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

D2.2 Final Requirements, Architecture Design, Business Models, and Data Models

Deliverable type	R (Report)
Dissemination level	P (Public)
Due date	31.12.2022
Submission date	31.12.2022
Lead editor	Juan Carlos Caja (TID)
Authors	Juan Pedro Fernandez-Palacios, Juan Carlos Caja, Oscar González-de-Dios, Pablo Armingol, Antonio Pastor (TID), Georgios P. Katsikas, Dimitrios Klonidis (UBITECH), Stanislav Lange (NTNU), Lluís Gifre, Pol Alemany, Ricardo Martínez, Michela Svaluto, Javier Vilchez, Ricard Vilalta (CTTC), Alberto Mozo, Luis de la Cal, Amit Karamchandani (UPM), Carlos Natalino (CHAL), Sebastien Andreina, Konstantin Munichev, Giorgia Mason (NEC), Min Xie, Jane Frances Pajo, Håkon Lønsethagen, Hanne Kristine Hallingby (Telenor), Achim Autenrieth, José Juan Pedreño Manresa (ADVA), Mika Silvola (Infinera), Michele Milano, Nicola Carapellese (SIAE), Javier Moreno, Sergio González, Esther Garrido (ATOS), Sebastien Merle, Peer Stritzinger (Stritzinger)
Reviewers	Adrian Farrel (ODC), Oscar González-de-Dios (TID)
Quality Check team	Adrian Farrel (ODC), Daniel King (ODC)
Work package	WP2

Abstract

This document reports the final version of the use case requirements, architecture design, business models, and data models that refer to ETSI TeraFlowSDN release 2.0. This document is an update of D2.1, based on comments from WP3, WP4, and WP5, and including lessons learned from TeraFlow SDN release 1.0. A new class of secure cloud native Software-Defined Networking (SDN) controller, called TeraFlowSDN (TFS) controller, is described that significantly advances the technology used in Beyond-5G (B5G) networks offering ground-breaking features for both flow management (service layer) and the integration of optical/microwave network equipment (infrastructure layer). This new

SDN controller will be able to integrate with the existing Network Functions Virtualisation (NFV) and Multi-Access Edge Computing (MEC) frameworks and incorporate security using Machine Learning (ML) and forensic evidence for multi-tenancy based on Distributed Ledger Technologies (DLT).

[End of abstract]

Revision History

Revision	Date	Responsible	Comment
0.1	25.07.2022	Editor	Initial structure of document
0.2	28.10.2022	All	Preliminary contributions and first draft
0.3	18.11.2022	All	Final contributions
0.4	13.12.2022	Editor	Internal review
0.5	25.12.2022	Editor	Apply changes after Technical and QA review
1.0	26.12.2022	Editor	Final version

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 document is an update of D2.1, based on comments from WP3, WP4, and WP5. This deliverable offers a final version of the use case requirements, architecture, design, and data models that refer to ETSI TeraFlowSDN release 2.0.

This document opens with an introduction that describes the goal of this delivery, how it relates to previous deliverables, and how this document is organised. The following parts describe the use cases, then a business case analysis based on potential cost reductions and new revenue prospects for telcos is also included. The fourth section summarises the received feedback from release 1. The next sections present the most recent and brand-new specifications for TeraFlowSDN controller, followed by the updated architecture. The document completes with the prospective topics, future work, and the conclusions. Internal workflows are provided as Annex, and they will be further detailed in D3.3, D4.2 and D5.2.

In summary, a brand-new class of secure cloud native Software-Defined Networking (SDN) controller, called ETSI TeraFlowSDN (TFS) controller, will significantly advance the technology used in Beyond-5G (B5G) networks offering ground-breaking features for both flow management (service layer) and the integration of optical/microwave network equipment (infrastructure layer). This new SDN controller will be able to integrate with the existing Network Functions Virtualisation (NFV) and Multi-access Edge Computing (MEC) frameworks and incorporate security using Machine Learning (ML) and forensic evidence for multi-tenancy based on Distributed Ledger Technologies (DLT).

In the first section, we focus on the use cases. The use cases help to identify the specific requirements. Figure 1 and Figure 2 show the detailed use cases per TeraFlowSDN component. Specific information about the use cases is provided within each of the sections describing the different scenarios.

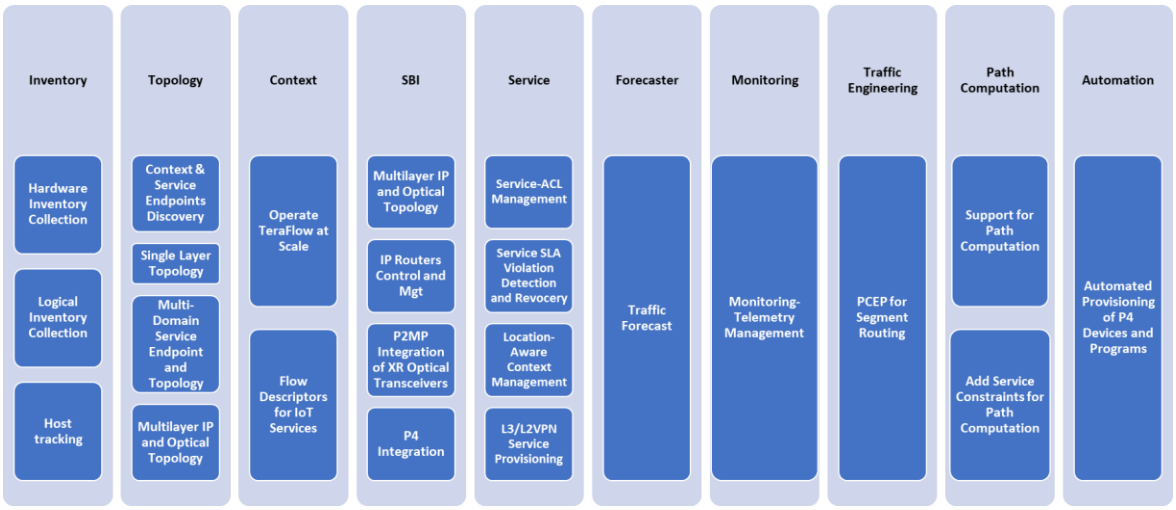


Figure 1 ETSI TeraFlowSDN Release 2.0 Use Cases (part 1/2)

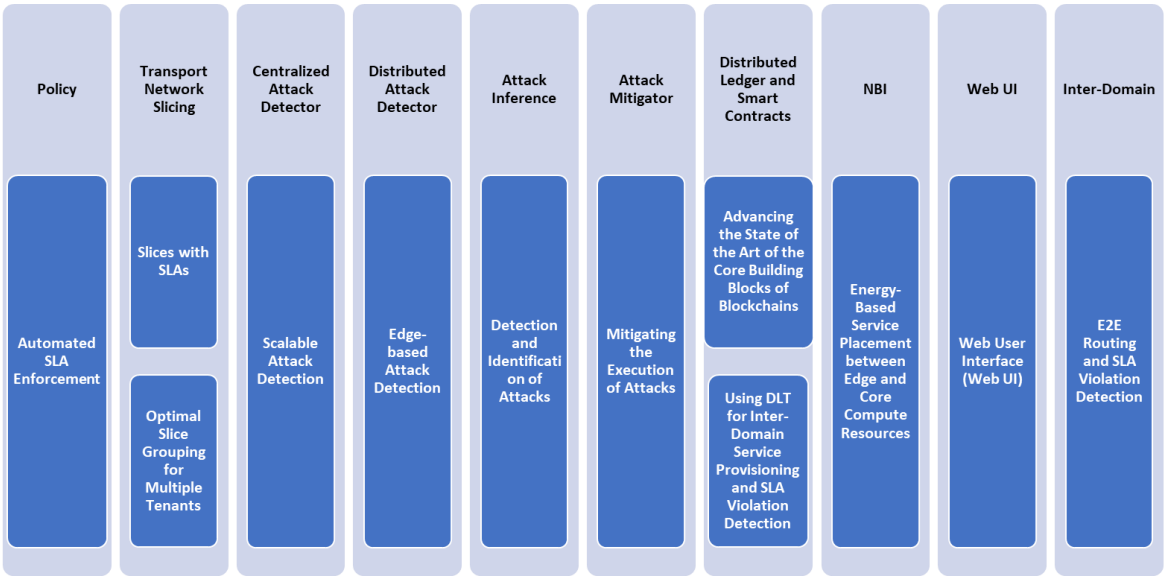


Figure 2 ETSI TeraFlowSDN Release 2.0 Use Cases (part 2/2)

Business model analysis for TeraFlowSDN controller is important, as we need to design a valid business model for the controller. We have suggested five preliminary business model canvases for four actor roles. They all indicate potential ways to extract revenue from potential customers in a regime with full compatibility, e.g., enabled by TeraFlowSDN. One key observation is the difference between a future hardware (HW), i.e., bare metal, provider and software (SW) providers. The HW provider will be in a classic manufacturing market where economies of scale drive competition, however, timely on-premises installation and SW configuration may significantly affect the market. Following a life-cycle management of HW, leasing may be an interesting revenue model for HW providers. The providers of SW, NetApps, and TeraFlowSDN related services have two main business models. One is labour intensive, addressing a market for consulting and integration. The other relies on Intellectual Property Rights (IPR), where it may be possible to use a license revenue model. We anticipate that the market for integration will be a mass market in the sense that most operators are potential customers. For NetApps and specific SDN features, we assume that the potential customers are fewer or smaller, and that this is will be a niche market, potentially with a premium price. The elaborated business model canvases and comparison of them will be subject to further analyses in the next phase.

We can summarise the received feedback from release 1.0 in two main groups: Requested new features, and a desire for simplification and more depth in describing the user and developer guidelines for TeraFlowSDN.

Moreover, TeraFlowSDN functional and non-functional requirements are presented. They serve as the basis for the ETSI TeraFlowSDN controller release 2.0. The functional requirements are classified into components, and the non-functional requirements include performance, usability, scalability, reliability, and portability.

The release 2.0 architecture is show in Figure 3. It is based on a micro-service architecture, and each component is detailed, later in this document, through a template that describes the main functionality, the operations for the component, and the internal data models. A protocol buffer is described for each of the components in order to model the services.

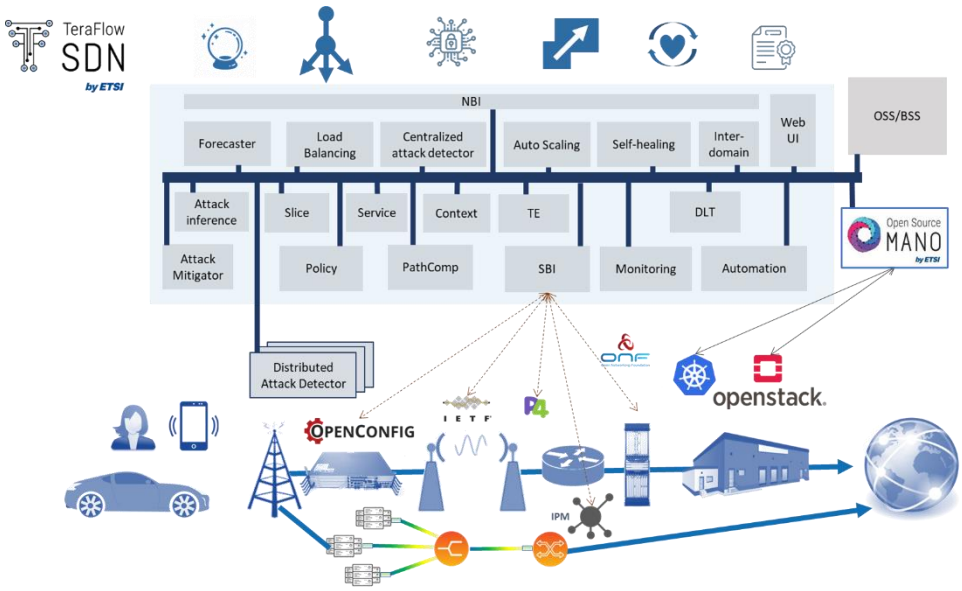


Figure 3 ETSI TeraFlowSDN Release2.0 Architecture

We have analysed the use cases, and Figure 4 lists the workflows for TeraFlowSDN and their internal components.

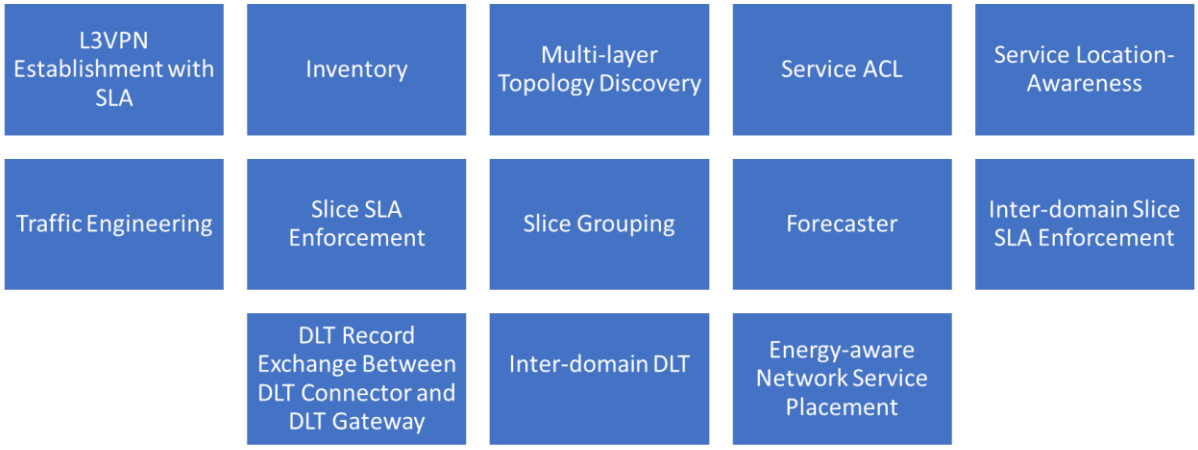


Figure 4 TeraFlowSDN Release 2.0 Workflows

This milestone has served the main purpose of simplifying the development of features and architecture for ETSI TeraFlowSDN release 2.0. Next steps will be provided within WP3 and WP4, and feedback to WP2 will be received in order to formulate the final version of the release 2.0 architecture.

Contents

Executive Summary 4
List of Figures 12
List of Tables 14
Abbreviations 15
1. Introduction 19
2. Use Cases 20
2.1. Inventory 20
2.1.1. Hardware Inventory Collection 20
2.1.2. Logical Inventory Collection 20
2.1.3. Host Tracking 20
2.2. Topology 20
2.2.1. Context and Service Endpoints Discovery 21
2.2.2. Single Layer Topology 21
2.2.3. Multi-Domain Service Endpoint and Topology 21
2.2.4. Multilayer IP and Optical Topology 21
2.3. Context 24
2.3.1. Operate TFS at Scale 24
2.3.2. Flow Descriptors for IoT Services 25
2.4. SBI (Formerly the Device Component) 26
2.4.1. IP Routers Control and Management 26
2.4.2. Microwave Integration 27
2.4.3. Point to Multi-Point Integration of XR Optical Transceivers 28
2.4.4. P4 Integration 29
2.5. Services 29
2.5.1. Service-ACL Management 30
2.5.2. Service SLA Violation Detection and Recovery 31
2.5.3. Location-Aware Context Management 31
2.5.4. L3VPN Service Provisioning 31
2.5.5. L2VPN Service Provisioning 33
2.6. Forecaster 33
2.6.1. Traffic Forecast 33
2.7. Monitoring 34
2.7.1. Monitoring-Telemetry Management 34
2.8. Traffic Engineering 35

2.8.1. PCEP for Segment Routing 35
2.9. Path Computation 36
2.9.1. Support for Path Computation 36
2.9.2. Add Service Constraints for Path Computation 38
2.10. Automation 39
2.10.1. Automated Provisioning of P4 Devices and Programs 39
2.11. Policy 39
2.11.1. Automated SLA Enforcement 39
2.12. Transport Network Slicing 40
2.12.1. Slices with SLAs 40
2.12.2. Optimal Slice Grouping for Multiple Tenants 41
2.13. Centralized Attack Detector 42
2.13.1. Scalable Attack Detection 42
2.14. Distributed Attack Detector 42
2.14.1. Edge-Based Attack Detection 42
2.15. Attack Inference 43
2.15.1. Detection and Identification of Attacks 43
2.16. Attack Mitigator 43
2.16.1. Mitigating the Execution of Attacks 43
2.17. Distributed Ledger and Smart Contracts 44
2.17.1. Advancing the State of the Art of the Core Building Blocks of Blockchains 44
2.17.2. Using DLT for Inter-Domain Service Provisioning and SLA Violation Detection 45
2.18. NBI (Including the Old Compute Component) 45
2.18.1. Energy-Based Service Placement between Edge and Core Compute Resources 46
2.19. Web User Interface (WebUI) 47
2.20. Inter-Domain 48
2.20.1. E2E Routing and SLA Violation Detection 48
3. Business Model and Ecosystem Analysis 49
3.1. The Ecosystem and Actor Roles 49
3.2. The Network Operator – The Customer 51
3.3. System Integrators Serving Operators 51
3.4. Operators and Regimes for Testing of HW 53
3.5. HW Provider 54
3.5.1. HW as Bare Metal 54
3.5.2. HW Retaining Some Dedicated Embedded SW 55
3.6. SW Provider 55

3.7. NetApp Provider 56
3.8. Provider of TFS Related Products and Services 57
3.8.1. TFS as a Service 57
3.8.2. Development of New TFS Features 58
3.9. Neutral Lab – Testing and Certifying 59
3.10. Summary 63
4. Feedback from Release 1.0 64
4.1. Feedback from Advisory Board 64
4.2. Feedback from Users 64
4.3. Feedback for TFS Questionnaire 65
4.3.1. TFS Questionnaire 65
4.3.2. Preliminary Feedback Analysis 65
5. Updated Requirements for the TFS Controller 68
5.1. Functional Requirements 68
5.1.1. Context 68
5.1.2. SouthBound Interface (SBI) 68
5.1.3. Service 68
5.1.4. Forecaster 69
5.1.5. Monitoring 69
5.1.6. Traffic Engineering 70
5.1.7. Path Computation 70
5.1.8. Automation 71
5.1.9. Policy 72
5.1.10. Transport Network Slicing 72
5.1.11. Centralized Attack Detector 73
5.1.12. Distributed Attack Detector 73
5.1.13. Attack Inference 73
5.1.14. Attack Mitigator 73
5.1.15. Distributed Ledger and Smart Contracts 73
5.1.16. NorthBound Interface (NBI) 74
5.1.17. Inter-Domain 74
5.1.18. Web User Interface 75
5.1.19. TFS Controller Security 75
5.2. Non-Functional Requirements 76
5.2.1. Performance 76
5.2.2. Usability 76

5.2.3. Scalability 76
5.2.4. Security 77
5.2.5. Reliability 77
5.2.6. Portability 77
6. Updated Architecture 78
6.1. Overall Architecture Update 78
6.2. Detailed Architecture 80
6.2.1. Context 80
6.2.2. SBI (Formerly Device Manager) 82
6.2.3. Service 82
6.2.4. Forecaster 83
6.2.5. Monitoring 84
6.2.6. Traffic Engineering 87
6.2.7. Path Computation 87
6.2.8. Automation (ZTP) 88
6.2.9. Policy Manager 90
6.2.10. Slice Manager 91
6.2.11. Centralized Attack Detector 92
6.2.12. Distributed Attack Detector 94
6.2.13. Attack Inference 94
6.2.14. Attack Mitigator 95
6.2.15. Distributed Ledger 96
6.2.16. NBI (Previously Compute) 97
6.2.17. Inter-Domain 98
6.2.18. Web User Interface (WebUI) 98
6.2.19. Cloud Orchestrator Features 98
7. Prospective Topics and Future Work 101
7.1. Regional Extensions to IETF Slicing 101
7.1.1. Interconnected Sliced Networks and Services 102
7.1.2. Mechanisms for Filtering and Classifying Traffic onto Slices and Regions 103
7.2. TFS as a Dedicated Technology SDN Controller 103
7.3. TFS as a Toolbox for Intent-Based Networking 104
7.4. TFS as a QKD SDN Controller and SDN Orchestrator 105
7.4.1. QKD SDN Controller 105
7.4.2. QKD SDN Orchestrator 106
8. Conclusion and Next Steps 107

9. References 108
Annex I: Workflows 110
L3VPN Establishment with SLA 110
L3VPN Establishment 111
L2VPN Establishment 111
Inventory 112
Multi-Layer Topology Discovery 113
Service ACL 113
Service Location-Awareness 114
Traffic Engineering 115
Slice SLA Enforcement 116
Slice Grouping 116
Forecaster 117
Inter-Domain Slice SLA Enforcement 118
DLT Record Exchange Between DLT Connector and DLT Gateway 119
Inter-Domain DLT 119
Energy-Aware Network Service Placement 120

List of Figures

Figure 1 ETSI TeraFlowSDN Release 2.0 Use Cases (part 1/2) 4
Figure 2 ETSI TeraFlowSDN Release 2.0 Use Cases (part 2/2) 5
Figure 3 ETSI TeraFlowSDN Release2.0 Architecture 6
Figure 4 TeraFlowSDN Release 2.0 Workflows 6
Figure 5 IP Topology 22
Figure 6 PLUG-ID Usage. Optical Domain, and Topology 23
Figure 7 IP Mono-Domain Topology 24
Figure 8 IP IGP and LLDP Augmentations 24
Figure 9 Microwave Integration 27
Figure 10 XR Optical Transceivers Integration 28
Figure 11 gNMI Telemetry Architecture 35
Figure 12 PCEP for Segment Routing New Requirements 36
Figure 13 Internal Architecture of Path Comp Component 37
Figure 14 Sequence Diagram for DLT 45
Figure 15 Illustration of Start and Endpoint of the Evolution of a Transport Network Ecosystem 49
Figure 16 Updated TFS Ecosystem 50
Figure 17 Sequences in Business Transaction between Operator and Providers of HW, SW, NetApps, and SDN Services 51
Figure 18 Illustration of System Integration and Testing Regimes in Business Transaction between Operator and Providers 53
Figure 19 Factors Affecting Business Models and Ecosystem Paths and-State – dots for illustration only 62
Figure 20 Stakeholders Involved in the Answers 66
Figure 21 Main Interest in TeraFlowSDN 67
Figure 22 Expected Usage of TFS 67
Figure 23 TFS Release 2.0 Architecture 78
Figure 24 Context Data Model 81
Figure 25 Device Internal Data Model 82
Figure 26 Forecaster Internal Data Models 83
Figure 27 TeraFlow Monitoring Component Updated Architecture 84
Figure 28 Monitoring Data Model 86
Figure 29 Path Computation Internal Models 88
Figure 30 Automation Component Architecture. 89
Figure 31 Automation Internal Data Models 89
Figure 32 Policy Component Architecture. 91
Figure 33 Policy Internal Data Models 91
Figure 34 L3 Attack Mitigator Output 93
Figure 35 L3 Centralized Attack Detector Internal Data Model 93
Figure 36 Attack Inference Data Model 95
Figure 37 L3 Attack Mitigator Output Data Model 96
Figure 3838 Optical Attack Mitigator Data Model 96
Figure 39 DLT Component Data Model 97
Figure 57 I2NSF Architecture 105
Figure 58 Depiction of an SD-QKD Network Showing a Set of SD-QKD Nodes 105
Figure 59 Use Case of SDN Orchestrator for QKD Network and OTN 106
Figure 40 L3VPN Provisioning with SLA 110

Figure 41 L3VPN Provisioning 111
Figure 42 L2VPN Provisioning 111
Figure 43 Inventory 112
Figure 44 Multi-Layer Topology Discovery 113
Figure 45 ACL Service Deployment 113
Figure 46 Service Location-Awareness Sequence Diagram 114
Figure 47 Traffic Engineering Sequence Diagram 115
Figure 48 Slice SLA Enforcement Sequence Diagram 116
Figure 49 Slice Grouping Sequence Diagram 116
Figure 50 Forecaster Sequence Diagram 117
Figure 51 Inter-Domain Slice SLA Enforcement Sequence Diagram 118
Figure 52 Inter-Domain SLA Violation Reaction Sequence Diagram 118
Figure 53 DLT Record Exchange Between DLT Connector and DLT Gateway Sequence Diagram 119
Figure 54 Inter-Domain Service Preparation and Activation with DLT Sequence Diagram 119
Figure 55 Inter-Domain SLA Violation Reaction with DLT 120
Figure 56 Energy-Aware Network Service Placement Sequence Diagram 120

List of Tables

Table 1 Business Model Canvas for System Integrator in the Transport Network 52
Table 2 Business Model Canvas for Providers of Bare Metal HW 54
Table 3 Business Model Canvas for Providers of Dedicated HW 55
Table 4 Business Model Canvas for SW Providers 56
Table 5 Business Model Canvas for NetApp Providers 57
Table 6 Business Model Canvas for Provider of TFS Package and Service 58
Table 7 Business Model Canvas for TFS Related Provider – Develop New Features 59
Table 8 Advantages and Disadvantages Regarding Handling of Devices to be Tested 60
Table 9 Business Model Canvas for a Provider Which is a Neutral Test Lab 61
Table 10 Business Model Canvas for the Actor Role Which Issues Certificates for Compatibility 62

Abbreviations

ACL	Access Control List
AM	Attack Mitigator
API	Application Programming Interface
AI	Artificial Intelligence
AS	Autonomous System
B5G	Beyond 5G
BGP	Border Gateway Protocol
CAD	Centralized Attack Detector
CD	Continuous Delivery
CE	Customer Edge
CI	Continuous Integration
DAD	Distributed Attack Detector
DLT	Distributed Ledger Technologies
DSCP	Differentiated Service Code Point
DWDM	Dense Wavelength Division Multiplexing
E2E	End-to-End
E-NNI	External Network-to-Network Interface
EVPN	Ethernet VPN
G/MPLS	Generalised Multiprotocol Label Switching / Multiprotocol Label Switching
gNMI	Google Network Management Interface
gRPC	Google RPC
GUI	Graphical User Interface
HL3	Hierarchical Level 3: Aggregation stage 2
HL5	Hierarchical Level 5: Access
HW	Hardware
IETF	Internet Engineering Task Force
IoT	Internet of Things
IP	Internet Protocol
IPR	Intellectual Property Rights
IS-IS	Intermediate System to Intermediate System protocol

KPI	Key Performance Indicator
L0	Layer 0
L2	Layer 2
L2VPN	Layer 2 Virtual Private Network
L3	Layer 3
L3NM	Layer 3 Network Model
L3VPN	Layer 3 Virtual Private Network
L4	Layer 4
LAG	Link Aggregation Group
LAN	Local Area Network
LLDP	Link Layer Discovery Protocol
LSP	Label Switched Path
MANO	Management and Orchestration
MEC	Multi-access Edge Computing
ML	Machine Learning
MPLS	Multiprotocol Label Switching
MQP	Managed Quality Path
MW	Microwave
NBI	North-Bound Interface
NEP	Node Edge Point
NFV	Network Functions Virtualisation
ONF	Open Networking Foundation
OPM	Optical Performance Monitoring
OSM	Open Source MANO
OSPF	Open Shortest Path First protocol
OSS	Operation Support System
P4	Programming Protocol-independent Packet Processors
PCE	Path Computation Element
PCEP	Path Computation Element Protocol
PE	Provider Edge
PiL	PoP Interconnecting Link

PoP	Point of Presence
QKD	Quantum Key Distribution
RAT	Radio Access Technology
RPC	Remote Procedure Call
RSVP	Resource Reservation Protocol
SBI	South-Bound Interface
SCS	Specialised Connectivity Services
SDN	Software-Defined Networking
SIP	Service Interface Point
SLA	Service Level Agreement
SLO	Service Level Objective
SME	Small and Medium Enterprises
SNMP	Simple Network Management Protocol
SR	Segment Routing
SW	Software
TAPI	Transport API
TC	Traffic Class
TDM	Time Division Multiplexing
TE	Traffic Engineering
TED	Traffic Engineering Database
TFS	ETSI TeraFlowSDN
TN	Transport Network
TSN	Time Sensitive Networking
TTL	Time To Live
UNI	User-to-Network Interface
VL	Virtual Link
VLAN	Virtual Local Area Network
VNF	Virtual Network Function
VPLS	Virtual Private LAN Service
VPN	Virtual Private Network
VPWS	Virtual Private Wire Service

VRF	Virtual Routing and Forwarding
VSI	Virtual Server Instance
WAN	Wide Area Network
WDM	Wavelength Division Multiplexing
WIM	WAN Infrastructure Manager
WP	Work Package
XR	Point-to-Multipoint coherent optical technology
ZTP	Zero-Touch Provisioning

1. Introduction

This document is an update of D2.1 [6], based on comments from WP3, WP4, and WP5. This deliverable provides a final version of the use case requirements, architecture, design, and data models that refer to ETSI TeraFlowSDN release 2.0. Furthermore, it also includes a business case analysis based on potential savings and new business opportunities for telcos. The final analysis of business cases will be provided in D6.4.

The ETSI TeraFlowSDN (TFS) 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 as well as provide revolutionary features for both flow management (service layer) and optical/microwave network equipment integration (infrastructure layer), while incorporating security using ML and forensic evidence for multi-tenancy based on DLT.

Use cases for IP and optical networks are addressed and demonstrated in commercial solutions based on standard interfaces. The network area for the solution is transport network scenarios integrated with (edge) computing and storage resources. ETSI TeraFlowSDN will adapt dynamically based on flows and applications. TFS covers a wide variety of networks, ranging from distributed edge-computing, through transport backhaul (including optical and microwave solutions), to the network core. TFS provides carrier-grade connectivity services for B5G networks.

2. Use Cases

This section reports use cases as well as extensions and updates to those already presented in D2.1 [6].

2.1. 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. Inventory use cases for optical and IP domains are further described in [18], [12].

2.1.1. 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 TFS Controller. These components might be line cards, transceivers, ports, etc.

2.1.2. 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 TFS 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 information regarding all logical protocols such as BGP, MPLS, and RSVP.

2.1.3. 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.

2.2. Topology

A set of abstractions has been defined to represent several views of the network topology. Context and Topology discovery use cases for optical and IP domains are further described in [18], [12]. There are two modes of operations:

- Polling mode - based on polling retrieval operations issued periodically and after each service creation to reconcile the actual state of the network.

- Event-triggered mode - based on an initial proactive synchronisation done from the NBI client module and a connection-oriented notification subscription session based on the NBI Notification mechanism.

In the following sections, we use “layer 0” to refer to transport using optical and microwave networks, “layer 2” to indicate transport in Ethernet networks, and “layer 3” for IP and MPLS packets. Multi-layer topologies include transport mechanisms that integrate multiple layers, 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, clusters vendor-specific devices, or administrative regions. A multi-domain topology includes the combined view of multiple SDN controlled domains.

2.2.1. Context and Service Endpoints Discovery

Technologies Involved	**Optical, IP, Microwave**
Type	Topology
Description	This use case retrieves all Service Endpoint [18] information available from the TFS Controller. It is intended to be performed by any NBI client controller, module, or application to discover the logical representation of the network as viewed by the TFS Controller

2.2.2. Single Layer Topology

Technologies Involved	**Optical, IP, Microwave**
Type	Topology
Description	This use case retrieves all topological information available from the TFS 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.

2.2.3. Multi-Domain Service Endpoint and Topology

Technologies Involved	**Optical, IP, Microwave**
Type	Topology
Description	This use case intends to define how the TFS Controller exposes a 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 TFS Controller to synchronise 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.

2.2.4. Multilayer IP and Optical Topology

Technologies Involved	**IP and Optical**
Type	Topology
Description	To represent the different relationships between IP and Optical network elements (physical or logical) to be consumed for different applications.

IP topology comprises node, ports, and L0/L2/L3 connections as shown in Figure 5.

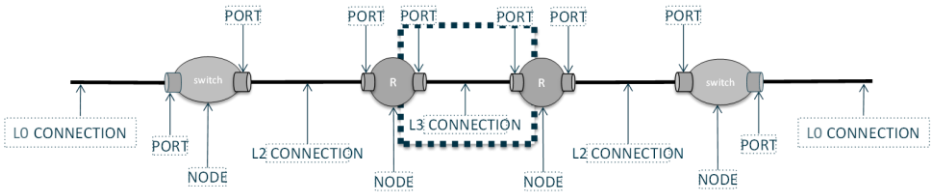


Figure 5 IP Topology

We consider two different topological views:

- Physical topology:
  - Nodes:
    - Physical Nodes
    - Geographical Location
  - Links:
    - Physical connections between routers (back-to-back connections)
  - Termination Points:
    - IP Interfaces
    - Loopback Interfaces
    - Ports
    - LAGs
- Logical topology: per-protocol topologies (e.g. OSPF, IS-IS, BGP,…), per-layer (e.g. IP connectivity, layer 2 connectivity),…

Topologies are stacked and related. One over the other (e.g., logical over physical, IP over optical).

RFC8345 (generic network YANG model) [30] provides the supporting-termination-point through the ietf-network:networks/network = {NetID}/node = {NodeId}/ietf-network:termination-point = {Tp-Id} attribute:

```
/nw:networks/nw:network/nw:node:
+--rw termination-point* [tp-id]
+--rw tp-id
+--rw supporting-termination-point
   +--rw network-ref
   +--rw node-ref
   +--rw tp-ref
```

A reference to an L0 network for a reduced TE Topology is added from RFC 8795 [25] to use the inter-domain-plug-id attribute based on a unique number that identifies, in the network, a connectivity supporting a given inter-domain TE link through ietf-network:networks/network = {NetID}/node = {NodeId}/ietf-network:termination-point = {te-tp-id}:

```
/nw:networks/nw:network/nw:node/nt:termination-point:
+--rw te-tp-id?   te-types:te-tp-id
+--rw te!
     +--rw inter-domain-plug-id?            binary
```

The implementation starts at the SIP where it is logically mapped to at least one topology NEP through the TAPI-topology:owned-node-edge-point/mapped-service-interface-point attribute:

```
/tapi-common:context:
      +--ro topology* [uuid]
         +--ro node* [uuid]
         |  +--ro owned-node-edge-point* [uuid]
         |  |  +--ro mapped-service-interface-point* [service-
interface-point-uuid]
         |  |  |  +--ro service-interface-point-uuid -> /tapi-
common:context/service-interface-point/uuid
```

From this requirement, it can be concluded that the Optical TE shows a SIP as the border between both worlds, and it must be augmented, with a new attribute called "inter-layer-plug-id" with its respective value as a "string", as presented in the following object:

```
/tapi-common:context:
      +--rw service-interface-point
      |  +--rw name* [value-name]
         +--rw [inter-layer-plug-id]
         +--rw value?                string
```

PLUG-ID Usage Example:

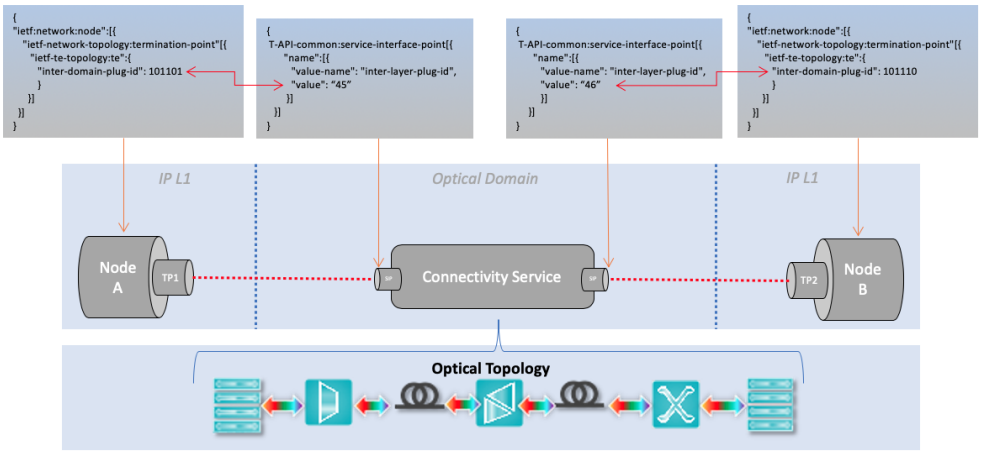


Figure 6 PLUG-ID Usage. Optical Domain, and Topology

TFS Target Objective	Objective 1.1 Accelerate innovation in transport (optical and microwave) and IP networks.
Research Actions	Part of Scenario 1 demo, to be demonstrated at OFC23.
TFS Architecture Update	RFC8345 [30] and the TAPI NBI in the core modules showing abstracted IP and optical topology views towards third party applications (e.g., external multilayer planning tools).

