application that aims to discover the component hierarchy of the equipment. Use cases for optical and IP domains concerning Inventory are further described in [TR547], [LIU21].

3.1.1. Hardware Inventory Collection

Technologies involved	**IP, Optical, Microwave**
Type	Inventory
Description	This use case consists of retrieving all hardware (physical) information about the equipment available from the TeraFlow SDN Controller. These components might be line cards, transceivers, ports, etc.

3.1.2. Logical Inventory Collection

Technologies involved	**IP, Optical, Microwave**
Type	Inventory
Description	This use case consists of retrieving all logical configuration information about the equipment available from the TeraFlow SDN Controller. Logical inventory refers to layer 3 and layer 2. There are three modes for this use case: • **Retrieve logical interfaces inventory** - focus on retrieving all the logical or virtual interfaces of specific equipment, such as sub-interfaces, VLAN interfaces, tunnel interfaces, and other non-physical interfaces. • **Retrieve logical resources inventory** - focus on retrieving all the logical or virtual resources of specific equipment such as system policies for routing, access, logging, security, etc. • **Retrieve logical protocols inventory -** focus on retrieving the information regarding all the logical protocols such as Border Gateway Protocol (BGP), Multiprotocol Label Switching (MPLS), Resource Reservation Protocol (RSVP).

3.1.3. Host tracking

Technologies involved	**IP**
Type	Inventory
Description	This use case retrieves all detected hosts (both at IP and MAC layers) and the latest detection time. This will allow identification of possible required flows and implementation of other use cases such as E2E service provisioning.

3.2. Topology

A set of abstractions has been defined to represent several views of the network topology. Use cases for optical and IP domains concerning Context and Topology discovery are further described in [TR547], [LIU21]. There are two modes of operations:

- Polling mode - based on periodic polling retrieval operations and after each service creation to reconcile the actual state of the network.
- Event-triggered mode - based on an initial proactive synchronization done from the NBI client module and a connection-oriented notification subscription session based on the NBI Notification mechanism.

In the Internet layering reference model, the communications between a computing system are split into seven different abstraction layers: Physical, Data Link, Network, Transport, Session, Presentation, and Application. In the following sections, we use layer 0 to refer to transport using optical and microwave networks and layer 3 for IP packets. Multi-layer topologies include multiple layer transport mechanisms, typically IP over optical/microwave networks.

From the SDN controller perspective, a network domain includes the network elements that the SDN controller manages and may be specific areas of technology, vendors or administrative regions. The multi-domain topology includes the combined view of multiple SDN controlled domains.

3.2.1. Context & Service Endpoints Discovery

Technologies involved	**Optical, IP, Microwave**
Type	Topology
Description	This use case retrieves all Service Endpoint [TR547] information available from the TeraFlow SDN Controller. It is intended to be performed by any NBI client controller, module, or application to discover the logical representation of the network done by the TeraFlow SDN Controller.

3.2.2. Single Layer Topology

Technologies involved	**Optical, IP, Microwave**
Type	Topology

Description	This use case retrieves all topological information available from the TeraFlow SDN Controller. It is intended to be performed by any NBI client controller, module, or application that aims to discover the logical representation of the network. There are two modes of operations foreseen for this use case: • **Polling mode** - based on periodic retrieval operations and after each service creation to reconcile the actual state of the network. • **Event-triggered mode -** based on an initial proactive synchronization done from the NBI client module, and a connection-oriented notification subscription session based on the NBI Notification mechanism.

3.2.3. Multi-Layer Topology

Technologies involved	**IP, Optical, and Microwave**
Type	Topology
Description	This use case consists of retrieving all topological information available from the TeraFlow SDN Controller to carry out multi-layer mapping.

3.2.4. Multi-Domain Service Endpoint and Topology

Technologies involved	**Optical, IP, Microwave**
Type	Topology
Description	This use case intends to define how the TeraFlow SDN Controller exposes the unified multi-domain topology, including its service mapping list and topological information. The discovery of this information is intended to be requested proactively from a TeraFlow SDN Controller to synchronize the information, which must be updated when the OSS requests it. UNIs and E-NNIs must be exposed as new service ports in the topological information.

3.3. Service

Two different kinds of service are considered:

- Layer 2/3 services: L2/L3VPN services
- Layer 0: Optical or Microwave services

In the L3VPN for 5G Services, the nodeB are directly connected to the cell site (HL5 layer). The cell site acts as a first aggregation layer, for nodes that share the same geographical location. The connections between the next Access Router layer (HL3) and cell site layer are made in a ring topology. The HL3 receives and aggregates traffic from the rings of the same state (geographical location). In the HL5 an L3VPN is created to receive the interfaces of each nodeB [MUS21]. L3VPNs are used to deploy 5G but fixed and enterprise services mainly because several traffic discrimination policies can be applied in the network to transport and guarantee the right SLAs to the mobile customers. They have been typically statically configured, and with the adoption of modern protocols, they can be dynamically configured on IP routers. IP routers are interconnected using underlying network elements, such as DWDM or Microwave transport networks. To provide E2E connectivity services, underlying network connections shall be controlled and managed.

3.3.1. Service Lifecycle

Technologies involved	**IP, Optical, Microwave**
Type	Service Lifecycle
Description	This use case is used to handle service lifecycle in a single domain. Two different kinds of service are considered: - L2/L3VPN services - Optical or Microwave services All service types shall be able to support multiple configuration parameters, as well as to include multiple constraints (e.g., traffic engineering).

3.3.2. End-to-End Service Lifecycle

Technologies involved	**IP**
Type	Service Lifecycle
Description	This use case consists of creating, modifying, and removing L2/L3VPNs without the need for manual configuration. It also allows the creation of E2E underlying connections without changes in the configuration during service provisioning.

3.4. Transport Network Slicing

For end-business customers (like OTT companies or enterprises), leased lines have the advantage of providing high-speed connections with low costs. On the other hand, traffic control for leased lines is very challenging in rapid changes in service demands. Carriers are recommended to provide network-level slicing capabilities to meet this demand. Based on such capabilities, private network users have complete control over the sliced resources allocated to them and could be used to support their leased lines, when needed. A transport network slice will consist of a set of endpoints (e.g., CEs), a connectivity matrix between subsets of these endpoints service level behaviours requested for each sender on the connectivity matrix [FAR21]. The connectivity between the endpoints might be point-to-point, point-to-multipoint, or multipoint-to-multipoint. Often these slices will be used to satisfy network behaviour defined in a Service Level Agreement (SLA) [FAR21].

3.4.1. Slice Life-Cycle Management

Technologies involved	**Optical, IP, Microwave**
Type	Transport Network Slicing

Description	This use case includes the transport network slice life-cycle management. Users may formulate transport network slices based on the demand for services and the time needed to schedule the resources from the entire network flexibly. Several services might be offered for a transport network slice, including L3/L2 VPN or VLAN constraints.

3.4.2. Vertical Dedicated Network

Technologies involved	**Optical, IP, Microwave**
Type	Transport Network Slicing
Description	Vertical industry slicing is an emerging category of network slicing due to the high demand for private high-speed network interconnects for industrial applications. The biggest challenge is implementing differentiated transport network slices based on the requirements from different industries. For example, in the financial industry, to support high-frequency transactions, the slice must provide the minimum latency and latency management mechanism. For the healthcare industry, reliability mechanisms are needed to ensure the delivery of HD video without frame loss. In addition, the network needs to support on-demand, large-bandwidth allocations for bulk data migration in data centres. In each vertical industry scenario, the bandwidth needs to be adjusted as required to ensure flexible and efficient network resource usage. This use case shall consider two different types of slices: Ultra-Reliable Low Latency Communications (URLLC) and Enhanced Mobile Broadband (eMBB).

3.4.3. Slice Isolation

Technologies involved	**Optical, IP, Microwave**
Type	Transport Network Slicing
Description	Customers require the network slice to be delivered isolated from any other network slices delivered to any other consumers. In other words, any changes (to network loads or events like congestions or outages) to the other network slices do not negatively impact the delivery of this network slice. Consumers often specify isolation as an SLO and may be associated with other SLOs, e.g., latency under traffic congestion. Isolation can be achieved by resource partitioning and/or robustness techniques, e.g., dedicated resources, shared resources with safeguards, or reserved backup paths. Examples include traffic separation via VPNs (L2/L3VPN, EVPN), interference avoidance via network capacity planning, traffic policing or shaping, prioritization in resource utilization, etc. In terms of security, each slice may have additional security SLOs, which should be executed in an isolated way. On the other hand, isolation is a means to strengthen security because it protects one slice from attacks that vary the traffic on other services or slices on the same underlay network. Complete resource isolation can be achieved by provisioning dedicated fibres, which is feasible, but very expensive. Therefore, physical splitting (e.g., in time or frequency) can be used. For instance, in optical networks, full lambdas can be isolated using Wavelength Division Multiplexing (WDM) or Time Division Multiplexing (TDM) techniques by assigning specific time slots to specific slices. The previous techniques are known as hard isolation. At the opposite end of the spectrum, soft isolation solutions rely on the simple separation of traffic delivery, such as simple MPLS or VLAN tagging. These mechanisms offer separation, but not isolation performance guarantees. The design of intermediate solutions between hard and soft isolation may be classified into two classes: i.) Link layer (Layer 1.5 / Layer 2) technologies such as Flex Ethernet (FlexE), dedicated queuing, or TSN;

	ii.) Network layer technologies such as MPLS-TE, Deterministic Networking (DetNet), Segment Routing (SR). This use case requires the provisioning of different isolation levels for Transport Network Slices, being hard or soft isolation the minimum set required.

3.5. Monitoring

This use case describes the necessary information (including alarms and events) collected directly from network elements through multiple protocols, such as NETCONF/YANG or by streaming telemetry (gRPC) feeding the event manager. The received data shall be stored, analysed, and retrieved to perform additional closed-loop automation and big data analysis.

3.5.1. Events Collection and Intelligent Grouping

Technologies involved	**IP**
Type	Fault Management
Description	This use case consists of sending alarms and relevant information related to the alarms. It shall also be possible to define operator-defined threshold-crossing alerts (TCAs).

3.6. Traffic Engineering

Traffic engineering (TE) allows the enforcement of traffic steering flows by leveraging MPLS tunnels or Segment Routing paths. This allows increasing the efficiency of using the network resources by correctly mapping the traffic flows to the available resources and improving network management, identifying issues and reacting to overcome difficult failure situations. Furthermore, the Path Computation Element (PCE) function has been defined to allow the performance of complex constrained based path computation over a network graph representation. This improves the application of TE policies in G/MPLS networks. Based on these functionalities, traffic engineering's primary purpose is to reduce overall operating costs through more efficient network resource usage, including link occupation, traffic rerouting, and network availability [MUS21].

3.6.1. PCEP for Segment Routing

Technologies involved	**IP**
Type	PCEP for Segment Routing (LSP Creation, modify and delete with SR)
Description	This use case consists of creating, modifying, and deleting segment routing LSPs on the available hardware, considering specific constraints and the available resources. For example, the constraints given to the PCE for the calculation of the LSP could be required latency, bandwidth consumption and hop count, and if the result should be a strict explicit path or a loose one.

3.7. Automation

Automation greatly facilitates several key networking operations. First, automated network management procedures include dynamic network resource discovery (i.e., topology discovery) and network configuration via modern open standards (e.g., gRPC, P4, NETCONF/YANG). Secondly, automated service provisioning enables programmatic creation and management of network services at a high-level. At the same time, a low-level configuration is populated to the underlying network elements through their association with network flows and automatically-verified policies.

Moreover, automation transforms manual network operations through a software-defined NOS, which manages the device lifecycle via programmable control loops. Some related NOS tasks include (i) automated NOS deployment and basic configuration as a response to newly added devices in the network; (ii) remote NOS upgrade when a new version is available; (iii) rollback recovery upon, e.g., version instability issues; and (iv) NOS migration and white box configuration across different vendors. Popular NOSes include, but are not limited to, Stratum, Cumulus, Open Network Linux (ONL), DENT, SONiC, OcNOS, Beluganos.

Finally, automation enhances system stability by recovering flow state in case of, e.g., a race condition and by detecting anomalous behaviours or attacks, while taking appropriate remedial actions.

3.7.1. Service Restoration

Technologies involved	**IP**
Type	Automation
Description	In case of connectivity service failure, service paths automatically reroute themselves whenever a change occurs in the routing table or in the status of a node or link.

3.7.2. Service Optimization

Technologies involved	**IP**
Type	Automation
Description	This use case facilitates the dynamic network operation and optimizes the network occupancy. This use case motivates the automatic administration, and thus modification, of Connectivity Services using a centralized network controller.

3.7.3. Protected Services

Technologies involved	**IP**
Type	Automation
Description	This use case consists of creating preconfigured protected connectivity services over dedicated resources that are deployed at the very same time.

3.7.4. Zero Touch Provisioning

Technologies involved	**Optical, IP**
Type	NOS life-cycle management
Description	ZTP is the process of deploying an NOS and a base configuration in a network element (a router or a transponder) so that the network element can enter in production without any human configuration. The ZTP process is done for the first time when the device is turned on in the network. Periodically, system vendors release new versions of their NOS. A very similar ZTP process can be done for this upgrade scenario. Nowadays, the NOS upgrade process is a vendor-dependent process, and the procedure differs between vendor solution. The ZTP and upgrade process presented in this part is a first step towards having a common procedure for open white box scenarios.

3.7.5. NOS Upgrade

Technologies involved	**Optical, IP**
Type	NOS Upgrade
Description	NOS developers continually add new functionalities and reduce the possible bugs detected in the software that the network elements run. These software artifacts can have several releases during a year; this demands the network operators are continually moving the devices-software to the latest stable version. The process to automatically move from one software release to another is what we define as the NOS Upgrade process.

3.8. Policy Enforcement

Network operators may express policy rules to be automatically enforced in the network by the TeraFlow OS.

Technologies involved	**Optical, IP**
Type	Policy
Description	A network operator may define a policy rule and associate it with an existing network service. The TeraFlow OS policy component will ensure that the correct network configuration will be automatically applied to the correct devices. This will be achieved through the interaction among the policy, context, and device components of the TeraFlow OS.

3.9. ML-Based Security

Security is vital in the TeraFlow OS as network operations will be done by software components virtually operating without human intervention. These three use cases exemplify the detection of different types of attacks that can appear in the control and data plane.

3.9.1. Detection At Edge Nodes

Technologies involved	**IP**
Type	ML-based security
Problem	Detection and identification of malware network flows crossing the data plane cannot be done in an ML-based central component due to scalability problems and slow response times. It would be desirable to implement a distributed solution in which ML components are deployed at edge nodes.
Description	This use case addresses the monitoring and detection of malicious network flows crossing the data plane by placing ML components at the edge of network to achieve scalability and increase the speed response during the detection process. To this end, a feature extractor is deployed at the edge of the network to collect and summarise packets. The flow statistics aggregated by the feature extractor are sent to an ML classifier. Based on the real-time identification of malicious flows, the ML model will report insights to the TeraFlow SDN controller to perform security assessments. A crucial task is to determine whether the feature extraction and aggregation processes can be done much more efficiently at the kernel level using high performance techniques (for example, extended-bpf technology) than doing it at the application level using standard OS interfaces.

3.9.2. Detection At The Cybersecurity Component

Technologies involved	**Optical, IP**
Type	ML-based security
Problem	Attack detection and identification based on monitoring data. In ML, attack detection can be realized by the anomaly detection task. Attack identification, in ML, can be realized by the classification task.
Description	This use case consists of continuously assessing the data plane security status of the network across optical and IP layers. First, layer-specific attack detection and identification models perform the monitoring data from optical and IP layers independently. Then, a security assessment is performed considering both layers' monitoring data and security status. Finally, if relevant, the security assessment performed by the edge nodes can also be considered. There are two tasks for this use case: • Attack detection – consists of determining whether an attack is being launched in the network or not. • Attack identification – consists of identifying, among the previously categorized attack types, which type is the currently detected attack; there is also the possibility that the attack cannot be categorized among the existing attack types, in which case the attack can be identified as unknown and later categorized by a specialist.

3.9.3. Adversarial Attacks to Resilient ML

Technologies involved	**IP**

Type	Security of the control plane.
Problem	ML components are usually deployed on top of SDN controllers to help them to identify complex patterns. However, new attacks can introduce small perturbations in the input that humans cannot easily detect and fool ML inferences.
Description	This use case addresses the design of ML algorithms that are deployed on top of the SDN controller and react with resiliency to sophisticated adversarial AI attacks. In particular, the set of ML components included in TeraFlow's netApp ecosystem will be able to protect themselves against the recently appeared adversarial attacks that try to fool ML algorithms. Resilience to adversarial attacks is added to ML algorithms by using existing open-source tools (e.g. Cleverhans and Foolbox). Only the models obtaining high accuracy against the corresponding attacks included in the open-source libraries will be selected to be part of the TeraFlow's netApp ecosystem. Although adversarial attacks are still in their infancy, this use case will consider several well-known sources of attacks such as adversarial perturbations, out of distribution black box attacks and white box attacks.

3.10. Distributed Ledger and Smart Contracts

These use cases address the design and development of permissioned blockchains. The reason for permissioned blockchains is twofold: higher throughput, scalability, and improved privacy. The focus will be on the security requirements and requirements for the distributed ledger component.

3.10.1. Wholesale of Transport Resources

Technologies involved	**IP**
Type	Distributed Ledger and Smart Contracts
Description	In the transport resource wholesale market, more minor local carriers and wireless carriers may rent resources from larger carriers ("carriers carrier"), or infrastructure carriers instead of building their networks. Likewise, international carriers may rent resources from respective local carriers and local carriers may lease their owned networks to each other to achieve better network utilization efficiency. From the perspective of a resource provider, it is crucial that a network service is timely configured to meet traffic matrix requirements requested by its tenants. Typically, a resource purchaser expects to use the leased network resources flexibly, just like they are self-constructed. Therefore, the purchaser is not only provided with a network service, but also the full set of functionalities for operating and maintaining it. In a flexible and independent manner, the purchaser also expects to schedule and maintain physical resources to support their own end-to-end automation using both leased and self-constructed network resources. This use case shall consider a marketplace for wholesale transport resources using Distributed Ledger Technologies (DLT) and smart contracts.

3.10.2. Resource Allocation Forensic Analysis

Technologies involved	**IP, Optical, Microwave**
Type	Distributed Ledger and Smart Contracts

Description	This use case focuses specifically on the data exchanged, and data stored by the distributed ledger component will be considered. Among others, the distributed ledger will record runtime information such as software status (e.g., software/firmware version) and runtime information (e.g., remote attestation, tamper detection). This use case will enhance device and component security, enable tamper detection, and ensure the independent verification of device status, history, and details.

3.11. Compute Integration

3.11.1. NFV Infrastructure Point of Presence (NFVI-PoP) interconnection

Technologies involved	**IP**
Type	Compute integration
Description	The WAN Infrastructure Manager (WIM) controller can be defined as a dedicated entity to handle the establishment, deletion and updating of the connectivity services between the network endpoints located at different NFVI-PoPs (i.e., data centers). In other words, the WIM controller takes over the lifecycle management of the end-to-end connections to be allocated over an underlying transport infrastructure (WAN) offering multi-site service connectivity. That said, the targeted Teraflow OS (acting as an SDN Controller implementation) becomes the WIM controller interacting with an NFV orchestrator for the sake of deploying network services requiring both cloud and networking resources. For the specific and selected implementation of a NFV orchestrator based on the ETSI OSM MANO release, the interworking between the OSM and the Teraflow OS (i.e., WIM controller) relies on an available YANG data model specified in the IETF L2/L3 VPN document [RFC8466]. To this end, the OSM offers a plugin that handles the lifecycle management of L2/L3 VPN connectivity services (i.e., creation, removal, and updating) over a defined REST API. Using this API, the OSM acts as the client, whilst the Compute component of the Teraflow OS operates as the server to receive, proceed, and trigger the L2/L3 VPN connectivity services demanded by the OSM orchestrator. This entails the interactions of the Compute component with other components (e.g., service) involved in the Teraflow OS solution.

3.12. Inter-Domain

An inter-domain transport network slice comprises several transport slices controlled by multiple SDN controllers.

3.12.1. Inter-domain connectivity services

Technologies involved	**IP**
Type	Inter-domain
Description	This use case focuses on providing dedicated QoS-aware inter-domain connectivity services and enabling interaction between TeraFlow OS instance and peer TeraFlow OS instances that manage different network domains to create E2E transport network slicing service. Before exchanging any requests, two peering TeraFlow OS instances need to authenticate each other. In the context of workflows related to the preparation and activation of inter-domain services, interactions with the service catalogue (available service templates) and the service inventory (existing service instances) are necessary. Moreover, in case a service

request spans multiple domains, the slice component involves the IDC by sending it an inter-domain subslice request. In addition to sending an inter-domain subslice request to the local slice component, the TeraFlow OS instance also forwards the inter-domain subslice request to the other domains that are involved in the E2E service.

3.13. Scenario – Use Case Matrix

In Table 1 below, the relationship between the proposed use cases and the D5.1 scenarios. This matrix will ease specific demonstrations in the scope of WP5.

