Deliverable

D3.1 SDN Preliminary Design Report

DISCRETION

Code:	DISCRETION-DME-TEC-TEC01-E
Issue:	2.0
Approval Date:	22/03/2023
Confidentiality Level:	EU Unclassified

Prepared by: DISCRETION Consortium

Reviewed by: Telefonica Team

Approved by: Catarina Bastos

Approval Signature:

This page intentionally left blank

Document Status Log

Issue	Section	Change Description	Date
1.0	All	First Version	20/10/2022
2.0	All 3.1 4.1 9 12	Minor phrasing and typo's correction Section 3.1 SDN in Military Scenarios moved to section 3.2 Section 4.1: Federated SDN Architecture reviewed and expanded Minor editions of section 9: SDN Based Software defined radio Review of section 12: Security Application	28/02/2023

Table of Contents

List of Tables

List of Figures

1. INTRODUCTION

1.1. Purpose

This v2.0 document presents the final results of architectural work on DISCRETION, as a continuation of the first v.1.0 document. Although some parts might suffer modifications based on the results and findings and of the different work packages, the general frameworks can be considered almost final and will suffer little changes.

The objective of the SDB Preliminary Design Report document is to define the preliminary architecture of DISCRETION, that aims at integrating and combining the technologies of software defined network (SDN) and quantum key distribution.

This, combined with legacy optical networks, targets to build a highly secure, scalable and resilient network control architecture to provide secures communications for strategic and tactical scenarios in defence applications and create a degree of autonomy for European defence forces, based on the mutual beneficial relationship between QKD and SDN.

On the one hand, QKD provides a continuous flow of keys whose security cannot be compromised computationally breaking an algorithm. QKD allows two remote parties, in this case linked by a quantum channel, to have a mechanism to share secret keys, and these keys cannot be observed or tampered with by an adversary without alerting the original parties. This provides a physical security plane to the SDN network that can benefit from the QKD keys to protect its control as well as data flows.

On the other, the SDN network allows with its flexibility the integration of QKD technologies in a network. SDN relies on simple key ideas: separation of control and data plane, centralization of network policies and programmability of the networks. SDN networks are more agile, flexible, and easier to manage, meaning that new services (e.g., QKD) and network elements devices may be deployed much faster. They are also easier to re-configure and to support interoperation among diverse networks.

However, SDN implementation in the context of defense also presents challenges, for instance when considering tactical networking and in particular when sharing information, not only due to its dynamical environment but also how to secure information having a centralized control of the network (easier to manage but also easy to attack).

Thus, it is important to develop an **SDN architecture that guarantees redundancy and resilience against network element failures and network link loss**, including cyberattacks perpetrated against the network, such as a quick capability to adapt to network topology updates due to insertion of new network elements.

An important part of the architecture is how to generate, store and distribute cryptographic keys to the Cypher Machines (CMs) developed by the project. CMs will be the components responsible for assuring data protection in DISCRETION. They shall enable real-time data encryption and decryption, using key material pre-shared keys provided by quantum key distribution when available, or fall back to classical key generation algorithms, thus serving as an interface between user plane overlay (red network) and underlying non-secure infrastructure (black network)

The SDN Preliminary Design Report leverages on the work conducted in WP2, in which User Stories were used to define User Requirements, along with System Requirements. Those requirements have been analysed in WP3, so that a suitable architecture can be devised for Discretion. Figure 1-1 illustrates the followed methodology.

Figure 1-1: Work Package 3 SDN Methodology Design

1.2. Scope

As stated, this document presents ongoing architectural work on DISCRETION, establishing the baseline for further developments, and a consistency check against the requirements developed in WP2.

The document is structured around the different blocks that form the SDN, QKD enabled architecture.

Section 0 presents the general framework, with a State of the Art of SDN in different applications, and a general introduction about and security concerns red-black network separation paradigm for defence applications.

Section 0 introduces the general architecture with the different network planes and building blocks composing the solution and the communication interfaces established between them as long as the type of architecture chosen for the project.

Section 0 expands the analysis of security aspects introduced in section 4, focusing on the communication interfaces between red and black network, in which there is a clear trade-off in terms of security vs. black network programmability.