New Requirements

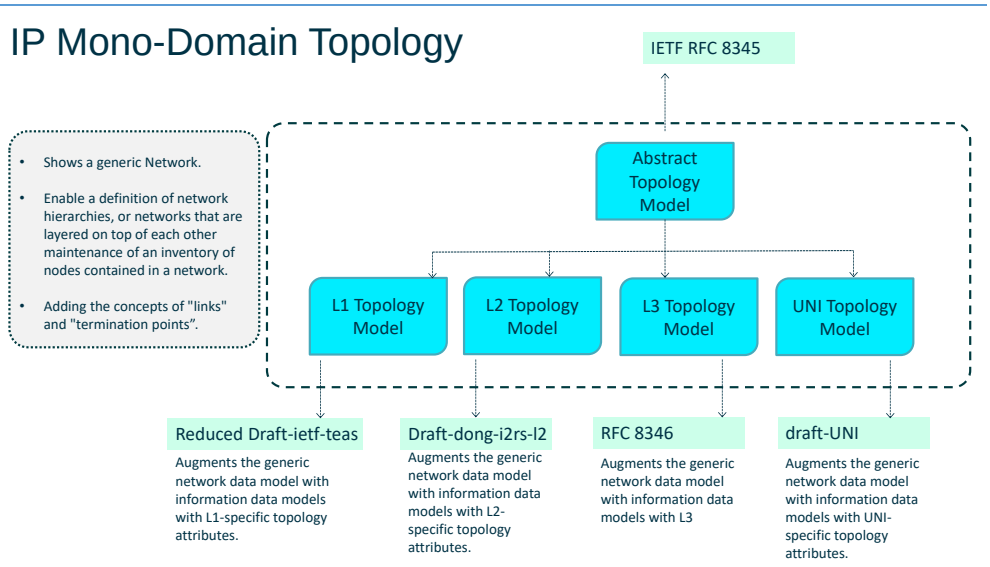


Figure 7 IP Mono-Domain Topology

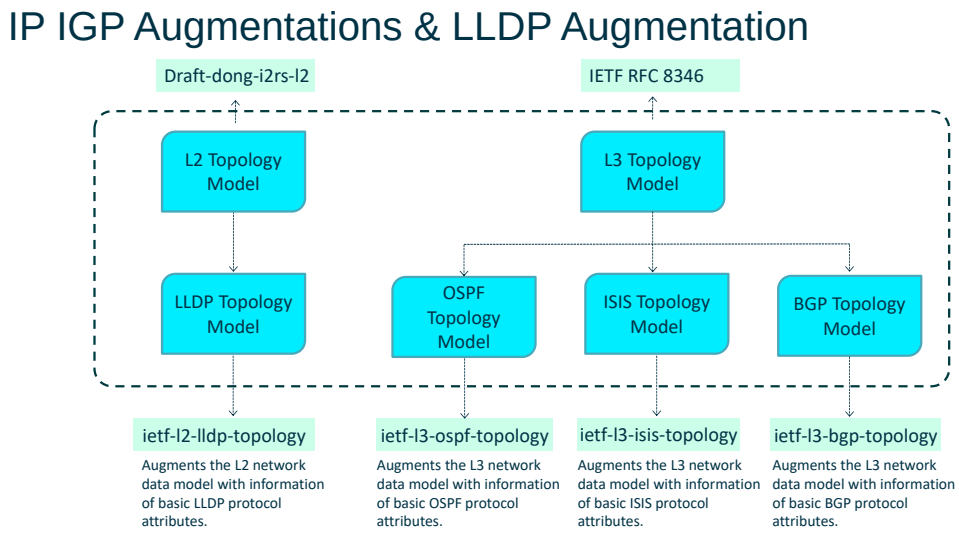


Figure 8 IP IGP and LLDP Augmentations

2.3. Context

In this section, we present two use cases that evolve the Context component architecture.

2.3.1. Operate TFS at Scale

Technologies Involved	**All**
Type	Context, Load Balancer
Description	This use case tests the TFS Controller under heavy request loading.
TFS Target Objective	Objective 1.2: Introduction of cloud-native flow management with a control plane latency below 10 milliseconds.

	Objective 2.1: Increase by an order of magnitude (x10) the flow processing capabilities of current SDN controllers. This results in the ability to handle a Tera of connectivity services. Provide resilience mechanisms for the TFS components (self-healing, auto-scaling, and load balancing).
New Requirements	Context Component needs to be able to replicate. This means that the internal database needs to be distributed.
Research Actions	Design of TFS as a cloud-native application. Detailed review of distributed cloud-native databases. Introduction of service mesh for micro-services.
TFS Architecture Update	No interfaces or sequence diagrams need to be updated.

2.3.2. Flow Descriptors for IoT Services

Technologies Involved	**IP, Optical**
Type	Context, Service, SBI

Description	Extend service protocol buffer definition in order to include the following information:
	• Timestamp, End-Point Location
	• ACL
	  ○ L0
	  ▪ Lightpath
	  ○ L2
	  ▪ Source-mac
	  ▪ Destination-mac
	  ▪ Ethertype
	  ▪ VLAN-id/VPLS
	  ○ L3
	  ▪ Source-address
	  ▪ Destination-address
	  ▪ DSCP
	  ▪ Protocol number (payload type)
	  ▪ Hop-limit (TTL)
	  ○ L4
	  ▪ Source-port
	  ▪ Destination-port
	  ▪ TCP flags
	  ▪ MPLS/SR
	  ▪ TC bits
	  ▪ TTL

	▪ Label values • Constraints o Time and Location o Duration o Calendar o End-Point Location o SLA Constraints o Isolation-level (e.g., disjoint paths, …) o E2E latency o Capacity o Recovery-path
TFS Target Objective	Objective 2.2: Ability to handle multiple technology flows (multi-RAT) and introduce the necessary flow descriptors for IoT services (across layers 2, 3, and 4). Introduce context-awareness in flow management. The introduction of follow-me network connectivity services shall generate at least a 10% reduction in the network flow requests that are generated due to network mobility.
New Requirements	Add support for constraints in the Service Component and the SBI. Add service constraints to path computation.
Research Actions	The use case will be demonstrated within the context of multiple scenarios. There is no specific research activity for the use case, but it will serve as a background for research activities related to specific flow descriptions and SLA validation.
TFS Architecture Update	It requires update of the protocol buffers used for Context. It requires dedicated handling of SLA constraints by Service and Path Computation Components.

2.4. SBI (Formerly the Device Component)

2.4.1. IP Routers Control and Management

Technologies Involved	**IP**
Type	SBI
Description	This use case considers the management and control of IP/MPLS routers which expose a Netconf API with OpenConfig YANG data models. The TFS controller will maintain a Netconf session with the device and will be able to: • Retrieve configuration and state data from the relevant data models of the device. • Edit configuration in the device to fulfil the different Service Provisioning, Traffic Engineering, Provisioning, Topology, and Inventory use cases.
TFS Target Objective	Objective 1.2: Beyond 5G integration with L3VPN/L2VPN up to the network edge. This objective requires a clear network programmability framework to provide cloud-scale network management capabilities. Objective 2.3: Adoption of novel protocols for inventory, alarms, telemetry, and provisioning. This objective includes the adoption of novel network equipment control and management paradigms, far beyond classical SNMP. The introduction of novel control and

	management interfaces, such as OpenConfig and gNMI, will provide the necessary benefits to TFS for supporting upcoming B5G networks.
New Requirements	Validation and verification of proper OpenConfig implementation of L3VPN services. L3VPN life cycle management, including use cases of SLA violation detection and recovery.
Research Actions	This use case will be demonstrated in Scenario 1, in demo paper for OFC23.
TFS Architecture Update	No architectural update needs to be performed.

2.4.2. Microwave Integration

Technologies Involved	**Microwave using SIAE MW Links**
Type	SBI and Service Component integration.
Description	Manage and configure MW devices to Add/Delete/Modify and monitor L3VPN E2E services involving routers at edge interfaces of the MW transport layer. Management of MW devices is achieved through an intermediate controller that exposes, in its NBI, the standard IETF Network Topology and Ethernet Transport Service models, and the ONF TR532 MW model.

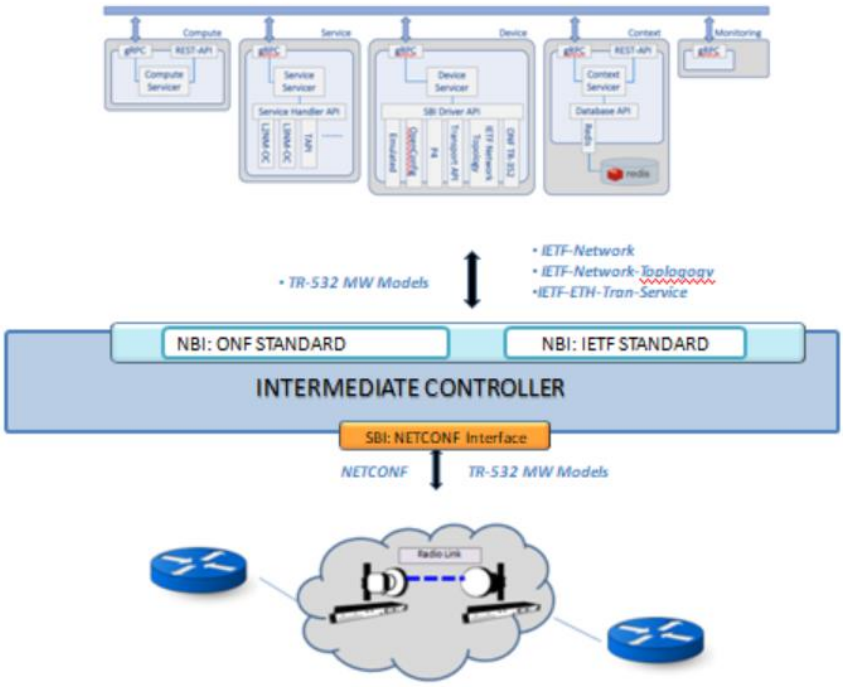


Figure 9 Microwave Integration

TFS Target Objective	Objective 1.1: Accelerate innovation in transport (optical and microwave) and IP networks, and ultimately help operators provide better connectivity for communities all around the world. Demonstrate the integration of the management of microwave transport networks in a multi-technology and multi-vendor environment.
New Requirements	Integrate MW links into the L3VPN workflow.

Research Actions	Possible demonstration in ICC Rome 23 or with ETSI ISG mWT Proof of Concept.
TFS Architecture Update	Several data models need to be updated, as the service protocol buffer needs to include: node_id_src, tp_id_src, node_id_dst, tp_id_dst, vlan_id. Device protocol buffer includes the following extensions: • MICROVAWE_RADIO_SYSTEM • DEVICEDRIVER_IETF_NETWORK_TOPOLOGY Include SBI support towards microwave SDN controller through IETF Network Topology YANG model.

2.4.3. Point to Multi-Point Integration of XR Optical Transceivers

Technologies Involved	**Optical (Point-to-multipoint DWDM with Infinera XR solution)**
Type	SBI
Description	This use case is for point-to-multipoint network optical network discovery and optical bandwidth allocations changes.

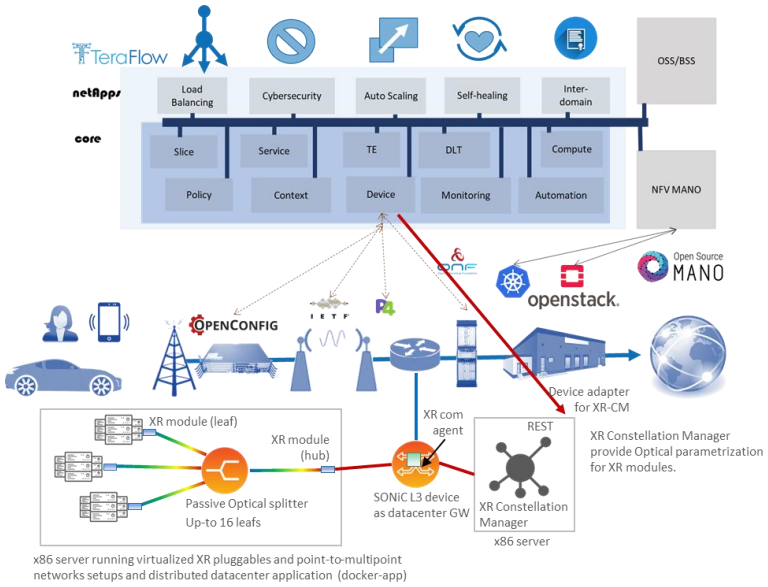


Figure 10 XR Optical Transceivers Integration

TFS Target Objective	Objective 1.1: Accelerate innovation in transport (optical and microwave) and IP networks, and ultimately help operators provide better connectivity for communities all around the world. Objective 2.3: Adoption of novel protocols for inventory, alarms, telemetry, and provisioning.
New Requirements	SBI and Service Component integration , and managing XR pluggables via the Infinera Intelligent Pluggable Manager (IPM).
Research Actions	Possible OFC23 regular paper, or OFC23 demo participation.
TFS Architecture Update	A new driver implementing the IPM REST API will be provided.

2.4.4. P4 Integration

Technologies Involved	Packet (Programming Protocol-independent Packet Processors (P4) domain-specific language).
Type	TFS device driver over software-based (i.e., bmv2 P4 switch) and/or hardware-based (i.e., Intel Tofino-2 switch or Xilinx Alveo SN-1000 SmartNICs).
Description	This use case demonstrates how TFS manages P4 devices.
TFS Target Objective	Objective 1.2: Adaptation of flow requirements by means of flow definition extensions and the introduction of network programable languages, such as P4. Objective 2.3: Adoption of novel protocols for inventory, alarms, telemetry, and provisioning.
New Requirements	**P4-01:** The P4 device driver requires a means to load an already-compiled P4 program, i.e., a pair of p4.info and a p4.json files. This will be made possible using the **settings argument of the device driver constructor. When the TFS P4 device driver is loaded, the P4 info and JSON files will be stored into the deployed instance of the driver, such that when a connection attempt is made towards a P4 device (i.e., using the Connect RPC), a new instance of the P4RuntimeClient will be created and the driver will be able to call the client's "set_fwd_pipe_config(p4info_path, p4_bin_path)" method. This method takes the two P4 files as input arguments and configures the P4 device through an RPC to the device's P4RuntimeServer. Obviously, this integration assumes that P4Runtime is the key dependency (TFS Controller on the client side, and the device on the server side). **P4-02:** After loading a given P4 program onto the target P4 device, the TFS P4 device driver should be able to configure the forwarding pipeline at runtime. This will be achieved using the SetConfig and DeleteConfig RPCs of the driver, which will be developed in a way that allows the TFS device driver to pass a list of forwarding rules for insertion or deletion respectively. **P4-03:** To monitor the runtime state of the pipeline, the TFS P4 device driver will employ the GetConfig RPC, which will fetch the installed rules from the switch and report them to the TFS Controller. **P4-04:** To allow zero-touch provisioning of P4 devices (provided by the Automation Component) when new, unconfigured devices appear in the network, the GetInitialConfig RPC method will be implemented. This RPC will return a minimal set of pipeline instructions, which, when enforced by the Automation Component (see the use case in Section 2.10) to a target P4 device, will allow this device to perform simple L2 forwarding, thus being immediately functional when it appears in the network. After this point, it is up to the network operator to install more advanced forwarding rules which may realise additional, and potentially more complex, network functions. With these new requirements in place, the TFS Controller will be able to fully-configure programmable P4 pipelines.
Research Actions	Euro P4 '22 demo or OFC '23 or P4 Workshop '23 or IEEE HPSR' 23 demo/tutorial.
TFS Architecture Update	The planned RPCs will be implemented in a way that no or only minor changes will be required to the device protobuf. Should any changes occur, they will be reported in the context of WP3 in MS3.3.

2.5. Services

In this section, we present use cases that evolve the Context component architecture. In the L3VPN for 5G Services, the nodeB (i.e., antenna) 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 region (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 can be applied in the network to transport and guarantee the right SLAs to the mobile customers mainly because they offer several traffic discrimination policies. Such L3VPNs have been typically statically configured, and with the adoption of modern protocols, they can now be dynamically configured on IP routers. IP routers are interconnected using underlying network elements, such as DWDM or MW transport networks. To provide E2E connectivity services, underlying network connections shall be controlled and managed [2].

2.5.1. Service-ACL Management

Technologies Involved	**IP**
Type	Service
Description	Configuration of network access control lists (i.e., filters, rules, etc.). ACLs are organised into ACL sets, with each set containing one or more ACL entries. ACL sets are identified by a unique name, while each entry within a set is assigned a sequence-id that determines the order in which the ACL rules are applied to a packet. Individual ACL rules specify match criteria based on fields in the packet, along with an action that defines how matching packets should be handled. Entries have a type that indicates the type of match criteria, e.g., MAC layer, IPv4, IPv6, etc. Functions Covered: Create the ACL: Configure the name or number for the ACL Provide a description Define the type (IPv4, IPv6, L2, MPLS, Mixed) Provide the ID (user defined identifier) Define the rules used to match the packet: Destination address DSCP Protocol Source address Destination port Source port Define the action for the matched packets: Forwarding-action (drop, accept, reject) Log-action (log-syslog, log-none) Define the sequence to apply the ACL (sequence-id) Apply the ACL to an interface (interface / sub-interface): Specify the direction to apply the ACL on the interface (ingress / egress) Set-Name • Type
TFS Target Objective	Objective 1.2: B5G integration with L3VPN/L2VPN up to the network edge.
New Requirements	OpenConfig extensions for Service ACL management.
Research Actions	Regular paper contribution at conference, in the scope of Scenario 1 or 3.

TFS Architecture Update	NBI support in core modules enabling standard integration with external provisioning tools through vendor agnostic APIs.

2.5.2. Service SLA Violation Detection and Recovery

Technologies Involved	**All**
Type	Service
Description	TFS provides continuous service monitoring and SLA guarantees. This use case provides the necessary requirements for SLA violation detection and service SLA guarantee.
TFS Target Objective	Objective 1.2: B5G integration with L3VPN/L2VPN up to the network edge. This objective requires a clear network programmability framework to provide cloud-scale network management capabilities.
New Requirements	TFS shall support SLA monitoring. TFS shall trigger a Policy, when SLA violation is detected. TFS shall provide service restoration using a break-before-make strategy.
Research Actions	To be demonstrated in Scenario 1, in OFC23 demo.
TFS Architecture Update	New workflow sequence needs to be included, considering the necessary control loop, between Monitoring, Policy and Service.

2.5.3. Location-Aware Context Management

Technologies Involved	**All**
Type	Inter-domain
Description	Service endpoints can be updated to support follow-me connectivity.
TFS Target Objective	Objective 2.2: Introduction of a new architecture to support massive IoT and new mobility paradigms. Objective 3.3: Support for context-awareness and follow-me applications in cross-border scenarios. Introduce context-awareness in flow management. The introduction of follow-me network connectivity services shall generate at least a 10% reduction in network flow requests that are generated due to network mobility.
New Requirements	Requested services can be constrained to a certain region or locations. Services can be updated due to change in endpoints.
Research Actions	To be demonstrated with Scenario 2.
TFS Architecture Update	Need for an updated service workflow. Inclusion of location in Endpoints and Location Constraints in Services.

2.5.4. L3VPN Service Provisioning

Technologies involved	**IP/MPLS**

Type	SBI, Service
Description	This use case allows the provisioning, modification, and deletion of an L3VPN service spanning one or more IP/MPLS routers via the TFS Controller using the L3NM data model (RFC 9182) [3]. The L3VPN creates a virtual routing network instance (usually known as VRF) in each of the routers involved in service deployment. The routing instance (VRF) created at every router allows routing information propagation between all of the sites involved in the service. The main functionalities covered in this use case are the following: • Provide a name and a description for the new L3VPN service. • The type and data plane encapsulation in this use case is limited to BGP based L3VPNs (type=L3VRF) over MPLS (encapsulation-type=MPLS). • Set the route distinguisher (RD) that should be used each VRF when it is signalled via BGP. In this use case, the RD has to be explicitly provided. • It is possible to configure a router-id to identify the routing device and used by BGP and OSPF to function in a routing instance. • Enable/disable the configured network instance on the network element. Note that some vendors' implementations do not allow disabling the configured network element so that the network instance will be enabled immediately after creation and kept enabled until it is deleted. • Label allocation (per prefix, per next-hop, or a single label per VRF) should be set to single label per VRF. • Enable the Address Families (AF) supported within the L3VPN. Note that some vendors may enable all families by default. • Configure Route Target for export and import. The controller will take care of creating the policies in the device automatically. • Create vpn_node (VRF) profiles to reuse when there are multiple VRFs. VPN topologies can be indicated (hub & spoke, full mesh, custom) as informative. The topology is achieved via a Route Target (RT) assigned for import and export. • Attach sub-interfaces to be bound to the L3VPN at L2 (single or double tagging, LAG members if applicable) and configure L3 information (IP address, loopback type interface). The physical interface configuration is out of scope of this use case. • Attach existing routing policies to the VPN for import and export • Control the VPN lifecycle using status variables such as pre-deployment, testing, or up. • Configure CE-PE routing, including static routes.
TFS Target Objective	Objective 1.2: B5G integration with L3VPN/L2VPN up to the network edge. This objective requires a clear network programmability framework to provide cloud scale network management capabilities. Objective 1.3 Automated service management for transport network slices.
Requirements	The device driver needs to send the necessary Netconf/OpenConfig queries to the IP/MPLS routers to create the service, VPN nodes, VPN access, and to add the corresponding routing protocols. The L3VPN VPN API exposing the L3NM data model can be consumed internally by TFS Apps (e.g., can be used to produce a network slice, or can be fully automated) or externally by an OSS.
Research Actions	To be demonstrated in Scenario 1 and demonstrate cloud-native features of the TFS Controller components (auto-scale, self-heal, load-balancing).

TFS Architecture Update	Novel component.

2.5.5. L2VPN Service Provisioning

Technologies involved	**IP/MPLS**
Type	SBI, Service
Description	This use case allows the provisioning, modification, and deletion of a Layer 2 VPN service spanning one or more IP/MPLS routers via the TFS Controller using a subset of the L2NM data model (RFC 9291) [4]. The types of L2VPN in scope are: Virtual Private LAN Service (VPLS) (RFC4761 [31] and RFC4762 [32]) Virtual Private Wire Service (VPWS) (Section 3.1.1 of RFC4664 [33]) Various flavours of EVPNs, such as VPWS EVPN (RFC8214 [34]) and EVPN over MPLS (RFC7432 [35]). The functionalities of the use case include • Create a new service with a name and indicate the type (VPLS, VPWS, EVPN) • Indicate the endpoints of the service • Attach the interface/sub-interface to the VPN at every node • To control the VPN lifecycle using status variables such as pre-deployment, testing, or up.
TFS Target Objective	Objective 1.2: B5G integration with L3VPN/L2VPN up to the network edge. This objective requires a clear network programmability framework to provide cloud scale network management capabilities.
Requirements	TFS customers may use various means to request a service that may trigger the instantiation of an L2NM. Customers may use L2SM (e.g., from the NBI Component) or more abstracted models such as a Network Slice Service (from the Slice Component). In both cases, L2NM may also be exposed directly for an operator's OSS. TFS needs to provide the necessary Netconf/OpenConfig commands to create/modify/delete the L2 VPN service.
Research Actions	To be demonstrated in Scenario 1 and demonstrate cloud-native features of the TFS Controller components (auto-scale, self-heal, load-balancing).
TFS Architecture Update	Novel component.

2.6. Forecaster

2.6.1. Traffic Forecast

Technologies Involved	**All**
Type	New component (Forecaster)
Description	Forecaster is a new component which is able to perform proactive SDN traffic optimisation 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, so as to reroute traffic before it happens).

TFS Target Objective	Objective 1.3: Proactive SDN traffic optimisation 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, so as to reroute traffic before it happens). Reduction by 25% of resource usage due to ML-based traffic optimisation.
New Requirements	A component shall be able to obtain the history of requested and serviced connectivity services with duration and capacity constraints. Forecaster shall include ML algorithms (prophet or AutoML) for predicting traffic forecasts. Traffic forecasts shall be analysed before determining whether to accept a new service request.
Research Actions	ICC regular paper.
TFS Architecture Update	New component. Context data models shall be extended including service timestamp, service duration, and service capacity. Policy Component requires a new action for service update.

2.7. Monitoring

The high density and heterogeneity of B5G networks will require extensive monitoring capabilities in order to gather the maximum number of metrics for network management, failure detection, and network optimisation purposes. For that reason, a highly scalable monitoring system is required by the TFS Controller, supporting novel monitoring, telemetry, and alarm interfaces, in order to cope with the stringent requirements of B5G systems.

2.7.1. Monitoring-Telemetry Management

Technologies Involved	**IP**
Type	Monitoring
Description	This use case will undertake the introduction of novel monitoring, telemetry, and alarm interfaces that will provide the necessary benefits to TFS for supporting the requirements of forthcoming B5G networks. This use case will demonstrate the integration of gNMI in the monitoring and telemetry procedures. The adoption of gNMI will allow the use of standards-defined interfaces using YANG and protocol buffer data models in addition to increasing the efficiency, reducing control latencies, and improving the network resource usage. This use case will also demonstrate the subscription and alarm system of the Monitoring Component, that is required for the proper operation of the rest of the TFS components.
TFS Target Objective	Objective 2.3: Adoption of novel protocols for inventory, alarms, telemetry, and provisioning.
Research Actions	Part of Scenario 1 demonstration.
TFS Architecture Update	Develop the required extensions in the TFS components to support gNMI dial-in monitoring. The required extensions are highlighted in red in Figure 11, while the extensions that are already developed are shown in green. The required extensions include the development of a new gNMI driver with a gNMI collector and integration with the already available monitoring loops.

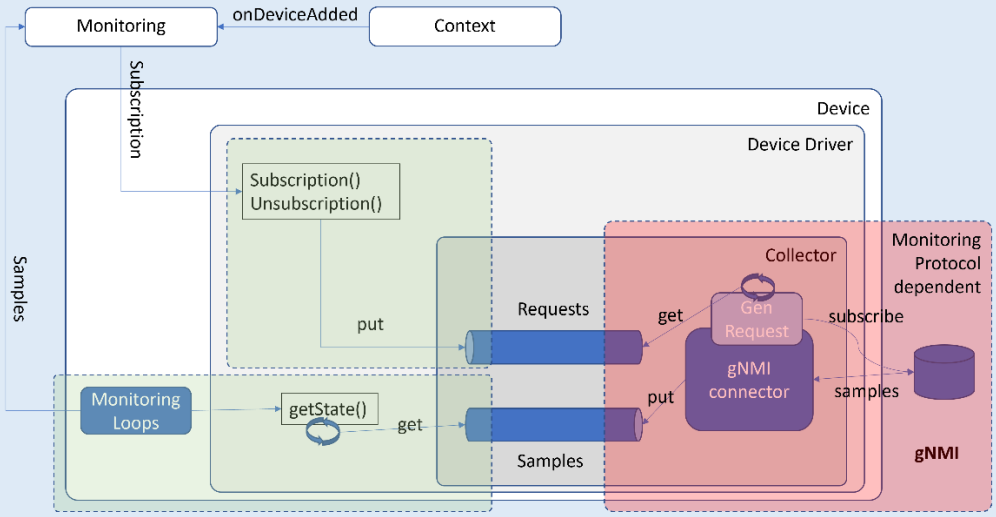


Figure 11 gNMI Telemetry Architecture

New Requirements	Develop a gNMI driver in the SBI, including a gNMI collector. Integrate the gNMI collector with the monitoring loops already developed in the SBI.

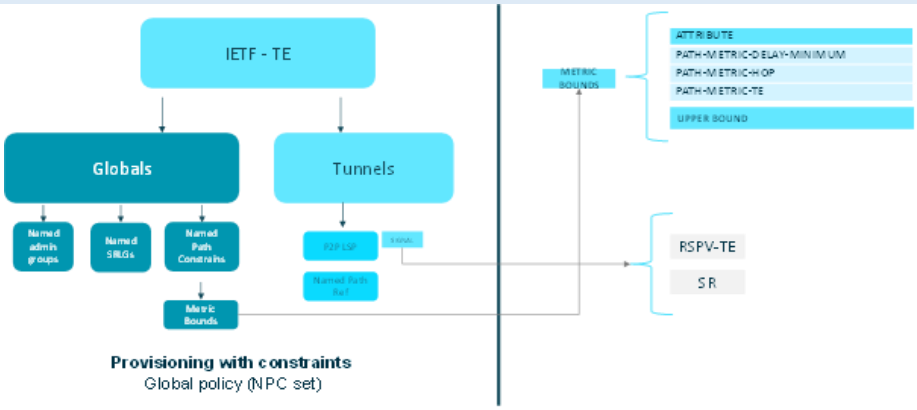
2.8. Traffic Engineering

The Traffic Engineering (TE) Component allows the enforcement of traffic steering by leveraging MPLS tunnels or Segment Routing paths. This makes it possible to increase the efficiency of network resource usage 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 PCE function has been defined to allow the performance of complex constraint-based path computation over a network graph representation. This improves the application of TE policies in G/MPLS networks. Based on these functionalities, one of the primary purposes of TE is to reduce overall operating costs through more efficient network resource usage, including link occupation, traffic rerouting, and network availability [13].

Inter-domain path computing was previously done by the PathComp Component, and for each domain in this path, the segment is delegated to the TE component for in-domain optimisation. The TE component subscribes to the relevant monitoring events regarding SLA compliance and may preventively modify the LSPs/SR paths to avoid SLA breaches.

2.8.1. PCEP for Segment Routing

Technologies Involved	**IP**
Type	PCEP for Segment Routing (path creation, modification, and deletion with SR).
Description	This use case consists of creating, modifying, and deleting segment routing paths on the available hardware for a given domain, considering specific constraints and the available resources. For example, the constraints given to the PCE for the calculation of the path could be required latency, bandwidth consumption, hop count, and whether the result should be a strict explicit path or a loose one.
TFS Target Objective	Objective 1.3: Automated service management for transport network slices.
Research Actions	Possible demo at NetSoft23.

TFS Architecture Update	NBI support in TE core module enabling standard integration with external orchestrators.
New Requirements	 *Figure 12 PCEP for Segment Routing New Requirements*

2.9. Path Computation

The PathComp Component is an element in the TFS Controller handling the route and transport resource selection for the incoming network connectivity services. It operates as a server, receiving requests from the Service Component and, interacting with the Context Component, retrieves information about the underlying topologies to seek feasible paths and resources fulfilling the demanded network service requirements, e.g., minimum bandwidth, maximum permitted latency, path disjointedness, etc. The PathComp Component is envisioned to support heterogeneous algorithms tackling diverse objective functions such as inter-domain path computation, i.e., deploying connectivity services with endpoints located at different domains, or energy-aware traffic routing aimed at rolling out the connectivity services with reduced energy consumption.

2.9.1. Support for Path Computation

Technologies Involved	**Inter-domain, Context Component, Service Component, Slice Component, Circuit-switched networks**
Type	Path and transport resource algorithms with multi-objectives and constraints
Description	The PathComp Component functionalities are queried on-demand by the Service Component at the time of provisioning a new network connectivity service or set of services. It could be also queried when updating an active network connectivity service whose requirements need to be modified. The inputs to the PathComp Component are: • Endpoints (e.g., PEs) to be interconnected • Service requirements, e.g., bandwidth, latency, path disjointedness, etc. • Algorithm identifier which is bound to a specific objective to be addressed by a determined algorithm, e.g., attain an efficient use of the transport resources, reducing the overall transport network energy consumption, etc. • Context information containing details of the involved topologies, connectivity, device and link attributes and constraints, etc. The PathComp Component may include a pool of specialised algorithms one of which is picked according to the algorithm identifier carried in the request. The output of the PathComp Component specifies: • No Path when the algorithm is unable to find a feasible path and transport resources fulfilling the network service requirements

- The explicit path and transport resources through which the connectivity service needs to be set up

The planned set (not exhaustive) of algorithms to be offered in the PathComp Component includes:

- Constrained K-Shortest Path (Yen) for handling different additive metrics, e.g., hops and maximum latency
- Energy-aware routing
- Disjoint working/backup path computations

The planned implementation (short term) is to enable a front-end (written in Python) to enable interaction with other TFS components (e.g., Service and Context) via a gRPC API. The back-end (written in C) interacts with the front-end using a REST API. The back-end is used to support "fast" and "intensive" execution of any selected algorithm. For the sake of completeness, a long-term future task will revisited the necessity of having a front-end/back-end architecture of the PathComp Component.

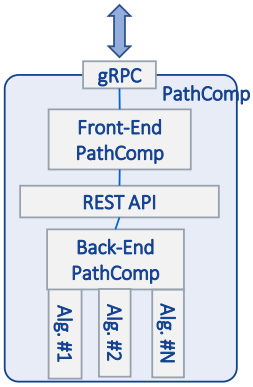


Figure 13 Internal Architecture of Path Comp Component

TFS Target Objective	The PathComp Component aims to offer a dedicated entity for computing the path/route and transport resources fulfilling diverse connectivity service requirements and tackling heterogenous network objectives. This specific functionality may be computationally intensive and should scale well with the size of the Context (underlying transport domains). Moreover, the PathComp Component represents a single and specialised entity where different algorithms can be hosted. This permits that any new algorithm to be used does not impact on other TFS components (e.g., Service). To this end, the PathComp Component relies on well-defined workflows and APIs to interact with other TFS components such as the Service and Context Components.
New Requirements	The PathComp component is a new core component being developed in the TFS Controller. The identified requirements are: • Interaction with the Service Component via gRPC (request and respond PathComp) specifying the endpoints, algorithm identifiers, service identifier, and service constraints • Interaction with the Context Component via gRPC (request and respond GetContext) to retrieve the Context (topologies, devices and links attributes, etc.) used as input to trigger the selected path computation and resource selection • Definition of the algorithms dealing with the identified network objectives within the project, e.g., energy-oriented routing, disjoint-path computation, etc.
Research Actions	The PathComp Component offers a complementary functionality (i.e., automatic path computation and resource selection) needed to perform operations such as the provisioning/updating of network connectivity services, slices, etc. An important aspect is that this function can be used to fulfil network service demands and also to target interesting networking objectives such as the reduction of energy consumption. Thus, the PathComp

	Component would be part of different research actions where the route and resource selection are key aspects.
TFS Architecture Update	The introduction of the PathComp Component will require that it interacts with the Service and Context Components. The interactions with the former (based on gRPC) will enable handling the PathComp request/response messages for selecting the route and the resources meeting an incoming network connectivity request. On the other hand, the interactions with the Context Component allow the PathComp Component to retrieve the topology information including the device and link attributes. This information is essential to trigger a selected routing algorithm triggered within the PathComp Component.

2.9.2. Add Service Constraints for Path Computation

Technologies Involved	**Calendar, disjoint, capacity, latency, Segment Routing, MPLS, …**
Type	Context, Service, SBI

Description	Extend service protocol buffer definition in order to include the following information:
	• L2VLAN-id/VPLS
	  ○ MPLS labels
	• L3
	  ○ DSCP
	  ▪ Protocol number (payload type)
	• L4
	  ○ MPLS/SR
	  ○ Label values
	• Constraints
	  ○ Time and Location
	  ○ Duration
	  ○ Calendar
	  ○ End-Point Location
	• SLA Constraints
	  ○ Isolation-level (e.g., disjoint paths,…)
	  ○ E2E latency
	  ○ Capacity
	  ○ Recovery-path

TFS Target Objective	Objective 2.2: Ability to handle multiple technology flows (multi-RAT) and introduce the necessary flow descriptors for IoT services (across layers 2, 3, and 4).
New Requirements	Add support for constraints in Service Component and SBI. Add service constraints to path computation.
Research Actions	To be demonstrated in Scenario 1 (restoration and intra-domain SLA enforcement) and Scenario 2 (inter-domain SLA enforcement).
TFS Architecture Update	It requires update of the protocol buffers of Context. It requires dedicated handling of SLA constraints by Service and Path Computation Components.