Please refer to D5.1 for a clear description of the proposed scenarios:

- Scenario 1: Autonomous Network Beyond 5G
- Scenario 2: Automotive
- Scenario 3: Cybersecurity

	Scenario 1	Scenario 2	Scenario 3
Inventory	X	X	
Topology	X	X	X
Service	X	X	X
Transport network slicing	X	X	
Monitoring			X
Traffic Engineering	X		
Automation	X		
Policy	X		
ML-based security			X
Distributed ledger and smart contracts		X	
Compute	X		
Inter-domain		X	

Table 1 Scenario – Use case matrix

4. Business Model and Ecosystem Analysis

Within TeraFlow, we work with business models from two approaches. First, designing business models is part of our project's regular exploitation activity within T6.1 and T6.3. Secondly, TeraFlow has a specific research task – T2.3 – which analyses and suggests how future ecosystems and business models will evolve and what they will look like - these activities enrich each other. This deliverable reports on the ecosystem analysis and makes references to D6.2.

In our analysis of the TeraFlow OS ecosystem, we rely on frameworks sourced from platform ecosystems and technological innovation systems [BER08] [Hek11]. Together, the frameworks offer well-known structures and driving factors in a growing ecosystem. The data input to our analysis includes the TeraFlow proposal, TeraFlow partners' exploitation plans, and literature. We especially appreciate the input from open-ended interviews with partners on topics described below.

As indicated in Figure 2, our analysis is structured with specific analytic components and their relationships. Ecosystems are complex market configurations; thus, we must simplify to provide a clear message. We first comment on the vision of a future market, and how the TeraFlow OS ecosystem is meant to unleash this market. We distinguish between the growth of the TeraFlow OS ecosystem and the growth of the transport network market that the TeraFlow OS enables. Then, we propose a diffusion pattern for the TeraFlow OS ecosystem and suggest roles filling it. Having the roles in mind, we structure insight on mechanisms that can both block and enable the growth of an ecosystem sorted according to well-known driving factors. Before the analysis, we summarise the main characteristics of an ecosystem.

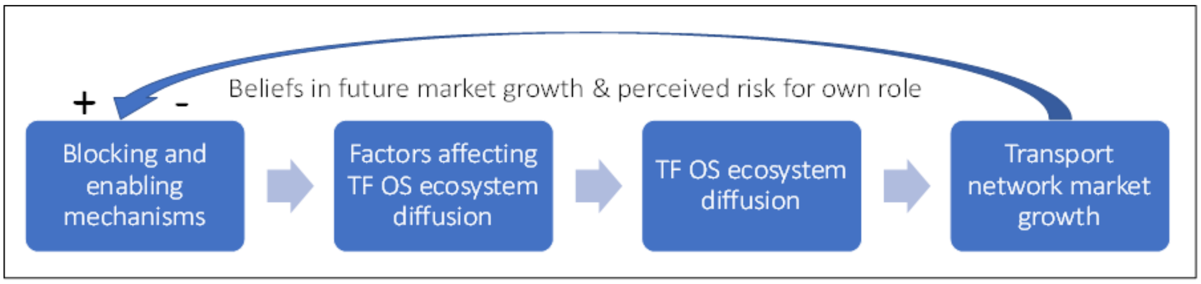

Beliefs in future market growth & perceived risk for own role
+
-
Blocking and enabling mechanisms
Factors affecting TF OS ecosystem diffusion
TF OS ecosystem diffusion
Transport network market growth

Figure 2 Components and relationships in an analysis of TeraFlow OS

An ecosystem [HAL21] is a business concept meant to capture market configurations where there are high interdependencies between roles, however, where one firm cannot directly control the others. That is, it is different from a pure market (no interdependencies and no control) and a value chain (interdependencies and possible to control). A mature ecosystem with settled roles and many firms has grown big from a small market with fewer and unsettled roles and firms, and high uncertainty. By nature, technologies and thus systemic markets with high interdependencies are characterized by strong reinforcement effects that explain the high uncertainty about a growth and growth path. Open interfaces and technologies make it easy to innovate, be interoperable, and grow in an ecosystem. While this is necessary to kick off an ecosystem, it is also necessary to make it attractive for firms to join an ecosystem by decreasing their perceived risk and increasing belief in profit opportunities in a future market.

4.1. TeraFlow OS and transport network market growth

TeraFlow is about developing an SDN controller for the transport network which is using OPEN and standard APIs enabling transport *network abstraction* and *vendor agnostic programmability,* thus full interoperability across network devices from different vendors and network applications (NetApps). Eventually, operators' ambitions with applying Teraflow OS are increased cost efficiency, flexibility and innovation, and decreased delivery times. Thus, an expected market pull and operator ability to serve a growing market is a backdrop for the TeraFlow OS initiative (please, also confer with Section 2 and D6.2). Implicitly, better interoperability across hardware, software, and operators is the only way to unleash this market growth - i.e., the ultimate argument for TeraFlow OS.

However, the argumentation for the future market contains the ambition of operators decreased costs, and thus, delivers two conflicting messages to stakeholders:

1. Total market decrease: Cost efficiencies for operator → market decrease for providers
2. Total market growth: Operators' innovation, addressing more needs → market growth for all

Eventually, beliefs or dis-beliefs in market growth and roles can mobilize and detract stakeholders. We emphasize this tension here, because our analysis shows that blocking and enabling mechanisms for the TeraFlow OS ecosystem seem to be highly affected by stakeholders' concerns about the constitution of a future market.

4.2. Diffusion of TeraFlow OS ecosystem

Figure 3 illustrates the diffusion of one ecosystem, with five phases [HEK11]. The vision of TeraFlow OS is to accelerate into **huge diffusion**; in its current form the TeraFlow OS ecosystem is an idea in a **pre-development phase**. Thus, the point of view in our analysis is an early phase of the TeraFlow OS ecosystem diffusion. From this angle, we identify and discuss the factors and mechanisms that are dominant in this phase, and what may bring the TeraFlow OS ecosystem into take-off and acceleration.

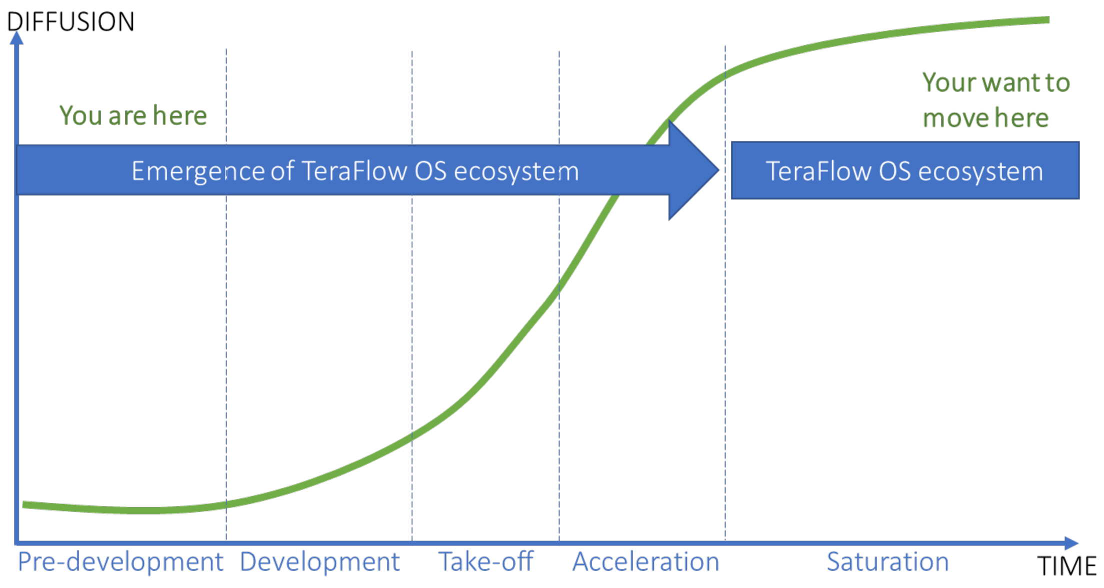


Figure 3 Idealized growth path for an ecosystem, positioning TeraFlow OS in pre-development phase (adjusted from [HEK11])

4.3. Roles in TeraFlow OS ecosystem

The following is an overview of the roles identified in the TeraFlow OS ecosystem, integrating roles of today and the future as depicted in Figure 4. We expect that the roles mostly will be the same, but the content of the role may change. As far as possible, we comment on this change.

The (network) **operators** implement TeraFlow OS in their operations of transport networks. Telefonica and Telenor are examples of operators. Figure 4 is a basic illustration of how TeraFlow OS affects one operator and its direct customers and providers. Currently, transport network operators are predominantly cost centres in telecommunication providers, who in turn sell fixed and wireless accesses and subscriptions to consumers and enterprises. However, in future, the operators of transport networks may be a profit and loss centre, selling services to many types of customers.

The **operator's products in the** future are Transport network slice (here denoted NaaS for the network as a service), or Layer2 Virtual Private Networks (L2VPN). These products are offered to different customers at a price dependent on, e.g., traffic volume or quality.

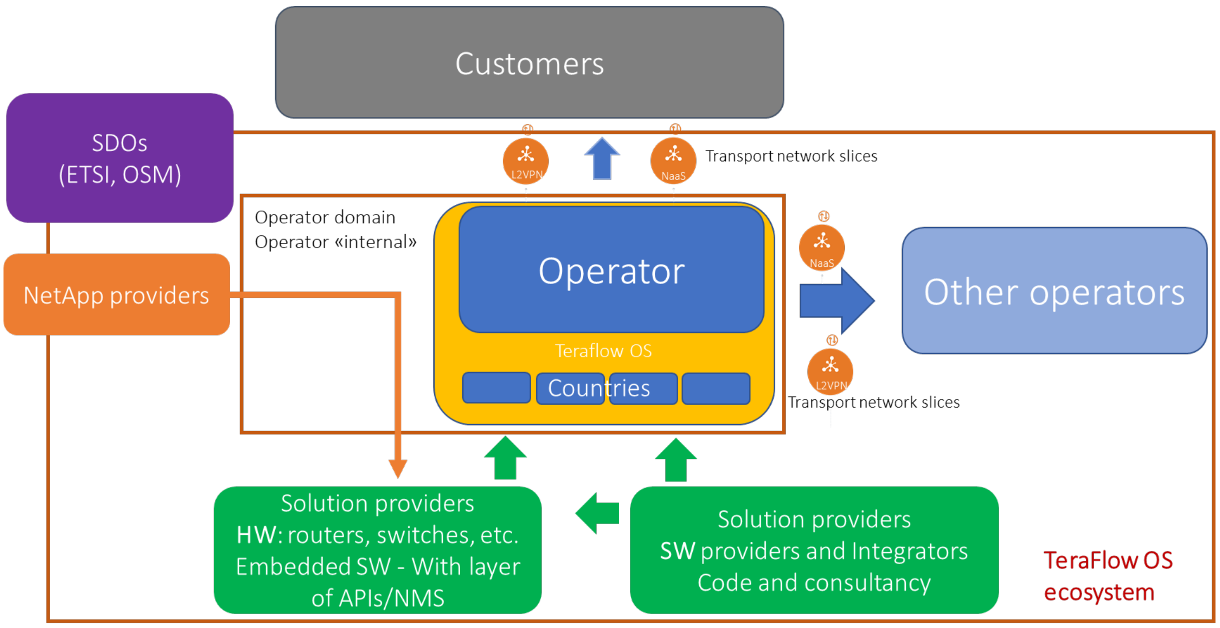


Figure 4 Roles in the TeraFlow OS ecosystem, perspective of one operator

The **operator's customers may be an entity that** is part of the same telecommunication provider and combines transport, core, and access networks – i.e., the communication service provider (CSP) role – or a digital service provider (DSP), which combine network components from different providers. Furthermore, **other operators** of transport networks are important customers. It is an additional challenge for some operators to operate the transport network across many countries and local operators. Such customers and other operators will have yet other customers who demand services and drive market growth.

The first set of **providers** to the transport network operator are solution providers who sell **hardware**, with embedded proprietary software and proprietary interfaces (network management systems). Infinera and SIAE are examples and partners in TeraFlow. Currently, this is the main way of selling hardware and software to operators, mostly with a price per hardware unit. The different provider-specific hardware interfaces constitute the big challenge that TeraFlow OS addresses and intends to replace vendor agnostic programmability. For traditional hardware providers, their role is shifting towards being providers of software, with new market terms and business models.

In the TeraFlow project there are also solution **providers** who sell only **software** in combination with hardware; however, their current role is marginal compared to hardware providers. It is expected that TeraFlow OS opens for smaller software providers and drives previous hardware providers into the role of software providers. Thus, software providers may deliver to, or be complements to the hardware providers indicated in Figure 4. Ubitech is a TeraFlow partner which deal with software and adjust it to hardware or help other software providers to deploy their applications on hardware or in networks.

The second set of **providers** are those developing and selling NetApps which can be used in the operation of the transport network. Currently, providers will have to align their NetApps to the proprietary interfaces offered by hardware providers. It is expected that the number of NetApp providers could increase if the TeraFlow OS ecosystem takes off. NEC is a partner in TeraFlow and an example of a potential provider of a security NetApp.

We suggest that **Standard Developing Organizations** (SDOs) have a distinct role in the TeraFlow OS, as a governance mechanism ensuring sustainability in the long term. For open-source initiatives such as TeraFlow OS, SDOs must be understood to include well-governed and trusted open-source communities such as OpenSource Mano.

The **provider of an SDN controller** is also a relevant role. Currently, actors in these roles provide complete systems which address the same domain as TeraFlow OS (for definitions and competitors, see D6.2). However, in future, as open-source software, TeraFlow OS (APIs for standard network and service modelling) would be sourced from a shared repository governed by a designated community (e.g., such as ETSI's opens-source project OSM) or implemented and published as fully interoperable standards by well-known SDOs. In this context, roles as **system and SW integrators** or **consultancies** may emerge and extract a significant share of the market for SDN controllers [BER08]. E.g., the partner Peer Stritzinger sees itself as a provider of solution integration in the TeraFlow context.

In Figure 4, TeraFlow OS is the network management system used on the interface between the operator and solution providers, and which NetApp providers also use (see also D6.2).

4.4. Interdependencies in TeraFlow OS ecosystem

Figure 4 depicts only one operator using TeraFlow OS. Ideally, more operators must use TeraFlow OS to put sufficient pressure on hardware providers to adhere to open interfaces. Furthermore, the market for independent NetApps becomes sufficiently attractive only when providers can replicate implementations across operators. Finally, the need for transport network interoperability between operators calls for a wider implementation. Thus, there is ***interdependency*** between roles and their motives in the TeraFlow OS ecosystem. It becomes important to reach a ***critical mass*** in the development, demand for and use of Teraflow OS to take off and ***mobilize and motivate*** yet other operators, providers, and SDOs.

4.5. Enabling and blocking mechanisms for TeraFlow OS ecosystem

Many enabling and blocking mechanisms will affect take-off and eventual acceleration of the TeraFlow OS ecosystem. In Table 2, we have sorted our input in this respect according to well-known driving factors for growth for technological systems: legitimation; profit opportunities; market formation vision; experimentation; knowledge exchange and development; open technologies; network effects/externalities [GAW14], [BER08], [HEK11]. Research has shown that these factors have a lot of explanatory power on an aggregated level for how an ecosystem evolves and should be catered to from a strategic point of view. The mechanisms that enable or block, for instance, a factor like experimentation, can vary and be case-specific for the TeraFlow OS. It should be noted that the relationships between mechanisms and factors are not mutually exclusive.

Furthermore, the input is what partners can imagine as potentially having positive or negative effects on TeraFlow OS diffusion, now and in future. At this stage in our analysis, we share the raw input in Table 2. Still, we already see that the patterns and focus of interest that emerge are consistent with TeraFlow OS being in a pre-development phase.

It should not come as a surprise that a lot of the input concerns the openness of technologies, and how beliefs and attitudes are either positive or negative for the evolution of TeraFlow OS. In line with expectations in an early phase [HEK11], the intention and relevance of building knowledge is motivating partners, but also seen as an obstacle if it is not shared efficiently in a larger community. It is well-known that stakeholders must be able to envision profit opportunities for themselves in early phases to be motivated; in our data we see that a lot of resistance towards TeraFlow OS can be found in a skepticism to future market growth and a profitable role for providers. Yet others cannot see that there is an alternative to accepting the openness TeraFlow OS introduces.

Table 2 Enabling and blocking mechanisms for diffusion of TeraFlow OS ecosystem (factors adapted from (Gawer & Cusumano, 2014; Bergek, Jacobsson, Carlsson, Lindmark, & Rickne, 2008; Hekkert, Negro, Heimeriks, & Harmsen, 2011))

Enabling mechanisms	Driving factors for TeraFlow OS diffusion	Blocking mechanisms
• Large providers: the market moves towards openness, open interfaces • Governance system for standards and open source is stable • Large operator takes lead in openness • Large operators ally for openness, have a forum for sharing • Providers see it as an advantage to be pluggable, implemented easily, interact with open interfaces - to be compliant • SDO's acknowledgement	**Legitimation (trust in sustainability over time)**	• HW and SW providers reluctant to adhere to open source • Operators do not know if TF OS is the winner – wait. Followers. • Political positions on standards (country, actor) • Standard/open source will not be sustained
• Success stories • Use-cases are good (need to be solved) • Operators start to use and purchase open HW and SW, market grows, and some providers increase their sales • Proprietary providers are lagging – decide to join the "open" trend to take part in market growth • Opens source business models: Proprietary on top, Semi open, Exclusive knowledge • SW developers' knowledge is unique – basis for profit extraction, e.g., consultancy • Large providers – think there eventually will be other business models • Small provider - With open source we can be part of the larger market – spin offs	**Profit opportunities (belief in own profit in future market)** **Market formation vision (belief: market will grow from small to large)**	• Providers (all) think appropriation level for open tech (profit opportunities) too low • Providers cannot see willingness to pay for coding, system and SW integration, consulting, debugging • Future markets are perceived as too small • Providers' price and profit decrease – fear future profit • Providers see only cost cutting – not market growth • Providers see only additional costs – and no benefits for them • Do not recognize the need for interoperability • Operators not interoperable and thus, too few customers to scale for small providers

Enabling mechanisms	Driving factors for TeraFlow OS diffusion	Blocking mechanisms
• Small provider – the market moves towards open, customers want this • Small provider – costs less when not developing/maintaining proprietary interfaces – network mngm. systems • Interfaces are open – still possible to profit from proprietary embedded SW • Interoperability is necessary to manage to grow the market (total, and for own firm) • Interoperability between operators is necessary to bring scale to providers		• Providers certain about short term disadvantages - uncertain about how things will evolve in long term • Operators will not invest or pay – till benefits demonstrated • Providers will not deliver – till sure they get paid
• Small provider – we can adhere to what is required – deliver specialized services • A platform for experimentation, testing, demonstrating innovations – a neutral lab • A prototype environment where it is easy to develop and implement • Certifications – from experimentation • Allowing experimentation and learning • Mobilize providers with use-case method – invite providers to solve issues • To test in an SDO environment	**Experimentation (will reduce uncertainty and enable innovation and growth)**	• Increased innovation leads to increased competition • Testing is expensive • Not possible to get competitors to contribute within same environment – assessing competitors' innovations and works is weird • No willingness to fund experimentation lab
• Large developer community – critical mass • Large providers want to learn – put pressure on providers • Open interfaces – demonstrated feasibility • Active sharing of insight with all stakeholders • Continuous collaboration in research projects • Use of code repositories like GitLab • Available SW development environment, with e.g., downloadable virtual machines, sample topologies, graphical interfaces • Code is well explained and advertised – good enablement of using code • Connect to – be part of larger community • Code is published, shared, maintained, and governed in a professional way • Common rules and recommendations for coding are established and shared	**Knowledge exchange and development (will build insight on all other driving factors, knowledge externalities)**	• Small developer community • Open-source code not published properly • Not properly implemented to SDOs • Developers do not risk joining a small project – coding is wasted • Code does not have the umbrella, the ecosystem outside your own project/code • It is a huge effort to come up to speed on open-source coding for domain • Operators are locked into knowledge associated with existing providers • Operators are locked into HW regime – have no SW expertise
• Operators adhere to use of, demand, and use open interfaces – from all providers • HW providers adhere to the use of open interfaces • Large developer community • Open source turns out to be far more efficient production of code • Interface standards are good – implementations are compatible/interoperable • It is easier to implement new interfaces and reprogram HW with network programmability – less costly to adhere to new standards and interfaces • Large providers loose contracts when not adhering to open interfaces /standards	**Open technologies (will ease ability to innovate and grow)**	• Large providers are too fond of licensing model – the potential customer lock-in is attractive • Not invented here syndrome • Open will not function anyway – stay with current (open=naive) • It is only semi-open – there is proprietary software on top • Exclusive knowledge • Code is eventually, not disclosed • HW and SW providers reluctance to adhere to open source • For providers, to implement a new open standard interface – is costly • Software is not good enough • Standards are only ideas – not realized • Open-source code/standards are not good enough – allow variety

Enabling mechanisms	Driving factors for TeraFlow OS diffusion	Blocking mechanisms
• Smaller providers innovation and competitiveness – pressure on large providers • Easy to join for small NetApp providers • Operators are positive to open-source – frontrunners • Well-managed implementation into SDOs • System functions across standards/SDOs • Compatibility between HW and SW • Mutual knowledge between HW and SW providers • Software is good • Black box components (closed embedded code) are easily provided in open environment • Ensure sustainability and maintenance by being part of and SDO • Ensuring standards are realized – not just ideas		• SW for too specific areas is high risk • Operators - do not adopt e.g., TeraFlow OS - think network is "baby" – good relationships with providers - dependent on providers - are fine to buy HW and press prices (success story) - think TeraFlow OS is too small - do not trust the "providers" - do not have the SW knowledge – will have to buy externally anyway - are bad at specifying – specify failures - do not have a big enough pain point - quality levels too uncertain, network assurance at risk • Resistance towards changing a stable system • Code is costly to maintain
• Observed operators and providers innovation diffusion – increased profits • Value (creation with) of open-source SW increases when shared • Traction for open/standard interface – not important that TeraFlow OS wins • Succeed/solve openness in IP network – other layers will follow • Many other components have access to, and start to use the interfaces	**Network effects/externalities (critical mass reached, leading to further increase)**	• No traction

4.5.1. Further work with TeraFlow OS ecosystem

It is essential to cultivate enabling and mitigate the blocking mechanisms to ensure the evolution of the TeraFlow OS ecosystem. Currently, the TeraFlow project can follow up on four themes:

- Aim to enable good exchange of and development knowledge about the technology in question in the wider community. This would motivate project partners and mobilize yet others to contribute to knowledge development
- Aim to clarify how TeraFlow OS can be ensured sustainability to decrease the perceived risk of contributing. E.g., it is risky to participate in an open-source initiative that will not take off. This would also increase the legitimacy of the initiative
- Establish a trustworthy story for how the total market may grow
- Elaborate on roles in TeraFlow OS ecosystem and assess their future business models and profit opportunities

More concretely, the next step in the ecosystem analysis will be to:

- Continue interviews and data collection and draw further learnings
- Further clarify the interdependencies in the TeraFlow OS ecosystem
- Describe potential future roles in more details, and how their business models can be aligned with the intentions of TeraFlow OS ecosystem.