Section 0 focuses on the robustness, resiliency and reliability provided by SDN, presenting the different alternatives for SDN implementation and the implications in scalability and availability, as long as the options available for DISCRETION and the chosen path for it.

The following sections concern themselves with the different network planes in DISCRETION, namely Optical plane (Section 7), QKD plane (Section 8), Software Define Radio plane (section 0), and Data Layer plane (Section **Error! Reference source not found.**).

Section 11 details the QKS KMS mechanisms to distribute and synchronize quantum generated keys to the different application.

Finally, Section 12 introduces the security application, and its role as an application for the management of the CMs and used keys, and as an application for SDR configuration.

1.3. Acronyms and Abbreviations

The acronyms and abbreviations used in this document are the following ones:

Acronym	Meaning
AI	Artificial Intelligence
ALTO	Application-Layer Traffic Optimization
AMQP	Advanced Message Queuing Protocol
API	Application Programming Interface
BGP	Border Gateway Protocol
BGP-LS	Border Gateway Protocol – Link State

CM	Cypher Machine
COP	Control Orchestration Protocol
COTS	Commercial of the shelf
CPU	Central Processing Unit
CV-QKD	Continuous Variables – Quantum Key Distribution
DDoS	Distributed Denial of Service
E2E	End-to-End
eMMP	enhanced Mobile Broadband
EMS/NMS	Element/Network Management System
ETSI	European Telecommunications Standards Institute
FB	Functional Blocks
GMPLS	Generalized Multi-Protocol Label Switching
gRPC	gRPC Remote Procedure Call (no other official origin from "g")
GSGCP	Greedy Sub-Graph Cover Problem
ID	Identifier
IETF	Internet Engineering Task Force
IGP	Interior Gateway Protocol
IP	Internet Protocol
ITS	Information Theoretic Security
JSON-RPC	JavaScript Object Notation – Remote Procedure Call
KEM	Key Encapsulation Mechanism

KMS	Key Management System
L2MN	Layer 2 VPN Service Network Model
L2VPN	Layer 2 Virtual Private Networks
L3NM	Layer 3 VPN Service Network Model
L3VPN	Layer 3 Virtual Private Networks
LLDP	Link Layer Discovery Protocol
LO	Local Oscillator
LSP	Link State Protocol
LVS	Load Variance-based Synchronization
MAC	Mandatory Access Control
MEC	Mobile Edge Computing
MILS	Multiple Independent Levels of Security
ML	Machine Learning
mMTC	massive Machine-Type Communications
MPLS	MultiProtocol Label Switching
NBI	NorthBound Interface
NFV	Network Function Virtualization
NGNM	Next Generation Network Management
NIB	Network Information Base
NOS	Network Operating System
ODL	Open DayLight
ONOS	Open Network Operating System

OOPT	Open Optical & Packet Transport
OSS	Operational Support System
OTN	Optical Transport Network
P4	Programming Protocol-independent Packet Processors
PCC	Path Computation Client
PCE	Path Computation Element
PCEP	Path Computation Element Protocol
PCN	Protected Core Networking
PCS	Protected Core Segment
PoC	Proof-of-concept
PPDR	Public Protection and Disaster Recovery
PQC	Post Quantum Cryptography
PSK	Pre-Shared Keys
QFP	Quantum Forwarding Plane
QKD	Quantum Key Distribution
QKDN	Quantum Key Distribution Network
QoS	Quality of Service
RBAC	Role Based Access Control
RNG	Random Number Generator
RPC	Remote Procedure Calls
SA	Security Application / Secure Application

SAE	Secure Application Entity
SBI	SouthBound Interface
SDN	Software Defined Network
SD-QKD	Software Defined Quantum Key Distribution
SLA	Service Level Agreement
SNMP	Simple Network Management Protocol
SR	Segment Routing
SR-TE	Segment Routing - Traffic Engineering
SRTP	Secure Real-time Transport Protocol
TAPI	Transport – API
TCAM	Ternary Content Addressable Memory
TED	Traffic Engineering Database
TELSP	Traffic Engineering Link State Protocol
TLS	Transport Layer Security
TRSP	Tactical Radio Setup Point
TSF	Target Security Functionality
UML	Unified Modelling Language
uRLLC	Ultra-Reliable Low-Latency Communication
UUID	Universally Unique Identifier
YANG	Yet Another Next Generation
ZTP	Zero Touch Provisioning