2.10. Automation

2.10.1. Automated Provisioning of P4 Devices and Programs

Technologies Involved	**IP using the Programming Protocol-independent Packet Processors (P4) domain-specific language.**
Type	Automation
Description	Automation Component to provide zero-touch provisioning of P4 devices (i.e., automation of P4 integration described in Section 2.4.4).
TFS Target Objective	Objective 1.2: Adaptation of flow requirements by means of flow definition extensions and the introduction of network programable languages, such as P4. Objective 2.3: Adoption of novel protocols for inventory, alarms, telemetry, and provisioning.
New Requirements	**AUT-01:** Automatically provision a P4 program in addition to the device itself. To meet this requirement, the following prerequisites should be met: • The P4 device driver should be able to connect to a P4 device (Connect RPC) and load a desired (pre-compiled) P4 program as described in Section 2.4.4. - The P4 device driver should implement the GetInitialConfig RPC, which allows the Automation Component to auto-provision a new P4 device through the ztpAdd RPC. Specifically, the Automation Component will fetch the Device object from the Context Component and then call the Device object's GetInitialConfig RPC to receive an initial configuration for this device. Once this configuration is received, the Automation Component will update its local Device object with the received configuration and attempt to set this configuration to the device. Upon success, the device will make its new configuration available through the Context Component. **AUT-02:** The ztpUpdate RPC will be implemented, to support provisioning automated configuration updates onto a target device. This RPC will be given a DeviceID and a list of configuration rules which will comprise the update. Upon invocation, the Automation Component will fetch the Device object from the Context database, apply the updated configuration rules to the local Device object, and call the ConfigureDevice RPC to enforce the updated configuration to the target device. This will also result in an updated Device object being automatically populated into the Context database. In the context of automated provisioning of P4 devices, this method will assist the ztpAdd with runtime device updates. **AUT-03:** The ztpDelete RPC will be implemented, aiming at automatically removing certain configuration from a target device. This RPC will follow an identical workflow to the ztpUpdate RPC, except that it performs deletion, rather than update of a given configuration. In the context of automated provisioning of P4 devices, this method will assist the ztpAdd with runtime device configuration removal.
Research Actions	Euro P4 '22 demo or OFC '23 or P4 Workshop '23 or IEEE HPSR'23 demo/tutorial
TFS Architecture Update	To realise AUT-02 and AUT-03, the ztpUpdate and ztpDelete RPCs may be updated with an additional input argument that describes the updated/deleted configuration to be applied to the device. The automation.proto will be updated accordingly. The rest of the RPCs will be implemented in a way that no or only minor changes will be required to the device protobuf. Should any changes occur, they will be reported in the context of WP3 at the upcoming MS33.

2.11. Policy

2.11.1. Automated SLA Enforcement

Technologies Involved	**IP**

Type	QoS Management
Description	The TFS Policy Management Component provides means to schedule network management operations in response to events, based on the event-condition-action policy model drafted by the IETF [10]. Such policies can be applied either at the level of individual devices (i.e., device-level policies) or across entire network domain (i.e., network-wide policies). **Policy Conditions** TFS provides a powerful mechanism to express policy conditions encoded as AND/OR-separated policy condition patterns, where each pattern is in turn expressed in the form of: Monitoring.KPI, numericComparisonOperator, Monitroing.KPIValue. For example, a policy condition could check whether "PacketLoss > 2%" on a given connection/service. To realise such policy conditions, the Policy Management Component relies on services provided by the Monitoring Component. Specifically, the alarm subsystem of the Monitoring Component (planned for the TFS 2.0 release) allows external entities to create conditional queries on monitoring data. When these conditions are met (e.g., a Monitoring KPI exceeds a specific value, falls within a certain range, etc.) the Monitoring Component raises alarms which are sent back to the origin component (i.e., the component that requested the conditional query) for internal consumption. When the Policy Management Component receives such an alarm on a conditional query, it triggers an action as a response. **Policy Actions** Reacting to (conditional) policy events means allowing the TFS Controller to apply certain actions to restore a given set of devices or services back to a desired state. The Policy Management Component offers a modular set of actions which can be technically expressed as enum action items mapped to (i) RPC calls to other TFS components (e.g., SBI, Service, etc.), and (ii) parameters to pass to these RPC calls so as to enforce the desired action.
TFS Target Objective	Objective 2.3: Adoption of novel protocols for inventory, alarms, telemetry, and provisioning
Research Actions	OFC '23 demo or IEE HPSR'23 demo
TFS Architecture Update	An updated policy.proto is part of the develop branch on GitLab. A new policy-condition.proto is part of the develop branch on GitLab. A new policy-action.proto is part of the develop branch on GitLab.
New Requirements	To be able to apply QoS-related policies, data plane devices should be configured accordingly. For this to happen, OpenConfig extensions are planned for QoS management in IP devices.

2.12. Transport Network Slicing

2.12.1. Slices with SLAs

Technologies Involved	**All**
Type	Slice
Description	A slice request will include a set of end-points, several connections between them, and a group of SLOs constituting the full SLA. The constraints described by the SLA shall be preserved during the slice lifespan.
TFS Target Objective	Objective 1.3: Automated service management for transport network slices.
New Requirements	Components shall provide slices and interface in terms with IETF draft-ietf-teas-ietf-network-slices. [21]

	Slice SLAs shall be mapped as technology-agnostic intents, regardless of the underlying implementation (e.g., L2VPN, L3VPN). Slices, once deployed, shall be monitored and enforced, in terms of SLA constraints.
Research Actions	To be tested and demonstrated in Scenario 1.

TFS Architecture Update	Extend slice data model or select from existing model (to be further discussed, best candidate for slice service request is draft-ietf-teas-ietf-network-slice-nbi-yang [29], realisation is proposed to be done using existing implementation of L2/3VPN services, Timestamp, End-Point Location.
	
	Required parameters for slice service request:
	• Timestamp
	• Slice tenant/owner
	• Set of end-points
	• Set of connections between end-points
	• Duration (may be indefinite)
	• SLA constraints (as a set of SLOs), e.g.:
	  ○ Guaranteed bandwidth
	  ○ Availability
	  ○ Maximum E2E latency
	  ○ Maximum packet drop
	  ○ Isolation level

2.12.2. Optimal Slice Grouping for Multiple Tenants

Technologies Involved	**All**
Type	Slice
Description	Similar slice requests can share underlying services. Clustering algorithm for slice grouping. Consider both paths and SLA constraints. SLA monitored by slice group.
TFS Target Objective	Objective 3.2: Provisioning of multi-tenant transport network slices. Improve network resource usage by 30% by adopting multi-tenancy resource allocation algorithms. Optimal slice grouping: trade-offs between economies of scale and limitations as to which SLAs can be grouped together need to be considered. Optimal grouping of slices is required to maximise KPIs, such as resource utilisation, utility of the connectivity, and energy efficiency. In this context, trade-offs between the resulting control plane complexity and differential treatment of SLA classes should be considered.
New Requirements	User can select if slice grouping is performed per-slice request. Slice grouping introduces a clustering algorithm for finding service optimisation while preserving slice SLA. Service (re-)optimisation is provided.
Research Actions	Research paper Q4 2022.
TFS Architecture Update	Update Slice service RPC to include Slice Grouping. Use novel Slice model with SLA constraints. Use Policy Component with action to update services to apply slice grouping. Describe Slice service operation modes: per-request or user-triggered.

2.13. Centralized Attack Detector

2.13.1. Scalable Attack Detection

Technologies Involved	**L3 and Optical ML-based detectors**
Type	ML-based security
Description	This use case introduces scalable and reliable security assessment of the services established using TFS. This use case focuses on two different types of service: optical and IP. For optical services, the Centralized Attack Detector (CAD) must collect monitoring samples made available by coherent optical transceivers, and use this information to detect and possibly identify attacks. For this type of service, security monitoring cycles are executed periodically (e.g., every 30 seconds, every 1 minute). The monitoring samples are collected through the device driver and stored in the monitoring component. For IP services, the CAD receives per flow statistics summaries from the Distributed Attack Detector (DAD). The CAD stores the flows statistics received within a certain time window (which can be parametrised) in a buffer. A ML model classifies each flow in the buffer as normal traffic or as being part of an attack, and a confidence level in the decision is obtained. If a flow is detected as being part of an attack with a confidence level equal or greater than a configurable threshold, the CAD sends the attack flow information to the AM Component in order to trigger an adequate action to mitigate the detected attack. In addition, for monitoring purposes, security monitoring cycles are executed periodically (e.g., every 5 seconds).
TFS Target Objective	Objective 4.1: Cyberthreat analysis and protection
New Requirements	Upon the creation, update, or deletion of services, the CAD must set up appropriate background information within TFS (e.g., new monitoring KPIs). For the optical service, at the end of a security assessment cycle, the security status associated with the services under monitoring must be reported as KPIs of the TFS Monitoring Component. For the IP service, no new requirements have been considered.
Research Actions	Will be provided as part of Scenario 3.
TFS Architecture Update	For the optical service, the interface provided by the Monitoring Component will be extended to provide functionalities needed to realise this use case. A new argument (service_id) has been added to L3AttackmitigatorOutput which is the object used to send the attack detection information from the CAD to the AM Component with the RCP function SendOutput. The new argument is used to provide the AM Component with the identifier of the service where the attack was detected. This identifier is later used by the AM Component to enforce the mitigation action only in the network service represented by the service_id. Another argument called endpoint_id has also been added to keep track of the endpoint the attack came from in order to take it into account when performing the mitigation.

2.14. Distributed Attack Detector

2.14.1. Edge-Based Attack Detection

Technologies Involved	**L3. ML-based feature aggregator**
Type	ML-based security
Description	The Distributed Attack Detector (DAD) monitors the network data plane for the presence of malicious network flows and is deployed at the network edge to improve scalability and response time in the attack detection process and enable real-time detection of malicious traffic. To this end, a feature extractor is deployed at the network edge to collect and generate statistical summaries of network flows. To do this, packets are aggregated into flow-level

	statistics, where each flow is an aggregate of packets belonging to the same packet flow (same source IP address, source port, destination IP address and destination port). This aggregation is performed for all new packets that arrive within a specific time window which can be configured. In this way, each aggregation of flow statistics is sent to the CAD to detect malicious traffic.
TFS Target Objective	Objective 4.1: Cyberthreat analysis and protection
New Requirements	New component definition and distributed allocation.
Research Actions	Part of scenario 3 demonstration.
TFS Architecture Update	A new argument (service_id) has been added to sendFlow which is the RPC used to send the network connection information from the DAD to the CAD. The new argument is used to provide the CAD with the identifier of the service to which the connection belongs to. A new argument (service_id) has been added to L3CentralizedattackdetectorMetrics which is the object used to send the network connection information from the DAD to the CAD with the RCP function SendInput. The new argument is used to provide the CAD with the identifier of the service to which the connection belongs. Another argument called endpoint_id has also been added to keep track of the endpoint the attack came from in order to take it into account when performing the mitigation.

2.15. Attack Inference

2.15.1. Detection and Identification of Attacks

Technologies Involved	**Optical**
Type	ML-based security
Description	This use case refers to the ability of performing inference over monitoring samples using AI/ML techniques and models in order to detect (and possibly identify) attacks being launched over services provisioned using TeraFlowSDN.
TFS Target Objective	Objective 4.1: Cyberthreat analysis and protection.
New Requirements	The AI/ML model shall not require any prior knowledge of the attacks it is detecting. Therefore, unsupervised or semi-supervised learning techniques will be adopted.
Research Actions	Possible demo at OFC'23.
TFS Architecture Update	A new module was created within the Cybersecurity Component to accommodate the functionalities of this use case. The gRPC interface was defined by a new protobuffer file.

2.16. Attack Mitigator

2.16.1. Mitigating the Execution of Attacks

Technologies Involved	**Optical, L3**
Type	ML-based security
Description	The Attack Mitigator (AM) Component prevents the execution of attacks identified by the CAD. The CAD detects per connection if an attack occurs in the network. If an attack is detected, the CAD sends a notification to the AM Component indicating the details of the attack.

	If the attack detected is at Layer 3, the CAD includes the source and destination of the attack flow and the confidence level of the detection, as well as other relevant information about the detection (timestamp, ML model identifier, flow identifier, etc.). If the attack detected is at the optical physical layer, the CAD includes the service identifier of the affected service. The role of the AM Component is to instruct some core components of TFS to enforce adequate actions that can mitigate the attacks on the network. For example, in case of detection of an attack at Layer 3, the AM Component could instruct the Service Component and SBI to drop the malign connection. To achieve this goal, the AM Component communicates with the Context Component to update the service on which the attack has been detected in order to create a new ACL ruleset in the service, and to implement a new ACL rule on the devices traversed by the attack connection. If the attack is detected in an optical service, the AM Component will mitigate the attack by using a make-before-break strategy. This strategy consists in creating a new service that uses a different set of resources, i.e., a link-and-node-disjoint path, moving the traffic into the new optical service, and deleting the old (under-attack) service.
TFS Target Objective	Objective 4.1: Cyberthreat analysis and protection.
New Requirements	Integration with Service and Context Components (and collaterally with the SBI and physical routers).
Research Actions	To be discussed with partners during the plenary meeting.
TFS Architecture Update	When the CAD detects that a connection to the TFS Controller is part of an attack, the CAD will inform the AM Component about the detection. For this purpose, a new RPC has been created to send the information about the attack detection, including the source and destination addresses, and the confidence level related to the attack detection, as well as other relevant information about the detection (timestamp, ML model identifier, flow identifier, etc.). When the information of an attack flow is received, the AM Component will trigger the mitigation procedure. To decide which action should be applied to the packets sent by the marked source address to mitigate the attack, the AM Component will call the GetMitigation method, which will return the most adequate mitigation action to be applied to the packets. In the release 2.0, this mitigation action will only consist of discarding packets transmitted from the source address (i.e., dropping the connection). In addition, using the identifier of the service where the attack was detected, the AM Component will update the corresponding service where the attack was detected by communicating with the Context and Service Components to create a new ACL ruleset in the service, and to implement a new ACL rule in the devices traversed by the attack connection in order to perform attack mitigation. To this end, a new RPC between the AM and Context Components is added (UpdateService) to describe the ACL rule to be implemented by the Service Component.

2.17. Distributed Ledger and Smart Contracts

2.17.1. Advancing the State of the Art of the Core Building Blocks of Blockchains

Technologies Involved	**All**
Type	DLT
Description	Improving scalability of blockchain. Improving smart contract security.
TFS Target Objective	Objective 4.2: • Optimised consensus algorithms for permissioned ledgers that scale above 100 nodes.

	• Automated patching of smart contracts.
New Requirements	-
Research Actions	Developing a new, generic methodology for scaling permissioned blockchains (associated research paper published at ACSAC'21). Developing a novel automated tool for the online detection of smart-contract vulnerabilities.
TFS Architecture Update	Research to be performed.

2.17.2. Using DLT for Inter-Domain Service Provisioning and SLA Violation Detection

Technologies involved	All
Type	DLT
Description	DLT Connector is the DLT Component of TFS. Through it, different DLT records can be stored in the blockchain, including events from SBI, Service Component, or Slice Component. The DLT Connector implements a generic protocol buffer offered by the DLT gateway to operate JSON encoded messages in the blockchain.  *Figure 14 Sequence Diagram for DLT* The sequence diagram is depicted in Figure 14.
TFS Target Objective	Objective 4.2: Distributed ledger technologies. In particular, the ledger will use blockchain mechanisms to protect data related to network resources and services of the various stakeholders. Another building block of the distributed ledger is dedicated smart contracts (which TFS will also provide) for accessing and updating the blockchain.
New Requirements	TFS user can select the amount of information that is shared through the blockchain. The TFS DLT Connector interacts with a DLT gateway using a generic protocol buffer and gRPC, using JSON encoded records. The TFS DLT Connector allows stream reception of new elements from the blockchain.
Research Actions	Demonstration at IEEE NFV-SDN 2022.
TFS Architecture Update	DLT Connector is part of TFS and will implement the protocol buffer towards the DLT gateway. DLT gateway will be provided by NEC as it talks directly with the blockchain.

2.18. NBI (Including the Old Compute Component)

The NBI includes the function of the old Compute Component. It allows TFS to interconnect to an external NFV Orchestrator. By doing so, the TFS can automatically handle the lifecycle management of network connectivity services between remote data centre/cloud sites. In other words, the NBI acts

as a front-end to receive, process, and trigger the creation/update/removal of network connections fulfilling the requirements of the network services (involving both cloud and network resources) managed by the NFV Orchestrator. Thus, the NFV Orchestrator serves as a client demanding network connectivity services from the NBI . The NBI then interacts with other components, such as the Service Component, to meet the requested operations (e.g., connectivity creation, update, or removal).

2.18.1. Energy-Based Service Placement between Edge and Core Compute Resources

Technologies Involved	NFV Orchestrator (OSM), Placement Component of the OSM, NBI, Service Component, PathComp Component, Context Component.
Type	Network services involving compute and network resources
Description	The NBI (including the function of the old Compute Component) provides an API translation and mapping between the OSM WIM connector and the Service Component. Specifically, the OSM relies on a REST API to register and request diverse network connectivity operations such as creation, updating, and removal. TFS release 2.0 deployment of the NBI enables it to receive and process the establishment of network connectivity services with specified endpoints. TFS r2.1 is expected to tackle advanced functionalities such as: i) deploying network services dealing with energy-oriented objectives; and ii) requesting disjoint network connectivity services (i.e., primary and backup). The placement of incoming network services (set of VNFs and VLs) entails that the NFV Orchestrator (OSM) decides the PoPs (either edge or core) as well as their interconnecting PoP links. To do this, TFS will leverage the Placement Component offered by the OSM frametwork. The Placement Component outputs the PoPs that host the VNFs, and the PoPs' interconnecting links (PiLs) that need to be deployed to deliver the VLs.. Such PoP and PiLs selection is done fulfilling the network service requirements (e.g., latency), but also considering energy consumption. For the latter, the Placement Component defines a (static) list of "Prices": 1) a price for deploying a VNF in a given PoP (i.e., pop_price); and 2) a price for rolling out a VL using a specific PiL (pil_price). The idea is to bind such prices to energy consumption. For instance, the pop_price could be set to the aggregated amount of the power consumed by all the servers in a PoP. Alternatively, pop_price may be tied to the PoP location characteristics (e.g., edge/core) assuming that an edge PoP consumes less energy that a core PoP. A pil_price could be bound to the minimum number of devices to be traversed for interconnecting a pair of PoPs over the underlying transport infrastructure. Therefore, minimizing the pil_price may favour reducing the number of involved network devices, which in turn could reduce the energy consumption. The role of the TFS Controller, and particularly the NBI, focuses on receving and processing every OSM-selected PiL (specifying the PoPs to be interconnected) throughout a Network Service VL that is being deployed. This represents a new network connectivity service setup request, which entails seeking a transport path and networking resources that meet the specific VL demands (e.g., bandwidth and latency). The selected path and networking resources must be computed targeting energy-oriented objectives. To do that, the dedicated TFS components such as the Service Component and the PathComp Component, will be enhanced to adopt algorithms/mechanims favoring the reduction of the consumed energy. These adopted energy-aware algorithms could foster deploying network connectivity services over devices with occupied resources (i.e., grooming strategies/statistical multiplexing), avoiding having large activedevices which consume lots of energy even with low resource usage, and shutting down devices as much as possible, etc. These algorithms/strategies will be conducted in the context of the PathComp Component where an important goal is to attain a "good" trade-off between transport resource utilisation and resulting energy consumption. Another relevant aspect related to the new improved capabilities of the TFS NBI is to receive and process nertwork connectivity requests from the OSM demading disjoint transport paths. The idea is that the REST API communicating the OSM and the NBI support the disjoint transport path for a single service request. This capability entails also improving the gRPC communication between the NBI and other TFS components such as the Service Component.

	Additionally, the PathComp Component is used to output (upon demand) a pair of disjoint transport paths.
TFS Target Objective	Leveraging the OSM Placement Component to select the PoPs and PiLs to attain an energy-aware deployment of the incoming network service requests. With respect to the TFS components, the aim is to continue supporting the automatic lifecycle management of network services where the NBI operates as the front-end of the TFS Controller to set up/update/delete the derived connectivity services. Besides fulfilling the network service requirements, such bandwidth, latency, and disjoint transport paths, other components in the TFS Controller will also handle energy-oriented objectives when selecting the transport resources for network connectivity services.
New Requirements	Utilisation of the available OSM Placement Component to select PoPs and PiLs for incoming network service requests targeting overall energy reduction. At the TFS Controller, the idea is to exploit the PathComp Component to select transport resources that fulfil the network connectivity requests (bound to the network services) and accomplish an effective energy reduction. This entails deploying specific energy-aware algorithms and strategies within the PathComp Component. Support in the OSM: The NBI API support of new connectivity service attributes and requirements such as disjoint paths for a single network connectivity service request. These new features and requirements also need to be mapped into the gRPC API between the NBI and Service Component, and to support in the PathComp Component the calculation of a pair of primary and backup paths meeting the service demands.
Research Actions	Both the energy-oriented service placement and the support of network connectivity services with enhanced requirements (e.g., path disjointedness) are planned to be presented in strategic international conferences (e.g., OFC, netsoft, etc.). The targeted conferences will depend on the adopted underlying transport infrastructure, i.e., either optical or packet switching technologies.
TFS Architecture Update	For energy-oriented service placement, the PathComp Component supports executing explicit transport path computation and resource selection to meet not only demanded service requirements in terms of bandwidth, latency, and disjoint paths, but also handling devised energy-aware routing computation. Thus, it is planned to add specialised algorithms dealing with such energy-aware objectives to the PathComp Component. This entails revisiting the Context Component attributes/data models of both devices and links to characterise their incurred energy consumption. This information is then used as input to derive the path and network resource selection. Another important aspect in this scope is the interactions between both the Service and PathComp Components to request/respond path computations specifying the objective functions and requirements. To support enhanced network connectivity service requirements demanded by the OSM to the TFS Controller (e.g., disjoint paths), the REST API communicating with both OSM and the NBI needs to be extended. This extension will also need to be covered through the rest of the TFS components such as between the NBI and Service Component, and between the Service and PathComp Components.

2.19. Web User Interface (WebUI)

The Web-based User Interface (WebUI) is one of the microservices that make up the TFS Controller. The WebUI enables a network operator to manually interact with the TFS Controller to perform configuration operations and inspect the state of the network.

Technologies Involved	**All**
Type	User Interface
Description	The WebUI enables a network operator to manually interact with the TFS Controller to perform configuration operations and inspect the state of the network.

TFS Target Objective	Objective 1: Adoption of SDN by telecom operators
New Requirements	The WebUI needs to be implemented as another TFS micro-service. The WebUI needs different sections: Home, Device, Link, Service, Slice, Grafana, and About.
Research Actions	WebUI will be present in all TFS demos.
TFS Architecture Update	A new component is introduced in TFS architecture: WebUI. It consumes REST NBI calls.

2.20. Inter-Domain

2.20.1. E2E Routing and SLA Violation Detection

Technologies Involved	**IP, optical**
Type	Context, Service, Monitoring, Policy, Inter-domain
Description	To demonstrate E2E routing of connectivity services and transport slices, and the enforcement of SLAs.
TFS Target Objective	Objective 1.1: Accelerate innovation in transport (optical and microwave) and IP networks and ultimately help operators provide better connectivity for communities all around the world Objective 1.3: Automated service management for transport network slices Objective 3.3: Inter-domain provisioning of connectivity services
New Requirements	• Create and manage inter-domain transport slices using collected per-domain topologies. (See also Section 8.) • Monitor inter-domain KPIs to validate fulfilment of SLAs. • Implement mitigation actions when an SLA is violated. • Path computation: calculation of domains and per-domain SLAs based on E2E SLA requirements.
Research Actions	To be demonstrated in Scenario 2
TFS Architecture Update	Policy, Slice, Monitoring, PathComp

3. Business Model and Ecosystem Analysis

In the context of TFS, D2.1 [6] suggests it is necessary to understand how the evolution of two different markets are intertwined in a new regime with disaggregated and programmable 5G and 6G networks. On the one hand, there is the ambition for a growing market for transport networks provided by network operators. On the other hand, there is the market for the different actor roles in the supply chain/ecosystem which deliver necessary components to create transport network services. The growth in the first is a prerequisite for growth in the second. However, the growth of the transport network market is dependent on operators' ability to mobilise and motivate the actor roles in the ecosystem to join a journey characterised using open source, eventually consensus-based standards, and new business models.

In the following, we assume that that the actor roles have accepted the above market conditions: technologies are disaggregated, transport network programmable, and markets are coordinated with fully standardised controllers, e.g., TFS. The future market and ecosystem for the transport network will be constituted by many business actor roles, such as the hardware provider and system integrator. It will have developed into one dominant configuration. I.e., the actor role business models will have stabilised, as will the relationships between them, as we illustrate in Figure 15. We consider that we are now in the beginning of an ecosystem evolution, with the ambition to: 1) understand the alternative paths of its evolution; and 2) imagine potential alternative future stable ecosystems.

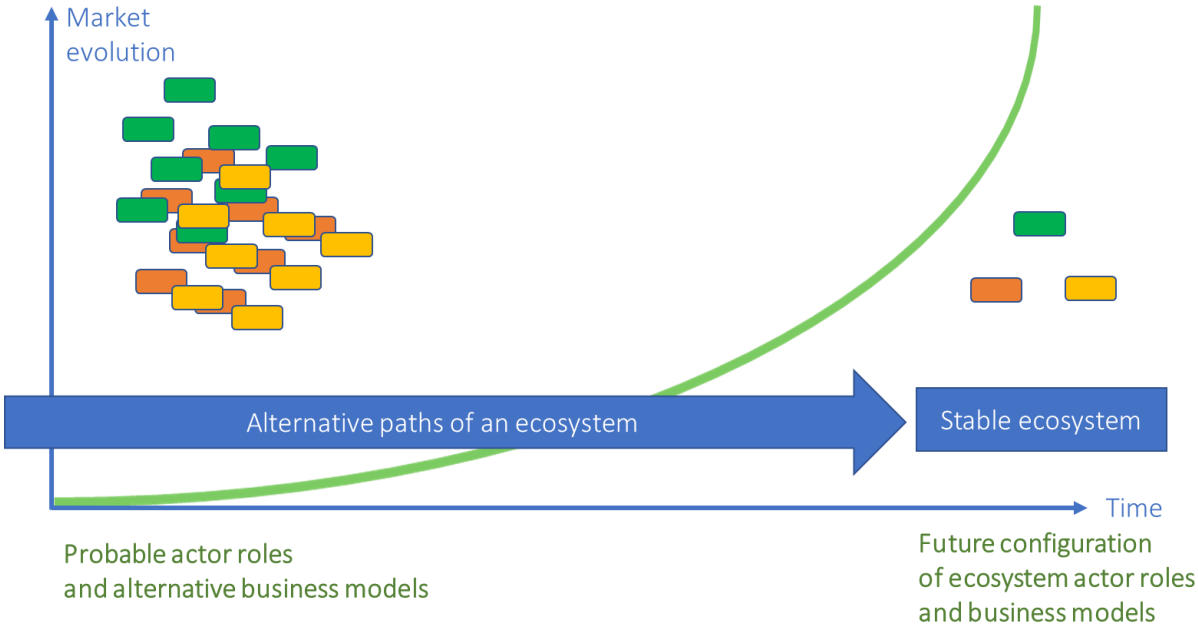


Figure 15 Illustration of Start and Endpoint of the Evolution of a Transport Network Ecosystem

To address this ambition, we list and illustrate existing and probable actor roles and suggest potential alternative business models. This includes an identification of underlying factors that could affect which business models are realised and how the ecosystem may evolve. How an ecosystem may take different paths will be further elaborated in the later TeraFlow deliverable D6.4.

The analyses in this section are based on secondary sources, and data from interviews with TeraFlow partners. Some data were already collected in D2.1 [6] and D6.2 [20]. New data were collected in plenary meeting workshop in Castelldefels on 25th May 2022, and partner interviews in the autumn of 2022.

3.1. The Ecosystem and Actor Roles

Figure 16 is an updated depiction of the TFS ecosystem. A thorough description of most actor roles can be found in D2.1 [6]. Our data suggest additional actor roles, e.g., we have indicated that there

will be system integrator in this market, and a testing regime which invites to new actor roles. In the following, we elaborate on the business models for the actor roles listed below. We will start by elaborating on the operator's role as customer, introduce system integrator as an important actor role, and comment on the changed testing regime. This will be the backdrop for discussing changes in other actor roles. Finally, we will comment on the emerging roles in a test environment, discussed further in D6.3 [7].

- System integrator of all components needed in transport network
- Hardware (HW) provider (often called vendors)
- Software (SW) provider (embedded in the HW)
- NetApp provider
- SDN provider: The provider of products and services based on TFS, in the form of:
    - System integration of TFS components (the controller), consulting
    - Development of new TFS features
- Test environment
    - Certifier for TFS interoperability
    - Tester

These actor roles for the transport network are well aligned with those of a disaggregated radio access network (RAN) [8] and other analyses of future 5G market ecosystems [9], [5], [1]. We do not elaborate on roles such as cloud providers or providers of semiconductors which are present in the open RAN ecosystem [8].

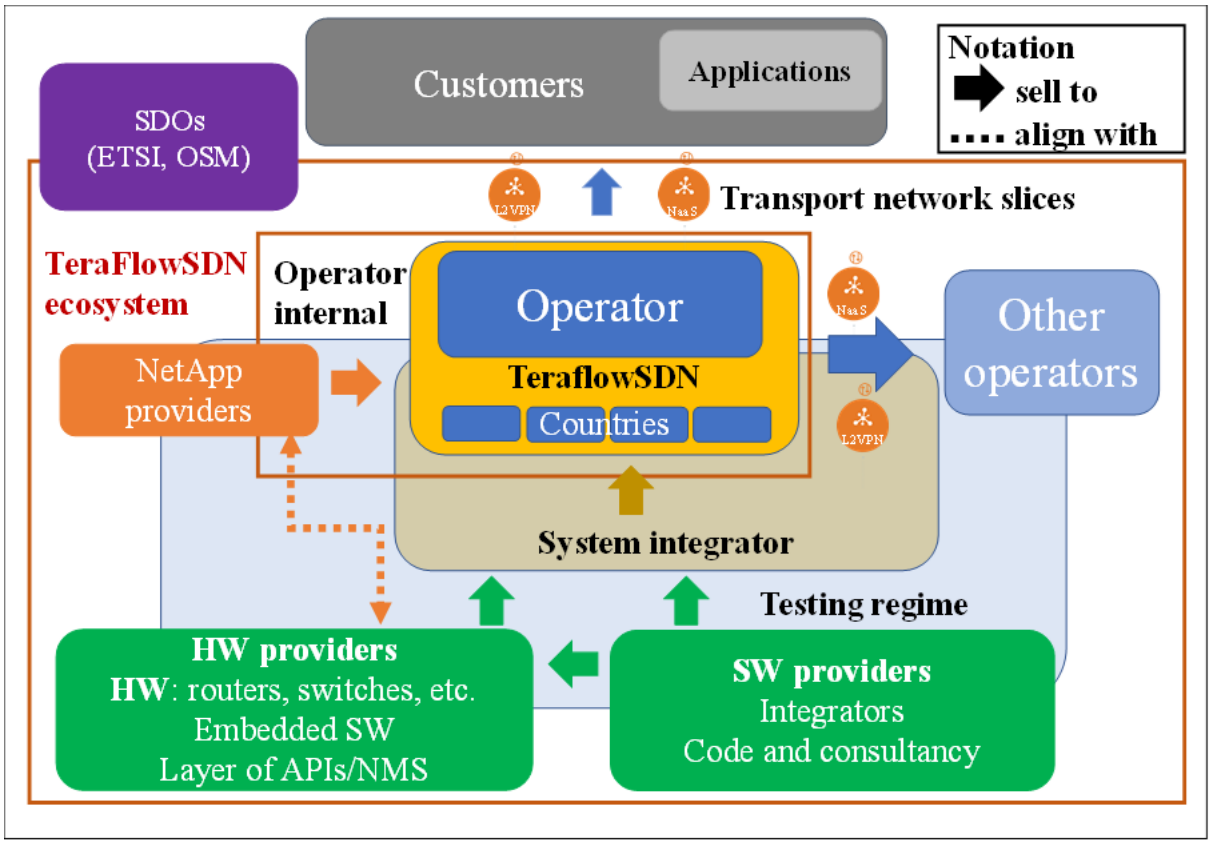


Figure 16 Updated TFS Ecosystem

We use a business model canvas [14] to structure different aspects of actor roles business models. To design business models is an iterative process, thus, the versions suggested in this document may in future be informed by new insight and changed accordingly.

Note that the programmable transport network enables operators to sell and manage transport network slices more efficiently and introduces new business models. This constitutes a second business flow from the operator point of view, which is aligned with the tasks of e.g., the TFS Slice, Inter-domain, and Distributed Ledger and Smart Contracts Components. For instance, the ability to interact with other SDN controllers and rent from or lease resources to other operators for intra-domain purposes, indicates a new business model for operators. Also, the ability for operators to peer and offer inter-domain transport slices is a new business model. Currently, we have not analysed in detail how this business flow and these models in turn affect other actor roles and their business opportunities. However, we can easily imagine that, e.g., a HW provider could lease HW to many operators in parallel, instead of selling HW to each.

3.2. The Network Operator – The Customer

All actor roles have in common that the network operator is their main customer. The different business models we elaborate can be positioned in a future business flow depicted in Figure 17. The linear flow underscores the initial ordering and installation of HW, and the role SW and NetApp providers can play in 1) the initial installation, and 2) support and reconfiguration. In addition, it is indicated how the resulting programmable transport network (i.e., SDN) can be managed with a controller like TFS, implying that standard interfaces are implemented for HW, SW, and NetApps. For each of the actor roles discussed, the potential flow depicted in Figure 17 is the basis for the suggestions on, e.g., key resources, partnerships, and value propositions.

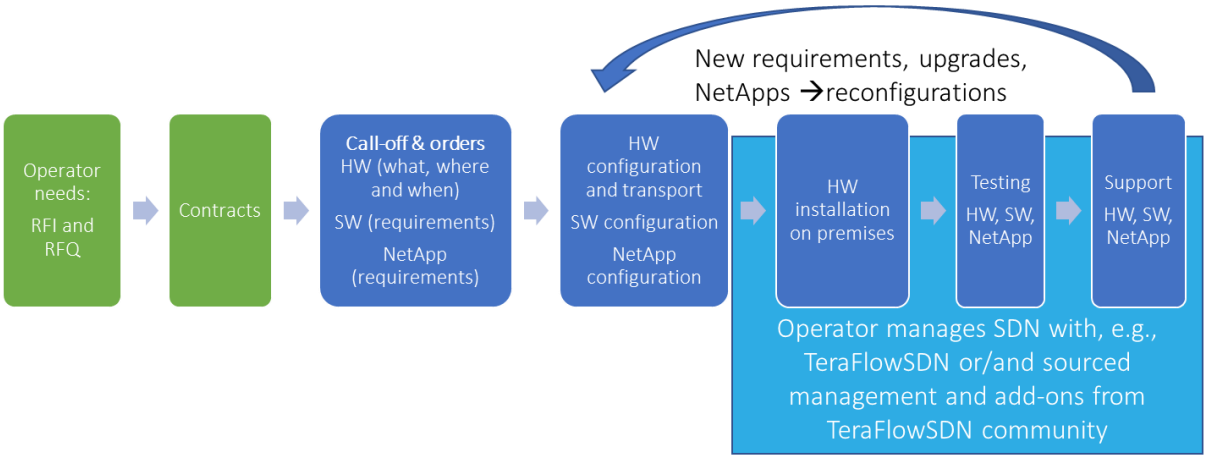


Figure 17 Sequences in Business Transaction between Operator and Providers of HW, SW, NetApps, and SDN Services

Our data suggest that the flow in Figure 17 must be complemented by additional actor roles and their business models, namely the system integrator and roles that cater to testing and certification. See Figure 17 and Figure 18. Here, we explain why these actor roles have been suggested as plausible in the future ecosystem and also suggest their business models.

3.3. System Integrators Serving Operators

As-is: In the current market, system integrators play a minor role in the delivery of a transport network to operators. Operators purchase HW from HW providers. The testing and implementation carried out by HW providers, for operators, constitute the "system integration" of the network.

To-be: The future disaggregated transport network introduces the potential for management and the combination of HW, SW, and NetApps from different providers. Still, it is suggested that, at least in the short and medium term, operators will use and rely on system integrators. That is, operators will shape RFQs and expect that the system integrators answer them. Only in the long term, it is plausible

that operators do system integration themselves. In general, outsourcing to system integrators is expected to become a trend if the market is disaggregated and changed [9].

Operators will continue to use system integrators because of complexity, lack of in-house competence, and risk-minimisation considerations. It is costly to hold this competence, and a huge effort to carry out the task, which may perhaps only be for limited periods of time. Even though operators may acquire both power and competence, they will often prefer to transfer responsibility and risks to a system integrator. The decision to use a system integrator may also be a result of history and culture.

Here we make a distinction between a system integrator and aggregator. A system integrator combines components, adds value, and sells directly to customers, i.e., it takes responsibility and risk that the components will function together. This role addresses one major concern of operators. An aggregator makes available many components so that they can easily be combined and used by others, i.e., it does not take responsibility for how the components function together. There may be room for an aggregator role in the long term, however, we do not see that it will be the one driving the market in the first instance.