- Clarify how a future TeraFlow OS can be governed and maintained

5. Requirements for the TeraFlow SDN Controller

In this section, the different requirements for the TeraFlow SDN Controller are introduced. They have been classified as:

- Functional Requirements: describe what the system must or must not do and can be thought of in terms of how the system responds to inputs.
- Non-functional Requirements: are requirements that specify criteria that can be used to judge the operation of a system rather than specific behaviours. They are contrasted with functional requirements that define specific behaviour or Functional Requirements.

In order to have them ordered, the same use case topics as described in Section 4 are used.

5.1.1. Inventory

- [REQ-INV-01] The SDN controller shall be able to recover information and state about hardware components of network elements and the logical configuration of the devices, including logical interfaces configuration from one node and all the parameters related to interface configuration (description, IP addressing, VLAN, etc.) of a node.
- [REQ-INV-02] The SDN controller shall offer the retrieved inventory to a client.

5.1.2. Topology

- [REQ-TOP-01] The SDN Controller shall provide a set of abstractions to represent several views of the network topology.
- [REQ-TOP-02] The SDN controller shall be able to collect the required information for providing network topology from the Network Devices or configuration files.

5.1.3. Service

- [REQ-SERV-01] The SDN Controller shall be able to manage the life cycle of L2VPN and L3VPN services [MUS21]. These are widely used to deploy 5G fixed and enterprise services mostly because several traffic discrimination policies can be applied in the network to transport and guarantee the right SLAs to the mobile customers.
- [REQ-SERV-02] An L3VPN service shall create a virtual routing and forwarding network instance (VRF) in each of the nodes involved in service deployment. This routing instance allows routing information to be propagated between the sites involved in the service.

5.1.4. Monitoring

Notification subscription requirement:

- [REQ-MON-01] TeraFlow OS shall allow external subscriptions to the notification service to enable visualisation of relevant data at a higher level of abstraction, thus assisting the TeraFlow OS in identifying potential problems and gaining dynamicity. Moreover, external agents (subscribers) shall be notified of events related to network/slice data (e.g., topology or connectivity) depending on the nature of the subscription, see below.

Topology monitoring requirements: It is expected that the SDN controller shall gather data on topology events and be able to use the notification system to inform subscribers. The basic preliminary actions to be monitored/notified in this regard are:

- [REQ-MON-02] Addition of a new topology element (e.g., topology, link, node, node edge point).
- [REQ-MON-03] Modification of parameters of existing elements in the topology (e.g., scaling/migration of resources).
- [REQ-MON-04] Status of operational changes of existing elements (e.g., up/down status), assuming both control plane level (network element-SDN controller) and data plane level (inter network components).
- [REQ-MON-05] Deletion of elements in the topology.

Connectivity monitoring requirements: The SDN controller will be able to collect data related to the connectivity of the elements in two different views: network or slice. In this regard, the main features to be monitored (or notified to subscribers) are:

- [REQ-MON-06] New connectivity-service element inserted/removed in/from the network/slice.
- [REQ-MON-07] Status change of existing connectivity-service element in the network/slice.
- [REQ-MON-08] Status change of the switching conditions of an existing connection element in the network/slice.

Microservice life-cycle monitoring requirements: The SDN controller will report on metrics that inform on the lifecycle operations of the microservices within the TeraFlow OS deployment. Functionalities that shall be monitored/notified according to the typical phases of a microservices life cycle are:

- [REQ-MON-09] Downshifting monitoring/notification, cloud-to-edge operations shall be notified to subscribers to guarantee cloud-to-edge migrations.
- [REQ-MON-10] Runtime monitoring/notification will be targeted to enable external monitoring of microservice’s runtime metrics usage, e.g., power, CPU, storage, memory, or bandwidth.
- [REQ-MON-11] Edge-to-cloud load balancing monitoring/notification, subscribers shall be notified of edge-to-cloud migration operations when required.

5.1.5. Traffic Engineering

- [REQ-TE-01] The SDN controller should configure the compatible devices through the device management component so they can connect to the PCE.
- [REQ-TE-02] The SDN controller should build a traffic engineering database (TED).
- [REQ-TE-03] The SDN controller should expose an API to create, modify and delete segment routing LSPs.

5.1.6. Automation

Network management requirements:

- [REQ-AUTO-01] Automatic topology discovery and inventory tracking of both physical and virtual network elements.
- [REQ-AUTO-02] Automated configuration and management through NBI (e.g., REST/gRPC) and SBI (e.g., NETCONF/YANG) interfaces based on open standards.

Service provisioning requirements:

- [REQ-AUTO-03] Automatic creation and management of network services specified via an open standard NBI, as well as communicating, through its SBI, with all the network elements needed to implement the service.
- [REQ-AUTO-04] Automated association of a network service to one or more flows.

Network operations' requirements:

- [REQ-AUTO-05] Automatically deploy a Network Operating System (NOS) and a base configuration when a new network element is added to the network, such that the network element enters in production without human configuration.
- [REQ-AUTO-06] Remotely upgrade the entire NOS or some of its components when a vendor releases a new version.
- [REQ-AUTO-08] Run-time and secure rollback version recovery when a NOS needs to be upgraded/downgraded in response to a detected problem.
- [REQ-AUTO-09] Run-time NOS migration from one vendor to another.
- [REQ-AUTO-10] Automatic provisioning of basic white box configuration, common across multiple white box vendors.

System stability requirements:

- [REQ-AUTO-11] Automatically perform rollback flow state recovery operations in case of a misconfiguration or race condition
- [REQ-AUTO-12] Automatically translate data stemming from detected attacks to specific remedy actions.
- [REQ-AUTO-13] Automatically detect network elements with anomalous behavior.

5.1.7. Policy

- [REQ-POL-01] The SDN controller shall allow network operators to easily create policy rules. Each policy rule will be comprised of high-level policy conditions (i.e., traffic selectors) and actions (i.e., traffic treatments) and will be triggered upon a network event.
- [REQ-POL-02] A policy rule shall be associated with a service ID.
- [REQ-POL-03] A policy rule shall be associated with a policy state, which indicates whether this policy is (i) planned for enforcement, (ii) actively enforced, or (iii) inactive.
- [REQ-POL-04] A policy rule shall be associated with a policy type, indicating whether this policy applies to a single device (I.e., device-level policy) or a network segment (i.e., network-wide policy).

- [REQ-POL-05] A policy rule shall be associated with a priority, which indicates the order of this policy in a broader list of policies applied to the data plane.
- [REQ-PO-L06] A policy rule shall be associated with a timeout, which indicates the amount of time for this policy to remain active. After the timeout expires, the state of this policy is put on hold, after a data plane update ensures that no flow rules of this policy remain active. In case of no timeout specified, a policy is characterized as permanent. To revoke such as policy, an explicit policy deletion must be issued by the TeraFlow controller.
- [REQ-POL-07] Once a policy rule is mapped and applied to a service, the SDN Controller will automatically associate this rule with a set of device IDs. This way, device, and service (re)configuration will be co-scheduled in order to enforce this policy.

5.1.8. Cybersecurity

Data gathering requirements:

- [REQ-CYB-01] The network traffic shall be captured by nodes themselves or based on mirroring techniques.
- [REQ-CYB-02] Telemetry and network flows are processed to extract statistical information including the information per packet, per network flow, and per protocol (TCP/UDP).
- [REQ-CYB-03] Statistical information extracted from telemetry and network flows is used as data source to ML-based Cybersecurity components.

Interface requirements:

- [REQ-CYB-04] The ML-based Cybersecurity solution notifies the TeraFlow SDN controller if it detects a malicious activity (i.e., network intrusions and harmful connections).
- [REQ-CYB-05] A classification of the malicious activity will be provided (e.g., a DDoS attack connection, a cryptomining attack connection, a vulnerability scanner attack, an unknown attack) jointly with a confidence value (e.g., a probability).
- [REQ-CYB-06] After a malicious activity is detected and sent to the TeraFlow SDN controller, a countermeasure should be provided to secure the network.

Machine Learning requirements:

- [REQ-CYB-07] Edge nodes can host ML-based inference engines to detect cybersecurity attacks locally.
- [REQ-CYB-08] Edge nodes hosting ML-based inference engines can provide predictions jointly with telemetry and statistical information.
- [REQ-CYB-09] ML-based inference engines will be resilient to adversarial attacks that will try to fool them to produce incorrect inferences.

5.1.9. Distributed Ledger and Smart Contracts

- [REQ-DLT-01] The TeraFlow SDN Controller shall be able to interact with other SDN Controllers using DLT to rent resources from respective local carriers and local carriers may lease their owned networks to each other to achieve better network utilization efficiency.
- [REQ-DLT-02] A network service is configured in a timely way to meet traffic matrix requirements requested by its tenants.

- [REQ-DLT-03] The distributed ledger will record device information such as software status (e.g., software/firmware version) and runtime information (e.g., remote attestation, tamper detection).
- [REQ-DLT-04] The distributed ledger will enable tamper detection and ensure the independent verification of device status, history, and details (remote attestation).

5.1.10. Transport Network Slicing

- [REQ-SLI-01] The TeraFlow SDN Controller shall provide transport network slice life-cycle management. Users may formulate transport network slices based on the demand for services and time to schedule the resources from the entire network's perspective flexibly. Several underlying services might be offered for a transport network slice, including L3VPN, MPLSVPN, or VLAN constraints.
- [REQ-SLI-02] The TeraFlow SDN Controller shall provide vertical industry slicing, which is a category of network slicing that is emerging due to the high demand for private high-speed network interconnects for industrial applications. In this scenario, the biggest challenge is to implement differentiated optical network slices based on the requirements from different industries.
- [REQ-SLI-03] Isolation shall be provided through resource partitioning and/or robustness techniques, e.g., dedicated resources, shared resources with safeguards, or reserved backup paths. Examples include traffic separation via VPNs (L2/L3VPN, EVPN), interference avoidance via network capacity planning, traffic policing or shaping, and prioritization in resource utilization.
- [REQ-SLI-04] Hard isolation can be achieved by provisioning dedicated fibres, which is feasible, but very expensive. Therefore, physical splitting (e.g., in time or frequency) can be used. For instance, in optical networks, where full lambdas can be isolated (WDM), or TDM techniques by assigning specific time slots to specific slices.
- [REQ-SLI-05] Soft isolation solutions shall rely on the simple separation of traffic delivery such as MPLS or VLAN tagging. These mechanisms offer separation, but not isolation performance guarantees. The SDN controller should be able to create soft network slices, via creation of multiple VRFs and VSIs on a network element (physical or virtual), as described in section 4 of the TIP MUST SBI Spec. [REQ-SERV-01], and [REQ-SERV-02] should be supported within these soft network slices.
- [REQ-SLI-06] The design of intermediate isolation solutions between hard and soft isolation may be classified into two classes: i.) Link layer (Layer 1.5 / Layer 2) technologies such as Flex Ethernet (FlexE), dedicated queuing, and TSN. ii.) Network layer technologies such as MPLS-TE, Deterministic Networking (DetNet), Segment Routing (SR).

5.1.11. Compute

From a macroscopic perspective, the Compute component is the front-end to take over of the interaction of the Teraflow OS instance with an external NFV orchestrator such as the OSM implementation. The aim is to automatically receive and process connectivity service requests from the NFV Orchestrator (e.g., creation, deletion, and removal) and then interact with other Teraflow OS components to meet such requested connectivity service operations. To this end, the following requirements need to be fulfilled:

- [REQ-COM-01] The Compute component should support a well-defined API (including a data model and protocol) to enable the interaction with an external NFV orchestrator implementation.
- [REQ-COM-02] The connectivity services requested from the NFV Orchestrator should support both L2VPN and L3VPN flows.
- [REQ-COM-03] The API between the NFV Orchestrator and the Compute Component should support the complete lifecycle management of the connectivity services. This encompasses the creation of point-to-point connectivity services specifying the service identifier, connection endpoints, transport layer and associated attributes (e.g., VLAN), traffic engineering details (e.g., guaranteed bandwidth, maximum tolerated latency, etc.), etc.
- [REQ-COM-04] The Compute component should keep track of the active connectivity services (e.g., in a dedicated repository) to handle the operations arriving from the NFV orchestrator such as removing or updating a specific connectivity service.
- [REQ-COM-05] The Compute component should offer a mapping function to “translate” the incoming connectivity service operation based on the defined API (i.e., protocol and encoding) to the commands and messages based on gRPC to interact with other Teraflow OS components (e.g., Service).

5.1.12. Inter-Domain

The Inter-Domain use case describes the interaction of a TeraFlow OS instance with peer TeraFlow OS instances which manage different network domains. The requirements include:

- [REQ-INT-01] Before exchanging any requests, two peering SDN Controller shall authenticate each other.
- [REQ-INT-02] After receiving an inter-domain subslice request from a peering SDN Controller, the SDN Controller analyses the order and – if the order can be fulfilled – uses the corresponding internal interfaces to (partially) satisfy the request.
- [REQ-INT-03] Similarly, when a modification of an inter-domain E2E service is requested, the SDN Controller shall use the corresponding internal interfaces to propagate the modification request.
- [REQ-INT-04] In the context of workflows related to the preparation and activation of an inter-domain service as well as the modification of services, interactions with the service catalogue (available service templates) and the service inventory (existing service instances) are necessary.

5.1.13. TeraFlow SDN Controller Security

Traditional SDN controllers, implemented as monoliths, provide authentication and authorization interfaces for external users and components (GUIs and APIs). Since those controllers are monoliths, securing the external interfaces is mostly sufficient for securing the controller. Internal communication between components is performed through programming-language-level calls and there are (mostly) no security concerns regarding those calls. But these might include programming languages errors, security of the hosts running the System, or even memory errors.

TeraFlow, on the other hand, adopts a cloud-native architecture. This means that different components of TeraFlow will communicate through standardized network protocols, and different components might be running in different machines in the network. Therefore, besides the usual authentication and authorization procedures commonly found in SDN controllers, TeraFlow shall ensure the secure message exchange among components.

- [REQ-SEC-01] The TeraFlow SDN controller shall provide means for external user/entity authentication. These users/entities can be people accessing the TeraFlow GUI, or external clients (automated code) using TeraFlow APIs.
- [REQ-SEC-02] The TeraFlow SDN controller authentication procedure shall maintain a record of the action permissions that are available for each one of the users/entities registered.
- [REQ-SEC-03] The TeraFlow SDN controller shall provide means for user authorization. This means that any component shall be able to obtain a handle for the currently authenticated user, including its permissions, to determine whether the current user has enough permissions to execute a given operation.
- [REQ-SEC-04] The TeraFlow SDN controller shall have mechanisms that allow the components to authenticate among themselves, therefore ensuring that the internal communication among components cannot be intercepted, modified, or falsely generated by a third-party malicious entity.

5.2. Non-Functional Requirements

We have analysed the following non-functional requirements: performance, usability, scalability, security, and portability.

5.2.1. Performance

- [REQ-PERF-01] The TeraFlow SDN controller shall provide an increase by an order of magnitude (x10) of the flow processing capabilities of current SDN controllers. This results in the ability to handle a Tera of connectivity services.
- [REQ-PERF-02] The SDN controller shall provide sufficient processing and sample rates of metrics that does not limit: i) the achievement of the objectives in the use cases; and ii) the timely identification and notification of potential problems, maximising the possibilities for mitigation. The expected rates shall be determined based on three criteria: i) objectives of the use case; ii) nature of the metric; and iii) priority of the metric.
- [REQ-PERF-03] Cloud-native flow management shall be provided with a control plane latency below 10 milliseconds. The SDN controller shall increase multi-layer resource allocation efficiency by 30% due to seamless deployment of VPN services.
- [REQ-PERF-04] Proactive SDN traffic optimization by means of ML algorithms (e.g., collection of real-time KPI data and use of ML to forecast where and when a problem is likely to occur, to reroute traffic before it happens). The SDN Controller shall provide a reduction of 25% resource usage due to ML-based traffic optimization.
- [REQ-PERF-05] The introduction of follow-me and context-aware network connectivity services shall generate at least a 10% reduction in network flow requests, which are generated due to network mobility.
- [REQ-PERF-06] Novel algorithms for latency budgets as a function of application requirements. These algorithms shall improve network consumption by 30% by providing joint strategies for allocation of compute and network resources.

- [REQ-PERF-07] Reduction of energy consumption by 30% thanks to algorithms that combine centralized computing elements and low-energy networks, as well as low-cost edge computational resources.
- [REQ-PERF-08] Improve network resource usage by 30% by adopting multi-tenancy resource allocation algorithms.

5.2.2. Usability

- [REQ-USA-01] The TeraFlow SDN controller shall provide a user interface that allows triggering a default service within seconds.
- [REQ-USA-02] The SDN controller shall provide a system to enable subscribers to visualize the metrics in a friendly and customizable manner according to their needs.

5.2.3. Scalability

- [REQ-SCA-01] The SDN controller shall provide autonomous replication of micro-services to support high numbers of incoming requests.
- [REQ-SCA-02] Optimized consensus algorithms for permissioned ledgers that scale above 100 nodes. Privacy-aware smart contracts for network management tasks, and in particular, the ones related to network resources and services, thus providing forensic evidence in multi-tenant scenarios.

5.2.4. Security

- [REQ-SEC-01] Experimentally verified identification of known physical-layer attacks with 99.9% or higher accuracy and of previously unseen attacks with 90% or higher accuracy with the inaccuracy fully compensated through window-based attack detection. Protection mechanisms will be able to interact with Flow Management, in the order of milliseconds, to create, modify, or remove potential flow threats. ML-based attack detectors shall be resilient to advanced threats, especially to adversarial attacks. 90% of attacks detected in less than 0.5 seconds.
- [REQ-SEC-02] Edge and central ML-based detectors operating in optical, network, and transport layers to promptly detect and mitigate attacks. Increase of protection reaction agility by reducing centralised response latency by 30%. ML-based threat detectors will use AutoML techniques to reduce model complexity, thus decreasing resource usage by 25% in comparison with current techniques.

5.2.5. Reliability

- [REQ-REL-01] The SDN controller shall monitor micro-services and per-flow status to apply healing mechanisms (e.g., component restart, flow redirection) both from a control and a data plane perspective.

5.2.6. Portability

Portability non-functional requirements refer to the usability of the same software in different environments.

- [REQ-PORT-01] The SDN Controller shall run on top of a Kubernetes cluster deployed over an UBUNTU 20.04 LTS server. The minimal required hardware shall be 4vCPU, 32Gb RAM and 1Tb HD.
- [REQ-PORT-02] User Interface shall be accessible for Firefox web browser and compliant with W3C standards.
- [REQ-PORT-03] Integration of the TeraFlow OS to support network visibility and management in compute infrastructure (e.g., Kubernetes, OpenNess, and Akraino). Moreover, TeraFlow will develop a specific plugin for NFV and MEC orchestrators to support integration with the TeraFlow SDN controller.

6. Proposed Architecture

This section presents the proposed architecture for the TeraFlow SDN controller. First, we will provide an overall view of the architecture. Secondly, we proceed with per-component architectures, with a special focus on RPC and data models involved. Finally, we present several workflows that involve multiple component interaction.

6.1. Overall Architecture

The cloud-native software architecture is based on container-based services (containers are a lightweight virtualization technique), which are deployed as microservices and managed on elastic infrastructure through agile DevOps processes and continuous delivery workflows. These micro-services are a software development technique that structures an application as a collection of interconnected and related services. In a micro-services architecture, services are simple and detailed, and the protocols are lightweight.

Figure 5 provides an overview of the proposed TeraFlow OS architecture. The TeraFlow OS is a cloud native SDN controller that is composed of multiple micro-services. Microservices interact with each other using a common integration fabric. Moreover, in the context of B5G networks, the TeraFlow OS can interact with other network elements, such as NFV and MEC orchestrators, as well as OSS/BSS. The TeraFlow OS controls and manages the underlying network infrastructure, including transport network elements (optical and microwave links), IP routers, as well as compute nodes at edge or public cloud infrastructures.