1.4. Definitions

The definitions of the specific terms used in this document are the following ones:

Definition	Meaning
Key	A parameter used in conjunction with a cryptographic algorithm that determines its operation in such a way that an entity with knowledge of the key can reproduce, reverse or verify the operation while an entity without knowledge of the key cannot. [NIST SP 800-57 PART 1 REV. 5]
Key Material	Random sequence of bits that is used to derive a key from it.
QKD link	Set of active and/or passive components that connect a pair of QKD modules to enable them to perform QKD and where the security of symmetric keys established does not depend on the link components under any of the one or more QKD protocols executed [r_etsi15].
QKD module	Set of hardware, software or firmware components contained within a defined cryptographic boundary that implements part of one or more QKD protocol(s) to be capable of securely establishing symmetric keys with at least one other QKD module [r_etsi15].
QKD network (QKDN)	Network comprised of two or more QKD nodes [r_etsi15].
QKD node	Set of QKD modules installed in the same location within the same security perimeter [r_etsi15]
QKD protocol	Operations on quantum and classical signals that allow two parties to agree on commonly shared bit strings between two ends of a QKD link that remain secret [r_etsi15].
QKD system	Pair of QKD modules connected by a QKD link designed to provide Quantum Key Distribution functionality using QKD protocols [r_etsi15]
Quantum Key Distribution (QKD)	Procedure involving the transport of quantum states to establish symmetric keys between remote parties using a protocol with security based on quantum entanglement or the impossibility of perfectly cloning the transported quantum states [r_etsi15].

2. RELATED DOCUMENTS

2.1. Applicable Documents

The following table specifies the applicable documents that shall be complied with during project development.

Table 1: Applicable documents

Reference	Code	Title	Issue
[AD 1]	DISCRETION-DME-PRB-001	DISCRETION PROPOSAL	2021
[AD 2]	DISCRETION-DME-TEC-SRD01-E	D2.1 USER AND SYSTEM REQUIREMENTS REPORT	2022
[AD 3]	DISCRETION-DME-TEC-TEC02-E	D4.1 QKD NODES PRELIMINARY DESIGN REPORT	2022
[AD 4]	DISCRETION-DME-TEC-TEC03-E	D4.2 CIPHER MACHINES PRELIMINARY DESIGN REPORT	2022

3. GENERAL FRAMEWORK

3.1. SDN in CSP Networks

Software Defined Networking (SDN) is a term that refers to the ability of software applications to configure and control the elements of a network dynamically in an adaptable, cost-effective, and manageable way.

SDN separates network control from data forwarding functions enabling the programming of the network through the control plane, providing abstraction of the underlying infrastructure to applications.

One of the first protocols used in SDN is the OpenFlow protocol. This protocol is a framework developed at Stanford University that enables network controllers with access to the forwarding plane of a packet-switching device (an OpenFlow switch). In other words, this protocol allows software servers to determinate the forwarding path that the packages should follow in a network of switches. OpenFlow operates on the network control plane, installing politics and rules for the traffic flows depending on the information of the packet's headers. [3]

It should be noted that this protocol was the forerunner of SDN. However, there are some limitations like the required size of the TCAMs or processing power in switches and CPUs, latency to install new rules in the forwarding path, the scalability of the OpenFlow control plane, or the fact that it should replace current routing protocols, which makes it difficult to implement in a Telco application. Today, the main use of SDN is in the less challenging management plane, for which protocols like the technology agnostic NETCONF/RESTCONF protocols are well suited, while control plane is still implemented by traditional routing protocols like IGP, MPLS or SR.

Nowadays, OpenFlow protocol is losing relevance also in control plane applications, and it is evolving to P4 [4].

There are many explanations for the popularity that the SDN paradigm has achieved in the telco industry. Nowadays, computer networks are more complex and harder to manage due to the increasing number of devices (e.g., routers and switches servers, load balancers…), network speed, security, resources…

There is a huge demand for telecommunication services and traffic, number of users and connected devices, and network usage are all growing fast.