One factor suggested to make operators build in-house competence is the lack of large system integrators who can carry the risk and be trusted: operators will not choose small firms or start-ups for such work. The implication from this is that, in a nascent market, an operator can also choose to build its own system integration competence. I.e., existing competence and market maturity is an *underlying factor* affecting the position a system integrator eventually will get.

HW providers could also take on the actor role as system integrator, but we cannot currently determine how this affects the role or the market evolution. It is also suggested that HW providers will prefer to use system integrators for their own purposes.

Table 1 suggests a business model canvas for a future transport network system integrator.

Table 1 Business Model Canvas for System Integrator in the Transport Network

<table>
<tr>
<td rowspan="2">Key partnerships
Providers of HW and SW which are components of the solutions.

Providers of controllers such as TFS.

Potential neutral test platform where compatibility in the integrated system can be pre-tested (provided by e.g., operators)</td>
<td>Key activities
Identify and capture customer key challenges – transfer to design.

Design systems with sufficient novelty and reliability.

Risk analyses: technological and financial.

Implement and run systems.

Efficient failure support.</td>
<td rowspan="2">Value proposition
Combines transport network components – design and implementation.

Has responsibility and carries the risk for functioning according to predefined levels.

One point of failure-support.

Proven record and large enough customer portfolio – trust position for service quality and endurance.</td>
<td>Customer relationships
Request for Information/Quotation
Long-term relationships.
Dedicated support
Potential high switching costs.</td>
<td rowspan="2">Customer segments
Large contracts:

Network operators.

Data centre operators.

Hyperscalers.</td>
</tr>
<tr>
<td>Key resources
Network architects, implementers, and managers.

Purchasing, contract, and logistics expertise.

Customer and partner/vendor portfolio management.</td>
<td>Channels
Owned digital provisioning

Partner channel</td>
</tr>
<tr>
<td colspan="2">Cost structure
Salaries. Management systems.</td>
<td colspan="3">Revenue streams
Complements: Per hour/project. Fixed yearly price for operation. Reseller fees (HW and SW).</td>
</tr>
</table>

Factors affecting ecosystem paths and end-state: operators' use of system integrators, or they are "doing-it-yourself" (DIY).

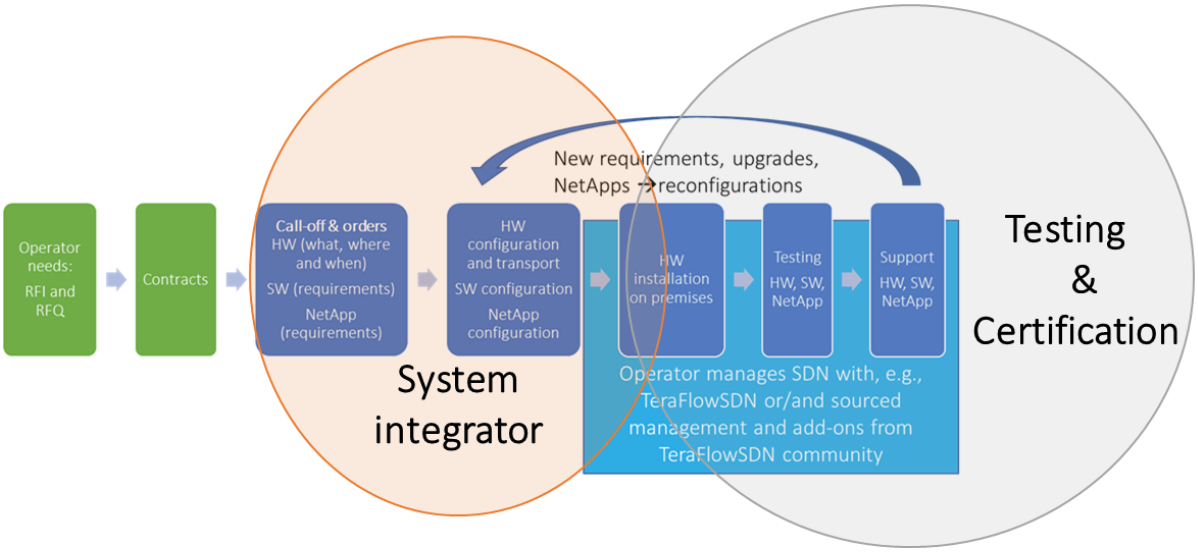


Figure 18 Illustration of System Integration and Testing Regimes in Business Transaction between Operator and Providers

3.4. Operators and Regimes for Testing of HW

As-is: Currently, the operators' normal procedure is to test HW before putting it into operation. This is under the condition that the operator has one or more preferred controllers or interfaces. The procedure follows steps such as:

- Operators publish Request for Information (RFI).
- HW providers answer to RFI. The answers include roadmap for HW availability and testing.
- Operators short-list HW providers.
- Operators and short-listed HW providers plan testing.
- Testing: HW is either certified, or not.

Testing, or certification of HW are usually paid by the providers, i.e., it is integrated into their pricing. Operators may carry out some additional local on-boarding and testing of HW before implementation.

To-be: With the deployment of one standard controller across all HW and operators, e.g., TFS, the current testing procedure could change. A situation with one standardised controller creates a predictability which opens for shifting the steps in the procedure. For instance, providers could carry out early pre-testing of HW, acquiring compatibility certificates which can be used across many operators and other HW providers. In this way, the previous shortlisting and operator managed testing would become redundant, saving time and costs for both operators and HW providers.

The introduction of a standardised controller and subsequent shift in procedure also introduces opportunities for new actor roles, such as a neutral lab and testing consultants. The potential changes in testing regimes could happen even in a market where operators use system integrators. Then, the system integrator would have a driving role in the testing regime.

Factors affecting ecosystem paths and end-state:

- Network operators and system integrators change the current procedure for testing HW, or they keep the current procedures.
- HW providers pay for testing, or operators (system integrators) pay.

Here, we have explained how testing is carried out traditionally by operators in the market, and briefly suggested why and how a standardised controller can change the procedures. The actor roles and business models for a new testing regime when TFS is implemented is elaborated in Section 3.9.

3.5. HW Provider

As-is: One current set of providers to the transport network operator are HW providers who sell HW with embedded proprietary SW and proprietary interfaces (network management systems). Infinera and SIAE are examples and partners in the TeraFlow project. Currently, this is the main way of providing HW and SW to operators, mostly with a price per HW unit. The different provider-specific HW interfaces constitute the big challenge that TFS addresses and intends to replace by introducing vendor agnostic programmability.

To-be: The role of established HW providers is shifting towards being providers of SW, with new market terms and business models. However, there is still a demand for HW. We suggest that there are two alternatives for a future HW provider: 1) bare metal or white box HW, and 2) HW which retains some embedded, dedicated SW. Both are aligned with the standard controller. Table 2 Table 3 are suggested business models for these two potential HW providers.

In both of the alternatives, the HW must still be tested to:

- Be compatible with the controller, e.g., the TFS Controller
- Be compatible (function well) with interfaces towards SW, HW, and embedded SW which is still running on dedicated HW.

To cater to efficient pre-certified and pre-configured HW, the ability to deliver on time and on location could be part of a value proposition.

Factors affecting ecosystem paths and end-state: All HW may turn into white boxes, or some HW might keep dedicated embedded SW (and functions) on the HW.

3.5.1. HW as Bare Metal

To-be: The HW market may become a mass-market (for bare metal) with a cost structure with economies of scale characteristics. The HW providers will be driven to chase high-volume manufacturing to be able to deliver cost-efficiently in a competitive market. Thus, the market will be driven towards fewer providers because of the advantages from large scale production. We suggest that alternative key competitive advantages can be the ability to deliver HW on premises in a timely way, with SW installed on the HW as required by the customers. This curbs the traction towards large scale, and opens opportunities for smaller and more flexible HW providers. Both aspects are catered to in the business model canvas. Furthermore, to support intra-domain operations between operators with the same HW, a new HW leasing business model is opened.

Table 2 Business Model Canvas for Providers of Bare Metal HW

<table>
<tr>
<td rowspan="2">Key partnerships
Providers of SW (SW to be installed on HW).

Providers of physical installation of HW on-premises.</td>
<td>Key activities
Production/manufacturing.

Logistics (for delivery and installation on-premises).

Routines for efficient pre-configuration of HW based on customer requirements.</td>
<td rowspan="2">Value proposition
Up-to-date HW which can be used interchangeably in network with alternative HW.

Frictionless provisioning and implementation of HW on network operators’ premises, on time (all geographies).</td>
<td>Customer relationships
Request for Information/Quotation.

Framework and call-off agreements.

Automated.</td>
<td rowspan="2">Customer segments

Mass market for HW: Network operators

Niche market for timely HW installation: Network operators</td>
</tr>
<tr>
<td>Key resources
HW designers.

HW production and supply chain facilities and management.</td>
<td>Channels
Owned digitalised call-offs.

Digital overview of deliveries.</td>
</tr>
</table>

	HW logistics		Physical delivery.	
Cost structure Cost-driven. Manufacturing, raw material. Costs for certification and testing.		**Revenue streams** Alternatives: Price per unit, price per unit installed, leasing agreements for HW on-premises.		

3.5.2. HW Retaining Some Dedicated Embedded SW

To-be: Despite the vision, demand for, and trend towards HW as bare metal, there are factors that hold back this progression. One argument is the difficult task of replacing specific HW with SW. Examples given for HW that may remain dedicated are within the optical networks. Interfaces, however, can be fully standardised enabling full compatibility with the rest of the network. There are more pragmatic explanations for the continued use of dedicated HW, e.g., for some operators it is more comfortable to point at one responsible HW provider when there are failures.

This version of a dedicated HW provider will be more labour intensive. Its key activities and resources are used for research and design, with good ability to capture future transport network requirements and to manage outsourced production of the HW. Their value proposition to network operators is to provide dedicated HW which can relieve customers of the risks they might encounter. We suggest that this type of HW provider will work hard to keep production costs for the advanced devices as low as possible, as they must carry labour-intensive research and design activities. The revenue streams will probably be per unit, with or without support contracts included.

Table 3 Business Model Canvas for Providers of Dedicated HW

<table>
<tr><td rowspan="2">Key partnerships
Network controller providers and consultancies.

Providers in the HW testing regime.

System integrators</td><td>Key activities
HW design.

HW research and development. Managing outsourced production.

Testing.</td><td rowspan="2">Value proposition
Premium HW dedicated to transport network tasks.

Relieving customers of perceived risks of failure.</td><td>Customer relationships
Request for Information/Quotation.

Framework and call-off agreements.</td><td rowspan="2">Customer segments
Market for dedicated HW and risk reduction: network operators</td></tr>
<tr><td>Key resources
HW designers.

Insight into transport network challenges and customer needs.</td><td>Channels
Owned digitalised call-offs

Digital overview of deliveries

Physical delivery</td></tr>
<tr><td colspan="3">Cost structure
Labour intensive research, development, and design.

Scalable production of IPR protected HW.

Costs of certification and testing.</td><td colspan="2">Revenue streams
Price per unit with/without support contracts.

Separate support contracts.</td></tr>
</table>

3.6. SW Provider

As-is: In the TeraFlow project there are also solution providers who sell only transport network SW in combination with HW. However, their current role is marginal compared to established transport network HW providers.

To-be: It is expected that TFS will open the market for smaller SW providers and drive previous HW providers into the role of SW providers. The previous embedded SW will move to run on-top of any HW. Thus, SW providers may deliver to, or be complementary to, the HW providers described above. Ubitech is a TeraFlow partner which deals with SW and adjusts it to HW and helps other SW providers

to deploy their applications on HW or in networks. Table 4 describes the business model for SW providers.

We suggest that the SW must also be tested in future, to:

- Be compatible with the TFS Controller.
- Be compatible (function well) with interfaces towards HW and embedded SW which is still running on dedicated HW.

The value proposition will be to deliver SW with operating network functionality, utilizing complementary capabilities of HW and other network components (physical cabling). The SW must be provisioned and managed efficiently, i.e., probably in a cloud-native environment.

Factors affecting ecosystem paths and end-state: With HW such as white boxes, SW providers will move into more prominent positions. With the continuation of dedicated HW, SW providers must find ways to co-exist.

Table 4 Business Model Canvas for SW Providers

<table>
<tr><td rowspan="2">Key partnerships
Providers of HW, on which SW shall be installed.

System integrators.

Network controller providers and consultancies.</td><td>Key activities
SW design and development.

SW testing.

SW support.

Provisioning automation.

Cloud-native environment</td><td rowspan="2">Value proposition
SW needed to get transport networks to function in specific ways.

SW can be managed by a standard SDN controller and interfaces, e.g., TFS.</td><td>Customer relationships
Request for Information/Quotation.

Framework and call-off agreements.

Automated.</td><td rowspan="2">Customer segments

Mass market for SW: Network operators</td></tr>
<tr><td>Key resources
SW developers.

SW testing.

SW support.

SW CI/CD platform.</td><td>Channels
Owned digital provisioning/call-offs.

Partner channels.</td></tr>
<tr><td colspan="2">Cost structure
Salaries, platform costs (servers etc.)

Costs of certification and testing.</td><td colspan="3">Revenue streams
Complements: 1) SW licenses. 2) Per hour/project for development, installation/provisioning, testing, support.</td></tr>
</table>

3.7. NetApp Provider

As-is: A second group 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 HW providers.

To-be: It is expected that the number of NetApp providers could increase if the TFS ecosystem takes off. NEC is a partner in TeraFlow and an example of a potential provider of a security NetApp. In Table 5, we have indicated that the NetApps are already implemented on HW in the HW delivery phase, but also that they can be implemented on HW that is already installed. We assume that the key activities and resources for NetApps are very much like those for generic, equipment-related SW, however, we would expect more IPR as a basis for a business model based on licensing. The market may become a

niche market with customised SW which in turn enables operators to differentiate. I.e., this is more of a premium market compared to the generic SW mass market described above.

Table 5 Business Model Canvas for NetApp Providers

<table>
<tr>
<td rowspan="2">Key partnerships
Providers of HW where NetApps shall be installed.

Providers of generic SW to be installed on HW, which Netapps must be compatible with.</td>
<td>Key activities
NetApp development

NetApp testing

NetApp support

Provisioning automation</td>
<td rowspan="2">Value proposition
NetApps customised for specific functionality on the HW and with SW.

Can be managed by standard SDN controller and interfaces, e.g., TFS</td>
<td>Customer relationships
Requests.

Long-term relationships.</td>
<td rowspan="2">Customer segments

Niche market for customised NetApp: Network operators</td>
</tr>
<tr>
<td>Key resources
NetApp developers

NetApp testing

NetApp support

NetApp CI/CD platform</td>
<td>Channels
Owned digital provisioning</td>
</tr>
<tr>
<td colspan="2">Cost structure
Salaries, platform costs (servers etc.).

Costs of certification and testing.</td>
<td colspan="3">Revenue streams: Complements: NetApp licenses.

Per hour/project for development, installation/provisioning, testing, support.</td>
</tr>
</table>

3.8. Provider of TFS Related Products and Services

As-is: The provider of an SDN controller is also an actor role with commercial opportunities. Currently, actors in this role provide complete systems which address the same domain as TFS (for definitions and competitors see D6.2, or e.g., Cisco and Sedona Systems).

To-be: However, in future, as open-source SW, TFS (with APIs for standard networking and service modelling) will be sourced from a shared repository governed by ETSI's Open-Source Group TeraFlowSDN (ETSI OSG TFS). In this context, roles such as system and SW integrators or consultancies may emerge and extract a significant share of the market for SDN controllers. Examples of similar commercial exploitation can also be found in the ETSI Open-Source MANO community (https://osm.etsi.org/wikipub/index.php/OSM_Ecosystem).

We suggest that there are two interesting business model canvases for providers who utilise TFS for commercial purposes. One is about turning TFS into a service which can be sold. A second role is about the development of new features (products and services) addressing niche demands or e.g., operators. These canvases are complementary, and one firm can use just one or both models.

3.8.1. TFS as a Service

The value proposition for TFS as a service builds on the observation of how an open-source group works: a feature request or fix is forwarded, and usually addressed by the proposer when time allows. This contrasts with a professional market, where the controller must be changed into a product (service). In the first instance, the different features (modules) must be assembled into a package, see [7] for detailed suggestions for a package. Furthermore, the product must be scalable and provide new and maintained code. Features must be provided with predictable attributes, attending to faults, and fixing of bugs. The service provided requires key resources such as IT/cloud environments and configuration capabilities. It all signals operators' requirements for reliable delivery of such services. Thus, it is a business opportunity to build a reliable product on top of TFS features. It is suggested that

this actor role probably should and would be taken by an existing large firm, for instance one that already is a system integrator.

In a nascent market, operators will have the choice to purchase a controller, or develop competence internally. This opens the same dilemmas as taking on the role as a system integrator. The risk is that it may be costly, and that the implementation of the controller will vary across operators and decrease its effect on compatibility. However, to kick-off the market, operators may have to build this role in-house.

Factors affecting ecosystem paths and end-state:

- Operators purchases controller, or operators choose to develop the competence internally.
- System integrator and provider of TFS as a service are the same, in an integrated close relationship, or the relationship is transactional.

Table 6 Business Model Canvas for Provider of TFS Package and Service

<table>
<tr>
<td rowspan="2">Key partnerships
Providers of HW which shall be managed by TFS.

Potential neutral test platform (provided by, e.g., operators)

Standard setting organisations, ETSI.

Universities, research institutes, developer communities.</td>
<td>Key activities
Assess customer challenges, with customer.

Alignment with other orchestration systems.

Alignment with different HW (bare metal switches).

Testing of resiliency, scalability.

Bug fixing.

Management of TFS.</td>
<td rowspan="2">Value proposition
Enable use /deployment of TFS so that transport networks can be programmed agnostic of HW and SW providers.

One point of service/failure: operator can rely on service provider.

Framework for enabling integration of new features with minimal downtime.

Tightly aligned with the purchases of HW and SW. E.g., system integrators and other key partners.</td>
<td>Customer relationships
Request for Information/Quotation

Requests.

Long-term relationships.

Dedicated personnel.

Assistance.

Potential high switching costs.</td>
<td rowspan="2">Customer segments

Mass market: Network operators. Data centre operators. Hyperscalers.

Niche market: Small enterprises and data centres.</td>
</tr>
<tr>
<td>Key resources
Package of TFS modules with additional service components.

Skilled developers

Code maintainers and fixers

Testing/validation

CI/CD platform</td>
<td>Channels
Owned digital provisioning

Partner channel</td>
</tr>
<tr>
<td colspan="2">Cost structure
Salaries, platform costs (servers etc.).</td>
<td colspan="3">Revenue streams
Complements: Per hour/project for development, installation/provisioning, testing, support.</td>
</tr>
</table>

In addition to the actor role as system integrator for TFS, we suggest that there is a role for consultancy regarding TFS. The value proposition would be to support the decisions about, use of, and maintenance of TFS, being a neutral highly competent party.

3.8.2. Development of New TFS Features

Furthermore, there is an opening for an actor role which develops new TFS features. Their customers will be operators or system integrators who want to differentiate how they carry out network management. These actor roles must develop a deep understanding of customer challenges and be

able to maintain the features developed. Their key resources are developers and the platforms enabling them to build and test features. Their costs are mainly salaries, while revenues can be per hour or project, or also for licensed features.

Table 7 Business Model Canvas for TFS Related Provider – Develop New Features

<table>
<tr>
<td rowspan="2">Key partnerships
Providers of HW which shall be managed by TFS.

Potential neutral test platform (provided by e.g., operators).

Universities, research institutes, developer communities.</td>
<td>Key activities
Understand customer challenges.

Develop new features with IPR.

Manage and support new features.</td>
<td rowspan="2">Value proposition
New features provided as distinct products, services on top of the TFS environment.

Address operators' specific challenges or ambitions with network management – enabling operator differentiation.</td>
<td>Customer relationships
Requests.

Long-term relationships.</td>
<td rowspan="2">Customer segments
Niche market: Network operators.

Data centre operators.
Hyperscalers.
System integrators.</td>
</tr>
<tr>
<td>Key resources
New features R&D

Developers

Testing/validation

CI/CD platform.</td>
<td>Channels
Owned digital provisioning.</td>
</tr>
<tr>
<td colspan="2">Cost structure
Salaries, platform costs (servers etc.).</td>
<td colspan="3">Revenue streams
Complements: Per hour/project, and licenses.</td>
</tr>
</table>

3.9. Neutral Lab – Testing and Certifying

As-is: The current testing environment for a standard SDN controller, such as TFS, is briefly described in Section 3.4, seen from the operator's perspective. The normal procedure is that device testing is required by operators and paid for by HW providers as an integral part of pricing and contracts.

To-be: The presence of a standard controller opens the prospect of a changed regime, where actor roles can emerge, value can be delivered to other actor roles, and revenues can be extracted. One important identified change is that the testing procedure stops being a bi-lateral concern between the customer (operator) and HW provider. Instead, both operators and HW providers may rely on third parties who carry out tests and state fulfilment to compatibility – i.e., certification.

In a TFS environment, all modules can be tested. If new HW is tested for its TFS compatibility, as a minimum the TFS module for the device is needed, i.e., the South-Bound-Interface module. In [7], a package of several modules is suggested for a provider of a test-lab: Context management, Monitoring, Southbound Interface, Service, Northbound Interface, WebUI, and Path computation. The HW to be tested, i.e., devices, must be handled as physical objects. TFS modules may also be used for testing against SW, and thus, handled logistically as SW.

The value proposition for all roles in a new testing regime rests on the benefits the operator and HW providers can extract. Operators always require that HW is tested and is compatible with their preferred controllers. In a situation where only one standard controller is used, the expectation is that HW is fully compatible – anything else is a waste of cost and time. Thus, HW providers would probably benefit from being pre-certified to answer one operator, but also to save costs and time because they carry out certification only once for multiple operators. This also opens opportunities for smaller providers, because the costs of testing against many controllers are removed. However, the emergence of a market where testing happens at a high scale might be delayed if different parties wait to see who picks up the bill. This is already suggested as a factor that will affect the evolution of the market.

One recent example of a testing regime is the Connectivity and Standard Alliance[2] (CSA) which manages certification of Things (e.g., Internet of Things) which needs to be compatible with the standard Matter. A HW provider sends products to a certified test provider which does the test and states fulfilment to standards (or otherwise!). The HW provider applies for a certificate for its product to the CSA. The HW provider pays for tests, and a fee for certification.

We find three emerging actor roles in a new TFS testing regime:

- Testers:
  - Neutral test labs
  - Consultancies
- Certifier – entity authorised to issue certification

We suggest that the testers can be smaller firms, labs, and consultancies. They can also do tests for other HW interfaces, e.g., OpenConfig or P4.

For the tester, a discussion is raised whether HW should be sent to a tester, or if SW should be sent to the HW provider. Pros and cons in this discussion are provided in Table 8, and this may affect the value propositions of the actor roles.

Table 8 Advantages and Disadvantages Regarding Handling of Devices to be Tested

Alternative testing arrangements	Advantage	Disadvantage
Lab: HW providers send devices to lab	Normal procedure – contracts protect against any issues. HW is tested in a neutral environment, can be approved as tested. Do not need e.g., an additional layer of approval when vendors do the testing themselves. HW-providers: save work and cost – can re-use the test with other operators.	HW providers do not trust lab, will not reveal highly sensitive devices. Physically sending of devices seems very inefficient.
Lab: Lab sends testing SW to HW providers	Efficient – logistics efficient. Easier to send SW and the vendor do the testing themselves. HW providers sends a report for certification.	It is hard/complex to learn the skills for deploying and using the Test SW. Needs an additional iteration with acceptance of the testing done by the vendor itself. Harder to protect against cheating.
Ecosystem of certified SW consultants who can do the testing remotely or on-premises	Consultants with expertise/know-how install SW and test it on devices. Flexible – can handle the whole "testing journey". Testing as a service.	Consultants' neutrality – they must not mix their roles with e.g., driver development. If not pre-certified – still need to get another acceptance of the testing from a "central certifying unit".
Controllers come to devices	For devices: must be compatible with many controllers - are open – controllers should come to them.	Controller: must handle high volume of devices – devices should come to them.

Based on the Table 8, some key issues emerge that potentially affect the evolution of a testing regime. To function, all parties in a testing regime need to trust that their own interests are not violated by

[2] Certification Process | Why Certify | - CSA-IOT

others or by the system. E.g., the neutrality of testers is key. It is also a returning discussion who should pay for testing. Eventually, testing is integrated into the price and operators are paying.

Factors affecting ecosystem paths and end-state: The institutions established for testing are quickly acknowledged as trustworthy, or issues with and violations of neutrality delay the necessary trust

In Table 10, we suggest business model canvases for the neutral test lab and certifier. Also refer to [7] for elaboration of TFS packages for these roles. We do not elaborate on how a consultancy firm can take on a role as a tester.

The neutral test lab creates value by saving time and costs for both operators and providers and enables more efficient market expansion by providing a generic compatibility. The lab should also be quick, transparent, and trusted to handle competitive sensitive HW. It will mainly be providers of HW and SW which purchases its services, and these customers should be handled in digital channels with dedicated accounts aligned with the logistics of customers physical equipment sent for testing. Key activities of the tester will be customer management, the testing, and reliable documentation of test results. The key resources supporting this is the lab itself, and predefined and transparent procedures for testing and handling of test results. Relevant TFS modules will compose a package for the neutral test lab. The neutral test lab must collaborate with a certifier and shipping companies. Also, the ETSI OSG TFS community may be a key partnership when new testing challenges must be handled. The cost structure is mainly salaries for testing experts, and testing platform CAPEX and OPEX. Revenues will probably be a fee for running a test, which may vary with HW and SW size and other characteristics.

Table 9 Business Model Canvas for a Provider Which is a Neutral Test Lab

<table>
<tr><td rowspan="2">Key partnerships
Provider of compatibility certificate.

Network operators who are pushing full compatibility and pre-testing.

Shipping/logistics companies.

ETSI OSG TFS – developers who can help with new challenges.</td><td>Key activities
Customer management: digital process for handling test requests, and logistics for physical HW.

Technical testing, documentation, and reporting.</td><td rowspan="2">Value proposition
Save time and costs for providers and operators with compatibility tests which are generally valid.

Efficient logistics.

Enable expansion of markets.

Mitigate lock-in to providers.

Can be trusted with competition sensitive HW.</td><td>Customer relationships
Transactional.

Lab serves many providers-Account-based, with log-in functions.</td><td rowspan="2">Customer segments
Providers of HW and SW.

Large and small providers..</td></tr>
<tr><td>Key resources
Laboratory, with testing equipment and expertise.
Interface with certifier.
Predefined procedure for testing.
Framework for handling testing requests, results, and inquires.
Relevant TeraFlowSDN components.</td><td>Channels
Digital channels for ordering and following tests.

Alignment of digital process and physical products.</td></tr>
<tr><td colspan="2">Cost structure
Salaries, testing platform costs (servers etc.).</td><td colspan="3">Revenue streams
Fee for carrying out tests on request.</td></tr>
</table>

A neutral test lab may or may not hold the position as a certifier. Thus, in Table 10, we have carved out the role of a certifier. A certifier will get a request from a HW or SW provider which has proven compatible via a neutral test lab. The certifier delivers value by being trusted and neutral, quick, and efficient. Trust is also being built by being transparent with easily accessible frameworks and testing results. HW and SW providers purchase issuing of certificates via a digital channel. The key activity is an efficient process for handling requests for certification, and the certification database which will form over time. The neutral test lab will be a key partnership, as well as operators who demand and are dependent on general compatibility and systems to achieve this. The costs will be the CAPEX and OPEX of a platform for certification issuing. Probably, the revenues will be a fee to get a certificate issued and registered in an available database.

<table>
<tr><td rowspan="2">Key partnerships
Network operators, dependent on compatibility.

Labs which provide testing.</td><td>Key activities
Process for receiving and handling certification requests.

Database with certified HW.</td><td rowspan="2">Value proposition
Neutral part – trusted by those who want to be tested, and those who need HW to be compatible.

Quick and efficient administration.

Transparency.

A database with issued certificates which is easily accessible.</td><td>Customer relationships
Appointed role as certifier.</td><td rowspan="2">Customer segments
HW and SW providers which need to be compatible.</td></tr>
<tr><td>Key resources
Management of predefined requirements for being certified.

Framework for handling certification requests, and inquiries.</td><td>Channels
Digital channel.</td></tr>
<tr><td colspan="2">Cost structure
Certification platform costs (servers etc.).</td><td colspan="3">Revenue streams
Fee for providing certification on request.</td></tr>
</table>

Table 10 Business Model Canvas for the Actor Role Which Issues Certificates for Compatibility

Figure 19 summarizes the factors affecting ecosystem paths and end-state. The actor roles, and their alternative business models will enter a process where the ecosystem evolves and eventually one dominant path may emerge. In retrospect, path-dependency is commonly seen as an effect of reinforcing effects in systemic markets. In forethought, actors, and actor roles, can prepare for and act to steer and shape the emerging paths to their benefits. Thus, they also need to consider which are the most attractive options. Alternative business models serve as one means which actors can use to assess future opportunities.

We have suggested a set of factors which could affect actor roles' business models and the ecosystem evolution. These are described in Figure 19 as dimensions with two extremes; a future reality can be anywhere between the extremes. Thus, the alternative combinations from which a stable ecosystem may emerge are vast. In our analyses, we commented on yet other underlying circumstances which affect business models and evolution, e.g., competence.

In the next phase of our analyses, we will apply the suggested business models and relevant factors to explore, discuss, and justify potential paths for an emerging ecosystem.

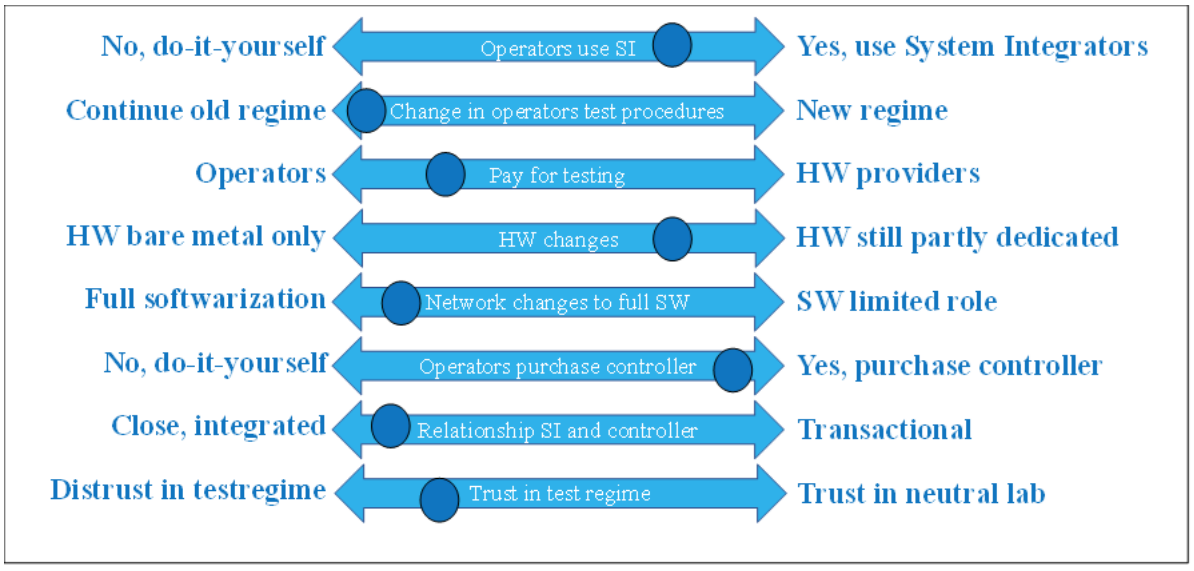


Figure 19 Factors Affecting Business Models and Ecosystem Paths and-State – dots for illustration only

3.10. Summary

We have suggested nine preliminary business model canvases for eight actor roles and indicated the existence of even more. They are all potential ways to extract revenues from customers in a regime with full compatibility, enabled by TFS.

One key observation is the difference between a future HW (bare metal) provider and the other SW providers. The HW provider will be in a classic manufacturing market where economies of scale drive competition, however, timely on-premises installation and SW configuration may significantly affect the market. Following a life-cycle management of HW, a leasing approach may be an interesting revenue model for HW providers.

The providers of SW, NetApps, and TFS-related services have two main business models. One is labour intensive, addressing a market for consulting and integration. The other relies on IPR where it may be possible to use a license revenue model. We anticipate that the market for system integration will be a large market in the sense that most operators are potential customers. For NetApps and specific TFS features we assume that the potential customers are fewer and smaller, and that this will be a niche market, potentially with a premium price.

The future of the fully compatible market we sketch seems to be dependent on the existence of actor roles which can provide compatibility testing and certification in a neutral manner. Thus, operators and providers must together mobilise, motivate, and push yet other stakeholders to take the roles as testers and certifiers. Current financial streams for testing – costs and revenues – must be re-directed into neutral labs. The willingness to invest in laboratories, expertise, testing procedures, and customer handling may emerge only when the potential revenues are identified. We have provided one first iteration of how this could be carried out.

How the mix of to-be actor roles, their business model canvases, and factors affecting their different potential paths in an evolving ecosystem will develop is not discussed here. It will be subject to further analyses in the next phase of the TeraFlow project and reported in a later deliverable.

4. Feedback from Release 1.0

In this section, we summarise the feedback received from release 1.0.

4.1. Feedback from Advisory Board

- "From the requirements perspective, which kind of interfaces shall TeraFlow support?"
  - [Noboru] For the southbound interfaces in your controller, specially to control optical transport networks, Netconf is necessary.
- "Automation: Are there network automation trends we are missing? What policy types would you like to see being supported by a controller?"
  - [Georgios] Do you think there are very important policy types to be considered in year 2 as potential policies to be supported by TeraFlow?
  - [Noboru] For automation, one missing part is how to manage, for example, automatic restoration. Do you considered to implement a restoration function in this automated SDN controller?
  - [Georgios] Reported in milestones in WP2, we foresee for year 2 to exploit existing Kubernetes features such as rollback recovery (e.g., in case of mistakenly applying a software upgrade, enable rollback mechanism to recover a stable version of the affected component). This can be seen as a way to mitigate errors coming from links and broken device configurations.
  - [Ricard] Maybe we can prioritise that for year 2. I think it makes a lot of sense.
- "Which use cases would it be interesting to demonstrate on this scenario?"
  - [Noboru] I'm interested in provisioning time. How long will it take to complete end-to-end provisioning in this demo including the optical layer?
  - [Noboru] I'm also interested in how to validate the scalability of the SDN controller in this demo.

4.2. Feedback from Users

Some users reported the need for clarifying and simplifying the deployment procedure. Comments:

- Facilitate/clarify the deployment of a development environment.
- Avoid defining environment variables in each script; instead, create scripts containing the environment variables that can be sourced and loaded in the development environment.
- Use a simplified environment instead of a complete Kubernetes environment. For instance, use a MiniKube-based environments (based or MicroK8s) for development.
- The tutorials and Wiki pages are confusing or not clear in some sections.
- Some Python requirements have the wrong version specified, and some scripts try to install conflicting versions of the requirements.

Some new features have also been proposed:

- Use a BGP Speaker to discover the devices in a network topology.
- Implement a Zero-Touch-Automation mechanism able to first connect to a whitebox through SSH, enable management protocols (e.g., Netconf, REST-API), configure the management IP address, port, and admin user credentials, and reconnect to the devices after being rebooted.
- Some users notified their interest in the P4 Device Driver and the supported features.

4.3. Feedback for TFS Questionnaire

In collaboration with the ETSI OSG TFS Marketing and Communications (MARCOM) Task Force, TeraFlow partners agreed to send a global scale questionnaire to recognise current trends, use cases and needs that probable TFS users might encounter. To this end, several questions were asked. In this section, we provide details for the questionnaire, as well as an analysis of the preliminary results (to be concluded by end of the year and reported in D6.4, M30).

Section 4.3.1 contains the text of the questionnaire as issued.