The TeraFlow OS cloud-native architecture provides multiple benefits which have already been clearly demonstrated in other cloud computing applications. The most important benefit is application resiliency, where microservices are monitored and restarted in case of misbehaviour. Another benefit is application scalability, which accommodates an increasing number of requests (i.e., load), with the deployment of new microservice instances when required. To detail the different TeraFlow OS functionalities (each one based on a single or multiple micro-service(s)), they have been divided into core and netApps functionalities. This classification is based on the degree of inter-relationship of these micro-services, as explained below.

TeraFlow core micro-services are tightly interrelated and collaborate to provide a complete smart connectivity service. Once a Transport Network Slice request is received, the Slice Manager translates this request to the Service component. Moreover, the slice request is recorded by the DLT component in the blockchain. The Service component computes the necessary connectivity services and requests the necessary network element configuration (e.g., NETCONF, P4, OpenFlow), or interacts with underlying SDN controllers through the Device Management component. These configurations are also recorded using the Distributed Ledger component. Policies per flow are computed in Traffic Engineering (TE) component and verified, and network elements are monitored for anomalous behaviour in the Automation and Policy Management components. The Context Manager is responsible for handling the distributed non-relational database that contains all necessary information (including slice and/or flow requests, network topology, and network elements' configuration).

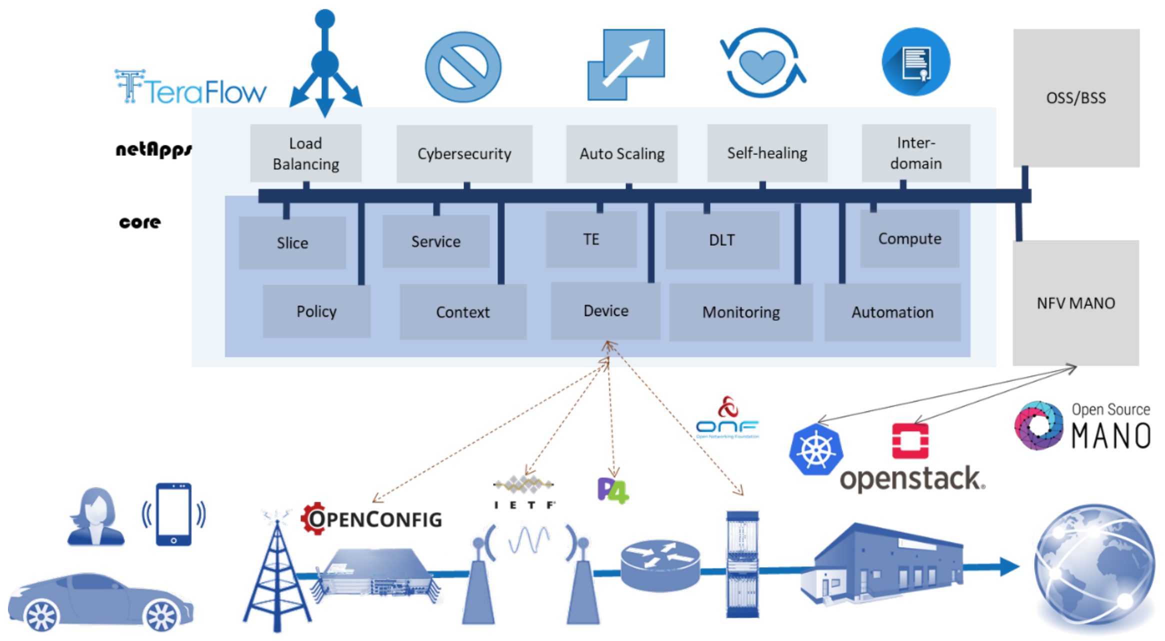


Figure 5 Overall proposed architecture

TeraFlow netApps consume TeraFlow core micro-services. The TeraFlow netApps provide the necessary carrier-grade features with a dedicated focus on: load-balancing, cybersecurity, auto-scaling, self-healing, and inter-domain smart connectivity services. Load-balancing allows the distribution of flow and slice requests among the micro-services component replicas. The Cybersecurity component provides AI/ML-based mechanisms to detect network intrusions and harmful connections, and it provides countermeasures to security incidents. Moreover, the Cybersecurity component will be able to protect itself against adversarial attacks that try to spoof the detector's ML components. The Auto-Scaling component focuses on the autonomous replication of micro-services to support high numbers of incoming requests. The Self-Healing component monitors micro-services and per-flow status to apply healing mechanisms (e.g., component restart, flow redirection) both from a control and a data plane perspective. Finally, the Inter-Domain micro-service allows the interaction of a TeraFlow OS instance with peer TeraFlow OS instances which manage different network domains.

6.2. Per Component Template

We have prepared the following template to provide the detailed information per component. This allows to detail the architectural design of the component and is used in the subsequent sections.

Name:	
Objective:	
Requirements:	
References:	
Responsible (and collaborators):	
Provided Operations:	(outcome1, outcome2) **operation** (input1, input2)

Internal models:	

6.3. Detailed Architecture

In this section we describe the components, organized as CoreApps and netApps.

Reference to protocol buffers available at: https://gitlab.com/teraflow-h2020/controller

6.3.1. CoreApps

CoreApps involve the following components: Context, Monitoring, Device, Service, Automation, Policy, Slice, and DLT.

6.3.1.1. Context

Name:	**Context**
Objective:	Stateful record of the necessary information. Provide the external NBI to obtain and manipulate TeraFlow status.
Requirements:	Shall include a non-SQL database (e.g., MongoDB). Internal interface shall be provided in gRPC. Shall provide NBI for external apps in REST HTTP.
References:	
Responsible (and collaborators):	CTTC, TID

Provided Operations:	ListContextIds(Empty) returns(ContextIdList) {}
	ListContexts(Empty) returns(ContextList) {}
	GetContext(ContextId) returns(Context) {}
	SetContext(Context) returns(ContextId) {}
	RemoveContext(ContextId) returns(Empty) {}
	GetContextEvents(Empty) returns(stream ContextEvent) {}
	ListTopologyIds(ContextId) returns(TopologyIdList) {}
	ListTopologies(ContextId) returns(TopologyList) {}
	GetTopology(TopologyId) returns(Topology) {}
	SetTopology(Topology) returns(TopologyId) {}
	RemoveTopology(TopologyId) returns(Empty) {}
	GetTopologyEvents(Empty) returns(stream TopologyEvent) {}
	ListDeviceIds(Empty) returns(DeviceIdList) {}
	ListDevices(Empty) returns(DeviceList) {}
	GetDevice(DeviceId) returns(Device) {}
	SetDevice(Device) returns(DeviceId) {}
	RemoveDevice(DeviceId) returns(Empty) {}
	GetDeviceEvents(Empty) returns(stream DeviceEvent) {}
	ListLinkIds(Empty) returns(LinkIdList) {}
	ListLinks(Empty) returns(LinkList) {}
	GetLink (LinkId) returns(Link) {}
	SetLink (Link) returns(LinkId) {}
	RemoveLink(LinkId) returns(Empty) {}

	GetLinkEvents(Empty) returns(stream LinkEvent) {}
	ListServiceIds(ContextId) returns(ServiceIdList) {}
	ListServices(ContextId) returns(ServiceList) {}
	GetService(ServiceId) returns(Service) {}
	SetService(Service) returns(ServiceId) {}
	RemoveService(ServiceId) returns(Empty) {}
	GetServiceEvents(Empty) returns(stream ServiceEvent) {}

Internal models:	Topology Device Link

Figure 6 is difficult to read, to this means it is available at: https://gitlab.com/teraflow-h2020/controller/-/blob/develop/proto/uml/context.png

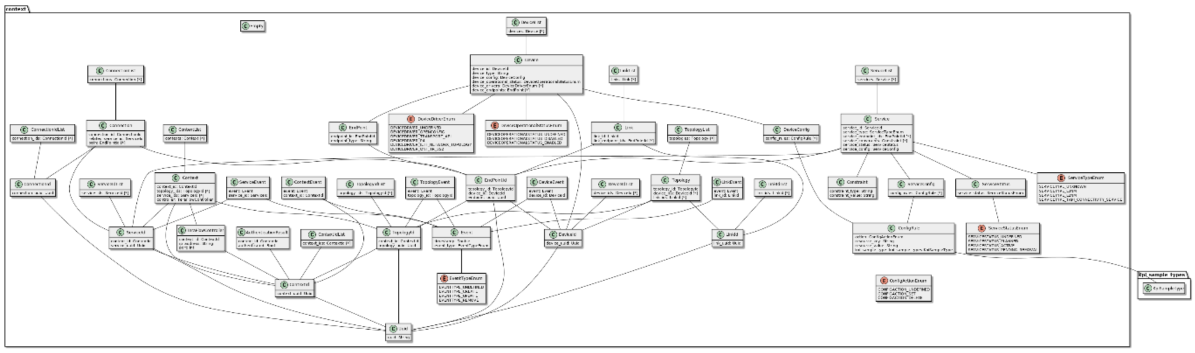


Figure 6 Context data model

6.3.1.2. Monitoring

The initial architecture of the TeraFlow monitoring component is depicted in Figure 7, it is formed by two main blocks: the Monitoring Core and the Metrics Database. Within the Monitoring Core four submodules can be found, the Subscription Manager, the Retriever, the Exporter and the Management Database.

The Retriever is in charge of pulling metrics from all the different monitored components, different components will implement different protocols/interfaces for the KPI provisioning, for that reason we have included a set of submodules within the retriever that will connects to the monitored elements using the required protocols.

All the extracted data will be stored in the Metrics Database, providing dimensional data with time series and key-value pairs in addition to powerful querying and aggregation mechanisms for accessing the collected data. The Metrics Database has direct integration with a data visualization tool offering a wide variety of visualization panels and real time graphics for better observability.

The Exporter submodule will be in charge of serving the collected metrics and KPIs to the different subscribers. Alongside, the Exporter submodule will also handle the configured alarms for triggering notifications based on thresholds and ranges.

The Monitoring Component will handle the subscriptions from other components, through the Subscription Manager submodule located in the Monitoring Core. The Subscription Manager will rely on the Management Database for maintaining the subscription registry in addition to the association between metrics and KPIs.

The following lines describe a more detailed view of the TeraFlow Monitoring component outlining the objectives, requirements, provided operations, and internal models.

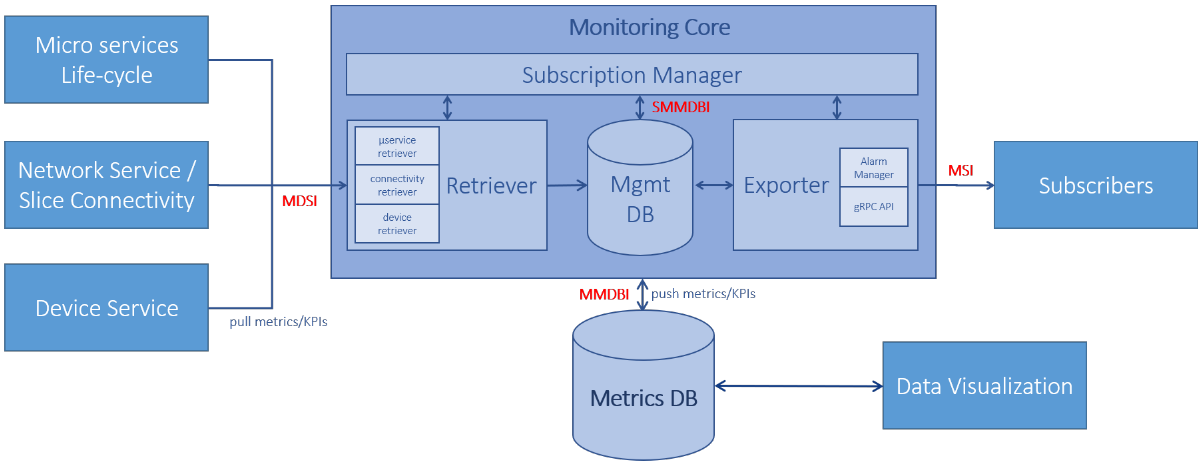


Figure 7 TeraFlow monitoring component initial architecture

Name:	**Monitoring**
Objective:	Provide sufficient information about network metrics (KPIs) and other relevant metrics to assist the life-cycle automation and high performance of the components.
Requirements:	Shall be able to monitor multiple KPIs. Shall be able to allow external subscriptions to the notification service. Shall be able to add, modify and visualize topology, connectivity and microservice life-cycle metrics.
References:	
Responsible(and collaborators):	ATOS
Provided Operations:	CreateKpi(GetKpiDescriptor) returns(KpiId) IncludeKpi (Kpi) returns () MonitorKpi (MonitorKpiRequest) returns () GetKpiDescriptor (KpiId) returns (KpiDescriptor) GetStreamKpi (KpiId) returns (stream Kpi) GetInstantKpi (KpiId) returns (Kpi)
Internal models: (see for reference Figure 8)	KpiDescriptor MonitorKpiRequest KpiId Kpi KpiValue KpiList

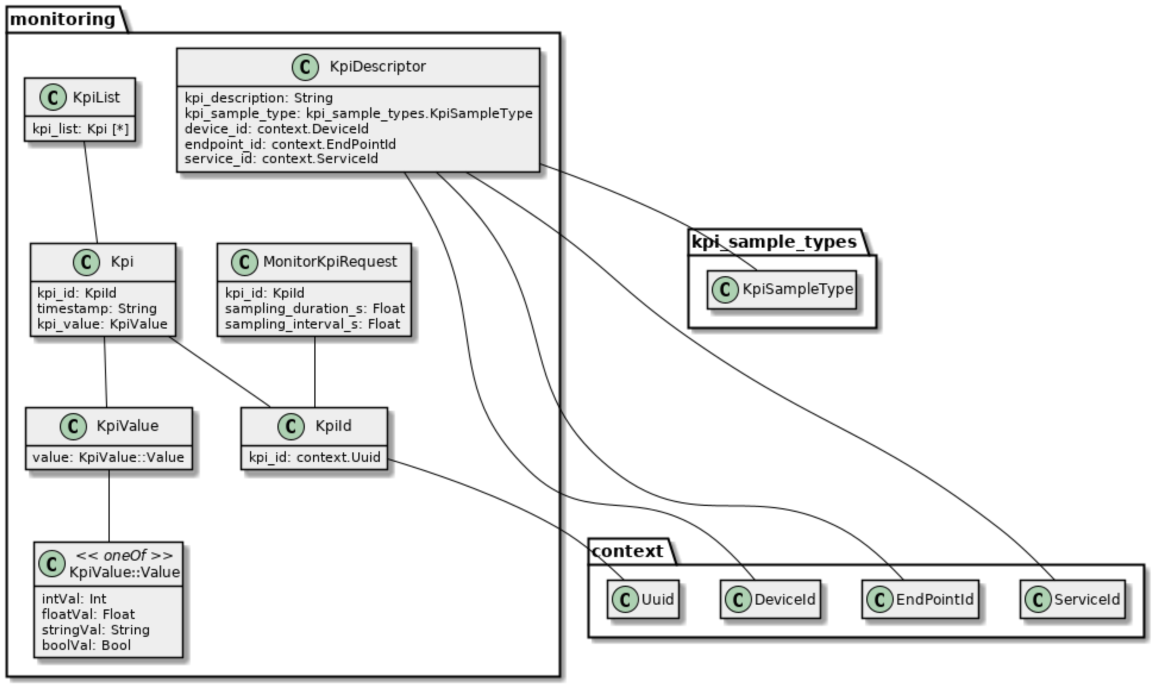


Figure 8 Monitoring data model

6.3.1.3. Device Manager

Name:	Device
Objective:	Provide inventory information and allow to configure and manage specific devices.
Requirements:	Shall be able to handle multiple types of devices. Shall be able to perform configuration and management
References:	OpenConfig, IETF
Responsible (and collaborators):	TID, SIAE, INF, VOL, UBI

Provided Operations:	AddDevice(Device) return (DeviceId)
	ConfigureDevice(DeviceConf) return (DeviceId)
	DeleteDevice(DeviceId) return (Empty)

Internal models:	Only provides operations. Data models belong to context

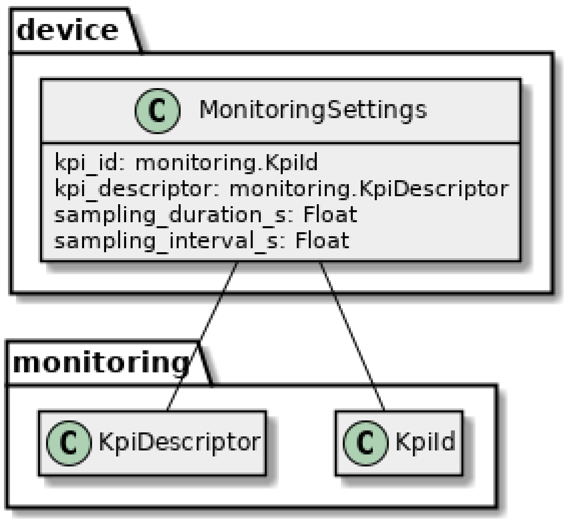

device
MonitoringSettings
kpi_id: monitoring.KpiId
kpi_descriptor: monitoring.KpiDescriptor
sampling_duration_s: Float
sampling_interval_s: Float
monitoring
KpiDescriptor
KpiId

Figure 9 Device internal data model

6.3.1.4. Service

Name:	**Service**
Objective:	Lifecycle management of multiple TeraFlow services.
Requirements:	Provide L3NM, L2NM and ONF Transport API lifecycle management support.
References:	
Responsible (and collaborators):	CTTC, INF, TID

Provided Operations:	CreateService (context.Service) returns (context.ServiceId) {}
	UpdateService (context.Service) returns (context.ServiceId) {}
	DeleteService (context.ServiceId) returns (context.Empty) {}
	GetConnectionList(context.ServiceId) returns (context.ConnectionList) {}

Internal models:	

6.3.1.5. Traffic Engineering

Name:	**Traffic Engineering**
Objective:	Manage Segment routing LSPs
Requirements:	• Create, modify, and delete segment routing LSPs • Should interact with the terraflow-os-context component to retrieve device and topology information. • Should interact with the teraflow-os-device component to configure compatible hardware to join the PCE.
References:	

Responsible (and collaborators):	STR

Provided Operations:	RequestLSP(service.Service) returns (service.ServiceState)
	UpdateLSP(service.ServiceId) returns (service.ServiceState)
	DeleteLSP(service.ServiceId) returns (context.Empty)

Internal models:	N/A

This component implements the creation, modification, and deletion of segment routing LSPs on the available hardware, considering the given constraints and the available resources. The constraints given to the PCE for the calculation of the LSP could be required or desired latency, bandwidth consumption and hop count, and if the result should be a strict explicit path or a loose one. Figure 10 shows the internal design of the component and how it relates with other TeraFlow components.