Concurrently, the operators are facing a tremendous growth in the number of servers and virtual machines, and thus, also in the need to be able to program, control and manage these networks dynamically and automatically. The Open Networking Foundation defines SDN architecture as [5]:

- **Programmable**: the network control is directly programmable due to the separation of the Control plane and the Data plane.
- **Agile**: Due to the separation of both Planes, the administrator of an SDN application can dynamically adjust the network configuration to satisfy continuously changing network requirements.
- **Centrally managed:** network control is centralized in software based SDN Controllers that keep a global view of the network.
- **Programmatically configured:** SDN permits to configure, manage, secure, and optimize the resources of the network using automated SDN programs, that may be implemented by network operation teams.
- **Open standards-based and vendor-neutral:** the control instructions are provided by SDN controllers instead of multiple, vendor-specific devices and protocols.

3.1.1. The evolution of Software Defined Network

Since the late 90s, with the accelerated growth of the Internet, it started to become necessary to provide new routing mechanisms in order to reduce network traffic congestions and latencies. To achieve this goal, some researching programs were formed, along with the idea of separating Control and Data plane in network devices, being this the seed for the first SDN designs.

Traditional (legacy) networks are complex and hard to manage and configure, often manually or with the use of low-level scripts, one element at a time, which leads to inefficient use of network resources and lack

of scalability [6]. SDN arises as a framework capable of overcoming these limitations due to the benefits that it can provide (separation of data and control plane, programmability, real time topology visibility, etc.). Figure 3-1 shows the separation between data layer and control layer in SDN deployments, as opposed to the monolithic structure of a legacy network.

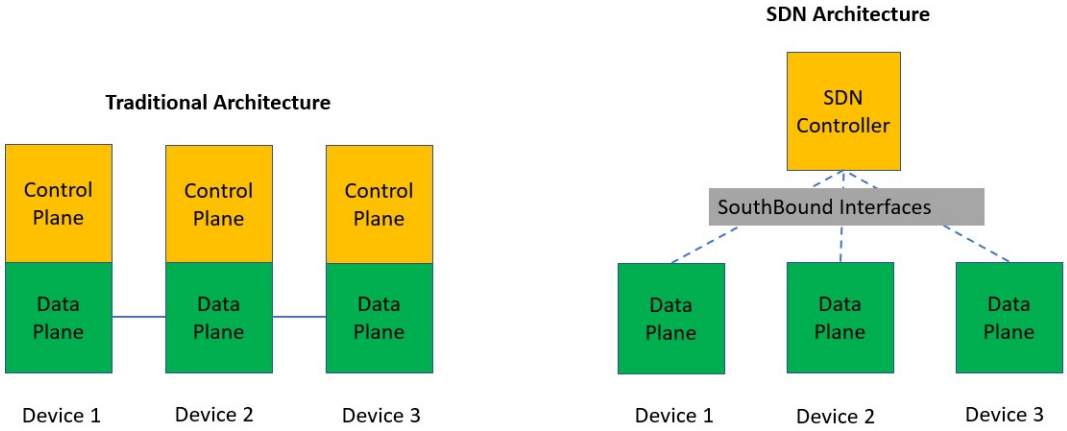

Figure 3-1: Traditional Architecture and SDN Architecture [7]

Also, in 2006, NETCONF [8] protocol appeared, providing a mechanism to install, manipulate and delete configuration of network elements. Later, in 2017, the RESTCONF [9] protocol appeared. Both protocols have grown in popularity and became the standard SDN protocols in IP and optical networks, and translate to network elements the information contained in data models implemented using the YANG [10] modelling language. These new protocols, added to the perspectives of SDN technology, inspired many researchers to move their sights towards the SDN field.

Some projects arose such as IDEALIST (Industry-Driven Elastic and Adaptive Lambda Infrastructure for Service and Transport networks) [11], STRAUSS (Scalable and efficienT oRchestrAtion of Ethernet services Using Software-defined and flexible optical networkS) [12] or MetroHAUL (METRO High bandwidth, 5G Application-aware optical network, with edge storage, compUte and low Latency) [13].