4.3.1. TFS Questionnaire

This is an open survey to the community in order to better know possible requirements, features, usages and research topics to be covered within ETSI TeraFlowSDN. Estimated response time is of 5-10 minutes. This survey will also help us to let you know the possibilities of joining our community. More info at: http://tfs.etsi.org

- We are in the process of finalizing TFS release 2. You can read more info in our blog post. Which are the features you would like to see implemented in upcoming releases?
- As NBI, release 2 will support IETF L2/L3VPN and IETF slice YANG data models. Which other NBI are of your interest? Why?
- As SBI, release 2 will support OpenConfig router data models, ONF Transport API, P4 and IETF TE for microwave. Which other SBI are of your interest? Why?
- We have demonstrated TFS in operator R&D labs. Are you interested in test and validation opportunities? Please detail interest.
- Which are your main interests for TeraFlowSDN?
    - Deploying TFS on my network
    - Develop TFS applications or extend TFS services
    - Provide support of TFS deployments
- We will be demonstrating several uses cases: transport network L2/L3 integration with beyond 5G networks, inter-domain connections between multiple operators, cybersecurity. Where do you foresee TFS deployed and to which use?
- Are you planning to download and use TFS?
    - I have already done it
    - I tried to, but it was too difficult (please comment on difficulties)
    - I will do in the next 6 months
    - No, I do not plan to do it
- Have you encountered any difficulty for using TeraFlowSDN?
- We will be happy to come back to you with more information. Please provide your e-mail, in case you wish to be contacted.

4.3.2. Preliminary Feedback Analysis

The questionnaire has been announced in Twitter and LinkedIn accounts, and sent to the following mailing lists:

- tb@5g-ppp.eu; sb@5g-ppp.eu; comms@5g-ppp.eu
- osg_tfs@list.etsi.org
- netsoft-list@ee.ucl.ac.uk
- ontc@comsoc.org
- tccc-announce@comsoc.org

Due to the need to close this section of the deliverable before the end of the questionnaire response period, we provide a preliminary analysis of the results received up to 15/12/2022. The final results of this questionnaire will be discussed in TFS Leadership Group in order to find synergies and possible new requirements for TFS release 3.0.

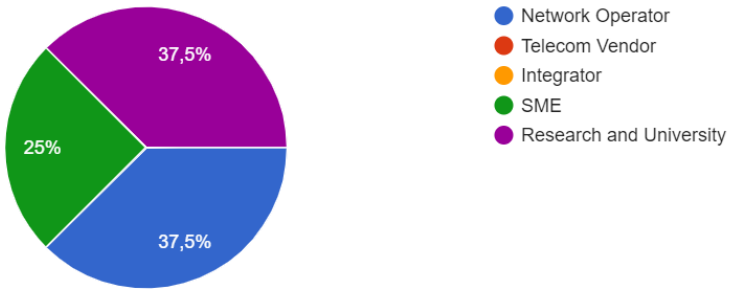


Figure 20 Stakeholders Involved in the Answers

- We are in the process of finalizing TFS release 2.0. You can read more info in our blog post. Which are the features you would like to see implemented in upcoming releases?
  - Improved support for smart traffic engineering,
  - Support for lightweight implementations for small ISPs (e.g., consider the deployment in small scenarios),
  - SDN-QKD including ETSI QKD-004 and QKD-015 components to manage integrated Quantum and optical resources,
  - Ability to balance path computation objective functions according to user's intent
- As NBI, release 2.0 will support IETF L2/L3VPN and IETF slice YANG data models. Which other NBI are of your interest? Why?
  - Inclusion of SBI support DiffServ/MPLS enabled routers. It should have a plug and play, drag and drop type of interface for easy deployment of NB services.
  - Intent based policies such as IETF i2NSF and IETF ALTO exposure.
  - IETF L0 Connectivity as a Service.
- As SBI, release 2.0 will support OpenConfig router data models, ONF Transport API, P4 and IETF TE for microwave. Which other SBI are of your interest? Why?
  - Telemetry from different sources: Prometheus exporters for IT resources, IPFIX,
  - All IETF SBIs
  - OpenFlow, gRPC, OVSDB
- We have demonstrated TFS in operator R&D labs. Are you interested in test and validation opportunities? Please detail interest.
  - Yes, we are interested to understand the effort that is needed for a SME and a small telecom operator to implement it in ONIE hardware based or P4 routers. We could also think to implement it in a live testbed.
  - Interested to know about, but not unable to assist. Would the TIP be interested in hosting this work?
  - Yes, for research
- Which are your main interests for TeraFlowSDN?

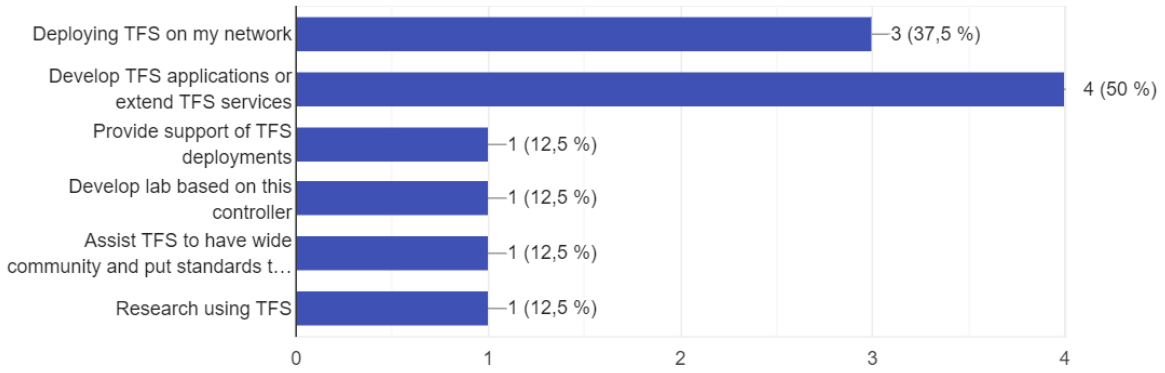


Figure 21 Main Interest in TeraFlowSDN

- We will be demonstrating several uses cases: transport network L2/L3 integration with beyond 5G networks, inter-domain connections between multiple operators, cybersecurity. Where do you foresee TFS deployed and to which use?
  - Operators / carriers transport network automation/programmability & seamless integration optical/IP devices (a single programmability endpoint).
  - We are interested in L2/L3, beyond 5G (i.e., 6G) and cybersecurity
  - Cybersecurity for research projects
  - Initial deployments for integrated, top-to-bottom EVPN and L3VPN services achieved over a multi-layer and multi-technology network
  - Transport access networks, enterprise networks connecting to multiple operators
- Are you planning to download and use TFS?
- Have you encountered any difficulty for using TeraFlowSDN?
  - Not yet tested
  - I have heard that the documentation/guidance is still immature.
  - I don't own an operational network

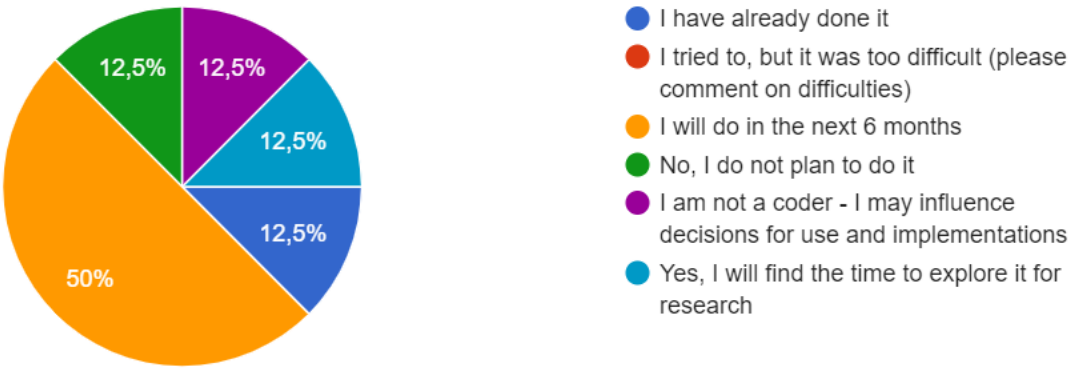


Figure 22 Expected Usage of TFS

5. Updated Requirements for the TFS Controller

This section provides the updated and new requirements for the TFS Controller. New requirements are in Green. Removed requirements in ~~red and strikethrough~~.

5.1. Functional Requirements

In this section, the different requirements for the TFS 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 2 are used.

5.1.1. Context

- [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.
- [REQ-INV-03] The Context Component needs to be able to replicate. This means that the internal database needs to be distributed.
- [REQ-INV-04] Add support for constraints in Service Component and SBI.
- [REQ-INV-05] Add service constraints to path computation.

5.1.2. SouthBound Interface (SBI)

The SouthBound Interface (SBI) was formerly named as the Device Component.

- [Updated-REQ-TOP-01] The SDN Controller shall provide a set of abstractions to represent several views of the network topology. This shall include representing the different relationships between IP and Optical network elements (physical or logical) to be consumed for different applications.
- [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.
- [REQ-TOP-03] Validation and verification of proper OpenConfig implementation of L3VPN services.
- [REQ-TOP-04] L3VPN life cycle management, including use cases of Service SLA violation detection and recovery.
- [REQ-TOP-05] Integrate MW link into L3VPN workflow.

5.1.3. Service

- [REQ-SERV-01] The SDN Controller shall be able to manage the life cycle of L2VPN and L3VPN services [13]. 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. Forecaster

- [REQ-FORE-01] A component shall be able to obtain the history of requested and serviced connectivity services with duration and capacity constraints.
- [REQ-FORE-02] The Forecaster shall include ML algorithms (prophet or AutoML) for making traffic forecasts.
- [REQ-FORE-03] Traffic forecasts shall be analysed before determining whether to accept a new service request.

5.1.5. Monitoring

General purpose monitoring requirements

- [REQ-MON-01] TFS shall allow monitoring of KPIs (metrics) of different kind and at different levels (network, compute, service, slice, etc).
- [REQ-MON-02] TFS shall be able to group/aggregate monitoring KPIs into KPI bundles in order to create different monitoring levels of abstraction.
- [REQ-MON-03] TFS shall provide data visualisation capabilities of the monitoring data.

Notification subscription requirements:

- [REQ-MON-04] TeraFlowSDN shall allow external subscriptions to the notification service to enable visualisation of relevant data at a higher level of abstraction, thus assisting the TeraFlowSDN 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.
- [REQ-MON-05] TFS shall allow the inclusion of an alarm system that enables the definition of monitoring alarms based on configured thresholds or value ranges of the monitoring data. This alarm system will notify the creator of a monitoring alarm when the monitored KPI exceeds the configured value range.

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-06] Addition of a new topology element (e.g., topology, link, node, node edge point).
- [REQ-MON-07] Modification of parameters of existing elements in the topology (e.g., scaling/migration of resources).
- [REQ-MON-08] 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-09] 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-10] New connectivity-service element inserted/removed in/from the network/slice.
- [REQ-MON-11] Status change of existing connectivity-service element in the network/slice.
- [REQ-MON-12] 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 TeraFlowSDN deployment. Functionalities that shall be monitored/notified according to the typical phases of a microservices life cycle are:

- [REQ-MON-13] Downshifting monitoring/notification, cloud-to-edge operations shall be notified to subscribers to guarantee cloud-to-edge migrations.
- [REQ-MON-14] 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-15] Edge-to-cloud load balancing monitoring/notification, subscribers shall be notified of edge-to-cloud migration operations when required.

5.1.6. 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.

5.1.7. Path Computation

Path computation request/response

- [REQ-PathComp-01] TFS shall support the exchange of request and response messages between the Service and PathComp Components. The request message contains the list of the endpoints (e.g., source and destination PEs), the objective function to be tackled by the targeted algorithm (i.e., algorithm Identifier), and the selected constraints/requirements to be met (e.g., bandwidth, latency, disjointed paths, switching capability, etc.).
- [REQ-PathComp-02] The response message should specify whether the path computation succeeds. If it does not succeed, it is recommended that the response message carry the reason/cause why the algorithm failed to find a feasible path where that information is available, if possible. If the path computation succeeds, the response message explicitly lists the set of devices and links forming the path satisfying the connectivity service. Additionally, other relevant path computation attributes are to be notified such as the resulting latency, number of hops, incurred consumed energy, etc.

Collecting context information:

- [REQ-PathComp-03] The PathComp Component should interact with the Context Component to retrieve an updated view of the involved topologies for a specific Context Id. Each topology is expected to grant an (abstracted) view of the underlying transport infrastructure encompassing the set of nodes, set of links, the node connectivity, node capabilities (e.g., edge points), attributes such as total and available capacity, link latency, link types (e.g., inter-domain, etc.), energy consumption on standby status, etc.

Path computation algorithms

- [REQ-PathComp-04] The PathComp Component can support a pool of distinct algorithms that, besides fulfilling the networking requirements of the connectivity service, may target specific networking objectives. These objectives can prioritise path and resource selection favouring overall resource utilisation or attaining reduced energy consumption. Thus, the selector of the

algorithm (or selected optimization objective) should be specified as a requirement to the PathComp Component in the request arriving from the Service Component.

- [REQ-PathComp-05] The PathComp Component within a TFS instance should be able to trigger an inter-domain path computation. This means that the PathComp Component is aware of selected and abstracted information from other domains controlled by peer TFS Controller instances. This abstracted domain information is used by the ingress TFS PathComp Component to select the domain sequence that eventually is traversed by the inter-domain network connectivity service.
- [REQ-PathComp-06] PathComp Component shall select transport resources that fulfil the network connectivity requests (bound to the Network Services) and accomplishes an effective energy reduction. This entails deploying within the PathComp Component specific energy-aware algorithms and strategies.
- [REQ-PathComp-07] Support is needed in the PathComp Component for the calculation of a pair of primary and backup paths meeting the service demands.

5.1.8. Automation

Network management requirements:

- [REQ-AUTO-01] Automatic topology discovery and inventory tracking of both physical and virtual network elements.
- [Updated-REQ-AUTO-02] Automated device-level configuration and management through NBI (i.e., gRPC) and SBI (i.e., all the SB protocols supported by the SBI) 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:

- [Updated-REQ-AUTO-05] Automatically deploy a base configuration when a new network element is added to the network, such that the network element enters into 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.
- [Updated-REQ-AUTO-10] Automatic provisioning of basic white box configuration, common across multiple white box vendors (e.g., P4 vs. OpenConfig).
- [New-REQ-AUTO-11] Automatic provisioning of device configuration at runtime, including both device configuration updates and removed device configuration.

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.

5.1.9. 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.
- [Updated-REQ-POL-02] A policy rule shall be associated with a service ID. When a service ID is provided, the Policy Component will query the Context database to automatically fetch the set of devices that this service is traversing. When a service ID is not provided, the Policy Component will iterate a list of devices in the policy rule object, to identify which devices should comply to the input policy.
- [Updated-REQ-POL-03] A policy rule shall be associated with a policy state, which indicates whether this policy is (i) inserted, (ii) validated, (iii) provisioned, (iv) actively enforced, or (v) failed. Moreover, the policy state machine will also capture states that classify an applied policy as effective or ineffective, so as to inform the network operator accordingly.
- [Updated-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). Policies applied to an entire Service are likely to be network-wide policies as a service typically traverses more than one device.

5.1.10. Transport Network Slicing

- [REQ-SLI-01] The TeraFlowSDN 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 TeraFlowSDN 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 prioritisation in resource utilisation.
- [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).

- [REQ-SLI-07] Slice SLA shall be mapped as a technology abstract intent, regardless of the underlying implementation (e.g., L2VPN, L3VPN). Slices once deployed shall be monitored and enforced, in terms of SLA constraints.
- [REQ-SLI-08] Slice grouping introduces a clustering algorithm for finding service optimisation while preserving slice SLA.

5.1.11. Centralized Attack Detector

- [REQ-CAD-01] The Centralized Attack Detector (CAD) shall subscribe to events related to services and setup the appropriate information related to the security assessment (e.g., monitoring KPIs) upon the creation/update/deletion of services.
- [REQ-CAD-02] The CAD shall orchestrate the security assessment loop at the optical layer by retrieving monitoring data, invoking the Attack Inference Component, and triggering attack mitigation upon the detection of attacks.
- [REQ-CAD-03] The periodicity of the CAD security assessment loop at the optical layer shall be set considering the periodicity of the monitoring cycle.
- [REQ-CAD-04] The CAD Component shall orchestrate the processing of the flow statistics data received from the DAD, invoking the ML model, and triggering the AM Component upon the detection of attacks.
- [REQ-CAD-05] The CAD Component shall detect attack flows with an accuracy equal or greater than 99% for known attacks.

5.1.12. Distributed Attack Detector

- [REQ-DAD-01] The Distributed Attack Detector (DAD) shall aggregate packets per flow, producing statistics that are later send to the Centralised Attack Detector (CAD) Component.

5.1.13. Attack Inference

- [REQ-AInf-01] The Attack Inference Component shall provide an interface that allows external components to provide samples and receive their status as normal or abnormal. Alternatively, when attack identification is possible, the attack class will be provided.
- [REQ-AInf-02] The Attack Inference Component shall use AI/ML techniques that allow for accurate and reliable detection and identification of attacks.
- [REQ-AInf-03] The Attack Inference Component shall use AI/ML model implementations that are efficient both in terms of computational complexity and response time.

5.1.14. Attack Mitigator

- [REQ-AM-01] The Attack Mitigator shall provide an interface that allows other components (such as the CAD Component) to notify it of the presence of an attack in the network. The notification shall contain details such as which layer (e.g., optical or Layer 3), the service under attack and the attack connection.
- [REQ-AM-02] The Attack Mitigator shall compute efficient mitigation strategies based on the characteristics of the detected attack.
- [REQ-AM-03] The Attack Mitigator shall coordinate with other components (i.e., Service and Context Components) the actions to be taken to mitigate the attack.

5.1.15. Distributed Ledger and Smart Contracts

- [REQ-DLT-01] The TFS Controller shall be able to interact with other SDN Controllers using DLT to provide inter-domain slices.
- [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).

5.1.16. NorthBound Interface (NBI)

From a macroscopic perspective, the NBI (previously the Compute Component) behaves as the front-end to take over the interactions of the TFS instance, for example with an external NFV orchestrator (e.g., OSM implementation). The aim is to automatically receive and process network connectivity service requests, for example from the NFV Orchestrator (e.g., creation, deletion, and removal) and then interact with other TFS components to meet such requested connectivity service operations. To this end, the following requirements need to be fulfilled. Note that unchanged requirements still refer to the Compute Component: those requirements must now be met by the functionality of the NBI.

- [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 TeraFlowSDN components (e.g., Service).
- [REQ-COM-06] The NFV Orchestrator-Compute Component API (based on REST) needs to be extended to support specific network connectivity service characteristics such as required protection level (e.g., disjoint paths), maximum tolerated latency, etc. These extensions are included in the request message and processed by the Compute Component.
- [REQ-COM-07] The Compute Component should map the new network connectivity service arriving from the NFV Orchestrator into the contents of the gRPC API connecting to the Service Component.
- [REQ-COM-08] Utilisation of the available OSM Placement Component to select PoP and PiLs for incoming Network Service requests targeting overall energy reduction.
- [REQ-COM-09] Support in the OSM NBI API for new connectivity service attributes and requirements such as disjoint paths for a single network connectivity service request. These new features and requirements need to also be mapped into the gRPC API between the NBI and Service Component.

5.1.17. Inter-Domain

The Inter-Domain use case describes the interaction of a TFS instance with peer TFS 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.
- [REQ-INT-05] Inter-domain Component shall trigger monitoring of inter-domain KPIs to validate fulfilment of SLAs.
- [REQ-INT-06] Inter-domain Component shall trigger the mitigation actions when an SLA is violated.
- [REQ-INT-07] Inter-domain Component shall interact with Path computation to select domains and per-domain SLAs based on E2E SLA requirements.

5.1.18. Web User Interface

The WebUI enables a network operator to manually interact with the TFS Controller to perform configuration operations and inspect the state of the network.

- [REQ-UI-01] The WebUI needs to be implemented as another TFS micro-service.
- [REQ-UI-02] The WebUI needs different sections: Home, Device, Link, Service, Slice, Grafana, and About. Each section shall provide information about internal TFS context state.

5.1.19. TFS Controller Security

Historically, SDN controllers have been implemented as monoliths and provide authentication and authorisation 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.

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

- [REQ-SEC-01] The TeraFlowSDN 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 TeraFlowSDN 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 TeraFlowSDN controller shall provide means for user authorisation. 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 TeraFlowSDN 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. Note that there are no modified or additional non-functional requirements presented in this document.

5.2.1. Performance

- [REQ-PERF-01] The TeraFlowSDN 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 optimisation 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 optimisation.
- [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 centralised 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 TeraFlowSDN 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 visualise the metrics in a friendly and customisable 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] Optimised 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 TeraFlowSDN 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 TeraFlowSDN controller.

6. Updated Architecture

6.1. Overall Architecture Update

For release 2.0, novel features have been proposed, and facilities have been consolidated and updated. Significant architectural work has been performed in the scope of WP2 in order to better provide a framework for the integration of the multiple micro-services that compose TFS. Figure 23 shows the updated TFS architecture for release 2.0.

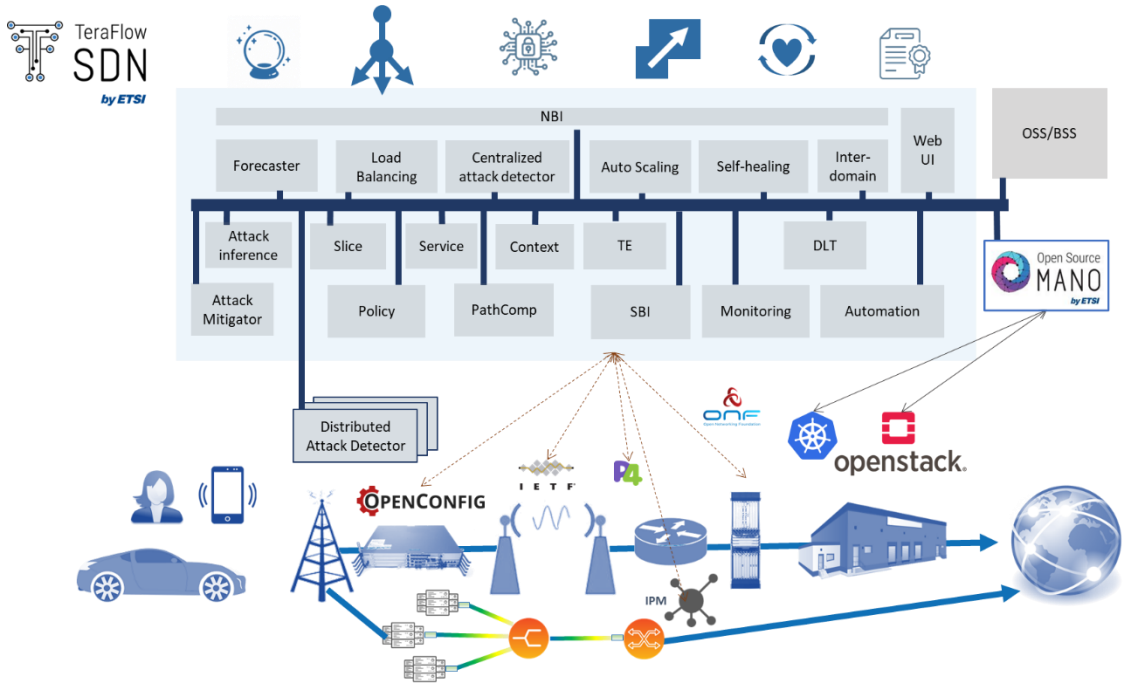


Figure 23 TFS Release 2.0 Architecture

In this release, focus has been provided on scalability and resilience of the controller, with a complete re-design of the Context Component to include a scalable database (i.e., CockroachDB) to support the stringent non-functional requirements.

Moreover, dedicated attention to network automation has been provided in this new release, with the inclusion and update of network automation workflows, including L3VPN establishment with SLA, multi-layer topology discovery, service Access Control Lists (ACLs), service restoration, service location-awareness, traffic engineering, slice SLA enforcement, slice grouping, forecasting, inter-domain slice SLA enforcement, inter-domain connectivity provisioning and SLA enforcement using DLT, and energy-aware network service placement.

Release 2.0 provides extended support for OpenConfig-based routers and interaction with optical SDN controllers through the ONF Transport API. Moreover, release 2.0 includes complete integration of microwave network elements (through the IETF network topology YANG model), and Point-to-Multipoint integration of XR optical transceivers and P4 routers. New features for P4 routers include: load a P4 pipeline on a given P4 switch; get runtime information (i.e., flow tables) from the switch; and push runtime entries into the switch pipeline.

SLA validation has been re-engineered through all the workflows, from device monitoring, up to service and slice life cycle management. Thus, the Slice, Service, Policy, and Monitoring Components and the SBI have been updated in order to support the necessary network automation workflows.

Moreover, slice grouping has also been introduced, as well as the PathComp Component. This component allows new use cases, such as energy-aware service placement.

Cybersecurity mechanisms have been updated, including novel components for attack detection, (either distributed or centralised), attack inference, and attack mitigation. Several novel use cases are supported. DLT has also been extended to interact with the Inter-Domain Component and make use of deployed Hyperledger Fabric.

An updated description of each component is provided below.

- The Context Component is responsible for the stateful record of the necessary information. It provides an internal API to obtain and manipulate TFS status. It is responsible for interactions with a cloud-scale database.
- The SouthBound Interface (SBI) (previously the Device Component) provides inventory information and allows configuration and management of specific devices though multiple SBI plugins including OpenConfig routers, ONF Transport API, XR constellation Driver, IETF Network Topology, and P4.
- The Service Component manages the lifecycle of multiple TFS services (including L3VPN and L2VPN network models).
- The Forecaster is a component able to perform proactive SDN traffic optimisation by means of ML algorithms, through the collection of real-time KPI data, and use of ML to forecast where and when a problem is likely to occur so as to reroute traffic before the problem happens.
- The Monitoring Component provides a subscription manager, which is in charge of offering subscription capabilities to the rest of the TFS Controller components. It also provides an alarm manager in order to provide an alarm system to the TFS Controller.
- The Traffic Engineering Component manages Segment Routing paths.
- The PathComp Component handles route and network resource selection, fulfilling network connectivity services and targeting a specific network objective (e.g., energy-efficiency, resource-efficiency).
- The Automation Component can automatically add/update/delete a physical or virtual device to/in/from the network with zero manual intervention, while ensuring that the correct device configuration parameters and device processing logic are installed, updated, or deleted.
- The Policy Component automatically translates high-level network policy rules to actual configuration applied to devices and services. A policy rule may generate configuration across an entire network domain, thus may need to configure multiple devices.
- The Slice Manager handles transport network slices with an SLA lifecycle, including slice monitoring and SLA violation recovery mechanisms. Moreover, a slice grouping algorithm is also included to increase resource utilisation efficiency.
- The Centralized Attack Detector coordinates the cybersecurity loop of the TFS Controller at both the L3 and optical layers.
- The Distributed Attack Detector is used for the security monitoring of layer 3 traffic. It detects and classifies attacks at remote sites (network edge) in a distributed fashion.
- The Attack Inference Component performs anomaly detection inference based on a set of samples. The implementation currently uses an unsupervised learning algorithm for anomaly detection.
- The Attack Mitigator Component is responsible for computing viable attack remediation solutions, depending on the attacks detected by other components.

- The Distributed Ledger Technology Component provides a distributed ledger gateway to record, query, and process relevant data for network management and detection of compromised edge-devices.
- The NBI (formerly the Compute Component) exposes a REST-based API for the export of information such as L2/L3VPN services or topology. It includes the function of the previous Compute Component, which allows the component to act as an ETSI OpenSource MANO (OSM) SDN/WIM connector.
- The Inter-Domain Component enables interaction between a TFS instance and peer TFS instances which manage different network domains to create E2E TN slicing services.
- The Web User Interface (WebUI) provides a graphical interface to easily navigate through internal TFS information, as well allowing manual service configuration.

6.2. Detailed Architecture

In this section we describe the components, organised as CoreApps and NetApps.

Reference to protocol buffers available at the controller directory of the TFS GitLab repository (https://labs.etsi.org/rep/tfs/controller).

6.2.1. Context

Name:	Context
Objective:	Stateful record of the necessary information. Provide the internal API to obtain and manipulate TFS status.
Requirements:	Shall include a cloud-scale database (e.g., CocroachDB). Internal interface shall be provided in gRPC.
References:	
Responsible (and collaborators):	CTTC, TID
Provided Operations:	rpc ListContextIds (Empty) returns (ContextIdList) {} rpc ListContexts (Empty) returns (ContextList) {} rpc GetContext (ContextId) returns (Context) {} rpc SetContext (Context) returns (ContextId) {} rpc RemoveContext (ContextId) returns (Empty) {} rpc GetContextEvents (Empty) returns (stream ContextEvent) {} rpc ListTopologyIds (ContextId) returns (TopologyIdList) {} rpc ListTopologies (ContextId) returns (TopologyList) {} rpc GetTopology (TopologyId) returns (Topology) {} rpc SetTopology (Topology) returns (TopologyId) {} rpc RemoveTopology (TopologyId) returns (Empty) {} rpc GetTopologyEvents (Empty) returns (stream TopologyEvent) {} rpc ListDeviceIds (Empty) returns (DeviceIdList) {} rpc ListDevices (Empty) returns (DeviceList) {} rpc GetDevice (DeviceId) returns (Device) {} rpc SetDevice (Device) returns (DeviceId) {} rpc RemoveDevice (DeviceId) returns (Empty) {} rpc GetDeviceEvents (Empty) returns (stream DeviceEvent) {} rpc ListLinkIds (Empty) returns (LinkIdList) {} rpc ListLinks (Empty) returns (LinkList) {}

	rpc GetLink (LinkId) returns (Link) {} rpc SetLink (Link) returns (LinkId) {} rpc RemoveLink (LinkId) returns (Empty) {} rpc GetLinkEvents (Empty) returns (stream LinkEvent) {} rpc ListServiceIds (ContextId) returns (ServiceIdList) {} rpc ListServices (ContextId) returns (ServiceList) {} rpc GetService (ServiceId) returns (Service) {} rpc SetService (Service) returns (ServiceId) {} rpc RemoveService (ServiceId) returns (Empty) {} rpc GetServiceEvents (Empty) returns (stream ServiceEvent) {} rpc ListSliceIds (ContextId) returns (SliceIdList) {} rpc ListSlices (ContextId) returns (SliceList) {} rpc GetSlice (SliceId) returns (Slice) {} rpc SetSlice (Slice) returns (SliceId) {} rpc RemoveSlice (SliceId) returns (Empty) {} rpc GetSliceEvents (Empty) returns (stream SliceEvent) {} rpc ListConnectionIds (ServiceId) returns (ConnectionIdList) {} rpc ListConnections (ServiceId) returns (ConnectionList) {} rpc GetConnection (ConnectionId) returns (Connection) {} rpc SetConnection (Connection) returns (ConnectionId) {} rpc RemoveConnection (ConnectionId) returns (Empty) {} rpc GetConnectionEvents(Empty) returns (stream ConnectionEvent) {}
Internal Models:	Context Slice Service Connection Device Link Topology

Figure 24 shows the Context Data Model and relationship between its objects. The figure is difficult to read in this document. It is made available at in the TFS GitLab repository at https://labs.etsi.org/rep/tfs/controller/-/blob/master/proto/uml/context.png for more detailed inspection.https://labs.etsi.org/rep/tfs/controller/-/blob/master/proto/uml/context.png

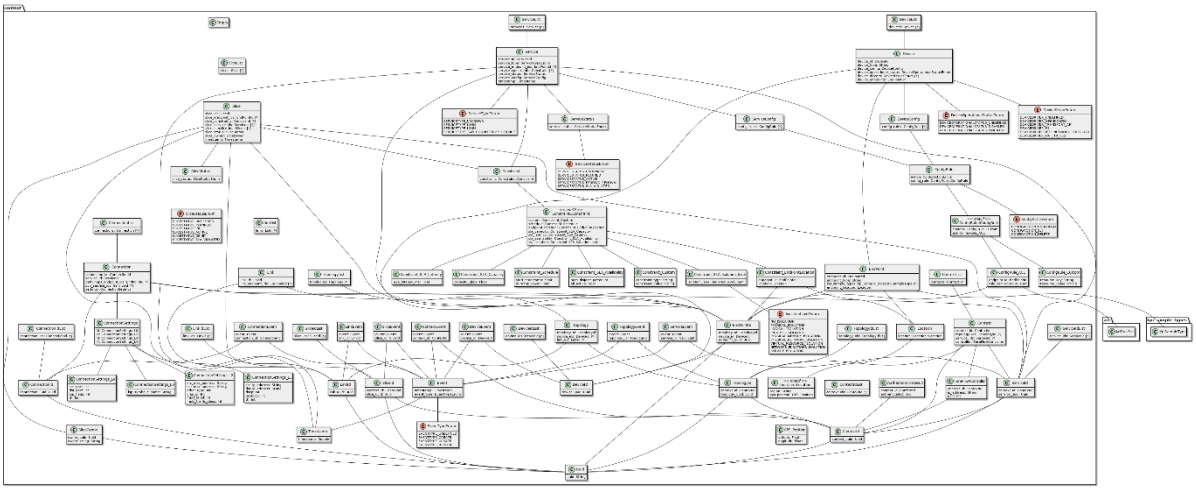


Figure 24 Context Data Model

6.2.2. SBI (Formerly Device Manager)

Name:	SBI
Objective:	Provide inventory information and allow configuration and management of specific devices though multiple SBI plugins including OpenConfig routers, ONF Transport API, IETF Network Topology, and P4.
Requirements:	Shall be able to handle multiple types of devices. Shall be able to perform configuration and management
References:	OpenConfig, ONF Transport API, IETF Network Topology, P4
Responsible (and collaborators):	TID, SIAE, INF, VOL, UBI
Provided Operations:	rpc AddDevice (context.Device) returns (context.DeviceId) {} rpc ConfigureDevice (context.Device) returns (context.DeviceId) {} rpc DeleteDevice (context.DeviceId) returns (context.Empty) {} rpc GetInitialConfig(context.DeviceId) returns (context.DeviceConfig) {} rpc MonitorDeviceKpi(MonitoringSettings) returns (context.Empty) {}
Internal Models:	Only provides operations. Data models belong to context

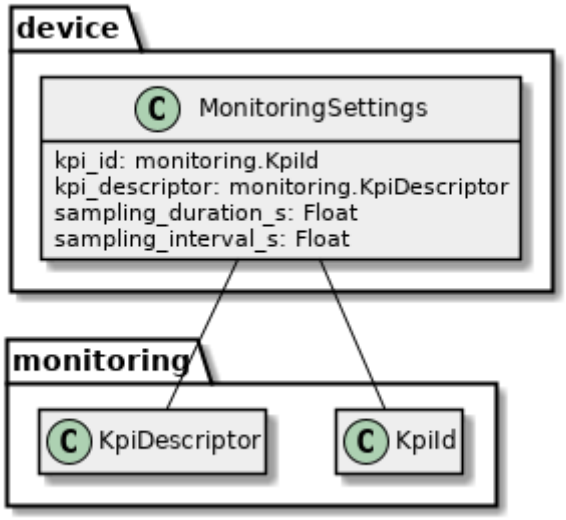


Figure 25 Device Internal Data Model

6.2.3. Service

Name:	Service
Objective:	Lifecycle management of multiple TeraFlow services (including L3 and L2 VPN network models).
Requirements:	Provide L3NM, L2NM lifecycle management support.
References:	L3NM, L2NM. [3], [4]

Responsible (and collaborators):	CTTC, INF, TID, UBI
Provided Operations:	rpc CreateService(context.Service) returns (context.ServiceId) {} rpc UpdateService(context.Service) returns (context.ServiceId) {} rpc DeleteService(context.ServiceId) returns (context.Empty) {}
Internal Models:	No internal models, as defined in Context

6.2.4. Forecaster

Name:	**Forecaster**
Objective:	Perform proactive SDN traffic optimisation 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, so as to reroute traffic before the problem happens).
Requirements:	[REQ-FORE-01] [REQ-FORE-02][REQ-FORE-03]
References:	-
Responsible (and collaborators):	CTTC, TID
Provided Operations:	rpc GetForecastOfTopology (context.TopologyId) returns (Forecast) {} rpc GetForecastOfLink(context.LinkId) returns (Forecast) {} rpc CheckService (context.ServiceId) returns (ForecastPrediction) {}
Internal Models:	Forecast ForecastPrediction

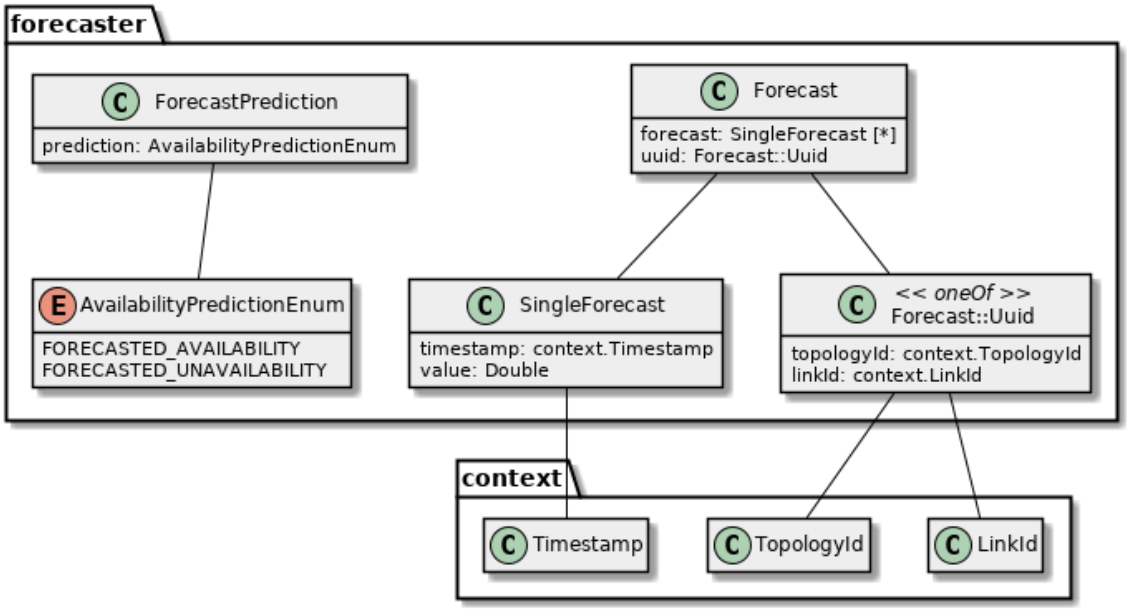


Figure 26 Forecaster Internal Data Models

6.2.5. Monitoring

The updated architecture of the TFS Monitoring Component is depicted in Figure 27. The new architecture shows the new Subscription Manager and Alarm Manager Components as well as the updated metrics database.