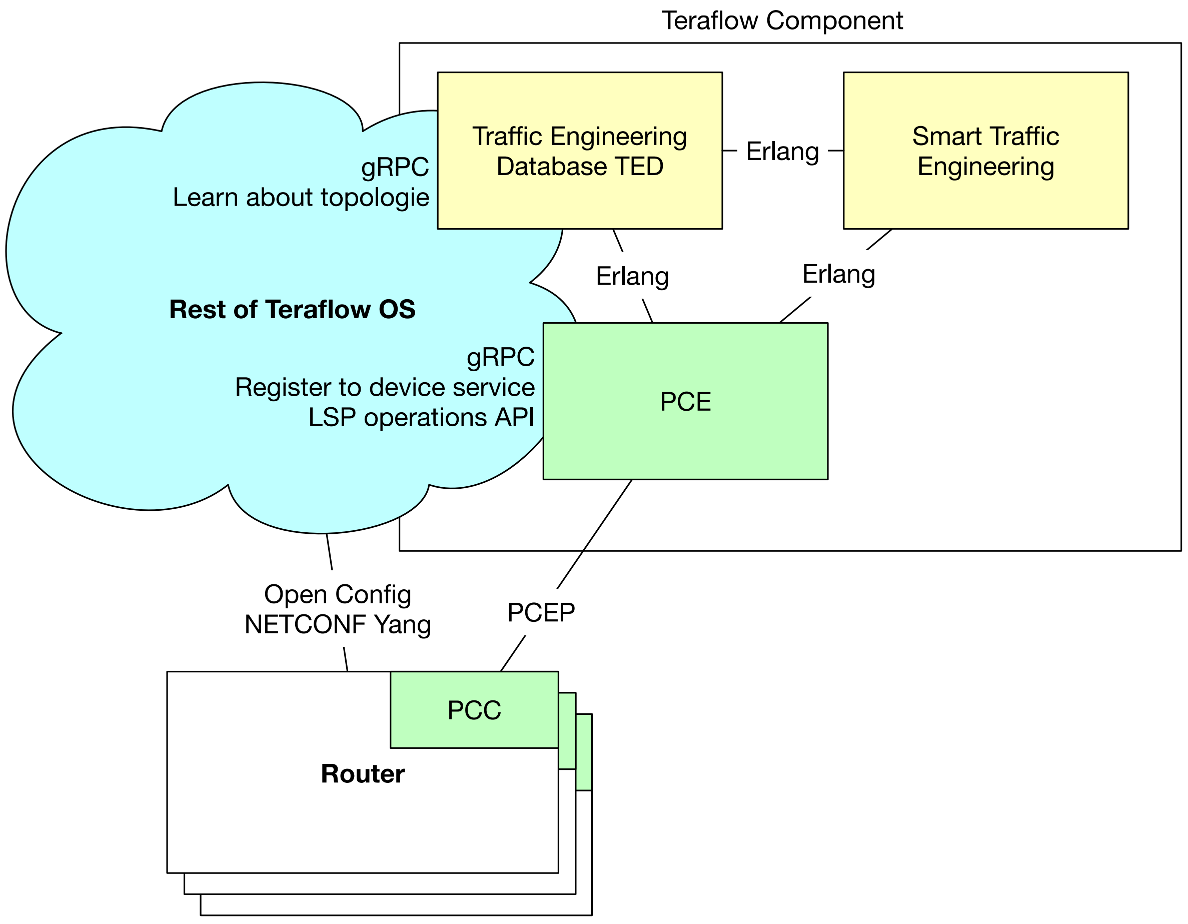


Figure 10 Traffic Engineering component

6.3.1.6. Automation (ZTP)

Name:	**Automation**

Objective:	Automatically add/update a physical or virtual device to/in the network with zero manual intervention or local configuration, while ensuring that the correct certificates, software, device configuration parameters, and device pipeline definition are installed. This component produces role-based device configuration, which is sent to the teraflow-os-config component via a southbound configuration protocol (e.g., gNMI or NETCONF).
Requirements:	• Shall be able to handle multiple types of devices. ○ Requires interactions with the teraflow-os-device component. • Shall be able to receive and validate a DeviceRole from its northbound interface ○ Such role-based configuration may be received by an external entity, either automatically or manually by a network operator. ○ A gRPC protocol buffer model may describe this API. • Shall be able to associate a valid *DeviceRole* with a physical or virtual device in the network ○ This is internal process once a northbound API call is received and before it is communicated to the configuration component. • Shall be able to send a DeviceRole to the configuration component through a southbound API. ○ Requires interactions with the teraflow-os-config component. ○ A gNMI API may be used for this inter-component communication. • Shall be able to perform device management. ○ Requires interactions with the teraflow-os-control component. ○ For table-based devices, a table pipeline setup is required. • Shall be able to monitor the behaviour of the underlying devices.
References:	
Responsible (and collaborators):	UBI, UPM, TID, TNOR

Provided Operations:	ZtpGetDeviceRole(DeviceRoleId) returns (DeviceRole) {}
	ZtpGetDeviceRolesByDeviceId(context.DeviceId) returns (DeviceRoleList) {}
	ZtpAdd(DeviceRole) returns (DeviceRoleState) {}
	ZtpUpdate(DeviceRole) returns (DeviceRoleState) {}
	ZtpDelete(DeviceRole) returns (DeviceRoleState) {}
	ZtpDeleteAll() returns (DeviceDeletionResult) {}

Internal models:	The main model of this component is the DeviceRole, which contains additional internal models, while exploiting models provided by other components (see for reference Figure 12).

Figure 11 shows the internal design of Automation component.

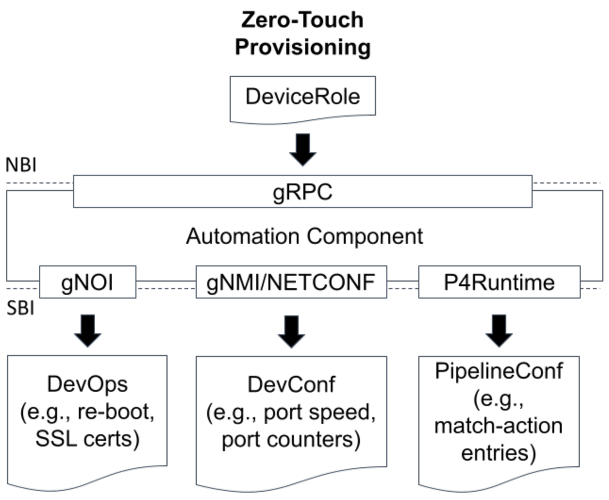

Zero-Touch Provisioning
DeviceRole
NBI
gRPC
Automation Component
gNOI
gNMI/NETCONF
P4Runtime
SBI
DevOps (e.g., re-boot, SSL certs)
DevConf (e.g., port speed, port counters)
PipelineConf (e.g., match-action entries)

Figure 11 Automation component internal architecture.

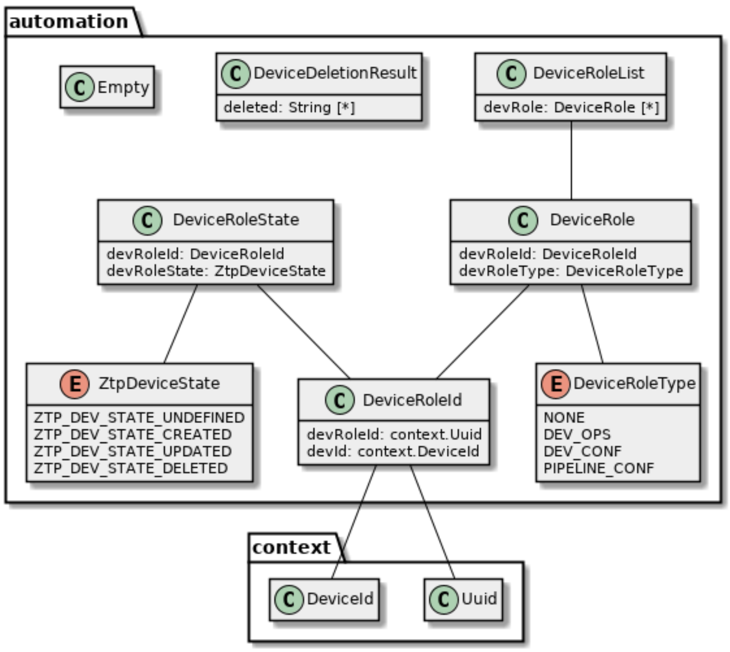


Figure 12 Automation data model.

6.3.1.7. Policy Manager

Name:	**Policy**
Objective:	Automatically translates high-level policy rules to actual configuration applied to either physical or virtual devices. A policy rule may generate configuration across an entire network domain, thus may need to configure multiple devices per policy. The component architecture is depicted in Figure 13.
Requirements:	• Shall be able to handle multiple types of devices. ○ Requires interactions with the teraflow-os-device component. • Shall be able to perform device management. ○ Requires interactions with the teraflow-os-device component. ○ For table-based devices, a table pipeline setup is required.

	• Shall be able to receive and validate a PolicyRule from its northbound interface. ○ Such policy rule may be received by an external entity, either automatically or manually by a network operator. ○ A gRPC protocol buffer model may describe this API. • Shall be able to verify that a given policy matches to a set of flows. ○ Requires interactions with the teraflow-os-context component to acquire the active flows. • Shall be able to enforce a policy. ○ Requires interactions with the teraflow-os-context component to install/update/delete flows according to a policy.
References:	
Responsible (and collaborators):	UBI, ODC, UPM, TID, TNOR

Provided Operations:	PolicyAdd (PolicyRule) returns (PolicyRuleState) {}
	PolicyUpdate (PolicyRule) returns (PolicyRuleState) {}
	PolicyDelete (PolicyRule) returns (PolicyRuleState) {}
	GetPolicy (PolicyRuleId) returns (PolicyRule) {}
	GetPolicyByDeviceId (context.DeviceId) returns (PolicyRuleList) {}
	GetPolicyByServiceId (service.ServiceId) returns (PolicyRuleList) {}

Internal models:	The main model of this component is the PolicyRule, which contains additional internal models, while exploiting models provided by other components (see for reference Figure 14).

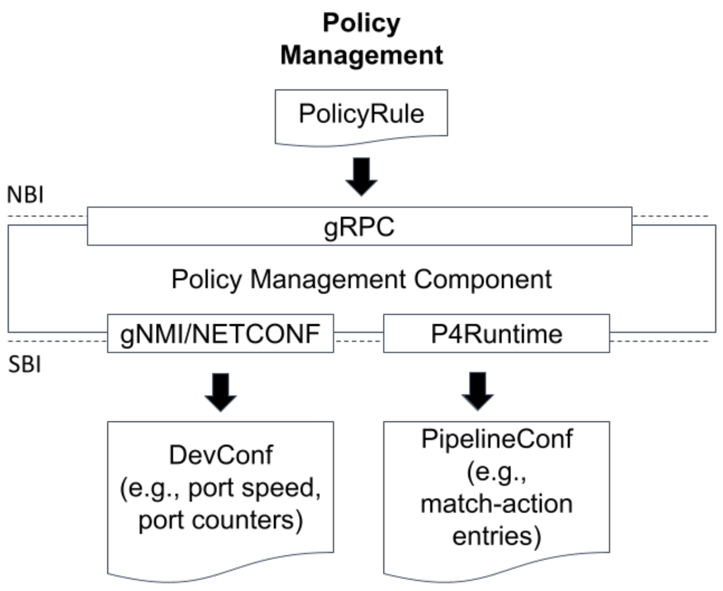

Policy Management
PolicyRule
NBI
gRPC
Policy Management Component
gNMI/NETCONF
P4Runtime
SBI
DevConf (e.g., port speed, port counters)
PipelineConf (e.g., match-action entries)

Figure 13 Policy component internal architecture.

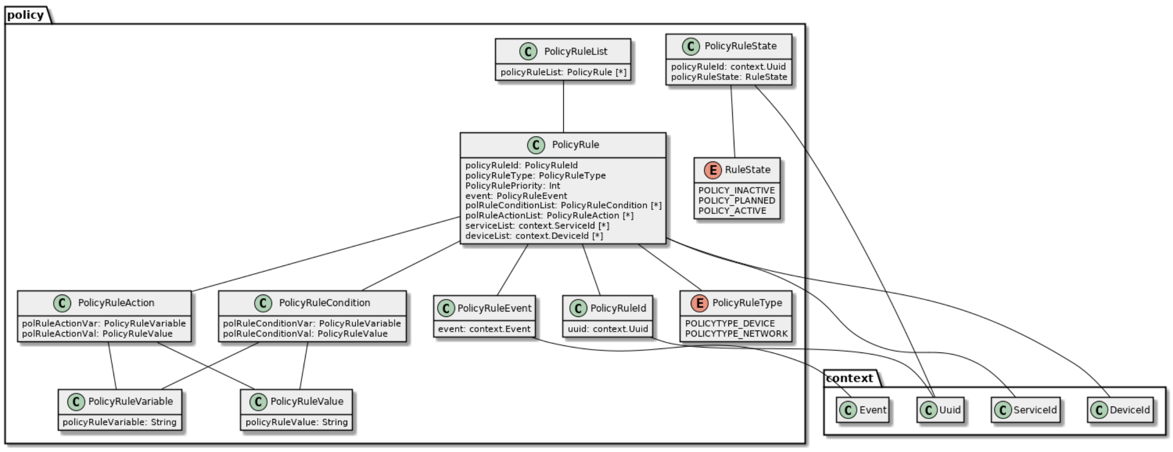

policy
PolicyRuleList
policyRuleList: PolicyRule [*]
PolicyRuleState
policyRuleId: context.Uuid
policyRuleState: RuleState
PolicyRule
policyRuleId: PolicyRuleId
policyRuleType: PolicyRuleType
PolicyRulePriority: Int
event: PolicyRuleEvent
polRuleConditionList: PolicyRuleCondition [*]
polRuleActionList: PolicyRuleAction [*]
serviceList: context.ServiceId [*]
deviceList: context.DeviceId [*]
RuleState
POLICY_INACTIVE
POLICY_PLANNED
POLICY_ACTIVE
PolicyRuleAction
polRuleActionVar: PolicyRuleVariable
polRuleActionVal: PolicyRuleValue
PolicyRuleCondition
polRuleConditionVar: PolicyRuleVariable
polRuleConditionVal: PolicyRuleValue
PolicyRuleEvent
event: context.Event
PolicyRuleId
uuid: context.Uuid
PolicyRuleType
POLICYTYPE_DEVICE
POLICYTYPE_NETWORK
PolicyRuleVariable
policyRuleVariable: String
PolicyRuleValue
policyRuleValue: String
context
Event
Uuid
ServiceId
DeviceId

Figure 14 Policy data model.

6.3.1.8. Slice Manager

Name:	**Slice**
Objective:	Handle Transport Network Slice lifecycle
Requirements:	[REQ-SLI-01] - [REQ-SLI-06]
References:	draft-ietf-teas-ietf-network-slices-03 draft-liu-teas-transport-network-slice-yang
Responsible (and collaborators):	(VOL), TNOR, UBI

Provided Operations:	CreateUpdateSlice (TransportSliceRequest) returns (SliceId) {}
	DeleteSlice (TransportSliceRequest) returns () {}

Internal models:	TransportSliceRequest (see for reference Figure 15).

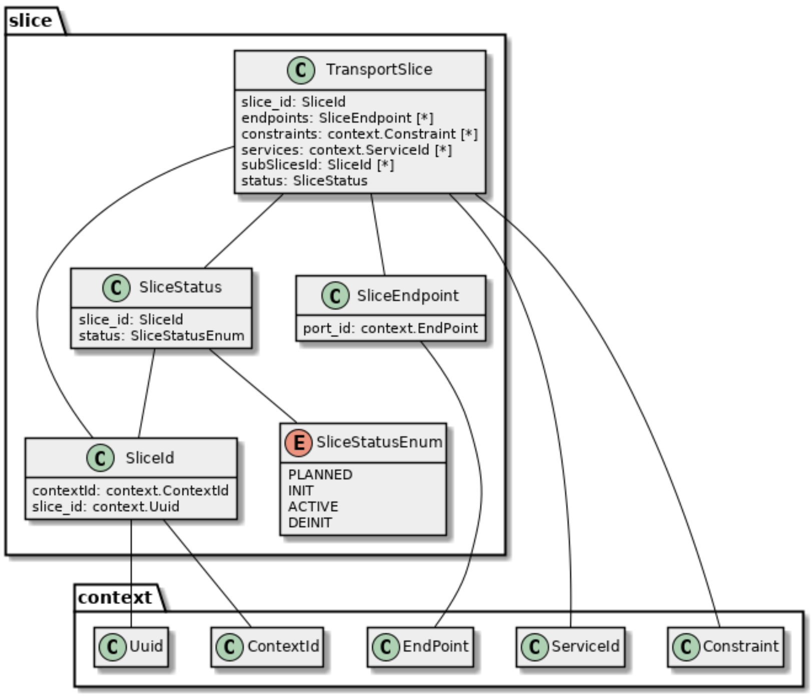


Figure 15 Slice data model

6.3.1.9. Distributed Ledger

Name:	**Distributed Ledger**
Objective:	Provide distributed ledger to record, query, and process relevant data for network management and detection of compromised edge-devices.
Requirements:	[REQ-DLT-01]--[REQ-DLT-04]
References:	
Responsible (and collaborators):	NEC, CTTC

Provided Operations:	RecordToDlt (DltRecord) returns (RecordStatus) {}
	GetFromDlt (DltRecordId) returns (DltRecord) {}

Internal models:	DltRecord, DltStatus (see for reference Figure 16).

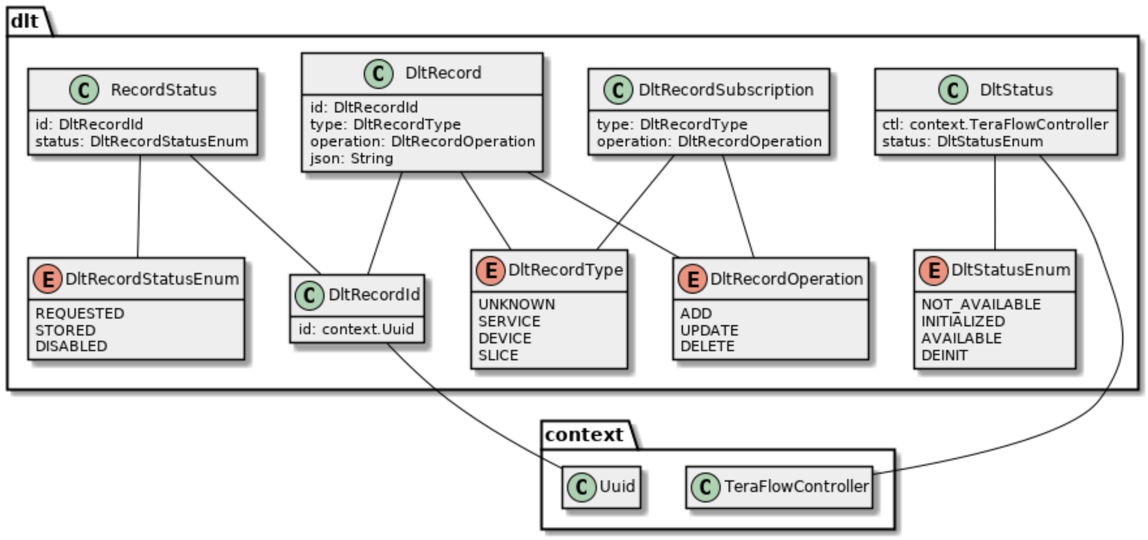


Figure 16 DLT component data model

6.3.2. netApps

NetApps involve the following components: Optical Centralized Attack Detector, Optical Attack Mitigation Processor, Attack Inference, L3 Centralized Attack Detector, L3 Distributed Attack Detector, L3 Fast Attack Mitigation Processor, Compute, and Inter-domain.

6.3.2.1. Optical Centralized Attack Detector

Name:	**Centralized Attack Detector**
Objective:	Provide physical layer attack detection at the optical layer and provide a consolidated attack detection mechanism based on the distributed attack detector.
Requirements:	• Shall consume/subscribe to security-related data from the TeraFlow Monitoring core component. • Shall process the optical KPIs (i.e., optical performance monitoring data) from the monitoring component and generate a security status of the optical channels. • Shall report detected attacks to the optical attack mitigation processor component. • Shall report security status to the TeraFlow Monitoring core component.
References:	
Responsible (and collaborators):	CHA

Provided Operations:	rpc **DetectAttack** (Empty) returns (Empty) {}
	rpc **NotifyServiceUpdate** (Service) returns (Empty) {}
	rpc **ReportSummarizedKpi** (KpiList) returns (Empty) {}
	rpc **ReportKpi** (KpiList) returns (Empty) {}

Internal models:	Only provides operations. Data models belong to service, monitoring, and optical attack mitigation processor.

6.3.2.2. Optical Attack Mitigation Processor

Name:	**Optical Attack Mitigation Processor**
Objective:	Compute attack remediation solutions based on the physical layer attack detection performed by other components.
Requirements:	• Shall receive attack notifications (e.g., from the optical centralized attack detection component); • Shall compute contextual attack responses; • Shall communicate with the Service, Automation & Policy Manager components to perform attack countermeasures.
References:	
Responsible (and collaborators):	CHAL
Provided Operations:	rpc **NotifyAttack** (AttackDescription) returns (AttackResponse) {}
Internal models:	This component contains two models: AttackDescription and AttackResponse (see for reference Figure 17).

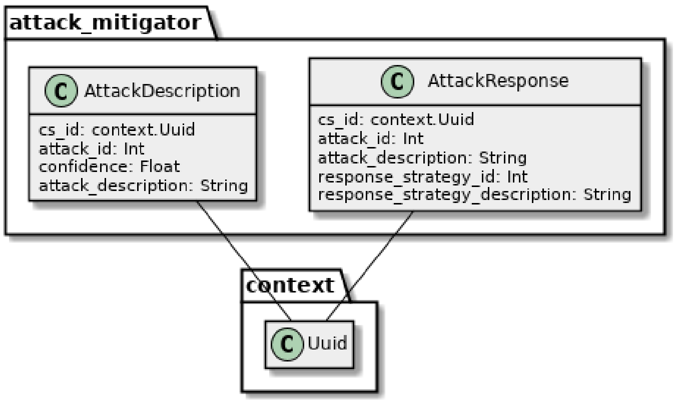


Figure 17 Optical attack mitigator data model

6.3.2.3. Optical Attack Inference

Name:	**Attack Inference**
Objective:	Performs the anomaly detection inference based on a set of samples. The implementation currently uses an unsupervised learning algorithm for anomaly detection (i.e., DBSCAN) that is used to detect attacks based on optical performance monitoring data. The design of this component is inspired by the design of the TensorFlow Serving component.
Requirements:	• Shall receive a set of samples (composed of a set of feature values) representing a monitoring window (defined by the centralized attack detector) over which the anomaly detection algorithm is run;

	• Shall return the cluster indices for each one of the samples, where the index -1 represents an anomaly, and cluster indices greater or equal to zero are considered normal samples.
References:	• DBSCAN Serving • TensorFlow Serving
Responsible (and collaborators):	CHAL
Provided Operations:	rpc **Detect** (DetectionRequest) returns (DetectionResponse) {}
Internal models:	This component contains four models: Metric, Sample, DetectionRequest and DetectionResponse (see for reference Figure 18).