The first of these projects aims to define an efficient (in cost and power) transport network architecture that could be used to work with dynamic and varying traffic scenarios, and that advanced the state of the art of SDN in Optical Flexi-Grid Networks [14] [15].

STRAUSS was orientated to define an efficient optical infrastructure for an Ethernet multidomain ecosystem. In this definition, the protocol used to communicate with the orchestrator was the COP (Control Orchestrator Protocol), working together with the OpenFlow control plane paradigm. Some of STRAUSS's most important state of art contributions were the bases for further projects as the ONF T-API solution [16] or projects that deployed multi-domain SDN orchestrators [17] [18].

MetroHAUL had the goal of creating a cost-optimal metropolitan network for heterogeneous 5G access networks infrastructures. The main result of this project was the definition of a multilayer SDN orchestration mentioned in other whitepapers as [19], [20] or [21].

In the beginning of the 2020's decade, to standardize the different approaches that appeared, the Telecom Infra Project created MUST (Mandatory Use case requirements for SDN for Transport). The main goals of this project are to define the requirements and strategy in order to select the best SDN standards and so, unify different SDN solutions under a common architecture. MUST presents a hierarchical SDN controller with specific Domain Level Controllers for each transport technology (IP, Optical and microwave). The Hierarchical Controller is in charge of providing the real-time control and of communicating with the OSS, providing it with a resource abstraction. Some of the modules that constitute the SDN Controller are Topology module, Telemetry, PCE, Real Time Network DB, Alarms and Events manager and Transport Connectivity Services.

A recent research project that it is also aligned with the MUST specifications is the ongoing Teraflow [22] H2020 research project with the goal of defining a Secure cloud native SDN Controller that could be standardized. More information about the TeraFlow SDN in section 3.5.1.

3.1.2. SDN Layers

Figure 3-1 shows a high level SDN architecture. SDN is based on the concept of the separation of control and data planes. The SDN controller is responsible for the control of the managed entities (underlying networks) via a specific (SouthBound) interface.

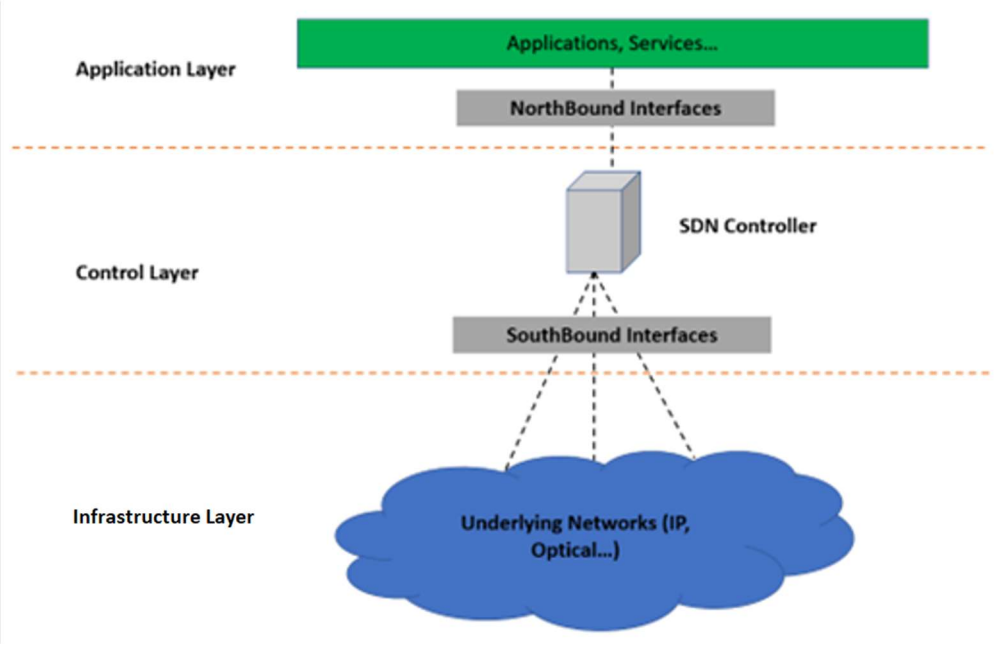

Figure 3-2: General Architecture of SDN [3]