The Subscription Manager is in charge of offering subscription capabilities to the rest of the components of the TFS Controller, these subscriptions can be created, updated, and deleted through a set of RPCs. The subscriptions can be configured with multiple parameters such as the start date, end date, the sampling interval, and the sampling duration.

The updated architecture also incorporates an Alarm Manager in order to provide an alarm system to the TFS Controller. The alarm system allows configuration of alarm timeouts, alarm frequency, and KPI value ranges. When the value of a monitored KPI exceeds the configured value range during the alarm frequency time period the Alarm Manager will notify the component that originally created the alarm.

The updated architecture also externalises the MngmtDB to the Context Component and updates from MetricsDB to QuestDB [17], a much more powerful and scalable database in order to achieve the high-performance capabilities required by the TFS Controller.

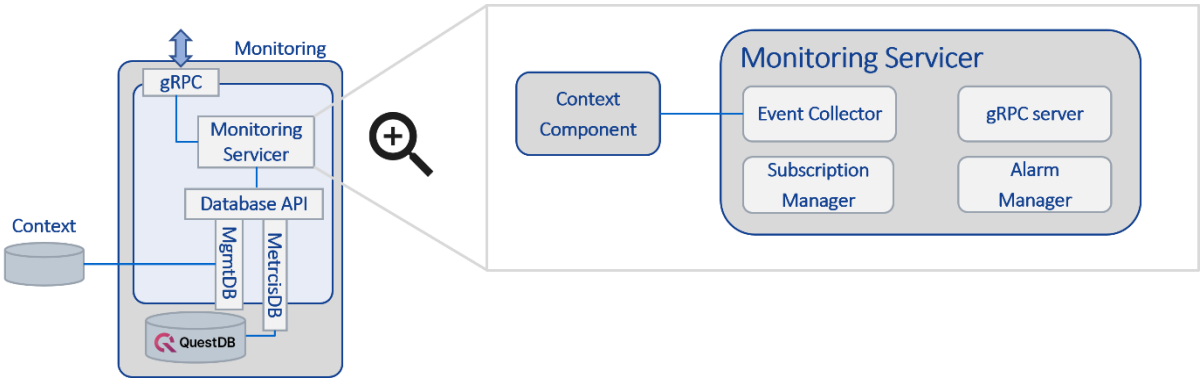


Figure 27 TeraFlow Monitoring Component Updated Architecture

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 group monitoring KPIs into KPI bundles. Shall be able to create, read, update, and delete KPIs and KPI bundles. Shall be able to allow external subscriptions to the notification service. Shall be able to configure alarms with KPI thresholds (value range). Shall be able to modify external subscriptions and alarms. Shall be able to add, modify, and visualise topology, connectivity, and microservice life-cycle metrics.
References:	QuestDB
Responsible (and collaborators):	ATOS
Provided Operations:	SetKpi (KpiDescriptor) returns (KpiId) DeleteKpi (KpiId) returns (context.Empty) GetKpiDescriptor (KpiId) returns (KpiDescriptor)

	GetKpiDescriptorList (context.Empty) returns (KpiDescriptorList) IncludeKpi (Kpi) returns (context.Empty) MonitorKpi (MonitorKpiRequest) returns (context.Empty) QueryKpiData (KpiQuery) returns (RawKpiTable) SetKpiSubscription (SubsDescriptor) returns (stream SubsResponse) GetSubsDescriptor (SubscriptionID) returns (SubsDescriptor) GetSubscriptions (context.Empty) returns (SubsList) DeleteSubscription (SubscriptionID) returns (context.Empty) SetKpiAlarm (AlarmDescriptor) returns (AlarmID) GetAlarms (context.Empty) returns (AlarmList) GetAlarmDescriptor (AlarmID) returns (AlarmDescriptor) GetAlarmResponseStream (AlarmSubscription) returns (stream AlarmResponse) DeleteAlarm (AlarmID) returns (context.Empty) GetStreamKpi (KpiId) returns (stream Kpi) GetInstantKpi (KpiId) returns (Kpi)
Internal Models: (see for reference Figure 28)	KpiDescriptor KpiQuery RawKpi RawKpiList RawKpiTable KpiId Kpi KpiValueRange KpiValue KpiList KpiDescriptorList SubsDescriptor SubscriptionID SubsResponse SubsList AlarmDescriptor AlarmID AlarmSubscription AlarmResponse AlarmList

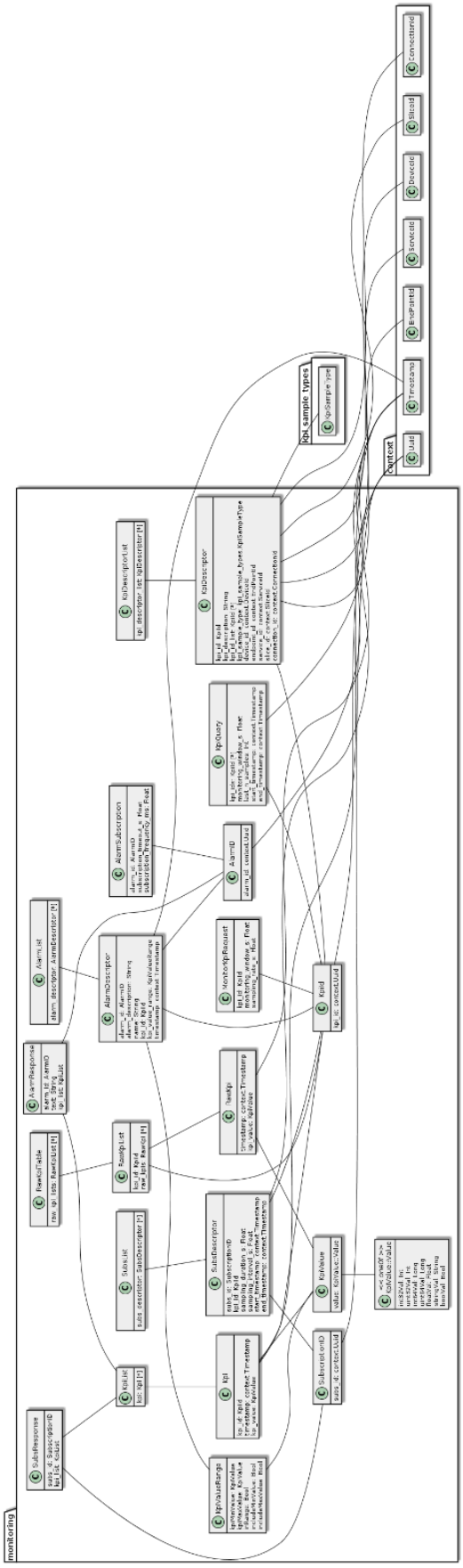


Figure 28 Monitoring Data Model

6.2.6. Traffic Engineering

Name:	Traffic Engineering
Objective:	Manage Segment Routing paths
Requirements:	Create, modify, and delete Segment Routing paths Should interact with the TFS Context Component to retrieve device and topology information. Should interact with the SBI to configure compatible hardware to join the PCE.
References:	IETF PCEP [26]
Responsible (and collaborators):	STR
Provided Operations:	rpc RequestLSP (context.Service) returns (context.ServiceStatus) {} rpc UpdateLSP (context.ServiceId) returns (context.ServiceStatus) {} rpc DeleteLSP (context.ServiceId) returns (context.Empty) {}
Internal Models:	N/A

This component implements the creation, modification, and deletion of Segment Routing paths on the available hardware, considering the given constraints and the available resources. The constraints given to the PCE for the calculation of the path could be required or desired latency, bandwidth consumption, hop count, and whether the result should be a strict explicit path or a loose path.

6.2.7. Path Computation

Name:	PathComp
Objective:	To handle route and network resource selection fulfilling network connectivity services and targeting a specific network objective (e.g., energy-efficiency, resource-efficiency, etc.)
Requirements:	• Receive the requests for computing one or more connectivity services specifying diverse network requirements • Host a pool of heterogenous algorithms targeting diverse network objectives • Compute feasible end-to-end paths (meeting demanded requirements and constraints) within either a single or multiple domains and technologies • Retrieve the Context
References:	-
Responsible (and collaborators):	CTTC
Provided Operations:	Compute(PathCompRequest) returns (PathCompReply)
Internal Models:	N/A

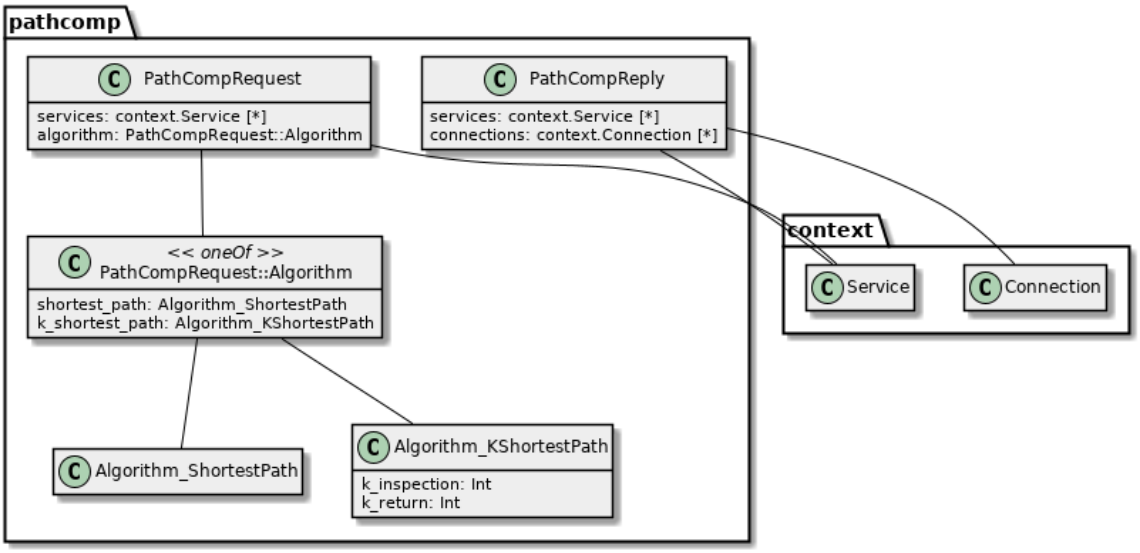


Figure 29 Path Computation Internal Models

6.2.8. Automation (ZTP)

Name:	Automation
Objective:	Automatically add/update/delete a physical or virtual device to/in/from the network with zero manual intervention, while ensuring that the correct device configuration parameters and device processing logic are installed, updated, or deleted. This component produces role-based device configuration, which is sent to the SBI via gRPC, and then communicated to the underlying devices via a southbound configuration protocol (e.g., gNMI, Netconf, P4) implemented by the respective device driver plugin. The Automation Component architecture is depicted in Figure 30.

Requirements:	• Shall be able to handle multiple types of devices.
	  ○ Requires interactions with the SBI.
	• Shall be able to receive and validate a DeviceRole from its northbound interface
	  ○ Such role-based configuration may be received by another TFS component and/or an external entity either automatically or manually.
	  ○ A gRPC protocol buffer model describes this API.
	• Shall be able to associate a valid DeviceRole with a physical or virtual device in the network.
	  ○ This is an internal process once a northbound API call is received and before it is communicated to the SBI.
	• Shall be able to send a DeviceRole to the SBI through a gRPC call.
	  ○ Requires interactions with theSBI.
	• Shall be able to perform device management.
	  ○ Requires interactions with the SBI.

References:	-
Responsible (and collaborators):	UBI, UPM, TID, TNOR

Provided Operations:	rpc ZtpGetDeviceRole(DeviceRoleId) returns (DeviceRole) {} rpc ZtpGetDeviceRolesByDeviceId(context.DeviceId) returns (DeviceRoleList) {} rpc ZtpAdd(DeviceRole) returns (DeviceRoleState) {} rpc ZtpUpdate(DeviceRole) returns (DeviceRoleState) {} rpc ZtpDelete(DeviceRole) returns (DeviceRoleState) {} rpc ZtpDeleteAll(Empty) returns (DeviceDeletionResult) {}
Internal Models:	The main model of this component is the DeviceRole, which contains additional internal models, while exploiting models are provided by other components (see Figure 31).

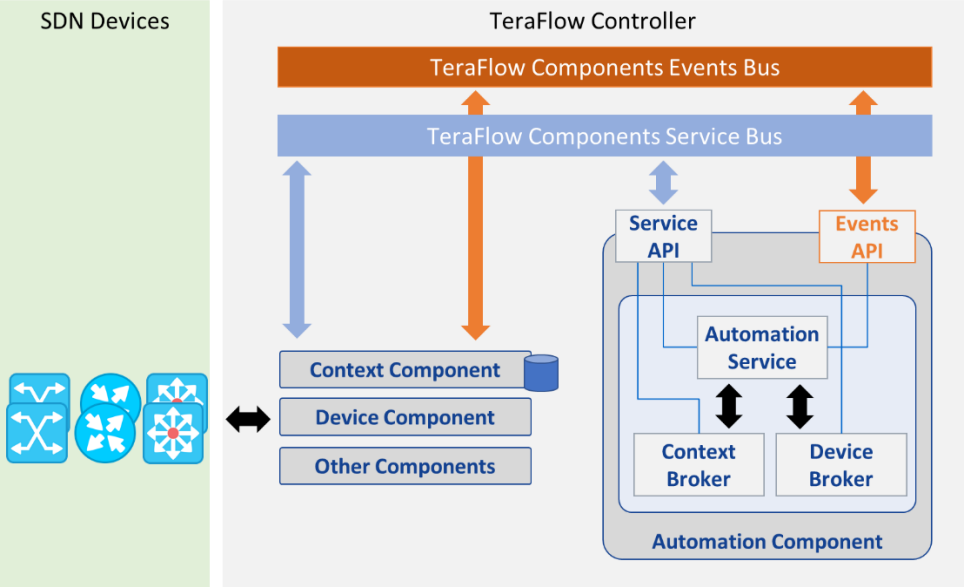


Figure *30* Automation Component Architecture.

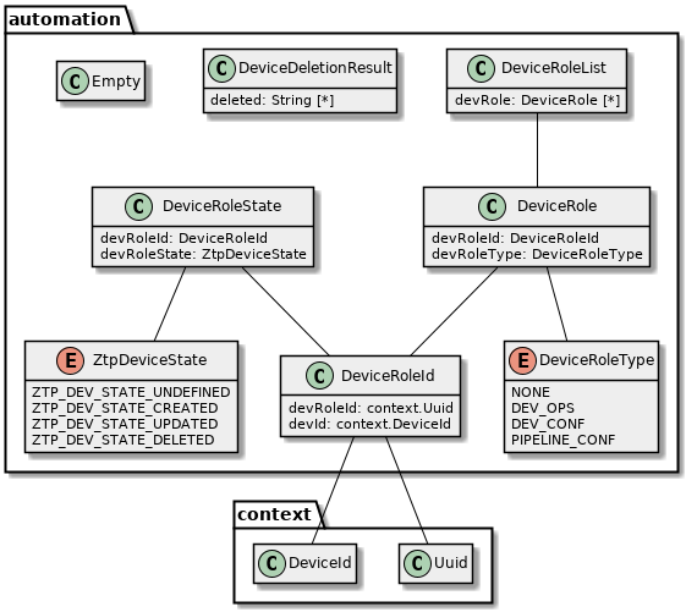


Figure 31 Automation Internal Data Models

6.2.9. Policy Manager

Name:	Policy
Objective:	Automatically translates high-level network policy rules to actual configuration applied to devices and services. A policy rule may generate configuration across an entire network domain, thus may need to configure multiple devices. The Policy Component architecture is depicted in Figure 32.

Requirements:	• Shall be able to handle multiple types of devices.
	  ○ Requires interactions with the SBI.
	• Shall be able to handle multiple types of services.
	  ○ Requires interactions with the TeraFlow Service Component.
	• Shall be able to receive and validate a PolicyRule from its northbound interface.
	  ○ Such a policy rule may be received by an external entity, either automatically or manually by a network operator.
	  ○ A gRPC protocol buffer model describes this API.
	• Shall be able to model service-level agreements (SLAs)
	  • A policy rule associates services with their key performance indicators (KPIs) and conditionally prescribes remedy actions when these KPIs fall outside a desired range of values.
	• Shall be able to assess service-level agreement adherence in real time
	  • The Policy Components uses Monitoring Services to create alarms on conditional KPI queries. Once a KPI reaches certain value or falls within a range of values, an alarm triggers the policy enforcement.
	• Shall be able to enforce a policy action in real time
	  ○ Upon the reception of an alarm, the Policy Component applies a remedy action to maintain a target SLA.

References:	-
Responsible (and collaborators):	UBI, ODC, UPM, TID, TNOR
Provided Operations:	rpc PolicyAddService (PolicyRuleService) returns (PolicyRuleState) {} rpc PolicyAddDevice (PolicyRuleDevice) returns (PolicyRuleState) {} rpc PolicyUpdateService (PolicyRuleService) returns (PolicyRuleState) {} rpc PolicyUpdateDevice (PolicyRuleDevice) returns (PolicyRuleState) {} rpc PolicyDelete (PolicyRuleId) returns (PolicyRuleState) {} rpc GetPolicyService (PolicyRuleId) returns (PolicyRuleService) {} rpc GetPolicyDevice (PolicyRuleId) returns (PolicyRuleDevice) {} rpc GetPolicyByServiceId (context.ServiceId) returns (PolicyRuleServiceList) {}
Internal Models:	The main model of this component is the PolicyRule, which contains additional internal models, while exploiting models provided by other components.

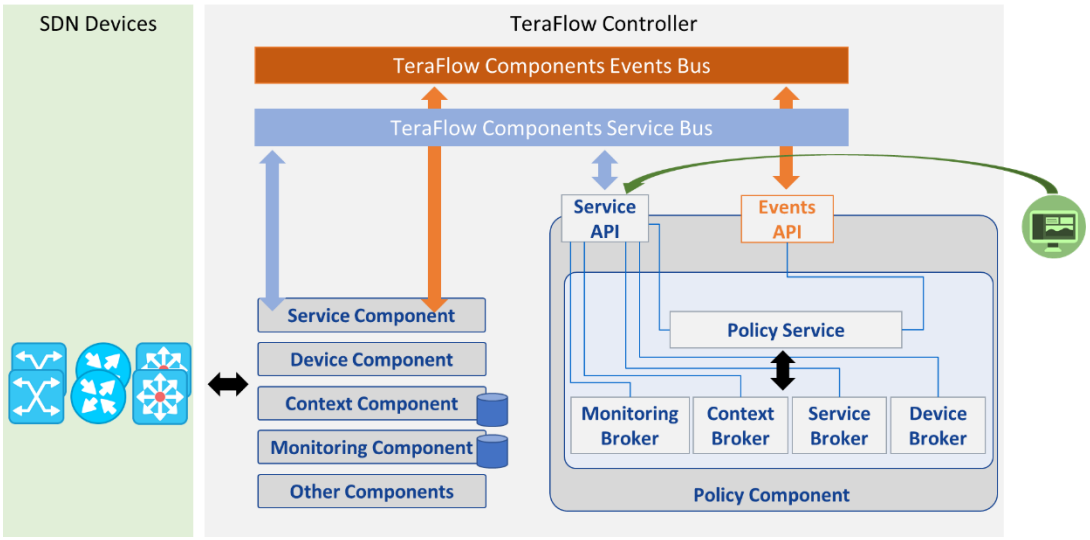


Figure 32 Policy Component Architecture.

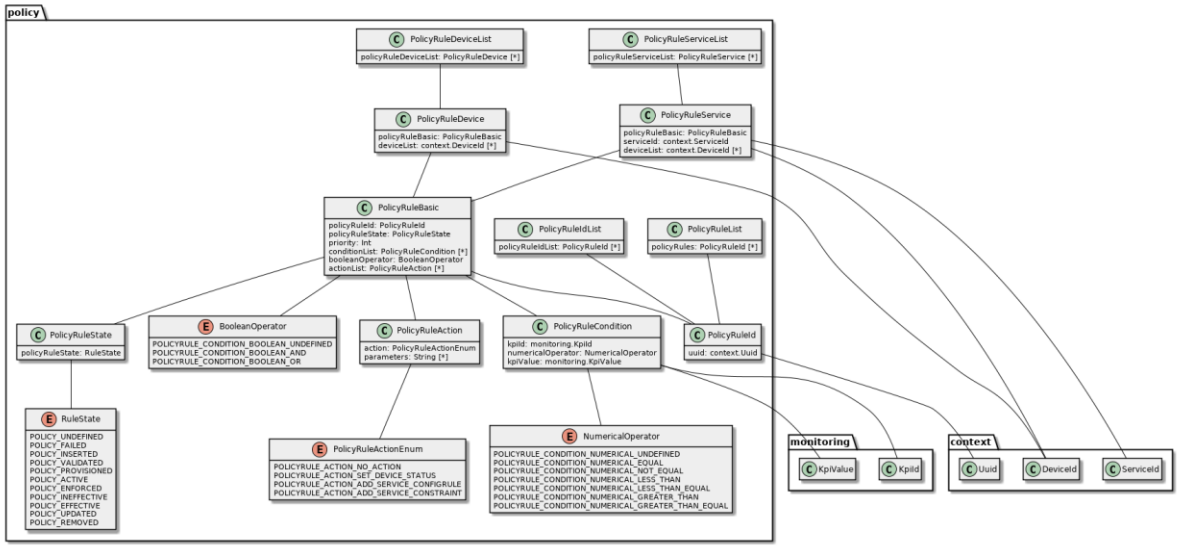


Figure 33 Policy Internal Data Models

6.2.10. Slice Manager

Name:	Slice
Objective:	Handle Transport Network Slice lifecycle
Requirements:	[REQ-SLI-01] - [REQ-SLI-06]
References:	IETF Network Slices [21]A YANG model for IETF Network Slices [12]
Responsible (and collaborators):	ADVA, CTTC, TNOR, NTNU, UBI
Provided Operations:	rpc CreateSlice(context.Slice) returns (context.SliceId) {} rpc UpdateSlice(context.Slice) returns (context.SliceId) {} rpc DeleteSlice(context.SliceId) returns (context.Empty) {} rpc OrderSliceWithSLA(context.Slice) returns (context.SliceId) {} // If slice with SLA already exists, returns slice. If not, creates it.

	rpc RunSliceGrouping (context.Empty) returns (context.Empty) {} // Optimises the underlying services and re-maps them to the requested slices.
Internal Models:	None. Located in Context

6.2.11. Centralized Attack Detector

Name:	**CAD**
Objective:	This component coordinates the cybersecurity loop of the TFS Controller. Within the project, we tackle the security assessment at two layers: optical and Layer 3 (L3). At L3, this component provides attack detection capabilities and a consolidated attack detection mechanism based on the input provided by the DAD. The CAD utilises ML algorithms to classify the input data received from the DAD and sends information regarding the address and port of the source and destination of the connection as well as the confidence level of the decision to the Attack Mitigator (AM). Only the connections that are detected as being part of an attack with a confidence level greater than a certain threshold are sent to the AM. In addition, the CAD collects the data sent by the DAD that will be processed periodically at a given time interval to produce relevant security-related KPIs that will be sent to the Monitoring Component. At the optical layer, this component implements the security assessment loop that retrieves OPM data from the Monitoring Component, and leverages on the Attack Inference to obtain attack detection. Upon a detection, it relies on the AM to compute and perform the mitigation strategy.
Requirements:	For the optical layer: • Shall consume/subscribe to service creation/update/deletion events triggered by the Context Component. • Shall consume/subscribe to security-related data from the Monitoring Component. • Shall process the summarised flow KPIs from the Monitoring Component and generate a consolidated data plane security status. • Shall report detected attacks to the Attack Mitigator. • Shall report security status to the Monitoring Component. For L3: • Shall consume network connection-related data from the DAD. • Shall report detected attacks to the Attack Mitigator. • Shall detect the summarised network flows that were detected as being part of a known attack with an accuracy equal or greater of 99%. • Shall collect real-time data from the DAD and process it periodically every 5-second time interval to produce relevant security-related KPIs. • Shall report security-related KPIs to the Monitoring Component.
References:	-
Responsible (and collaborators):	UPM, TID (L3), CHAL (Optical)
Provided Operations:	For the optical layer: rpc **DetectAttack** (Empty) returns (Empty) {} For L3:

	rpc **SendInput** (L3CentralizedattackdetectorMetrics) returns (Empty) {}
Internal Models:	L3AttackmitigatorOutput is sent to the AM and L3CentralizedAttackDetectorMetrics is received from the DAD. See for reference Figure 34 and Figure 35, respectively.

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

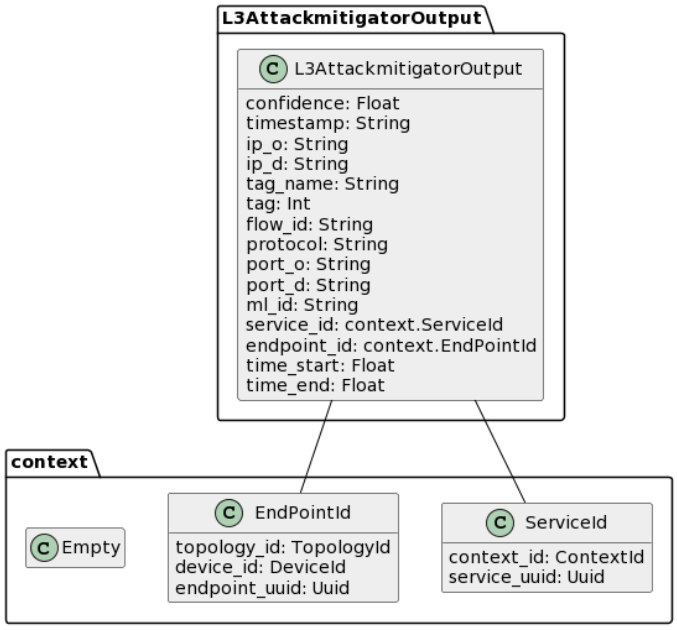


Figure 34 L3 Attack Mitigator Output

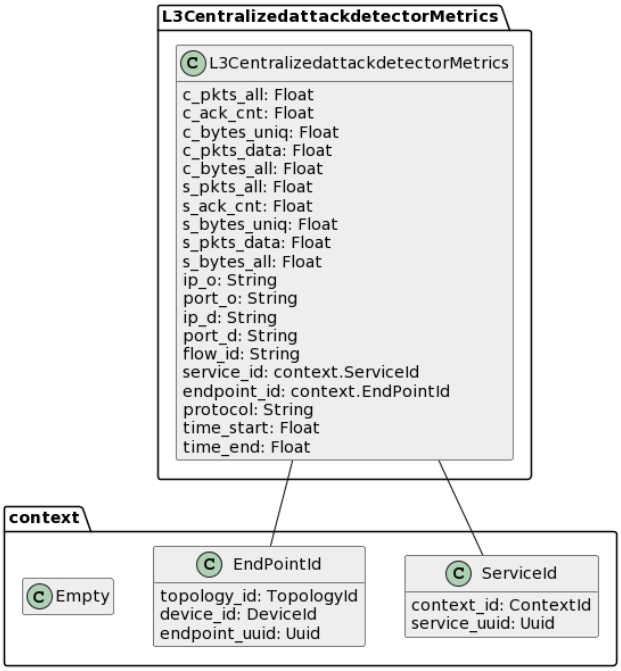


Figure 35 L3 Centralized Attack Detector Internal Data Model

6.2.12. Distributed Attack Detector

Name:	DAD
Objective:	The DAD monitors the network data plane for the presence of malicious network flows. For this purpose, this component integrates a feature extractor that is deployed at the network edge to collect and generate statistical summaries of network flows. To that end, packets are aggregated into flow-level statistics, where each flow is an aggregate of packets belonging to the same packet flow (same source IP address, source port, destination IP address, and destination port). This aggregation is performed each time new packets arrive. In this way, flow statistics at each instant in time are sent to the CAD to detect malicious traffic. For aggregation into flows, the DAD uses the tstat (TCP STatistic and Analysis Tool: http://tstat.polito.it/) tool, which allows recollection of transmitted packets and the production of statistical summaries. These summaries are encoded and stored in a compact text format that is sent to the CAD for the purpose of attack detection at the flow-level.
Requirements:	• Shall receive monitored network packets from the packet processor devices. • Shall generate summarised flow statistics and send them to the CAD.
References:	-
Responsible (and collaborators):	UPM, TID
Provided Operations:	This component does not deploy a gRPC server.
Internal Models:	L3CentralizedAttackDetectorModelMetrics is sent to the CAD. See for references Figure 35.

6.2.13. Attack Inference

Name:	Attack Inference
Objective:	Performs 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 OPM data. The design of this component is inspired by the design of the TensorFlow open source project's Serving Component.
Requirements:	• Shall receive a set of samples (composed of a set of feature values) representing a monitoring window (defined by the CAD) 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 36).

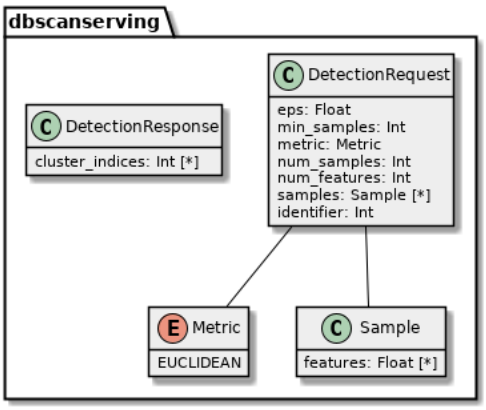


Figure 36 Attack Inference Data Model

6.2.14. Attack Mitigator

Name:	**AM**
Objective:	This component is responsible for computing viable attack remediation solutions, depending on the attack detected by other components. When dealing with Layer 3 attacks, it receives the attack detection information (see Figure 34) from the CAD. Then, it communicates with other service-related TFS components (i.e., Service, Context) to perform attack mitigation. When dealing with attacks at the optical layer, it receives information about the service on which the attack was detected, obtains relevant information from the Context Component, and coordinates with service-related TFS components (i.e., Service, Context) to perform attack mitigation.
Requirements:	• Shall receive attack detection notifications from the CAD. • Shall compute relevant attack mitigation strategies corresponding to the service where it was detected (i.e., Layer 3 or optical). • Shall communicate with other components integrated in the core layer of the TFS (i.e., Service, Context) to perform attack countermeasures.
References:	-
Responsible (and collaborators):	UPM, TID (L3), CHAL (Optical)
Provided Operations:	L3: rpc **SendOutput** (L3AttackmitigatorOutput) returns (context.Empty) {} rpc **GetMitigation** (context.Empty) returns (context.Empty) {} Optical: rpc **NotifyAttack** (AttackDescription) returns (AttackResponse) {}
Internal Models:	L3AttackMitigatorOutput (see for reference Figure 37 and AttackDescription and AttackResponse used by the AM's Optical sub-component (see for reference Figure 38).

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

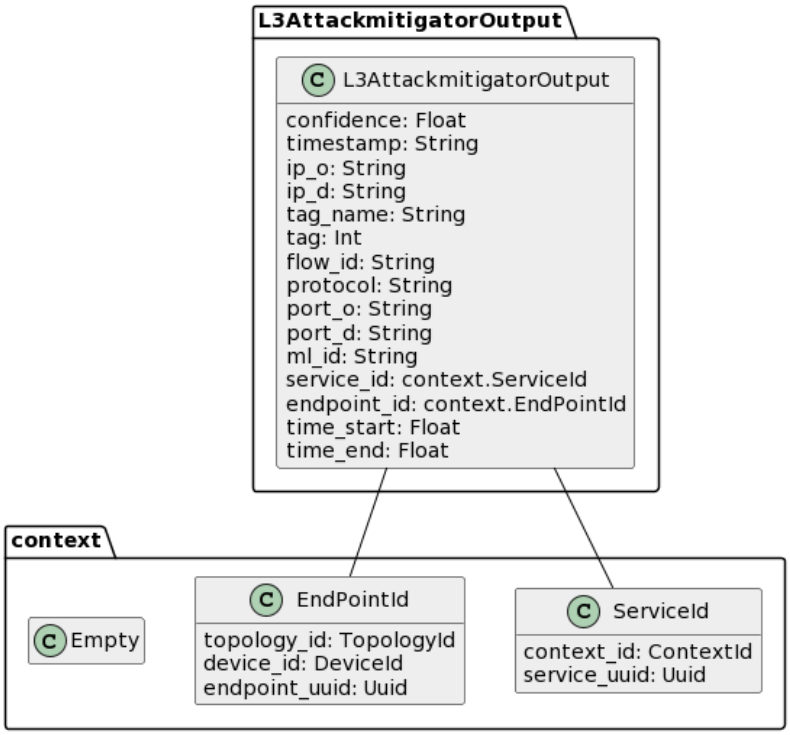


Figure 37 L3 Attack Mitigator Output Data Model

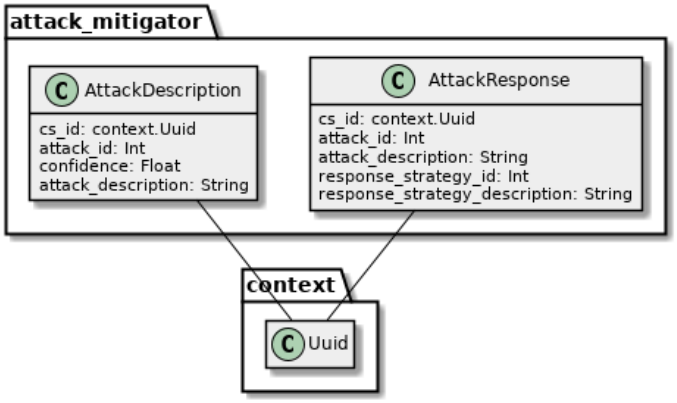


Figure 38 Optical Attack Mitigator Data Model

6.2.15. Distributed Ledger

Name:	DLT
Objective:	Provide a 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 39).