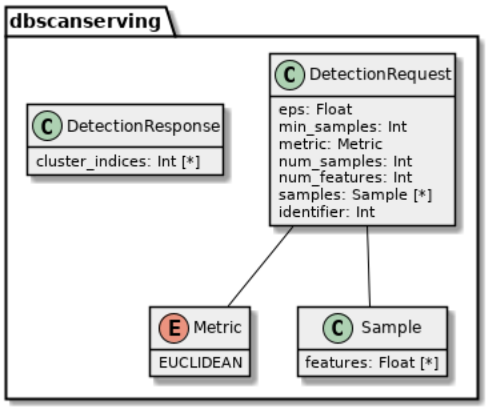


Figure 18 Attack inference data model

6.3.2.4. L3 Centralized Attack Detector

Name:	**L3 Centralized Attack Detector**
Objective:	This component provides attack detection capabilities at the IP layer and a consolidated attack detection mechanism based on the Distributed Attack Detector. The Centralized Attack Detector utilizes machine learning algorithms to classify the input data received from the Distributed Attack Detector component and sends the predictions to the Attack Mitigator.
Requirements:	• Shall consume/subscribe to security-related data from the TeraFlow Monitoring core component. • Shall process the summarized flow KPIs from the monitoring component and generate a consolidated data plane security status. • Shall report detected attacks to the fast attack mitigation processor component. • Shall report security status to the TeraFlow Monitoring core component.
References:	
Responsible (and collaborators):	UPM, TID, NEC

Provided Operations:	rpc **SendInput** (L3CentralizedattackdetectorMetrics) returns (Empty) {}
	rpc **GetOutput** (Empty) returns (L3CentralizedattackdetectorModelOutput) {}

Internal models:	L3CentralizedAttackDetectorModelOutput and L3CentralizedAttackDetectorMetrics (see for reference Figure 19 and Figure 20).

The L3 Centralized Attack Detector uses the l3_centralizedattackdetector.proto file for the protobuf messages which correspond to the following models.

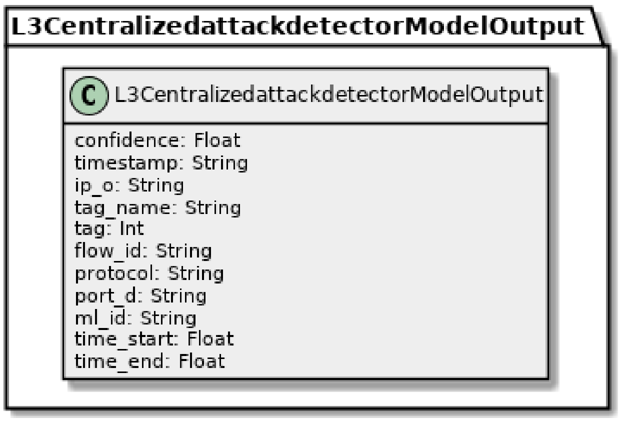


Figure 19 L3 Centralized Attack Detector Model Output

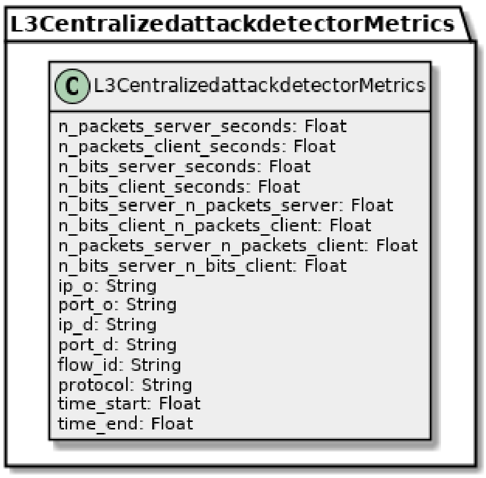

L3CentralizedattackdetectorMetrics
C L3CentralizedattackdetectorMetrics
n_packets_server_seconds: Float
n_packets_client_seconds: Float
n_bits_server_seconds: Float
n_bits_client_seconds: Float
n_bits_server_n_packets_server: Float
n_bits_client_n_packets_client: Float
n_packets_server_n_packets_client: Float
n_bits_server_n_bits_client: Float
ip_o: String
port_o: String
ip_d: String
port_d: String
flow_id: String
protocol: String
time_start: Float
time_end: Float

Figure 20 L3 Centralized Attack Detector internal data model

6.3.2.5. L3 Distributed Attack Detector

Name:	**Distributed Attack Detector**
Objective:	Detect attacks at remote sites (network edge) in a distributed fashion and classify them. The focus of attack detection in this component is the data plane attacks by analysing packets.
Requirements:	• Shall receive detailed monitoring data from packet processor devices.

	• Shall generate summarized flow KPIs and send them to appropriate/subscribed components such as the centralized attack detector. • Shall report detected attacks to the fast attack mitigation processor component.
References:	
Responsible (and collaborators):	UPM, TID
Provided Operations:	This component does not deploy a gRPC server.
Internal models:	Only provides operations. Data models belong to service, monitoring, and fast attack mitigation processor.

6.3.2.6. L3 Attack Mitigator

Name:	**L3 Attack Mitigation Processor**
Objective:	This component is responsible for computing viable attack remediation solutions, depending on the detected attack by other components. It receives per connection information from the Centralized attack detector. In the current version, the attack mitigator has only been tailored to communicate correctly with the Automation component and a placeholder has been crafted in place of the mitigation strategy (i.e., a print trace).
Requirements:	• Shall consume/subscribe to security-related data from the TeraFlow Monitoring core component. • Shall receive attack notifications from the centralized and distributed attack detection components. Currently only receives from the distributed attack detection component. • Shall communicate with the Automation & Policy Manager component to perform attack countermeasures. This is not yet implemented.
References:	
Responsible (and collaborators):	UPM, TID, NEC

Provided Operations:	rpc **SendOutput** (L3AttackmitigatorOutput) returns (context.Empty) {}
	rpc **GetMitigation** (context.Empty) returns (context.Empty) {}

Internal models:	L3AttackMitigatorOutput (see for reference Figure 21).

The L3 Attack Mitigator uses the l3_attackmitigator.proto file for the protobuf messages which correspond to the following model.

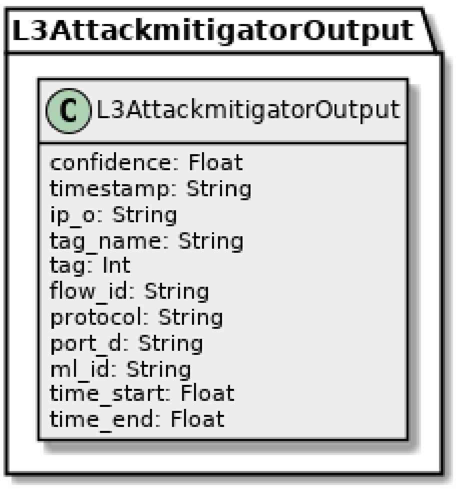


Figure 21 L3 Attack Mitigator Output data model

6.3.2.7. Compute integration

Name:	**Compute**
Objective:	It allows ETSI OpenSource MANO (OSM) SDN/WIM connector. It implements the standard IETF RFC 8466 "A YANG Data Model for Layer 2 Virtual Private Network (L2VPN) Service Delivery". It provides the endpoints and the necessary details to request the Layer 2 service. It supports L2VPN lifecycle management.
Requirements:	Allow OSM to perform the necessary WIM operations.
References:	RFC 8466 (ietf-l2vpn-svc.yang)
Responsible (and collaborators):	CTTC

Provided Operations:	rpc CheckCredentials (context.TeraFlowController) returns (context.AuthenticationResult) {}
	rpc GetConnectivityServiceStatus (context.ServiceId) returns (context.ServiceStatus) {}
	rpc CreateConnectivityService (context.Service) returns (context.ServiceId) {}
	rpc EditConnectivityService (context.Service) returns (context.ServiceId) {}
	rpc DeleteConnectivityService (context.Service) returns (context.Empty) {}
	rpc GetAllActiveConnectivityServices (context.Empty) returns (context.ServiceIdList) {}
	rpc ClearAllConnectivityServices (context.Empty) returns (context.Empty) {}

Internal models:	Only provides operations. Data models belong to context and service.

6.3.2.8. Inter-Domain

Name:	**Inter-domain**

Objective:	Enable interaction of a TeraFlow OS instance with peer TeraFlow OS instances which manage different network domains to create E2E TN slicing services.
Requirements:	[REQ-INT-01] - [REQ-INT-09] [align more with current capabilities outlined in the workflows]
References:	IETF: https://datatracker.ietf.org/doc/rfc8299 , https://datatracker.ietf.org/doc/rfc8453/ https://datatracker.ietf.org/doc/draft-ietf-teas-ietf-network-slices/02/
Responsible (and collaborators):	TNOR, NTNU, CTTC
Provided Operations:	• Authenticate (context.TeraFlowController) returns (context.AuthenticationResult) {} • LookUpSlice(slice.TransportSliceRequest) returns (SliceResponse) {} //Slice component or from interdomain component • OrderSliceFromCatalog(slice.TransportSliceRequest) returns (SliceResponse) {} • CreateSliceAndAddToCatalog(slice.TransportSliceRequest) returns (SliceResponse) {}
Internal models:	Only provides operations. Data models belong to context and service.

The inter-domain component allows the interaction of a TeraFlow OS instance with peer TeraFlow OS instances which manage different network domains. The inter-domain communication services can be classified into two types: inter-domain between administrative domains vs. inter-domain between technology domains (within one administrative domain), as shown in Figure 23. Depending on the type, different governance models are needed before defining the micro-services and corresponding requirements. In this section, we focus on Type I (Figure 23).

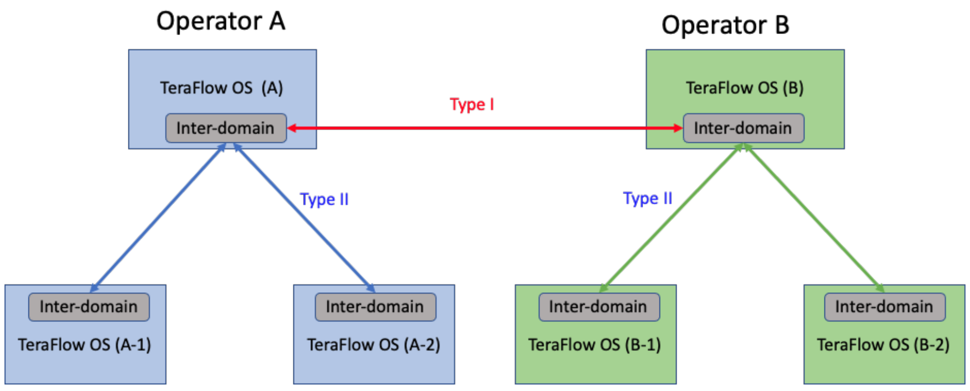


Figure 22 Different types of inter-domain communication.

To achieve objectives, the proposed inter-domain component leverages existing data models related to the context and slice components. The main operations include the authentication required prior to the information exchange between inter-domain components as well as interaction with the slice catalogue. Interactions with the slice catalogue include querying the catalogue (LookUpSlice), requesting the instantiation of an existing slice (OrderSliceFromCatalog), and the creation of a new slice as well as its inclusion into the catalogue (CreateSliceAndAddToCatalog).

6.3.3. Cloud Orchestrator Features

In this section, we present several features that are not provided by any component, but they are related to the selected technology (Kubernetes). These features are: auto-scaling, self-healing, and load balancing.

6.3.3.1. Auto-Scaling

The Auto-Scaling component focuses on the autonomous replication of micro-services to support high amount of load in terms of incoming requests.

The Horizontal Pod Autoscaler (HPA) automatically scales the number of Pods in a Pod deployment based on observed metrics (e.g., CPU utilization). Through integration with external mechanisms, it can also support autoscaling using custom metrics support, such as application-provided metrics.

6.3.3.2. Self-Healing

The Self-Healing component monitors micro-services and per-flow status to apply healing mechanisms (e.g., component restart, flow redirection) both from a control and a data plane perspective.

We will use two features from Kubernetes for providing self-healing mechanisms: liveness and readiness probes. The kubelet uses liveness probes to know when to restart a container. The kubelet uses readiness probes to know when a container is ready to start accepting traffic. Both features will be included in each Pod by usage of the Google Health grpc [GRPC21].

6.3.3.3. Load Balancing

Load-balancing allows the distribution of flow and slice requests among the micro-services component replicas.

Kubernetes implements load balancing as a load distribution mechanism. It uses two methods of load distribution, which are easy to operate through the kube-proxy feature.

7. Data Models

This section describes the selected data models to be used as both external (including NorthBound and SouthBound) and internal interfaces.

7.1. External Interfaces and Data Models

The external interfaces are the ones offered to be consumed from external components of TeraFlow and typically use NETCONF or RESTCONF protocols.

7.1.1. NorthBound Interfaces (NBI)

Several interfaces are proposed to be used as NorthBound Interfaces, being summarized in L3/L2 network models, Traffic Engineering (TE) tunnels, and IETF Transport Network Slices.

7.1.1.1. IETF L3/L2 Network Model

Network Models [BAR21a] are used to interact with the network controllers. Such models are used to instantiate the service in the network. Network models are used to provide an intermediate level of abstraction between what the customer requires, and the configuration implemented in the underlying network nodes. It provides the notion of a network service, which really is a set of configurations in the devices that happen to realize the customer needs. It is important to highlight that the services of the network models only exist in the controller. In the network nodes, protocols, routing instances, tunnels, routing profiles, access control lists, etc., for example, are configured to implement such service. Also, note that due to the separation of roles, the assignment of resources, for example, selecting the right interface for a customer or choosing an address or taking the decision of giving or not certain bandwidth is a role of the service orchestration layer. The network models are a good base to define the Resource-Facing services used by OSS applications.

The L3NM model provides a network-centric view of L3VPN services [BARb]. L3NM is meant to be used by a Network Controller to derive the configuration information that will be sent to relevant network devices.

The L2NM YANG module provides representation of the Layer 2 VPN Service from a network standpoint [BAR21c]. The module is meant to be used by a Network Controller to derive the configuration information that will be sent to relevant network devices.

7.1.1.2. IETF Traffic Engineering Tunnels

TE [SAA21] is a YANG data model for the configuration and management of Traffic Engineering (TE) tunnels, Label Switched Paths (LSPs), and interfaces have been part of the TEAS working group in the IETF. The model is divided into YANG modules that classify data into generic, device-specific, technology agnostic, and technology-specific elements.

7.1.1.3. IETF Transport Network Slice

A transport network slice will consist of a set of endpoints (e.g., CEs), a connectivity matrix between subsets of these endpoints service level behaviours requested for each sender on the connectivity matrix [FAR21]. The connectivity between the endpoints might be point-to-point, point-to-multipoint or multipoint-to-multipoint. Often these slices will be used to satisfy network behaviour defined in a Service Level Agreement (SLA) [FAR21].

Several data models might be considered, but our first implementation and preliminary results might be obtained from [LIU21].

7.1.2. SouthBound Interfaces (SBI)

In this section, the provided SouthBound interface are discussed. More information is provided in D3.1 in Device component architecture description.

7.1.2.1. ONF Transport API Driver

We consider the use of ONF Transport API 2.1 to interact with Open Line System (OLS) Controllers in charge of managing underlying optical transport networks.

7.1.2.2. ONF TR-532 Microwave

We consider the use of a REST API to interact with a SDN Microwave (MW) Domain Controller, which has a network-level view of the underlying MW domain. The SDN MW controller interacts with microwave network elements using ONF TR-532 data models. Details are provided in D3.1.

7.1.2.3. OpenConfig

Routers will be configured and controlled using OpenConfig parameters that are implemented in NETCONF server. It will be supported by DRX-30 IP/MPLS routers.

7.1.2.4. ONOS P4

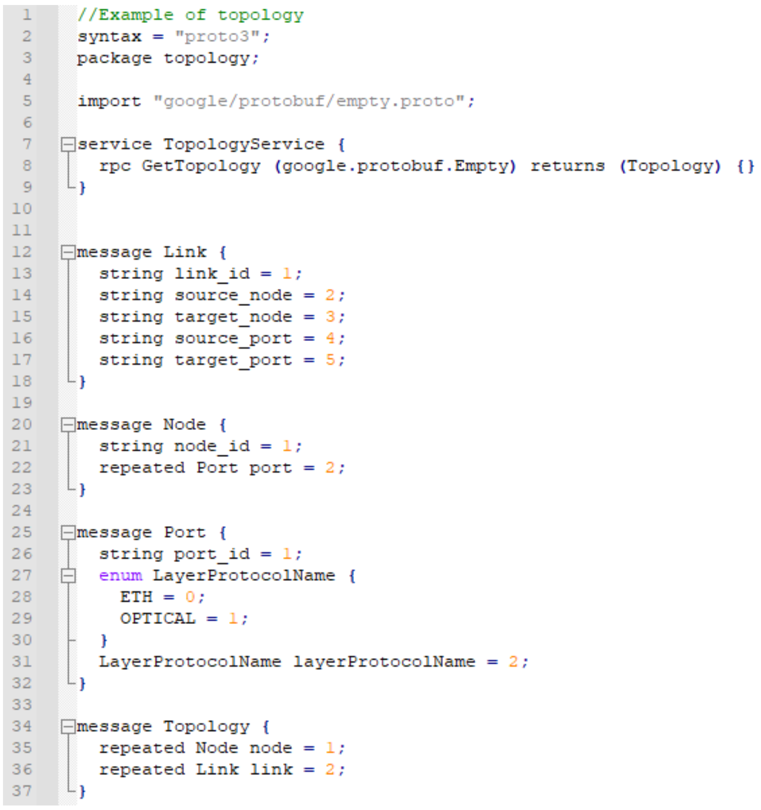
TeraFlow embraces the ONF initiatives around P4 by integrating ONOS, Stratum, and P4 into the TeraFlow ecosystem. In D3.1, we describe how the TeraFlow SDN controller will cooperate with ONOS to manage next-generation SDN whiteboxes based on the P4 paradigm.

7.2. Internal Data Models

Google Remote Procedure Calls (gRPC) [BRE19] is a protocol based on HTTP/2 as a transport protocol and it uses protocol buffers encodings for transported messages. As it is based on HTTP/2 transport protocol and uses byte-oriented encoding, it introduces low latency. gRPC has been used in highly scalable and distributed systems. It has been decided to be used as internal protocol in TeraFlow SDN Controller. Data models for gRPC are described using Protocol Buffers.

Protocol Buffers are a language-neutral, platform-neutral extensible mechanism for serializing structured data [PRO21]. They allow to model the data structures in a similar manner as in YANG. Its encoding in byte-oriented messages increases the efficiency compared to XML/JSON encodings. An Open-Source framework is provided to handle protocol buffers in multiple languages, such as python, Go, C++, Java, Erlang.

Figure 23 provides an example of a topology information model. In this example, a topology service is described, and it offers a GetTopology RPC that provides a topology structure. Each structure is described in message keyword. Topology message provides a list of node messages (a list is declared using repeated keyword) and a list of link messages. A node message includes an identifier and a list of port messages. Port messages include a port identifier and a layer protocol name. Link messages include a link identifier and references to source and target nodes and ports.



```
//Example of topology
syntax = "proto3";
package topology;

import "google/protobuf/empty.proto";

service TopologyService {
  rpc GetTopology (google.protobuf.Empty) returns (Topology) {}
}


message Link {
  string link_id = 1;
  string source_node = 2;
  string target_node = 3;
  string source_port = 4;
  string target_port = 5;
}

message Node {
  string node_id = 1;
  repeated Port port = 2;
}

message Port {
  string port_id = 1;
  enum LayerProtocolName {
    ETH = 0;
    OPTICAL = 1;
  }
  LayerProtocolName layerProtocolName = 2;
}

message Topology {
  repeated Node node = 1;
  repeated Link link = 2;
}
```

Figure 23 Example of protocol buffer

All protocol buffers are part of TeraFlow SDN source code, and they are available at: https://gitlab.com/teraflow-h2020/controller/-/tree/develop/proto

8. Use Case Workflows

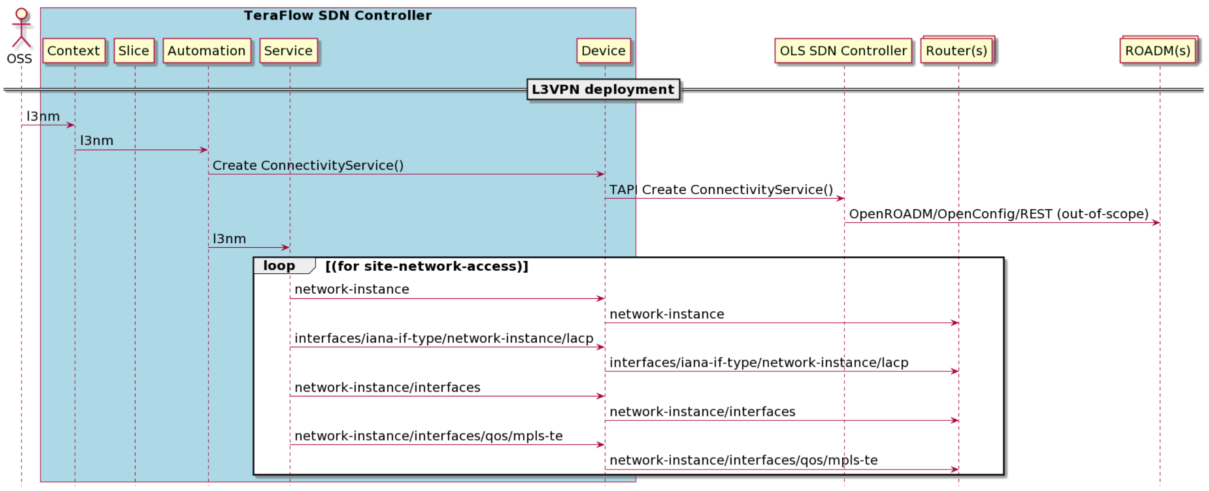
In this section, we provide several specific workflows to demonstrate the interaction among multiple components, as well as the underlying infrastructure devices, other controllers, etc.