In an SDN architecture there are three different layers:

- Infrastructure layer, composed by the managed/controlled network elements (switches, routers…)
- Control layer, composed by control elements, mainly the SDN controller. There are two mainly functions in this layer:
  - **Control Plane** is the responsible for taking decisions on how the packets should be forwarded by one or more network devices. Include topology discovery and maintenance, Packet route selection and path failover mechanisms. The control plane focuses on the forwarding plane and less on the operational plane.
  - **Management Plane** is the responsible for monitoring, maintaining, and configuring network devices, taking decisions based on e.g., the state of the device. The management plane focuses on the operational plane and not on the packet forwarding plane.
- Application layer, where applications and services define network behaviour and request services to the SDN controlled network through the Northbound interface. Applications include use cases such as network topology discovery, or network provisioning, among others.

Configuration between layers is carried out through the NorthBound and SouthBound Interfaces, with the configuration transport protocol RESTCONF [9] and NETCONF [8] respectively.

- Southbound interface is used to configure the underlying network, specifically the network devices, through the protocol mentioned above (NETCONF). The main goal of this interface is to provide communication and management between the SDN controller, nodes, physical/virtual switches, and routers. In addition, this interface, through the Real Time Database Topology Module that belongs to the SDN controller, allows the routers to discover network topology (represented by a graph) and to determine the optimal path through the Path Computation Element (PCE) [23].

NETCONF is the main API used in this project to configure the underlying network. In addition, BGP [24] (Border Gateway Protocol), SNMP [25] (Simple Network Management Protocol), MPLS (Multiprotocol Label Switching) [26] are also supported.

- Northbound interface is the API established between the SDN controller and the application layer. This allows a network administrator to access the SDN to realize the configuration or the retrieval of information from the SDN controller.

The SDN controller is responsible for the management of the forwarding traffic rules of SDN devices. Main networks services offered are: (i) Network topology management, (ii) collection of the network traffic information, (iii) notifications management, (iv) responsible for the configuration of the underlying network, (v) responsible of the determination of the shortest path with Path Computation Element, (vi) responsible of provide secure mechanisms to the network and (vii) monitorisation of the network information.

3.1.3. Software Defined Networking use cases

There is a list of essential uses cases, for creating, managing, and maintaining the networks, that all transport networks typically require. This list is covers topics such as:

- Network Creation: Automatic device day-0 and day-1 configuration. In other words, Network creation is the process of initially configuring the nodes in the network enabling the communication between them for different protocols and the automatization of the workday routine in the network infrastructure configuration, setting them ready for service.
- Service Provisioning: Establish connectivity services in the network for specific customers such as the L3NM [27] IETF model for Layer 3 VPN service provisioning within IP/MPLS Network), optical channels connectivity provisioning, etc.
- Inventory Support: Recovering information and state about hardware components and logical configuration of network elements through a controller, module, or application which objective is to discover the network hierarchy.
- Topology: Retrieval of Network Element information through a controller, module, or application to discover the logical representation of the underlying network for planning purposes.
- Traffic Engineering: Enforcement of traffic steering flows by MPLS tunnels or Segment Routing paths. This increases the efficiency of network resources usage due to mapping correctly the traffic flows using the available resources.
- Support to Performance Management: Retrieval of online telemetry information from underlying network nodes to assure network performance or detect bottlenecks.
- Support to Fault Management: Handling all provided network element information of faults, errors, failures, etc. Retrieved information is provided to the Management System.