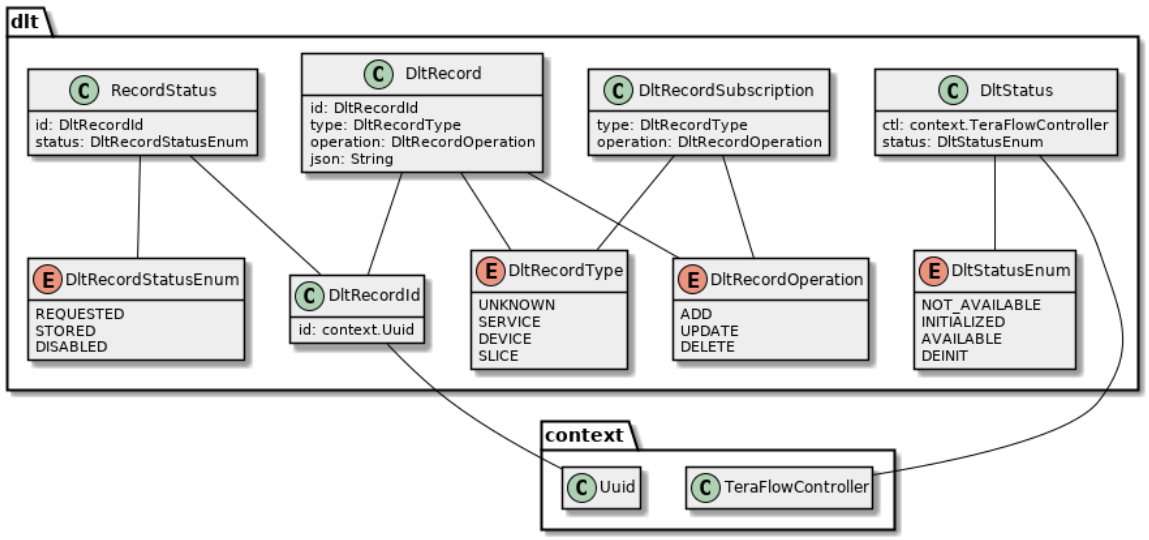


Figure 39 DLT Component Data Model

6.2.16. NBI (Previously Compute)

Name:	NBI
Objective:	It supports the 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" [23]. 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) [23]
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 the Context and Service Components.

6.2.17. Inter-Domain

Name:	Inter-domain
Objective:	Enable interaction of a TeraFlowSDN instance with peer TeraFlowSDN instances which manage different network domains to create E2E TN slicing services.
Requirements:	[REQ-INT-01] - [REQ-INT-09]
References:	RFC 8299 [27] RFC 8453 [28] IETF Network Slices [21]
Responsible (and collaborators):	TNOR, NTNU, CTTC

Provided Operations:	rpc Authenticate (context.TeraFlowController) returns (context.AuthenticationResult) {} // Deprecated
	rpc RequestSlice (context.Slice) returns (context.SliceId) {}
	rpc LookUpSlice (context.Slice) returns (context.SliceId) {}
	rpc OrderSliceFromCatalog (context.Slice) returns (context.Slice) {}
	rpc CreateSliceAndAddToCatalog(context.Slice) returns (context.Slice) {}
	rpc OrderSliceWithSLA (context.Slice) returns (context.SliceId) {} // If slice with SLA already exists, returns slice. If not, create it.
	rpc UpdateSlice (context.Slice) returns (context.Slice) {}

Internal Models:	Only provides operations. Data models belong to Context and Service Components.

6.2.18. Web User Interface (WebUI)

Name:	Web User Interface (WebUI)
Objective:	The WebUI enables a network operator to manually interact with the SDN controller to perform configuration operations and inspect the state of the network.
Requirements:	[REQ-UI-01] The WebUI needs to be implemented as another TFS micro-service. [REQ-UI-02] The WebUI needs different sections: Home, Device, Link, Service, Slice, Grafana and About. Each section shall provide information about internal TFS context state.
References:	
Responsible (and collaborators):	CTTC, Chalmers University
Provided Operations:	
Internal Models:	Only provides web-based HTTP interface. Data models belong to the NBI and Context Components.

6.2.19. Cloud Orchestrator Features

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

6.2.19.1. Auto-Scaling

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

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

6.2.19.2. Self-Healing

Self-Healing 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 [11].

6.2.19.3. 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. Prospective Topics and Future Work

7.1. Regional Extensions to IETF Slicing

Section 2.5 presents use cases for services and introduces service information models applicable to TFS. Future work may extend these concepts in a way that we describe as producing regional network slices. We recognise two levels of service concepts and their abstractions: i) the Managed Quality Path (MQP) level deals with semi-static, but still dynamic, infrastructure-oriented topologies; and ii) the Specialised Connectivity Service (SCS) level deals with highly dynamic sessions (end-user flows or tunnels), given the pre-established MQP level.

The complex, multi-dimensional problem of delivering connectivity services that meet SLAs for multiple QoS requirements is made more complicated when we seek to deliver those services for open public access at Internet scale. Providing an SCS for on-demand connectivity with support for multiple modes of traffic (beyond best effort) is a problem that cannot be addressed with today's tools and techniques, and the current operational approaches tend to predicate against providing these services. This topic is discussed further in [https://doi.org/10.36227/techrxiv.19690570.v1] which leads to proposals to develop network slicing based on Abstraction and Control of Traffic Engineered Networks (ACTN) [24] to provide regional network slices that can assist in delivery of MQP and SCS.

This section introduces the further work that could be performed, building on TeraFlow, to address these requirements. To set the scene, we first consider various service levels in different sectors or markets, as well as traffic modes in the context of delay performance, complemented by resilience levels. The figures below are examples and for illustration and do not provide a final answer on required traffic types and modes.

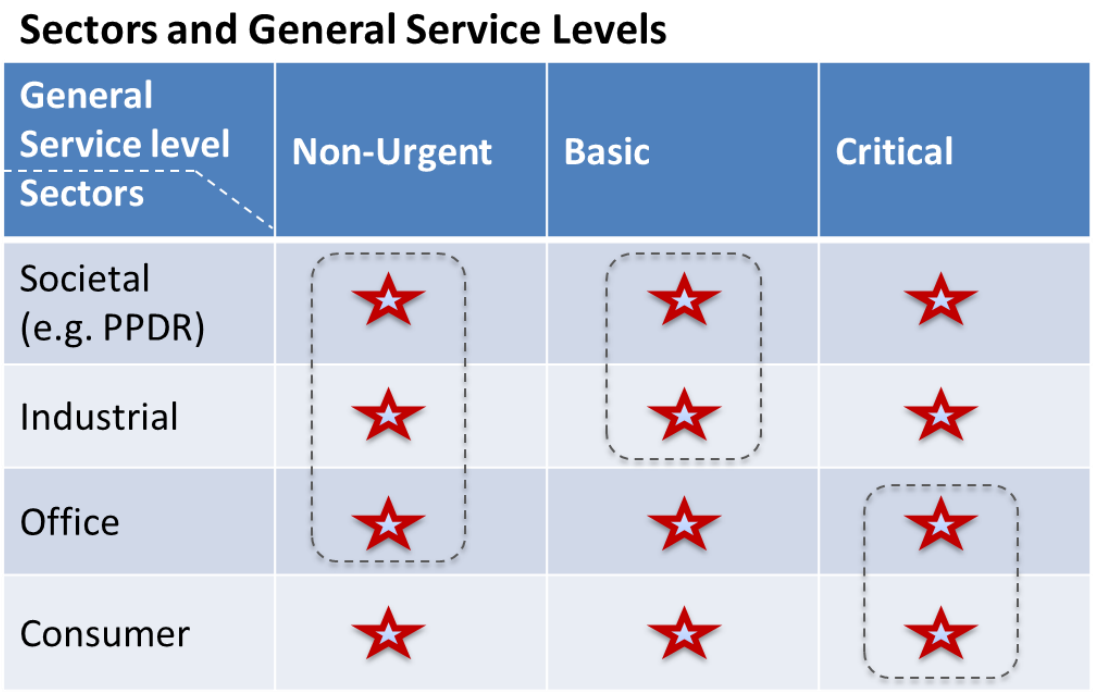


Figure 57 Exposing the Urgency of Different Service Levels According to User Type

Figure 57 shows how there is already a requirement for all or most service levels in all sectors (or markets). Even in the consumer market (considering consumer end-points) we foresee the need for a "consumer critical" service level, e.g., in relation to eHealth. Moreover, the illustration suggest that some traffic modes can be shared or be common across sectors or markets. For instance, the non-

urgent service level (e.g., traffic with elastic delay requirements in a context of non-critical / non-urgent setting) can be realised by a traffic mode common to the Office, Industry, and Societal sectors (markets). Indeed, the societal sector (considering, e.g., public protection and disaster relief – PPDR) can operate just fine through a shared "non-urgent" traffic mode with Industry and Office. Thus, the problem is already upon us: all network users have demands across all service levels and meeting those demands may require network operators to differentiate traffic in order to guarantee the different quality requirements. We anticipate that a limited number of traffic modes are sufficient in a metro or backbone transport network. However, a more fine-grained set of traffic modes may be needed when considering the (radio) access network.

Next, we consider latency performance and resilience.

Multi-Service Internetworking, Latency and Resilience

Latency level / Resilience level	Elastic	Basic	Bounded
Extra high		★	★
Higher	★	★	★
Basic	★	★	★

Figure 58 *MultiSservice Latency and Resilience in the Internet Backbone*

Figure 58 shows anticipated, inter-related requirements for latency and resilience in the context of transport network backbone and inter-AS traffic exchange. Only services that can accept elastic latency can tolerate the absence of extra-high resilience levels. Conversely, it is only with the availability of resilience that high guarantees of basic or bounded latency can be made.

In order to address these requirements in an operational manner, deployments may utilise slicing techniques. Using IETF network slices as an underlying approach and utilising ACTN as a control and management platform may offer an operational technique that is both scalable and realistic.

The subsections below identify the potential future research topics. They also present a possible roadmap for introducing this work through standards bodies (in particular, the IETF) and to the ETSI OSG TFS project. Where relevant, we also address how these topics relate to the 3GPP service concepts that can be linked to the IETF network slice, such as Access Point Name (4G) and Data Network Name (5G).

7.1.1. Interconnected Sliced Networks and Services

Initial work on network slices within the IETF has focused on delivering slices within a single domain. That has been sufficient for the consideration of simple 5G deployments, but more complex scenarios will require delivering end-to-end network slices across multiple networks and of different network technologies. This is a need both in the 3GPP architecture where the IETF technology network slice is used as a transport slice, and in the more general case where a network slice is delivered as a service to a customer.

Network slices may be interconnected and inter-related in several ways:

- A single operator's network may consist of several domains (ASes or areas) of the same technology that are separately sliced and that are combined to form a single end-to-end network slice service.
- An end-to-end network slice service may be delivered to a customer using resources belonging to multiple network operators. This situation is a little like the way that VPNs are achieved using cooperating networks.
- The "carrier's carrier" scenario sees one service provider deliver a network slice service to another service provider which, in turn, uses the resources of that network slice to deliver another network slice service to a customer.
- Just as different technology networks are used to build connectivity within other technology networks (for example, IP over optical), so network slices of one type of network may provide the connectivity necessary to build a network slice service in another technology network.

In fact, all four of these options may be combined arbitrarily and in complex arrangements so that network slices may be composed hierarchically and in series to construct the services delivered to customers.

This will form a rich area for research, but is also an important factor in how the TFS system can be developed and made more strategic.

7.1.2. Mechanisms for Filtering and Classifying Traffic onto Slices and Regions

An important aspect of how network slices can be useful is determining how traffic is forwarded onto each slice, considering even (destination) regions of a slice. The notion of an (IP packet destination) region is a concept to be explored and further defined. At a generic level it is a set or range of end-point addresses, e.g. a set of IP prefixes. Traffic filtering and classification forms a central element to many Internet systems from BGP (where Flowspecs are used to describe what traffic should be filtered onto which route) through PCE, Segment Routing, and MPLS-TE (where Flowspecs are used to describe the traffic that should be forwarded onto tunnels, paths, and LSPs). Even the IETF's service function chaining architecture includes a conceptual component called a Classifier that has the job of determining what traffic is sent onto which service chain.

The topic of traffic filtering and classification has been largely undiscussed in the development of network slices and network slice services, but it will be crucial to the value of the technology. TFS will need a way to identify this in far more detail than the simple "attachment circuit" approach used in the early specifications.

7.2. TFS as a Dedicated Technology SDN Controller

During the analysis of preliminary results, as well as with informal conversations with multiple potential ETSI TFS members and participants, the need for using TFS as an SDN controller for a specific technology has appeared. To this end, Scenario 1 (more details in D5.2) includes the decoupling of TFS as End-to-End SDN orchestrator and TFS as IP SDN controller. The modularity and flexibility of the TFS architecture allows someone to quickly implement technology-specific plugins and thus instantiate TFS as a dedicated technology SDN controller. Some of the proposed technologies follow:

- Optical SDN Controller. Shall be able to directly interact with optical equipment, such as ROADMs and OXCs, and offer ONF Transport API or IETF L0 Connectivity as a Service.
- MPLS SDN Controller, by including SBI support for DiffServ/MPLS enabled routers.
- L2 SDN controller to support OpenFlow and OVSDB.
- RAN controller.

7.3. TFS as a Toolbox for Intent-Based Networking

Several protocols have been suggested as an evolutionary path towards Intent-Based Networking (IBN). In this section we present ALTO and I2NSF as possible TFS extensions.

The Application-Layer Traffic Optimization (ALTO) protocol focuses on improving traffic management. Moreover it was designed for distributed (i.e., Peer-to-Peer (P2P)) applications scenarios such as file sharing, live media streaming, etc. The ALTO protocol aims to assist in the management of today's networks from the application-layer point of view, especially on tasks such as routing policies, link availability, network topologies, and so on. Application-layer elements may benefit from the information coming from lower layers, for example by adapting themselves and becoming more network-efficient and improving their performance.

The architecture of how the different elements in a network region interact with this protocol is illustrated in Figure 59. The different elements and their relationships are shown. Applications receive network resource information via the ALTO protocol, making it feasible that users may check how their requested service is performing and so, react by requesting an improvement from the service provider.

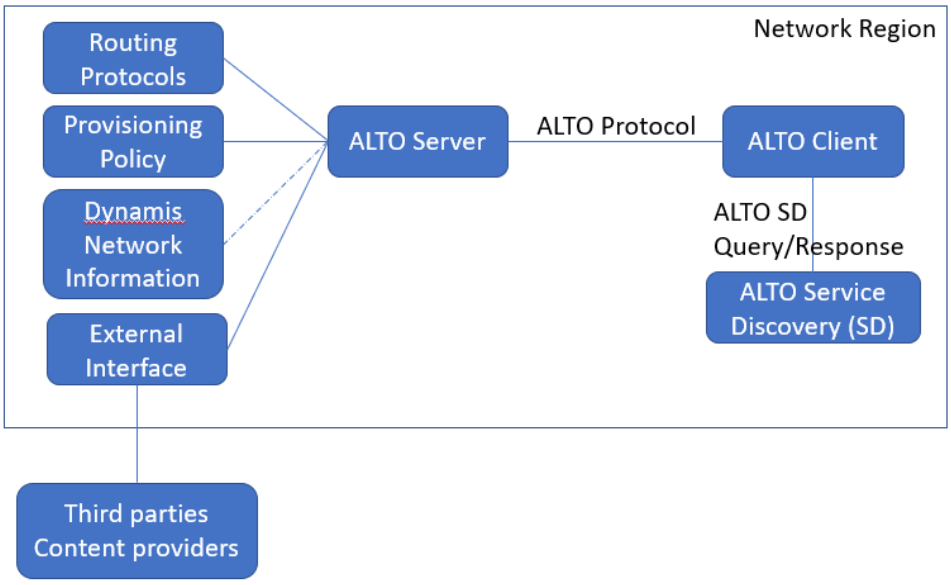


Figure 59 ALTO Architecture

The Interface to Network Security Function (I2NSF) is a framework that aims to describe and propose a group of interfaces and data models to control and monitor physical and virtual Network Security Functions (NSFs), allowing clients to define rules. Due to the wide range of security vendors and open-source technologies, I2NSF works on flow-based NSFs that manage and control packets/flows like flow filtering, deep packet inspection, etc.

The IETF's I2NSF working group has defined a reference model (Figure 40) with the following set of interfaces for the interaction between the users, network operators, developers, and the deployed NSFs:

- I2NSF Consumer-Facing Interface: This allows the users to define, manage, and monitor security policies for specific flows inside an administrative domain. The users do not need to know about the location and implementation of the NSFs, thus the request conveys only the

intent. Examples include blocking flows that match a set of specifications, and enforcing specific I2NSF policies for a particular flow.

- I2NSF Registration Interface: Due to the existence of multiple vendors, it is necessary for them to register the capabilities of their NSFs so that they can be used properly by the system.
- I2NSF-Facing Interface: Its main objective is to define and constantly check the enforced flow-based security policies by one or more NFSs.

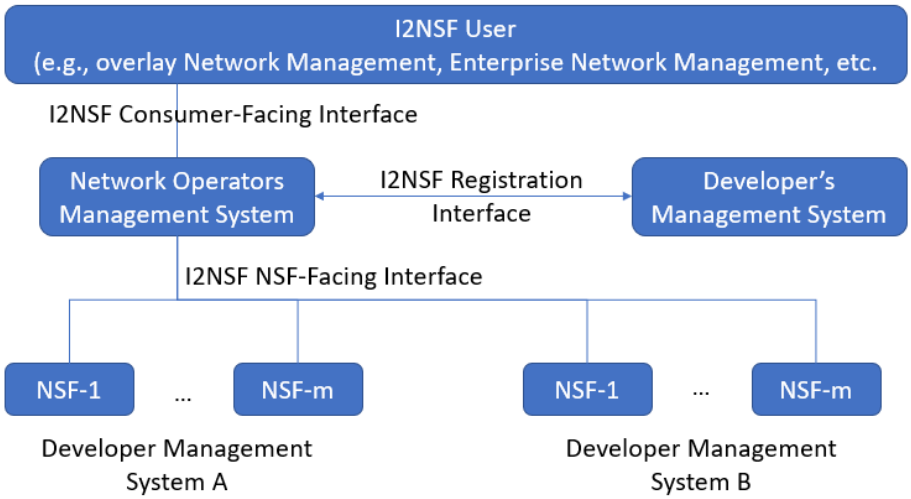


Figure 40 I2NSF Architecture

7.4. TFS as a QKD SDN Controller and SDN Orchestrator

One of the many technologies that can also be included is the capability to control and manage a dedicated Quantum Key Distribution (QKD) network, using an SDN agent in each QKD node, and the capability to control and manage end-to-end connectivity through a multi-layer network with QKD and Optical Transport Network (OTN) SDN controllers, through SDN orchestration.

7.4.1. QKD SDN Controller

Figure 41 depicts the high-level architecture of a software-defined QKD (SD-QKD) network as a set of connected nodes under the control of the SDN controller. One of the nodes is shown in more detail with the fundamental components which are required to build an SD-QKD node in order to illustrate the typical flow of information between components. The interface between the SD-QKD nodes and TFS Controller is described in [15].

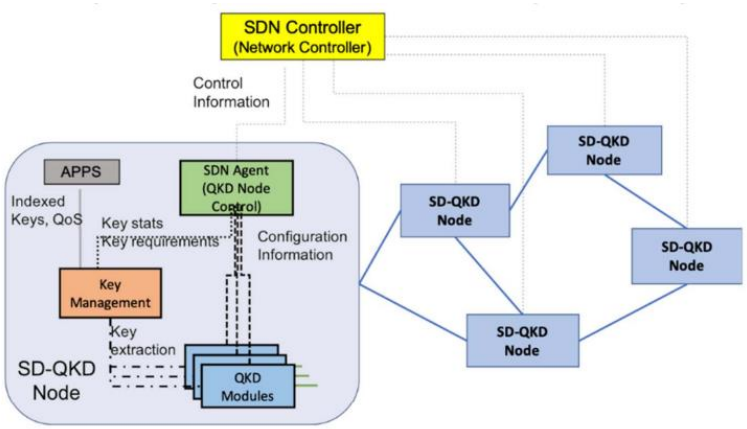


Figure 41 Depiction of an SD-QKD Network Showing a Set of SD-QKD Nodes

7.4.2. QKD SDN Orchestrator

An SDN orchestrator can provide end-to-end connectivity through multiple QKD network domains from multiple vendors via the SDN controllers of each QKD network. It can also provide E2E connectivity to QKD and classical network domains, as shown in Figure 42.

In order to deliver QKD-derived keys to secure application entities through an OTN, E2E connectivity can be orchestrated. To this end, network operators could choose to deploy a QKD network that is separate from a classical OTN and to operate and manage each network separately. Using TFS as an SDN orchestrator, both network domains would be aware of nodes that belong to each other for QKD-derived keys to be delivered to secure application entities in the OTN. To this end, TFS would need to be extended with an SBI plugin that implements the orchestration interface described in [16].

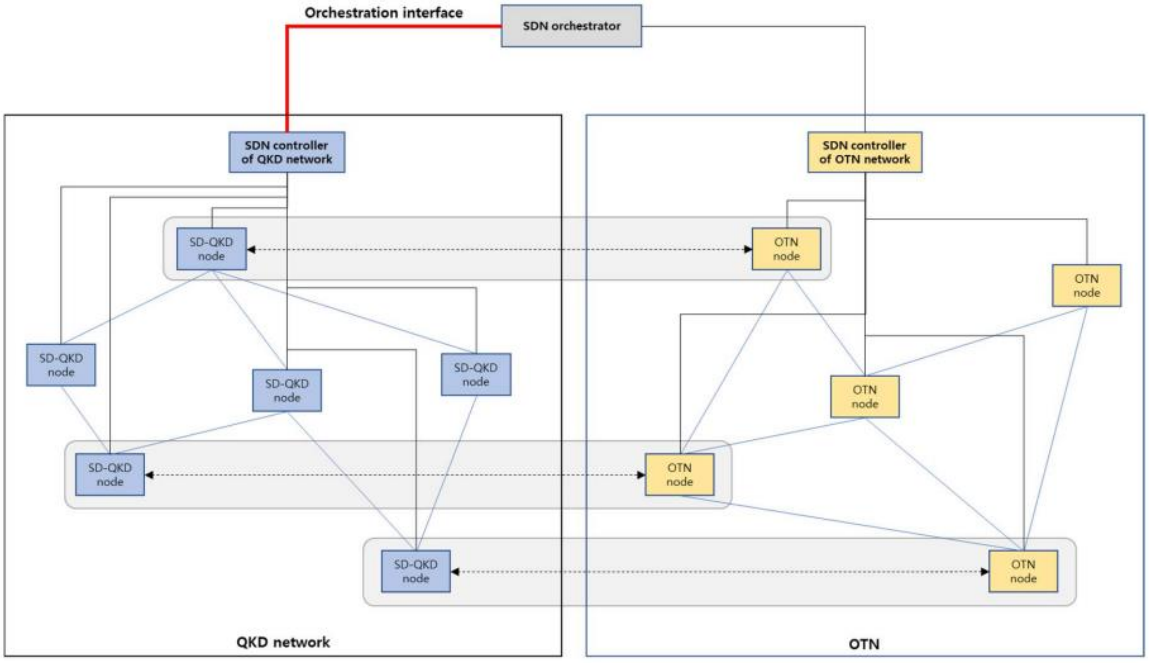


Figure 42 Use Case of SDN Orchestrator for QKD Network and OTN

8. Conclusion and Next Steps

This milestone includes all the necessary design descriptions to start providing the initial set of ETSI TeraFlowSDN features for release 2.0. TFS follows an agile methodology to produce three main software releases: v1.0 was released at M12, v2.0 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).

To support the development of software and the ability to make releases with minimal impact on the running services, TFS has adopted a CI/CD strategy, which is described in D5.1. The major software releases will deliver the main functionalities, while the minor releases will provide bug fixes and possibly additional features required by the currently operating experiments based on the three scenarios in D5.1. MS3.3 and MS4.2 will document the code freeze of the new programmed features, before integration tests.

9. References

[1] D. Abecassis, M. Kende, S. Osman, I. Kriegler, C. Gabriel, R. Kompany. May 2021. *"The economic impact of open and disaggregated technologies and the role of TIP"*. Analysys Mason report, for the Telecom Infra Project (TIP). https://cdn.brandfolder.io/D8DI15S7/at/cjwpw3jp9qvnwmsz9w297jrx/Analysys_Mason_-_The_economic_impact_of_open_and_disaggregated_technologies_and_the_role_of_TIP_full_report_-_2021-05-19.pdf. Accessed: 2022-12-23.

[2] 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.

[3] S. Barguil, et al., "A YANG Network Data Model for Layer 3 VPNs", IETF RFC 9182, February 2022. https://www.rfc-editor.org/rfc/rfc9182.html. Accessed: 2022-12-23.

[4] M. Boucadair, et al., "A YANG Network Data Model for Layer 2 VPNs", IETF RFC 9291, September 2022. https://www.rfc-editor.org/rfc/rfc9291.html. Accessed: 2022-12-23.

[5] Blind, K., Böhm, M., "The Relationship Between Open Source Software and Standard Setting", Thumm, N. (Ed.), EUR 29867 EN, Publications Office of the European Union, Luxembourg, 2019, ISBN 978-92-76-11593-9, doi:10.2760/163594, JRC117836

[6] TeraFlow Project Deliverable D2.1 "Preliminary requirements, architecture design, techno-economic studies and data models"

[7] TeraFlow Project Deliverable D6.3 "Exploitation Plan and Roadmap"

[8] "Building an open RAN ecosystem for Europe for Europe to lead in this essential innovation". Deutshe Telekom, Orange, Telecom Italia (TIM), Telefónica, Vodafone. November 2021. Self-published paper. https://cdn.brandfolder.io/D8DI15S7/at/jqrzjsgs267m6vxq55qp7/Building-open-ran-ecosystem-europe.pdf. Accessed: 2022-12-23

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

[10] Wu, Q., et al., "A YANG Data model for ECA Policy Management", IETF Internet-Draft , February 2021, work in progress. https://datatracker.ietf.org/doc/draft-ietf-netmod-eca-policy/. Accessed: 2022-12-23.

[11] "Health checking your gRPC servers on GKE", https://cloud.google.com/blog/topics/developers-practitioners/health-checking-your-grpc-servers-gke, Accessed: 2022-12-23

[12] Liu, X., et al., "IETF Network Slice YANG Data Model", IETF Internet-Draft, March 2022, work in progress. https://datatracker.ietf.org/doc/draft-liu-teas-transport-network-slice-yang/. Accessed: 2022-12-23

[13] "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: 2022-12-23.

[14] Osterwalder, A., & Pigneur, Y., "Business Model Generation", 2009. Self-published.

[15] ETSI, Quantum Key Distribution (QKD); Control Interface for Software Defined Networks, ETSI GS QKD 015 V2.1.1, April 2022.

[16] ETSI, Quantum Key Distribution (QKD); Orchestration Interface for Software Defined Networks; ETSI GS QKD 018 V1.1.1, April 2022.

[17] "Fast SQL for time series". https://questdb.io/. Accessed:2022-12-23.

[18] "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&modificationDate=1639671964711&api=v2, Accessed: 2022-12-23.

[19] Wu, Q., et al., "A Framework for Automating Service and Network Management with YANG", IETF RFC 8969, January 2021. https://www.rfc-editor.org/rfc/rfc8969.html. Accessed: 2022-12-23.

[20] TeraFlow Project Deliverable D6.2 "Exploitation Plan and Roadmap"

[21] Farrel, A., et al., "A Framework for IETF Network Slices", IETF Internet-Draft, December 2022, work in progress. https://datatracker.ietf.org/doc/draft-ietf-teas-ietf-network-slices. Accessed: 2022-12-23

[22] Wen, B., et al., "A YANG Data Model for Layer 2 Virtual Private Network (L2VPN) Service Delivery", IETF RFC 8466, October 2018. https://www.rfc-editor.org/rfc/rfc8466.html. Accessed: 2022-12-23.

[23] D. King, J. Drake, H. Zheng, and A. Farrel, "Applicability of Abstraction and Control of Traffic Engineered Networks (ACTN) to Network Slicing", IETF Internet-Draft, September 2022, work in progress. https://datatracker.ietf.org/doc/draft-ietf-teas-applicability-actn-slicing. Accessed 2022-12-23.

[24] Liu, X., et al., "YANG Data Model for Traffic Engineering (TE) Topologies", IETF RFS 8795, August 2020. https://datatracker.ietf.org/doc/rfc8795/. Accessed: 2022-12-23.

[25] Vasseur, J.P., and Le Roux, JL., "Path Computation Element (PCE) Communication Protocol (PCEP)", IETF RFC 5440, March 2009, https://datatracker.ietf.org/doc/rfc5440/. Accessed: 2022-12-23.

[26] Wu, Q., et al., "YANG Data Model for L3VPN Service Delivery", IETF RFC 8299, January 2018, https://datatracker.ietf.org/doc/rfc8299/. Accessed: 2022-12-23.

[27] Ceccarelli, D., and Lee, Y., "Framework for Abstraction and Control of TE Networks (ACTN)", IETF RFC 8453, August 2018, https://datatracker.ietf.org/doc/rfc8453/. Accessed: 2022-12-23.

[28] Wu, B., et al., "IETF Network Slice Service YANG Model", IETF Internet-Draft, October 2022, work in progress. https://datatracker.ietf.org/doc/draft-ietf-teas-ietf-network-slice-nbi-yang. Accessed: 2022-12-23.

[29] Clemm, A., et al., "A YANG Data Model for Network Topologies", IETF RFC 8345, March 2018, https://datatracker.ietf.org/doc/rfc8345/. Accessed: 2022-12-23.

[30] Rekhter, Y., and Kompella, K., "Virtual Private LAN Service (VPLS) Using BGP for Auto-Discovery and Signaling", IETF RFC 4761, January 2007, https://datatracker.ietf.org/doc/rfc4761/. Accessed: 2022-12-23.

[31] Lasserre, M., and Kompella, V., "Virtual Private LAN Service (VPLS) Using Label Distribution Protocol (LDP) Signaling", IETF RFC 4762, January 2007, https://datatracker.ietf.org/doc/rfc4762/. Accessed: 2022-12-23.

[32] Rosen E., and Andersson, L., "Framework for Layer 2 Virtual Private Networks (L2VPNs)", IETF RFC 4664, September 2006, https://datatracker.ietf.org/doc/rfc4664/. Accessed: 2022-12-23.

[33] Boutros, S., et al., "Virtual Private Wire Service Support in Ethernet VPN", IETF RFC 8214, August 2017, https://datatracker.ietf.org/doc/rfc8214/. Accessed: 2022-12-23.

[34] Sajassi, A., et al., "BGP MPLS-Based Ethernet VPN", IETF RFC 7432, February 2015, https://datatracker.ietf.org/doc/rfc7432/. Accessed: 2022-12-23.

Annex I: Workflows

This section depicts the sequence diagrams between internal and external TFS components and related elements.