8.1. L3VPN Service Provisioning

To configure a L3VPN, the TeraFlow SDN Controller first creates an instance of the VPN service, specifying attributes, import and export profiles, and the list of VPN endpoints that participate in the service. In a second step, the TeraFlow SDN Controller communicates with each of the network elements hosting the endpoints, such as virtual or physical routers, and for each of them:

- configures a VRF
- attaches (logical or physical) interfaces to the VRF
- configures the routing protocols to enable PE-PE and PE-CE communications over the relevant interfaces
- configures any other additional parameters (QoS, etc).

Figure 24 shows an example of this workflow.



TeraFlow SDN Controller
OSS
Context
Slice
Automation
Service
Device
OLS SDN Controller
Router(s)
ROADM(s)
L3VPN deployment
l3nm
l3nm
Create ConnectivityService()
TAPI Create ConnectivityService()
OpenROADM/OpenConfig/REST (out-of-scope)
l3nm
loop
[(for site-network-access)]
network-instance
network-instance
interfaces/iana-if-type/network-instance/lacp
interfaces/iana-if-type/network-instance/lacp
network-instance/interfaces
network-instance/interfaces
network-instance/interfaces/qos/mpls-te
network-instance/interfaces/qos/mpls-te

Figure 24 L3VPN Service Provisioning sequence diagram

8.2. Monitoring

The monitoring component collects metrics from the different components and devices, producing valuable data that can be used by other services and components to improve service performance, detect anomalies, or just for analytic purposes.

Figure 25 depicts an exemplary workflow for monitoring a generic device. This workflow involves different entities: the SDN Agent that is the generic device that will be monitored; the Context component that stores the configuration and attributes of the network elements managed in TeraFlow; the Device component that interfaces with the monitored devices; the monitoring component with its databases (MgmtDB and MetricsDB), the data visualization platform of the monitored data and the user that visualizes these data.

In first place, the Monitoring component starts to listen the Context Management service about new events in the devices, these events will be accessed in the queue *EventQueue* and will be translated to *KpiRequests*. Then, the Monitoring service process and registers each *KpiRequest* as a KPI in the Management DB and just after that, requests to start monitoring each KPI sending a *MonitorDeviceKpi* to the Device component. The Device service interfaces between the TeraFlow OS and the SDN agents, retrieving the information from the end devices and sending the samples to the Monitoring service through the *IncludeKpi*. Subsequently, the received samples are introduced in the MetricsDB by the Monitoring service to be lately accessed and visualized through the Data Viewer platform.

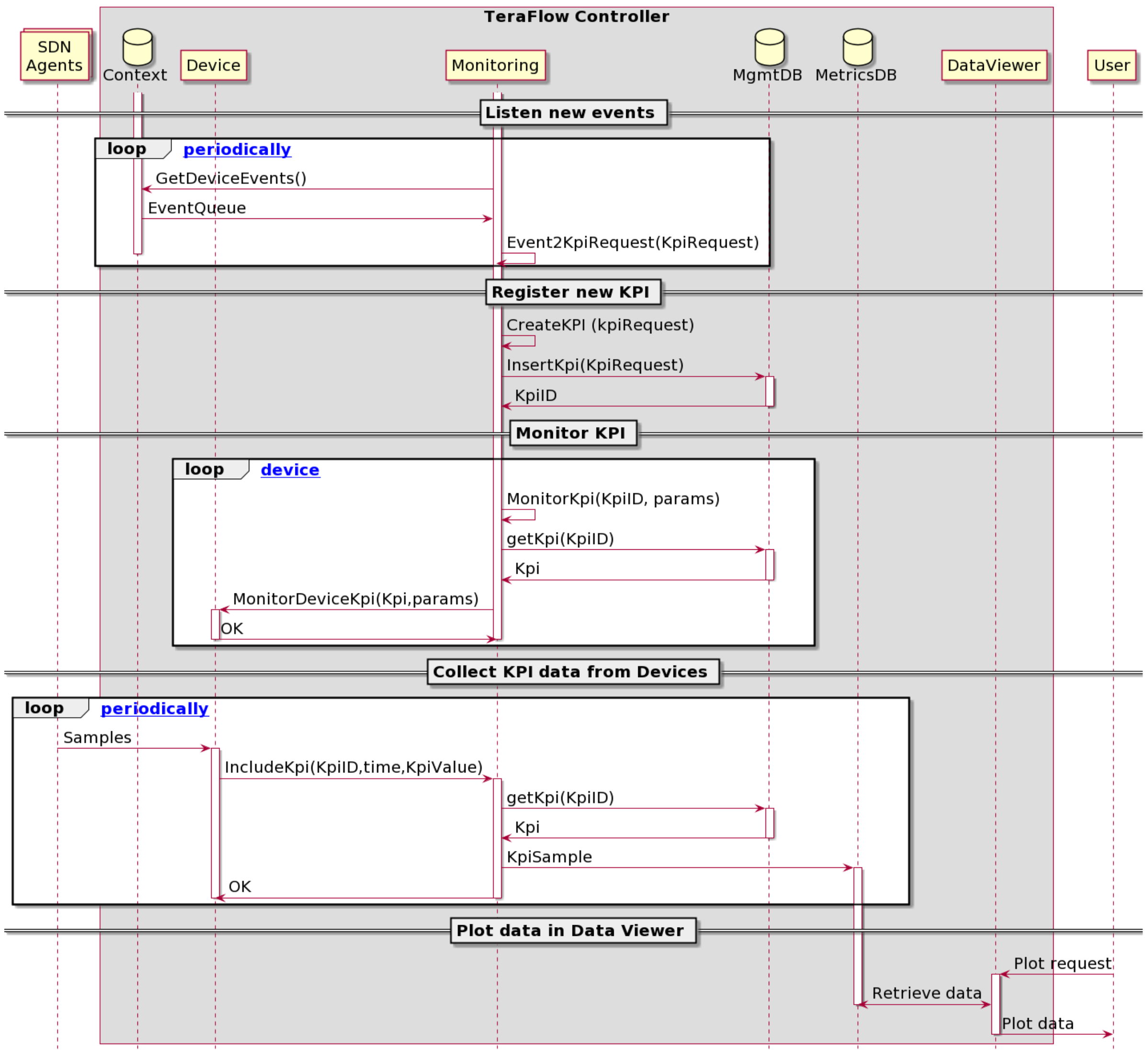


Figure 25 Monitoring sequence diagram

8.3. Service Restoration

Upon the deployment of a service, the TeraFlow controller invokes service monitoring routines which allow the service owner to select specific KPIs, subscribe to these KPIs, thus retrieve relevant monitoring information. To avoid manual effort upon the reception of an out-of-bounds performance metric for a given service, monitoring information can also be streamed towards an automated service restoration module, as shown in Figure 26. This module queries the monitoring service to collect a set of KPIs for a given service (also through the device service). Then, an internal routine of the automated service restoration module checks the values of the retrieved KPIs against a set of “expected” KPI boundaries. If an out-of-bounds KPI is detected, the service restoration module performs the necessary service update, while ensuring that the affected devices are reconfigured via the device service.

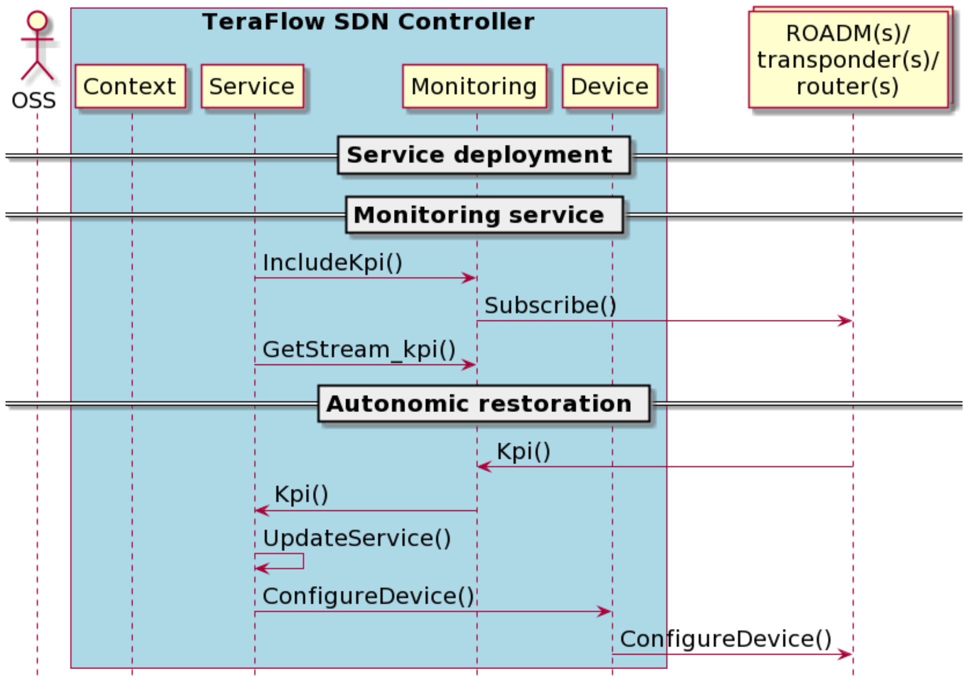


Figure 26 Service restoration sequence diagram.

8.4. Traffic Engineering

Figure 27 shows the different phases of the TE component: Building Traffic Engineering Database (TED), Registering PCE, Requesting Label Switched Paths (LSP) and Deleting LSP.

The TED is constructed by requesting the topological information to Context component and then building the TED internally in TE component. Later, the TE component requests to Context the available devices and it registers as PCE to each router device, that acts as PCC.

An SR LSP might be requested from Service component, resulting in a best path computation, and later triggering of the LSP to the routers. Finally, the SR LSP might be deleted from Service component and the Delete LSP request is forwarded to the routers.

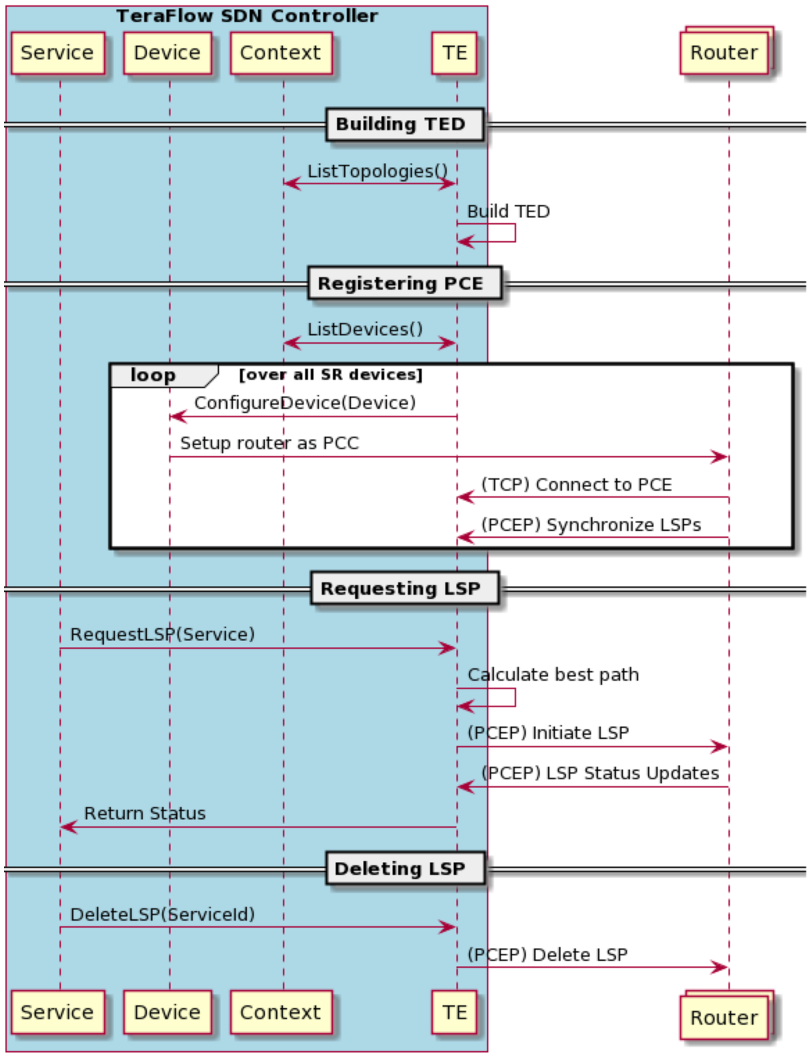


Figure 27 Traffic Engineering Sequence Diagram

8.5. Automation

Figure 28 visualizes the necessary steps for adding a new device under the management umbrella of the TeraFlow SDN controller in a fully automated manner. The process begins with the Policy component issuing an “AddDevice” RPC to the Device component. This method initiates a connection with the requested device, and if successfully connected, the device driver obtains the device configuration. This configuration is turned into a DeviceConfig object and pushed to the Context database. Upon success, a “DEVICE_ADD” event is generated, notifying that a new device is associated with a TeraFlow device driver plugin. As a result, the Context component generates a notification through the “Events’ API”. This event is received by the Automation component, thus the ztpAdd RCP is automatically triggered as shown in Figure 28. First, the Automation component requests the new Device object from the Context database, which in turn results in a “getDevice” call to the Device component. Then, if this device is not already configured, the Automation component requests this device’s initial configuration parameters by issuing a “getInitialConfig” RPC to the Device component through the Context component. Upon the reception of an updated DeviceConfig object, the Automation component loops through the configuration entries of its local object and updates the relevant entries according to the newly fetched DeviceConfig object. Next, the updated device object is pushed to the Device component and stored to the Context database via a "configureDevice" RPC. Upon success, the Automation component flips the DeviceStatus bit of the Device object to "ENABLED" and the respective ZTPDeviceState to "CREATED", while also generating relevant events. Finally, these events can be consumed by other TeraFlow OS components to begin e.g., monitoring routines for this device. Note that, if the DeviceStatus of the newly arrived device is already ENABLED, the device is already provisioned, thus no action is taken by the ztpAdd RPC, while a relevant warning is issued.

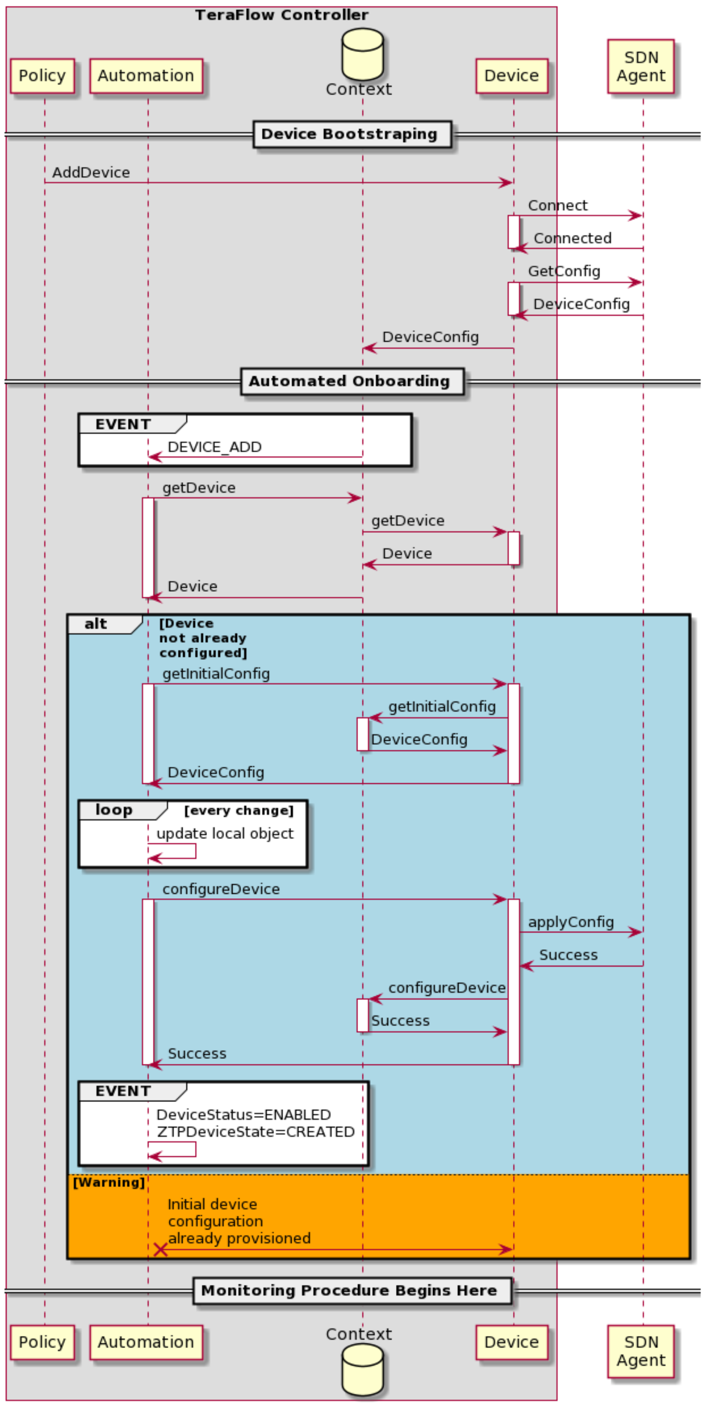


Figure 28 Zero-Touch Provisioning of a new device into TeraFlow OS.

8.6. Optical and L3 Centralized and Distributed Attack Detection

The centralized and distributed attack detection and mitigation workflow will provide TeraFlow with a continuous assessment of the security status of optical and IP services.

The distributed attack detector will be placed at remote sites, e.g., central offices, edge datacentres, and will receive IP traffic monitoring (e.g., by traffic mirroring) from co-located packet processors. The monitoring of IP traffic is expected to generate considerable traffic between the packet processors and the respective distributed attack detector. By deploying one instance of this component at each remote site, TeraFlow avoids transmitting the monitoring traffic across the network, reducing the network load, and potentially reducing the attack detection time. Once an attack is detected, the distributed attack detector notifies the attack mitigation process, providing a comprehensive characterization of the attack properties. Periodically, the distributed attack detector will also report summarized KPIs to the centralized attack detector with the purpose of enabling a holistic security assessment. This approach of sending summarized KPIs also removes the need to sending complete information about IP traffic to the centralized controller.

The centralized attack detector will consolidate the information from multiple instances of the distributed attack detector with the optical performance monitoring performed over optical services. This will enable the centralized attack detector to monitor the optical services for physical-layer attacks, while also having a view on the security status of IP traffic. Based on the summarized KPIs received from the distributed attack detector, the centralized attack detector can also perform attack detection on the IP layer by using the consolidated IP summarized KPIs. Upon an attack detection, the centralized attack detector sends an attack description to the fast attack mitigation processor for it to compute a mitigation for the attack.

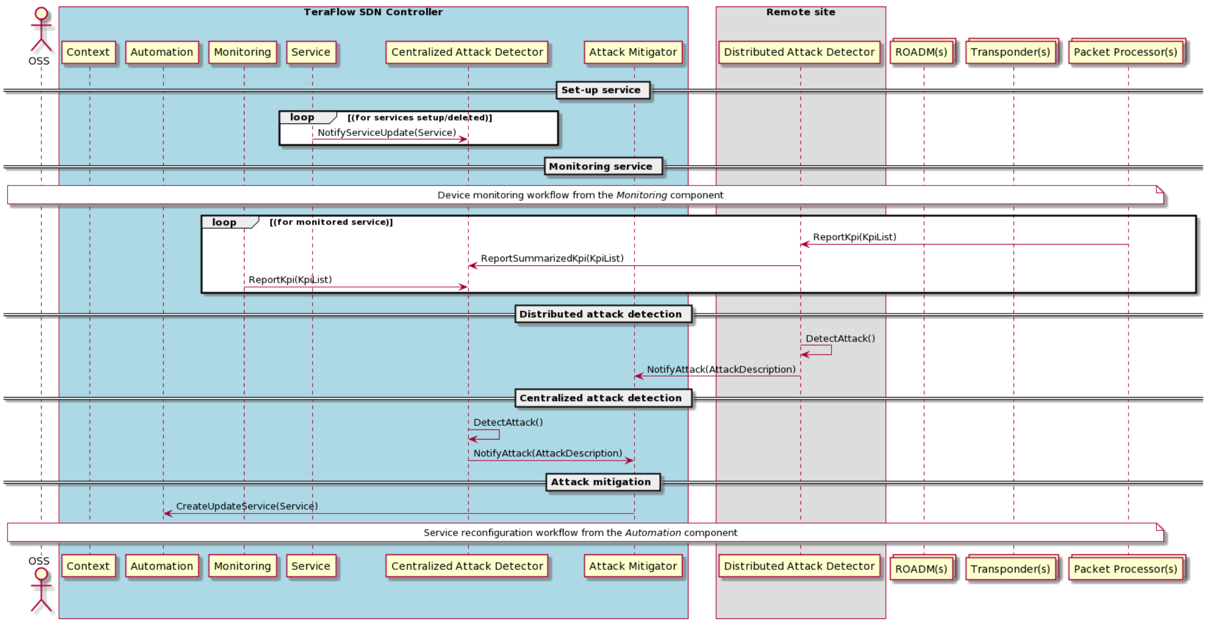

TeraFlow SDN Controller
Remote site
OSS
Context
Automation
Monitoring
Service
Centralized Attack Detector
Attack Mitigator
Distributed Attack Detector
ROADM(s)
Transponder(s)
Packet Processor(s)
Set-up service
loop
[(for services setup/deleted)]
NotifyServiceUpdate(Service)
Monitoring service
Device monitoring workflow from the Monitoring component
loop
[(for monitored service)]
ReportKpi(KpiList)
ReportSummarizedKpi(KpiList)
ReportKpi(KpiList)
Distributed attack detection
DetectAttack()
NotifyAttack(AttackDescription)
Centralized attack detection
DetectAttack()
NotifyAttack(AttackDescription)
Attack mitigation
CreateUpdateService(Service)
Service reconfiguration workflow from the Automation component

Figure 29 Centralized and distributed attack detection and mitigation

The fast attack mitigation processor, upon receiving an attack notification, is responsible for computing a viable attack mitigation, depending on the detected attack. For instance, if the attack detected is on the IP layer, one possible mitigation action is to block the flow that is carrying the malicious traffic. Another example is in the case of an optical physical layer attack, where a possible approach would be to establish a new optical service avoiding the current spectrum and/or links/nodes. The realization of the attack mitigation strategy will be coordinated with the Automation component, which will be responsible for performing the necessary actions across the devices in the network to make sure that the mitigation takes place.

In an initial implementation, the optical and the L3 workflows presented in Figure 29 will be implemented separately, i.e., the optical workflow will have its own components and the same for the IP/L3 workflow. This design decision is reflected in Section 6.3.2, where the component architecture for the optical and L3 workflows are described. In a future iteration, the centralized attack detector component for optical and L3 will be integrated to enable a cross-layer security assessment framework.

8.7. DLT and Smart Contracts

TeraFlow will deliver a permissioned distributed ledger that utilizes blockchains for network management. It will use blockchain technology to provide a trustworthy and resilient platform for storing, querying, and processing critical data about network resources and services owned and governed by different network entities. It will be privacy aware as well as transparent, resulting in an open, traceable, and fair sharing of network resources and services between stakeholders.

The use of blockchains replaces centralized network management consisting of conventional database management systems. Major advantages are the elimination of trusted third parties that maintain the databases with single point of failures, and data provenance including data immutability and traceability.

Smart contracts provide a universal basis to automate, simplify, and secure network management tasks that involve possibly sensitive data from multiple stakeholders of the network. Trust and multi-tenancy are improved in the SDN controllers by introducing novel security mechanisms through the usage of smart contracts and secure consensus algorithms. Network and device data is not stored and processed in a central location; instead, a blockchain, which stores the data and the operations on the data, is copied and spread across multiple nodes, and each node updates its blockchain to reflect a requested change, often by executing a smart contract.

The DLT operations are depicted in the sequence diagram in Figure 30. Services perform dual functions of reading from the DLT component and writing to the DLT component. The GetDLTstatus operations retrieves information from the blockchain and RecordtoDLT operation is used to write to the blockchain and the recorded information is shared among the DLT network components.

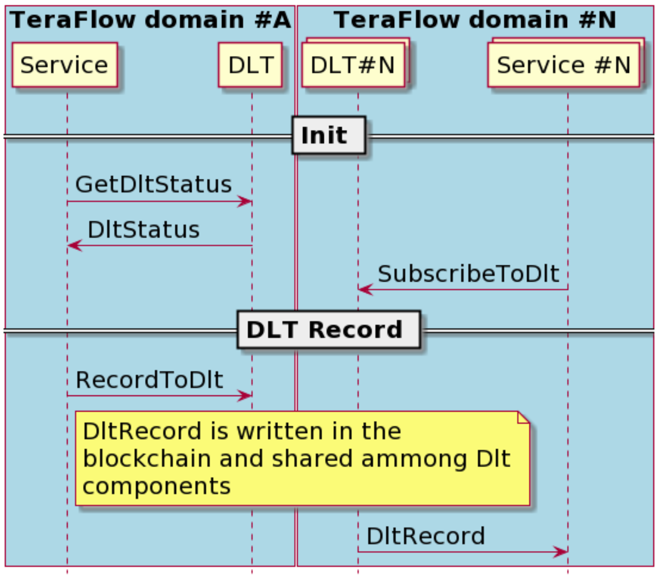


Figure 30 DLT sequence diagram

8.8. Compute integration

Figure 31 illustrates the integration of the Compute component of the Teraflow OS to interact with an external NFV Orchestrator, e.g., based on an OSM implementation. Specifically, it is depicted the integration to allow the NFV orchestrator requesting the creation of connectivity services over an underlying transport infrastructure (WAN) controlled/programmed by the SDN controlled offered by the Teraflow OS. By doing this, remote virtualized network functions (hosted at different cloud premises) and associated to a specific network service can be interconnected over the WAN. This integration between the NFV Orchestrator and the Compute component is supported over a defined REST API. Besides the creation of the network connectivity services, the API also supports other operations for handling the complete lifecycle management of such connections, i.e., deletion or update of active network services).

For the operation of creating network connectivity service, different steps constitute the resulting workflow between the NFV Orchestrator and the Compute component. The step labelled by 1) is bound to the request sent by the NFV Orchestrator (OSM) via a POST message to trigger the connectivity service (i.e., *create_connectivity_service*()). This is referred to as the creation of the *vpn service*. The POST message carries a set of required JSON-encoded objects, namely, the *vpn_id* (which contains an uuid), the *vpn_svc_type* (set to virtual private wire service, vpws), the *svc_topo* (set to any-to-any) and the *customer-name* (carrying OSM). After processing this message, the Compute component creates an entry for that network connectivity service. To unambiguously refer to that entry a service identifier (serviced) is created and set to the received *vpn_id.* Next, the Compute component responds to the NFV Orchestrator that the registration of the vpn service succeeded (i.e., step labelled by 2))

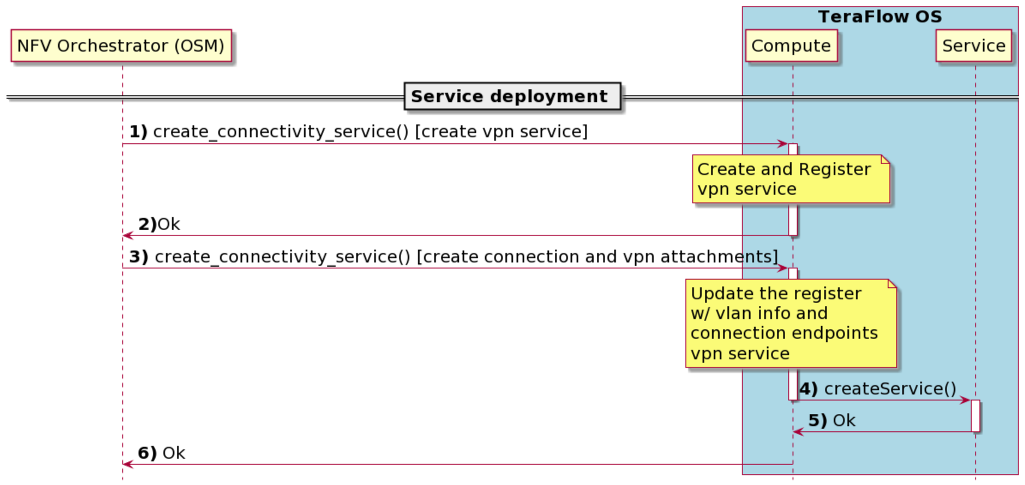

TeraFlow OS
NFV Orchestrator (OSM)
Compute
Service
Service deployment
1) create_connectivity_service() [create vpn service]
Create and Register vpn service
2)Ok
3) create_connectivity_service() [create connection and vpn attachments]
Update the register w/ vlan info and connection endpoints vpn service
4) createService()
5) Ok
6) Ok

Figure 31 NFV Service deployment using WIM

Next, the OSM sends another POST message in step 3) with the required information about the connection service and its vpn attachment endpoints. This entails specifying the *connection_point* (i.e., vlan), and the endpoints using the *bearer* and the *site*. These network connectivity service details are then updated at the Compute component and used to construct the gRPC createService message (step 4). Once the network connectivity service is successfully established by other Teraflow OS components (step 5)), the Compute component responds to the NFV Orchestrator informing that the network connectivity service is active, i.e., step 6).

8.9. Inter-Domain Services

Key functions of the inter-domain component are related to service lifecycle management, including the preparation and activation of a service as well as its modification during run time. Furthermore, monitoring of service KPIs across domains ensures that end-to-end requirements are met.

Figure 32 displays the workflow regarding service preparation and activation. The workflow is initiated by a customer which can be any entity consuming TeraFlow services such as the OSS or other management domains including end-to-end service management. The customer’s request is handled by the slice component which forwards the end-to-end transport slice request to the inter-domain component. The inter-domain component, in turn, decomposes the end-to-end transport slice into per-domain sub-slices and requests their creation in the respective TeraFlow domains through the corresponding inter-domain components. This inter-domain communication is performed in a secure manner by mutual authentication prior to the exchange of sub-slice requests. In case an appropriate sub-slice can be provided, the remote inter-domain component informs the requesting inter-domain component and the latter can order the slice. Otherwise, the creation of a corresponding slice is initiated alongside its insertion into the catalogue and establishment of connectivity, triggering interaction with slice and service components, respectively. Finally, the same procedure of sub-slice creation and connection establishment is performed at the local TeraFlow domain (domain #A in the figure).

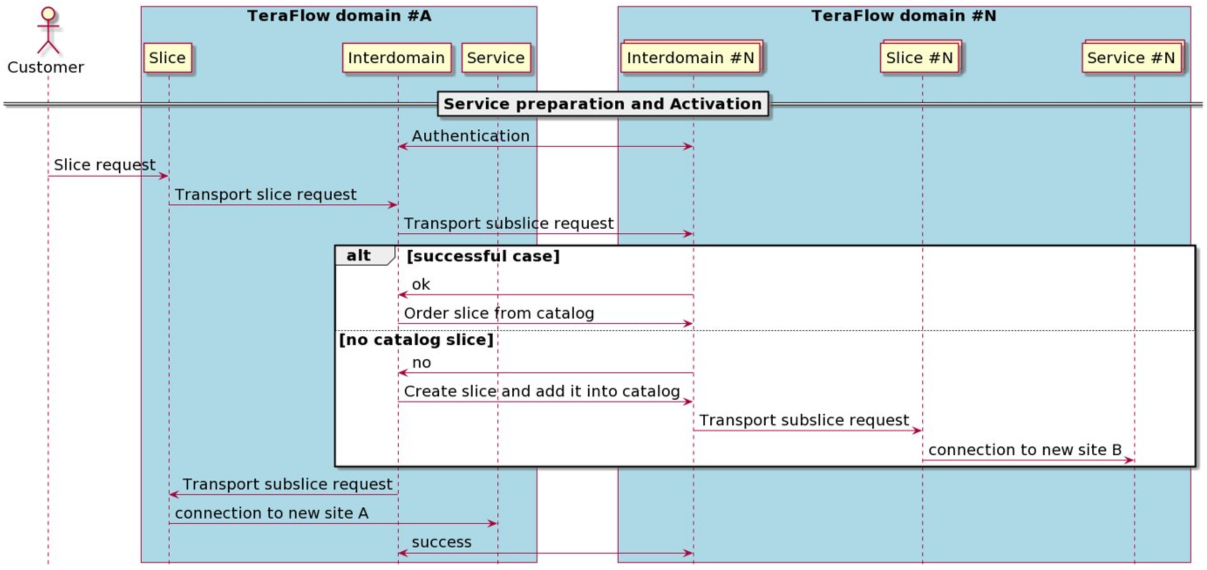


Figure 32 Inter-domain service preparation and activation

Figure 33 outlines the workflow of modifying a service. The process is triggered by an inter-domain component which sends a service modification request to a remote inter-domain component. Upon receiving this request, the remote inter-domain component evaluates whether the modification (e.g., an increase of bandwidth) can be supported. Depending on the outcome of the evaluation, the modification is either carried out by propagating the request to the involved slice and service components, or the requesting inter-domain component is informed about the negative outcome.

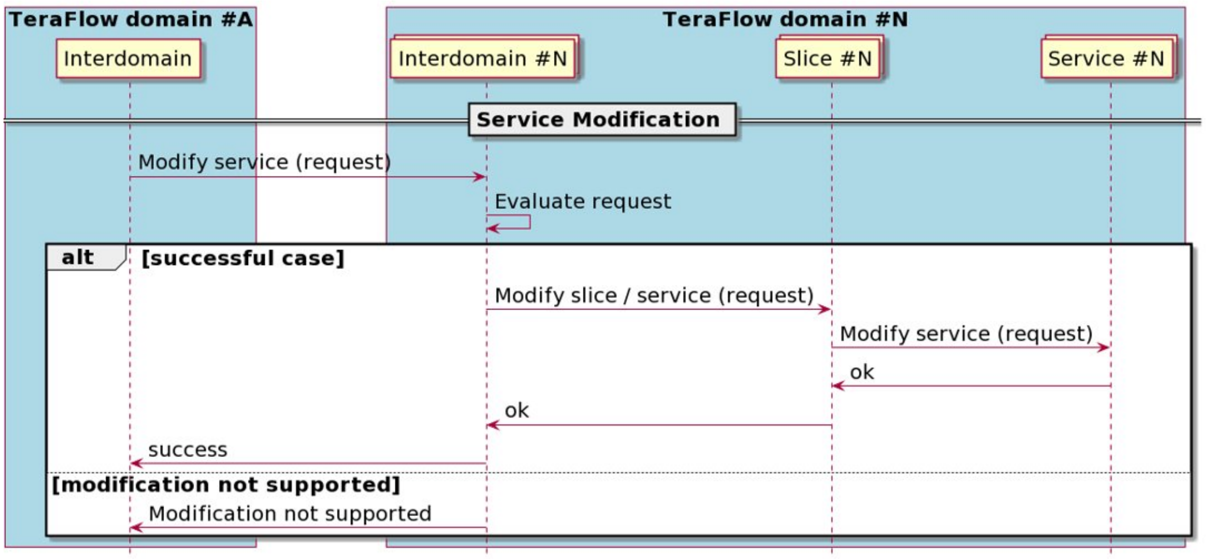


Figure 33 Inter-domain service modification

Finally, the inter-domain component leverages the monitoring service to obtain service- and/or device-level KPIs on a per-domain basis to ensure that end-to-end requirements are met. To this end, the inter-domain component subscribes to the relevant KPIs from the monitoring component in its domain and synchronizes them with remote inter-domain components. This process is summarized in Figure 34.

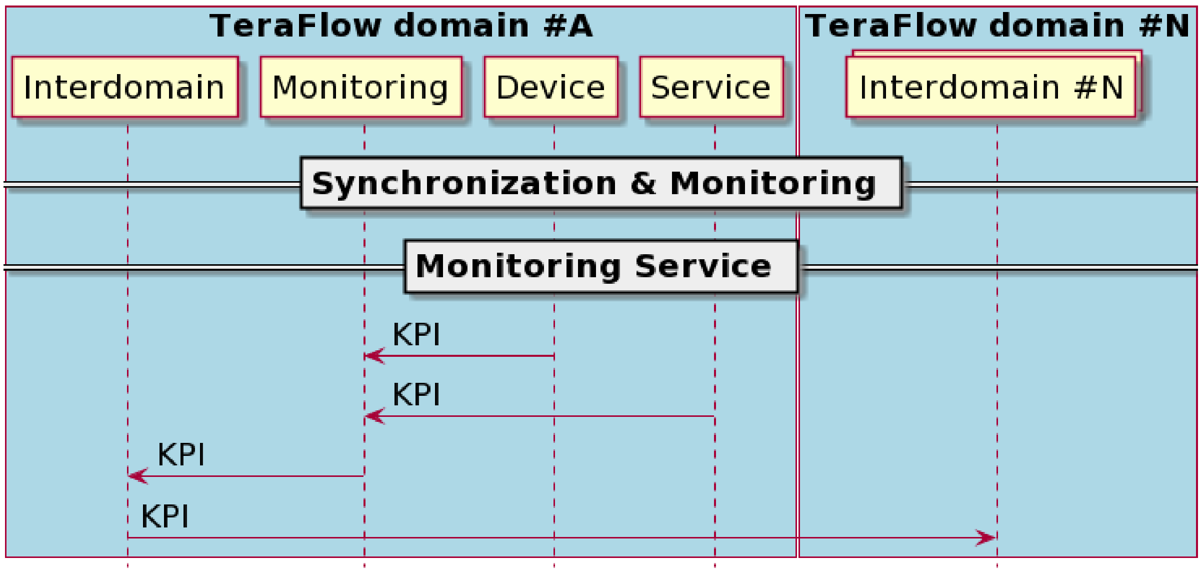


Figure 34 Inter-domain service synchronization/monitoring

9. Conclusions and Next Steps

This deliverable includes the methodology to be applied in TeraFlow, initial use case definitions, requirement elicitation, the draft architecture to be used in WP3 and WP4 for the release of TeraFlow OS v1. In addition, data models for the different components and interfaces are also detailed and use case-specific workflows have been described.

This deliverable includes all the necessary architecture definitions and detailed design descriptions to provide the initial TeraFlow features. TeraFlow follows an agile methodology to produce three major software releases: v1 will be released at M12, v2 will be released at M24, and v2.1 will be released at M30. These releases will cover the length of the technical work packages (WP3 and WP4).

For a detailed view of the performed work and further work of the different presented components, please refer to D3.1 and D4.1, which provide more technical details and preliminary obtained results of the different components. Also, the expected work for Y2-Y3 is included.

To support software development and the ability to make releases with minimal impact on the running services, TeraFlow will adopt a CI/ CD strategy described in D5.1. The major software releases will deliver the main functionality, while the minor releases will provide bug fixes and possibly additional features required by the currently operating experiments.

References

[BAR21a] S. Barguil, O. González-de-Dios, V. López, K. Pardini, R. Vilalta, Automation of Network Services for the Future Internet , Chapter in Design Innovation and Network Architecture for the Future Internet, published by IGI Global, April 2021.ISB: 9781799876465.

[BAR21b] S. Barguil, et al., A Layer 3 VPN Network YANG Model, IETF draft, October 2021. https://datatracker.ietf.org/doc/html/draft-ietf-opsawg-l3sm-l3nm-18, Accessed: 2021-12-28

[BAR21c] S. Barguil, et al., A Layer 2 VPN Network YANG Model, IETF draft, November 2021. https://datatracker.ietf.org/doc/html/draft-ietf-opsawg-l2nm-12, Accessed: 2021-12-28

[BER08] A. Bergek, S. Jacobsson, B. Carlsson, S. Lindmark and A. Rickne, "Analyzing the functional dynamics of technological innovation systems: A scheme of analysis," Research Policy, vol. 37, pp. 407-429, 2008.

[BRE19] E. Breverman et al., Optical Zero Touch Networking—A Large Operator Perspective, OFC 2019.

[EC21] European Comission, "5G supply market trends – Final Report," European Commission, Brussels, 2021.

[FAR21] Farrel, A., Gray, E., Drake, J., Rokui, R., Homma, S., Makhijani, K., Contreras, L. M., and J. Tantsura, "Framework for IETF Network Slices", IETF draft TEAS working group, August 2021. https://datatracker.ietf.org/doc/draft-ietf-teas-ietf-network-slices, Accessed: 2021-12-28

[GAW14] A. Gawer and M. A. Cusumano, "Industry Platforms and Ecosystem Innovation," Journal of Product Innovation Management, vol. 31, no. 3, pp. 417-433, 2014.

[GRPC21] Health checking your gRPC servers on GKE, https://cloud.google.com/blog/topics/developers-practitioners/health-checking-your-grpc-servers-gke, Accessed: 2021-12-28

[Hal21] H. K. Hallingby, S. Fletcher, V. Frascolla, A. Gavras, I. Mesogiti and F. Parzysz, "5G ecosystems," 5G Infrastructure Association, 2021.

[HEK11] M. Hekkert, S. Negro, G. Heimeriks and R. Harmsen, "Technological Innovation System Analysis – a manual for analysts," Universiteit Utrecht, Utrecht, 2011.

[LIU21] IETF Network Slice YANG Data Model, https://datatracker.ietf.org/doc/draft-liu-teas-transport-network-slice-yang/, Accessed: 2021-12-28

[MUS21] "MUST IP SDN Controller SBI Technical Requirements", Telcom Infra Project (TIP), https://cdn.brandfolder.io/D8DI15S7/as/mfbj6nm7w38xnbvrmcnbp9t6/MUST-IP-Controller-SBI-Requirements-Document-v10_FINAL_VERSION_WEBSITE.pdf, Accessed: 2021-12-28.

[PRO21] Protocol buffers, https://github.com/protocolbuffers/protobuf, Accessed: 2021-12-28.

[RFC8466] A YANG Data Model for Layer 2 Virtual Private Network (L2VPN) Service Delivery, IETF RFC 8466.

[SAA21] T. Saad, et al., A YANG Data Model for Traffic Engineering Tunnels, Label Switched Paths and Interfaces, IETF draft, https://datatracker.ietf.org/doc/draft-ietf-teas-yang-te/, Accessed: 2021-12-28.

[TR547] "TR-547 – TAPI v2.1.3 Reference Implementation Agreement v1.1", Open Networking foundation (ONF), https://wiki.opennetworking.org/download/attachments/259719184/TR-547-TAPI_ReferenceImplementationAgreement_v1.1.zip?version=2&amp;modificationDate=1639671964711&amp;api=v2, Accessed: 2021-12-28.