L3VPN Establishment with SLA

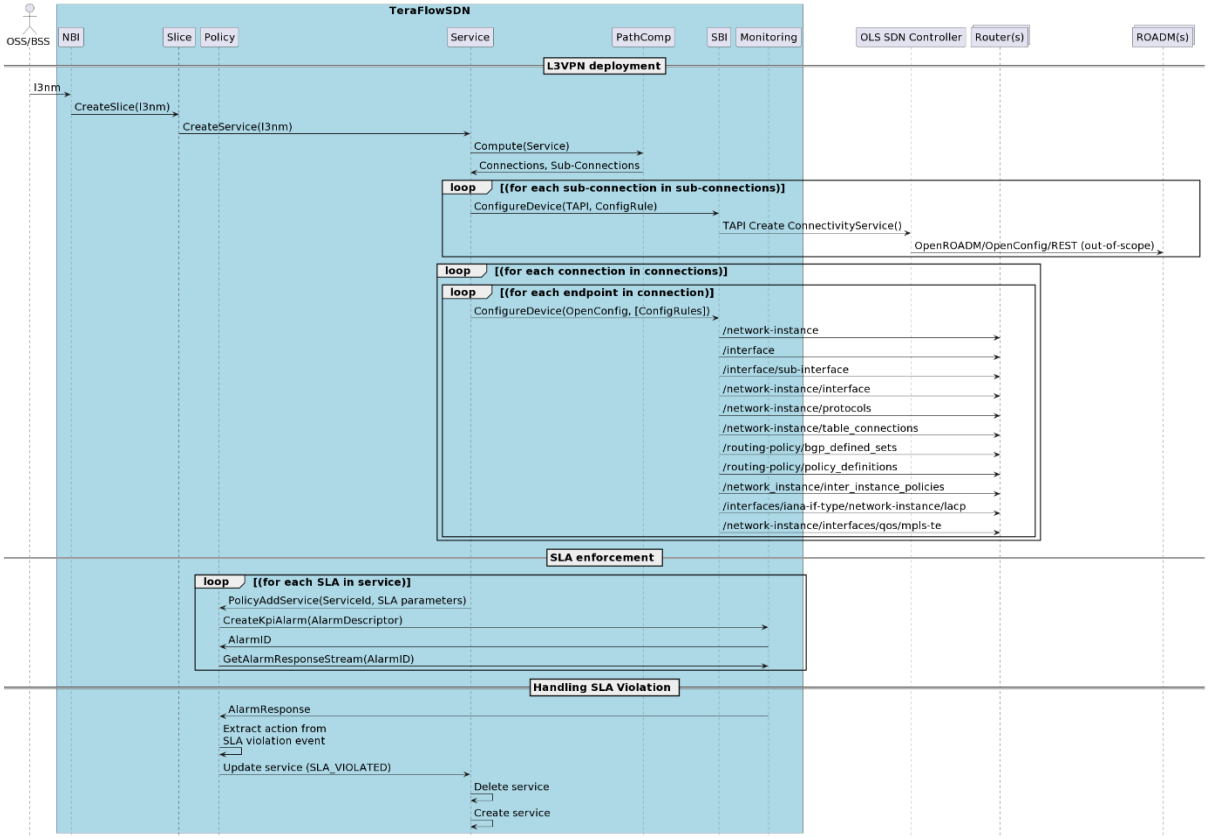


Figure 43 L3VPN Provisioning with SLA

L3VPN Establishment

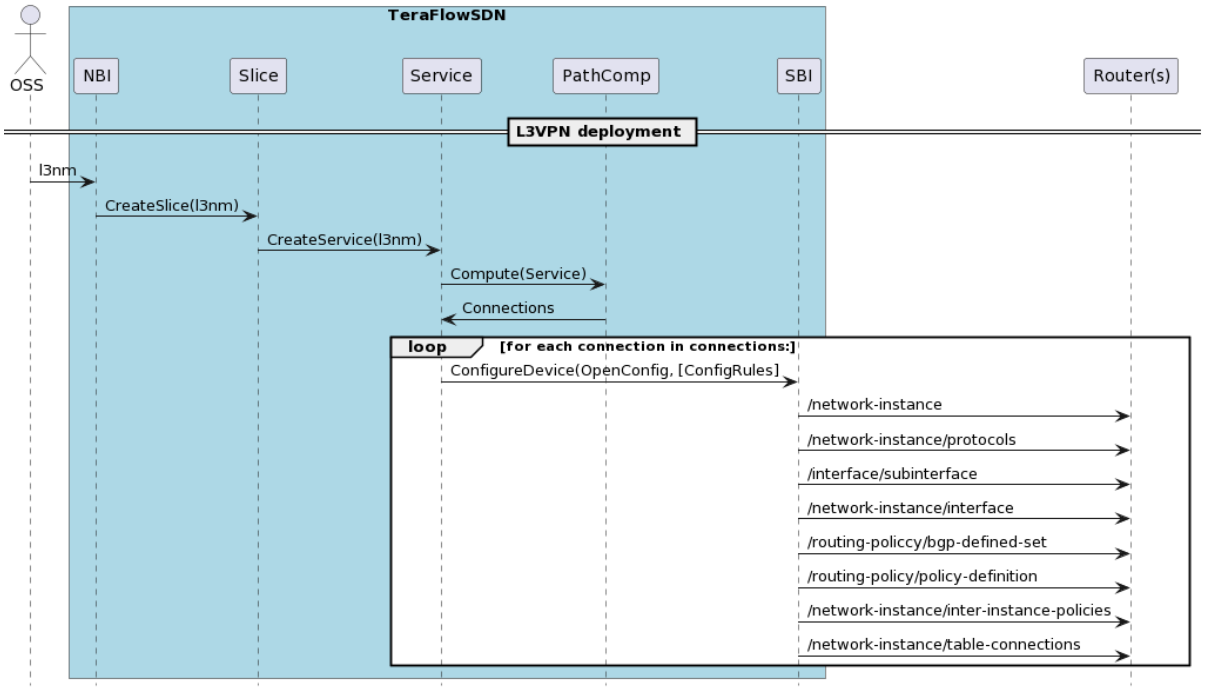


Figure 44 L3VPN Provisioning

L2VPN Establishment

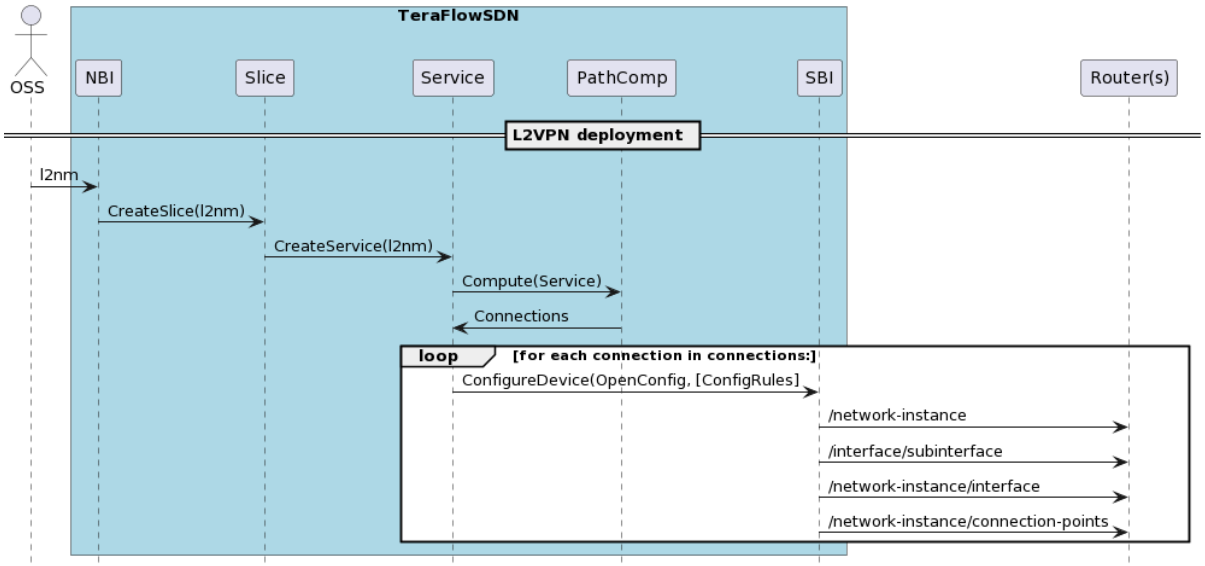


Figure 45 L2VPN Provisioning

Inventory

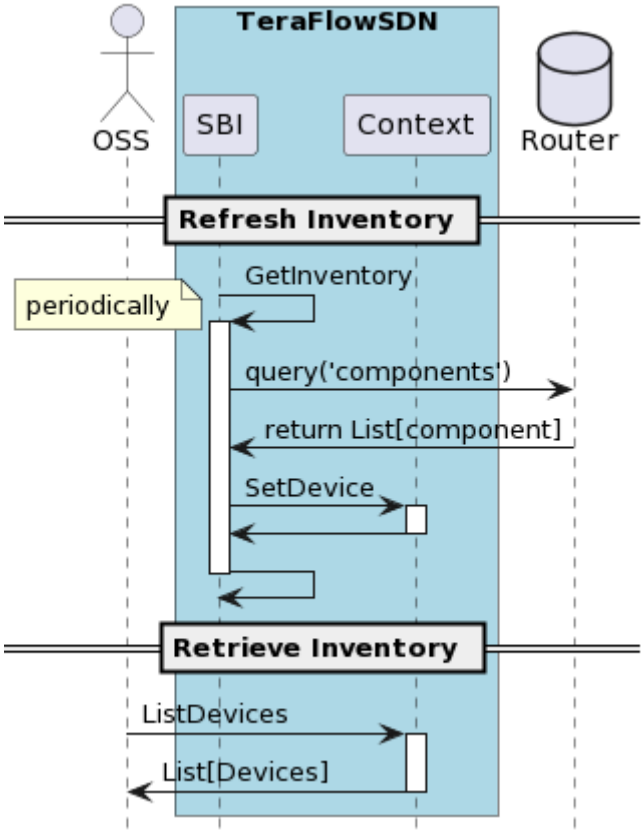


Figure 46 Inventory

Multi-Layer Topology Discovery

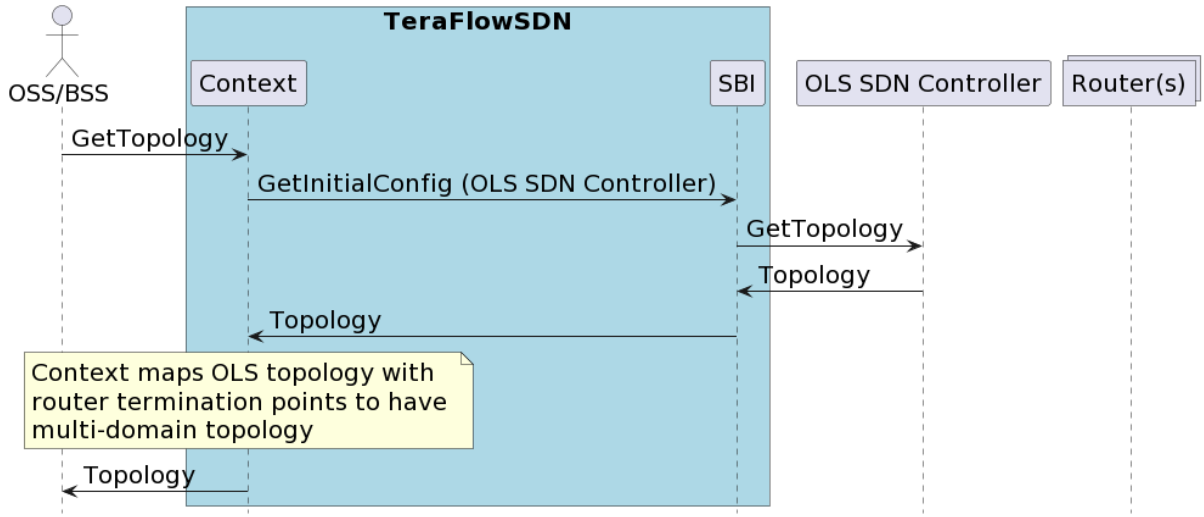


Figure 47 Multi-Layer Topology Discovery

Service ACL

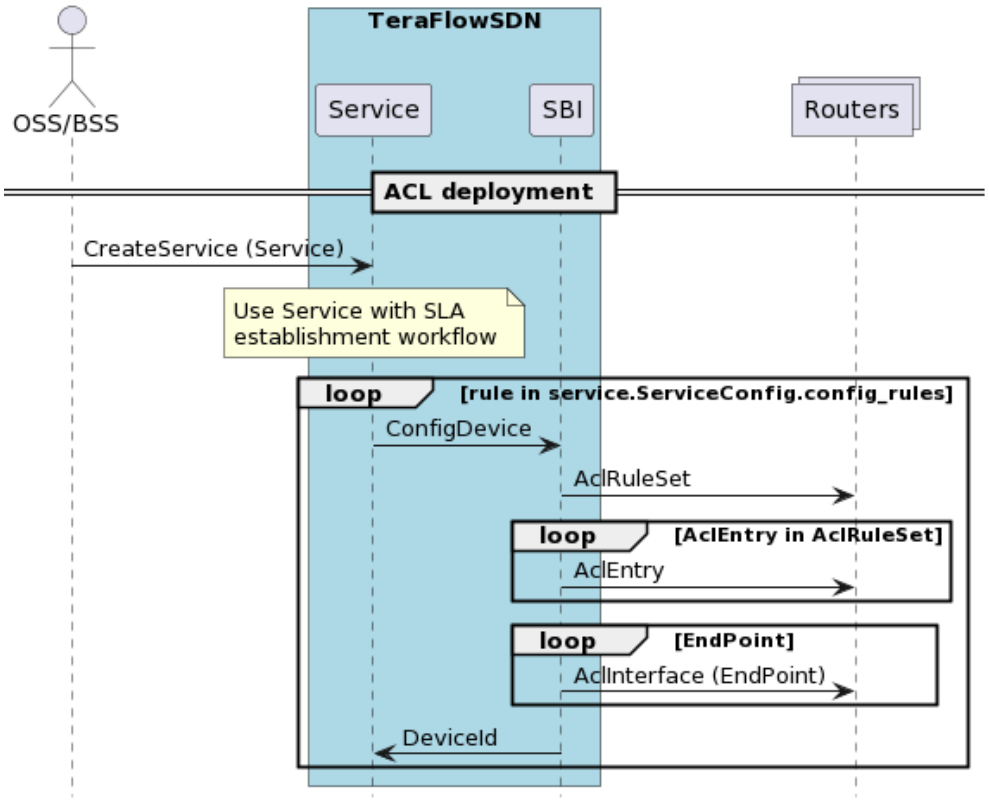


Figure 48 ACL Service Deployment

Service Location-Awareness

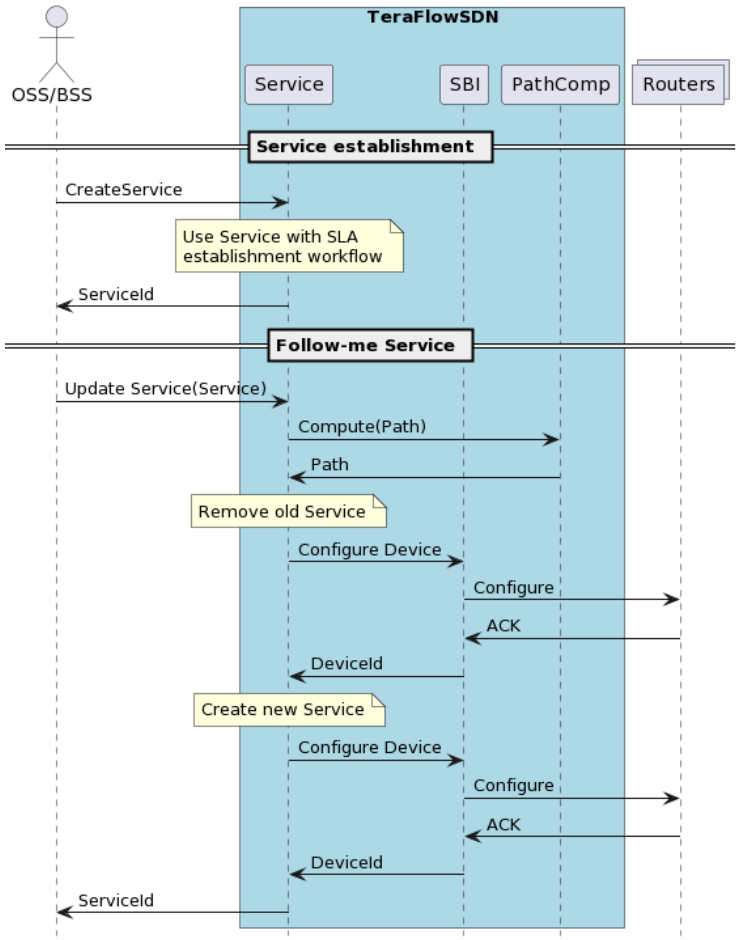


Figure 49 Service Location-Awareness Sequence Diagram

Traffic Engineering

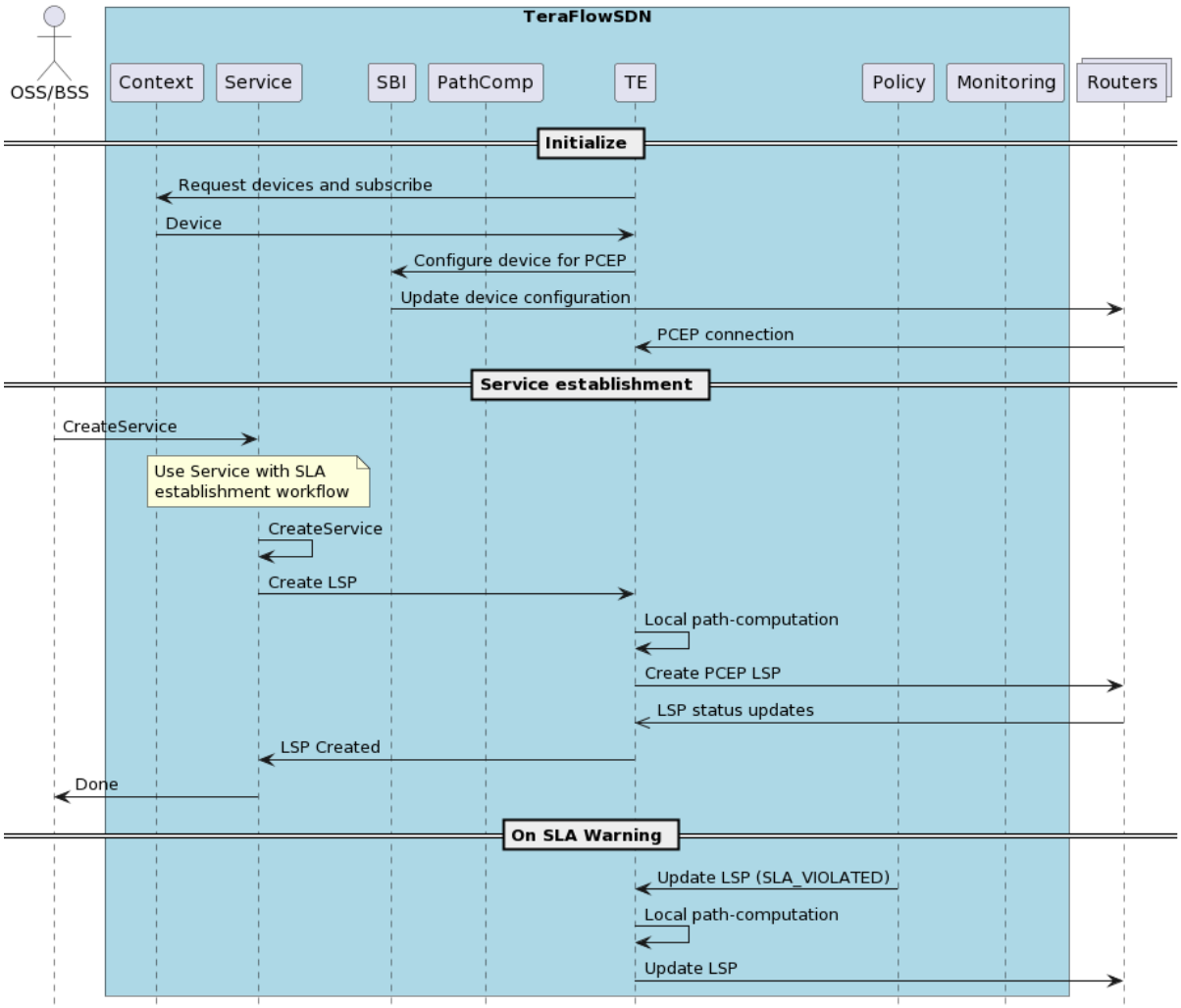


Figure 50 Traffic Engineering Sequence Diagram

Slice SLA Enforcement

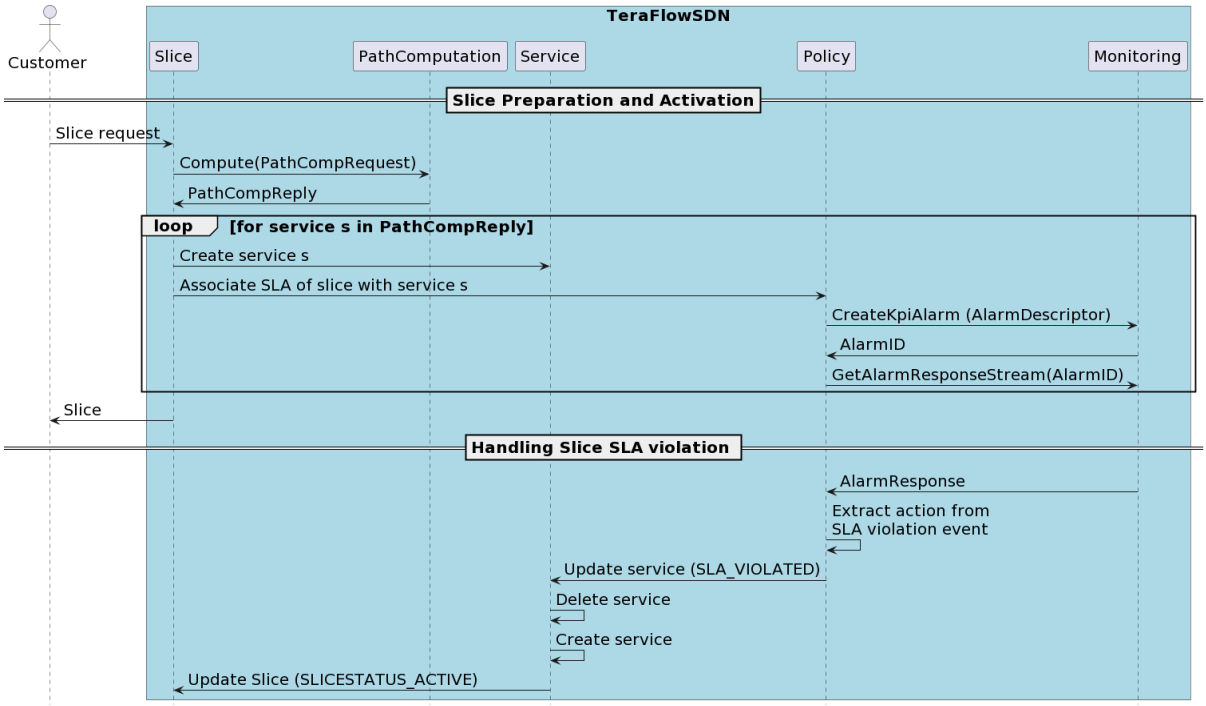


Figure 51 Slice SLA Enforcement Sequence Diagram

Slice Grouping

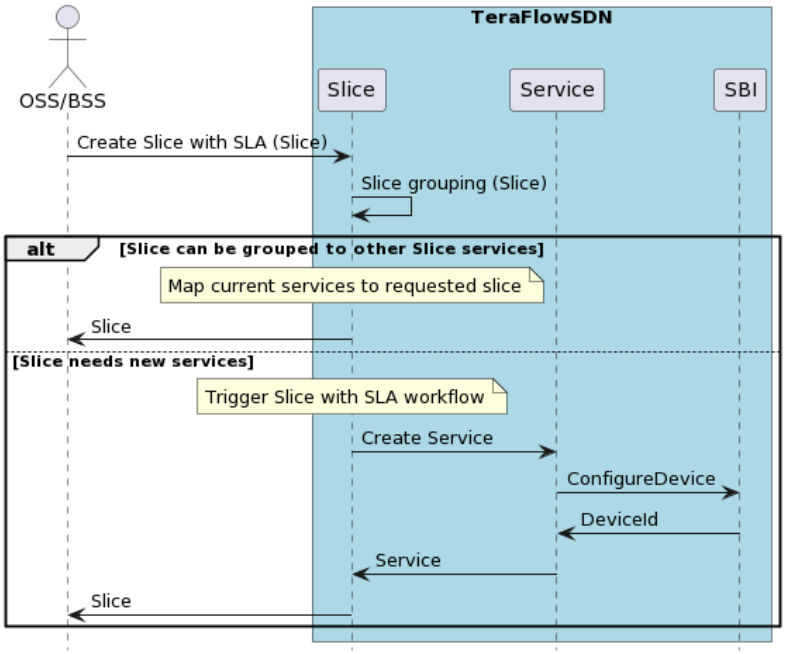


Figure 52 Slice Grouping Sequence Diagram

Forecaster

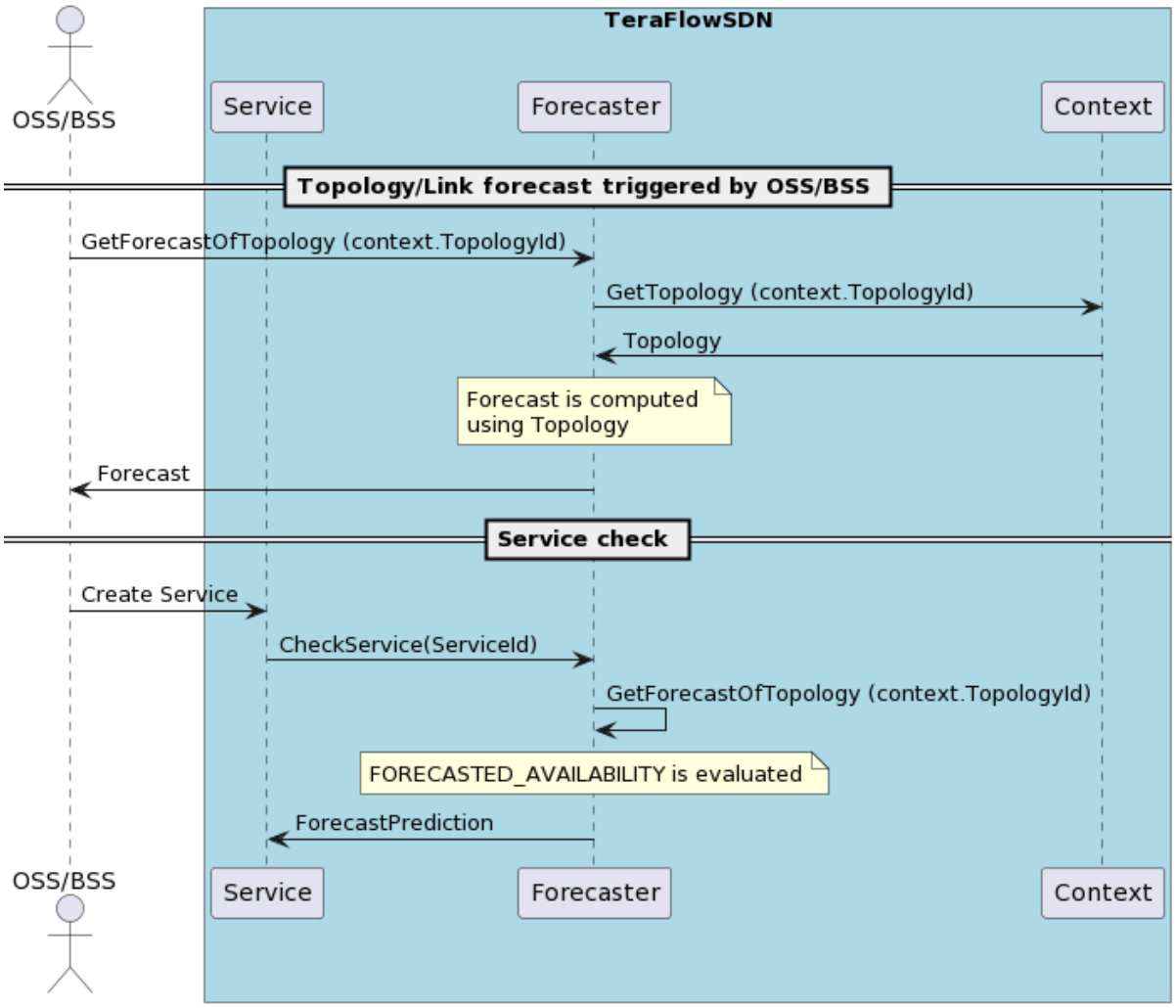


Figure 53 Forecaster Sequence Diagram

Inter-Domain Slice SLA Enforcement

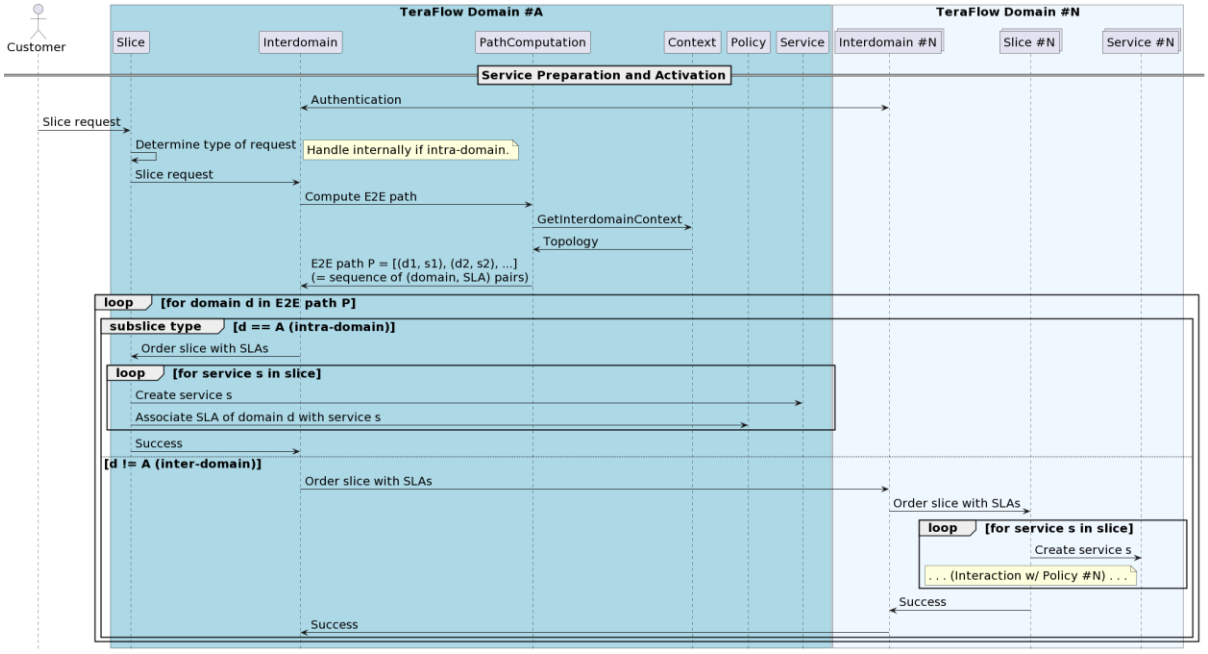


Figure 54 Inter-Domain Slice SLA Enforcement Sequence Diagram

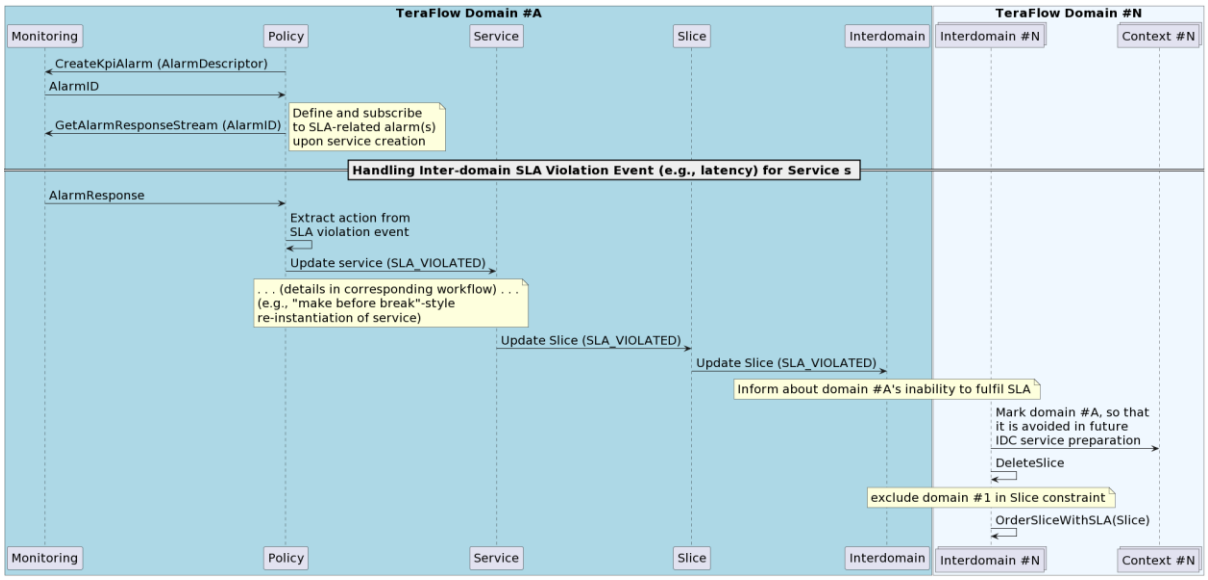


Figure 55 Inter-Domain SLA Violation Reaction Sequence Diagram

DLT Record Exchange Between DLT Connector and DLT Gateway

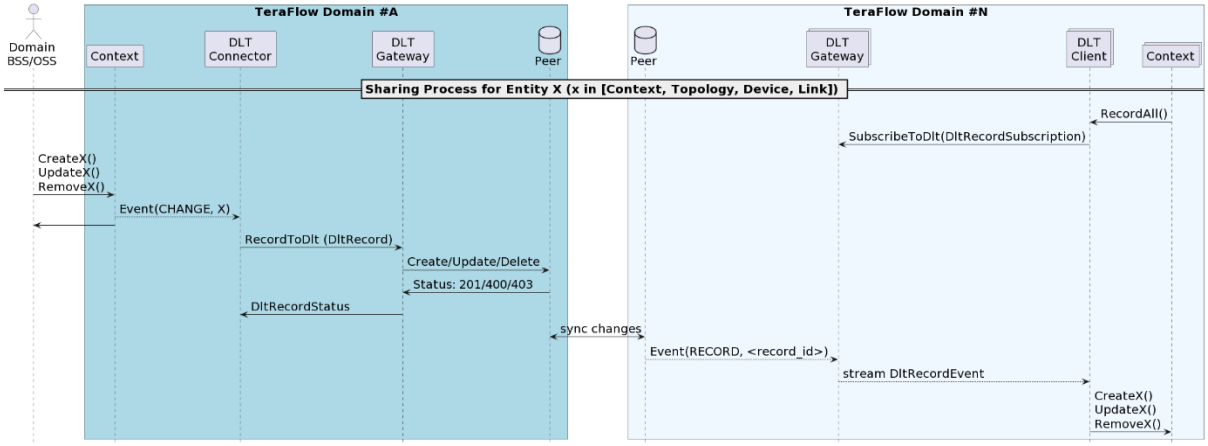


Figure 56 DLT Record Exchange Between DLT Connector and DLT Gateway Sequence Diagram

Inter-Domain DLT

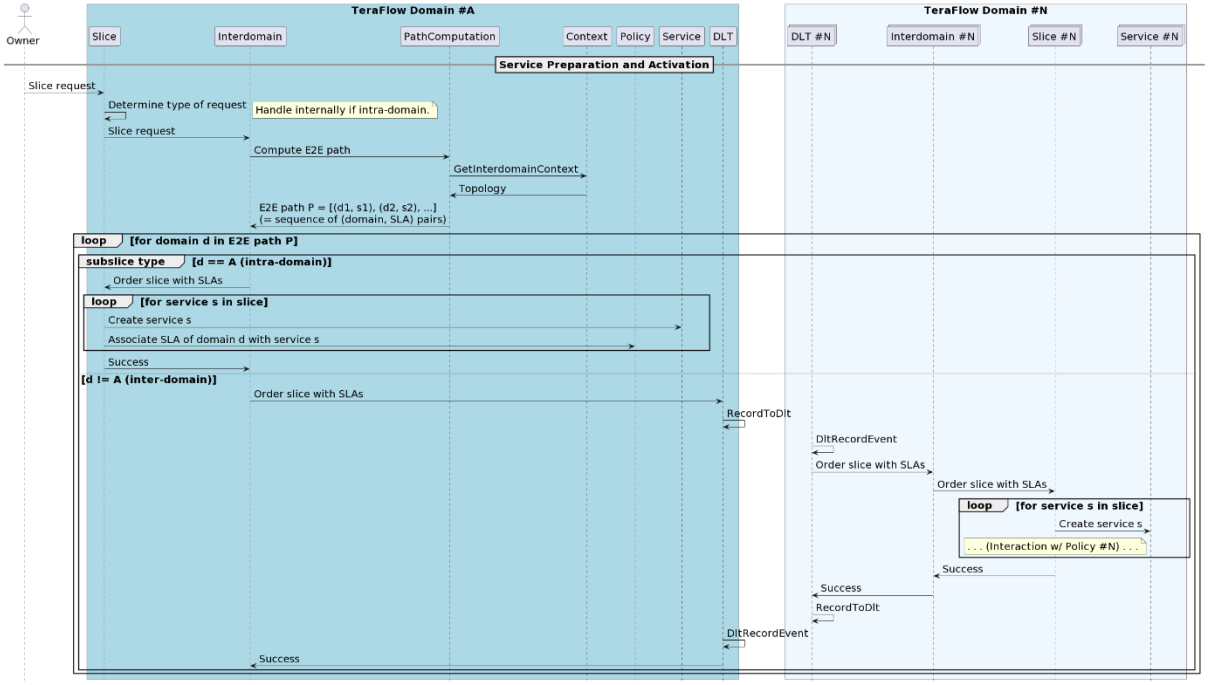


Figure 57 Inter-Domain Service Preparation and Activation with DLT Sequence Diagram

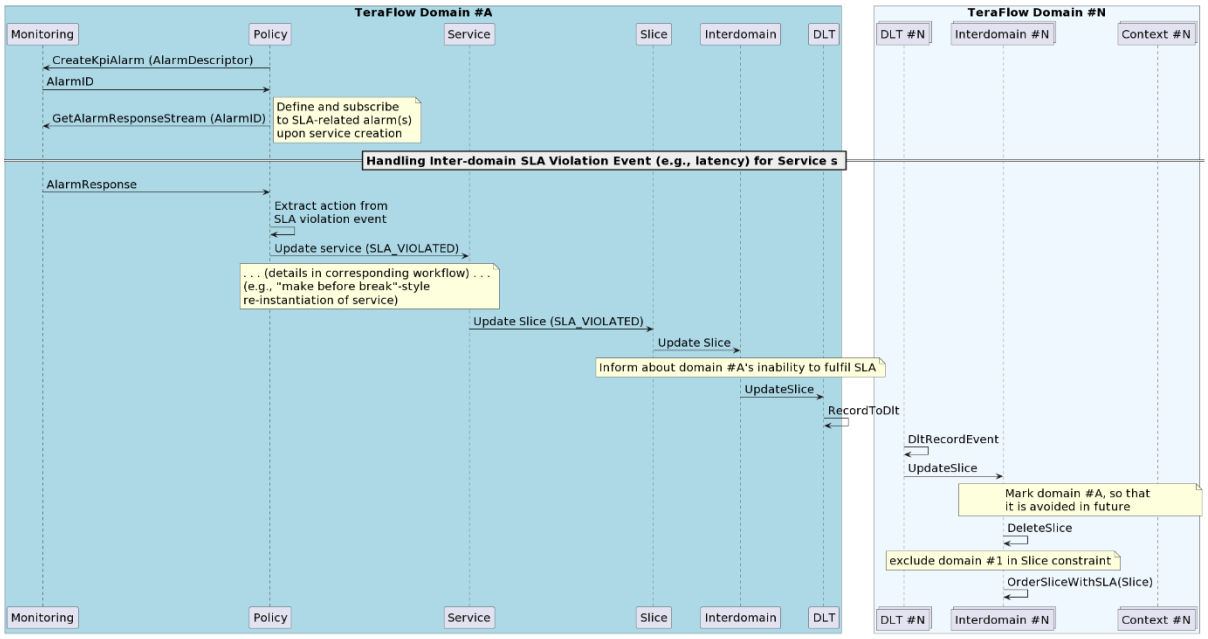


Figure 58 Inter-Domain SLA Violation Reaction with DLT

Energy-Aware Network Service Placement

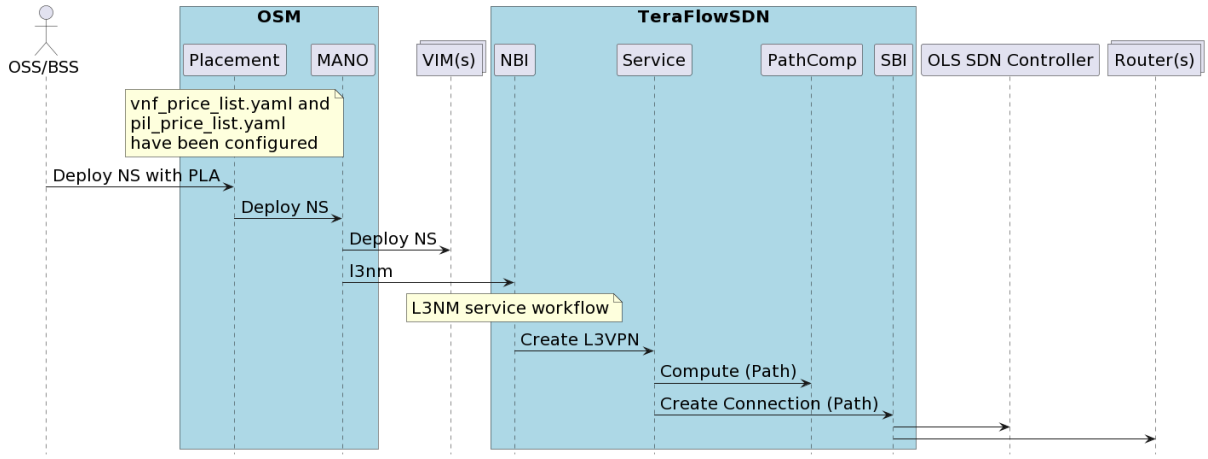


Figure 59 Energy-Aware Network Service Placement Sequence Diagram