Some examples of additional and relevant use cases for future evolution that are receiving attention by the industry are:

1. **Autonomous Networks using AI:** Combination of SDN (technology that enables automation in network) and AI (Artificial Intelligence is a technique that basically describes the capability to simulate or imitate the human reasoning as intelligent behavioural) to allows Dynamic network adaptation based on data information to cope with Traffic evolution, early failure detection, optimal routing, dynamic restoration, …

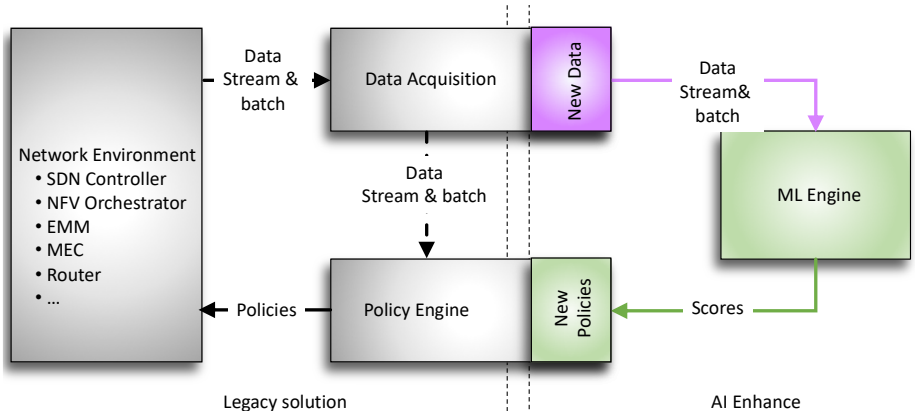

Figure 3-3: MAPE (monitor-analyse-plan-execute) loop implementation example.

Figure 3-3 depicts an exemplary of how an AI machine Learning (ML) based engine can obtain new data and process this information to create new policies that will be applied in the Network Environment (SDN Controller, Routers…).

2. **5G Slicing:** Network slicing is a key technology for 5G defined by NGNM [28] enabling efficient resource sharing of common Infrastructure between different services. These slices will permit network service providers to overcome the big challenge of upcoming 5G services: how to support and operate different kind of services with very distinct needs (i.e. eMBB, nMTC and uRLLC) onto the same physical infrastructure, that are created dynamically and transparently to the users and to the overall network orchestration mechanisms.

   5G envisages a mixture of centralized and distributed network functions. Hence, the transport networks shall be agnostic of whether the actual RAN topology is centralized or distributed for each slice.

   Basically, the goal is to assign logically network resources (including functions) to different customers or tenants that will perceive such resources as dedicated. Each slice will behave and appear as a full-functional network.

   Network slicing can be defined as the next form of providing wholesale services. Different situations can be considered here, leading to slices with different capabilities, specifically in terms of their management and control capabilities. It is possible to consider the following models [29]:

   - Internal slices, understood as the partitions used for internal services of the provider, retaining full control and management of them.
   - External slices, being those partitions hosting customer services, appearing to the customer as dedicated networks.

   The physical transport network links shall carry the traffic corresponding to all network slices that are using such link, a thus SDN Transport Network control will have to deal with specific requirements of each slice (throughput, delay, and jitter), as well as suitable inter-slice information exchange mechanisms to avoid congestion issues.

3. **SDN enabled Quantum Key Distribution (QKD):** Quantum Key Distribution technology generates keys based on transmitted QuBits (Quantum State representation, whatever its physical implementation such as the polarization of a photon, pulse phase…) using quantum technologies over a transmission medium to encrypt the communications through the generated QKD keys. ETSI proposes to use SDN orchestration [30] to deliver QKD keys to secure applications entities in an Optical Transport Network (OTN).
   In this use case, an operator requests service provisioning between two endpoints using QKD from the SDN orchestrator. Before, it is necessary that the SDN orchestrator requests the topology and inventory information of the underlying QKD network through the SDN QKD controller. Once the SDN orchestrator has the information of the QKD network, it must request a new path to transmit

the QKD keys between both endpoints and then the SDN QKD controller will choose the best path. With this configuration, an SDN orchestrator can manage multi-QKD network domains from multiple vendors.

3.1.4. SDN and network security

In both commercial and military contexts, users of the network need to have a minimum-security guarantee. In addition to the traditional threats, in SDN the control plane's decisions will affect a greater number of devices than in the traditional network scheme, so it is more important to prevent and mitigate any existent vulnerability. There are different Security Issues in SDN:

- Authenticity: There is the need of guaranteeing that entities in the network are the ones they claim to be.
- Attacks in the forwarding plane: network traffic flows through network elements can be disturbed by routers or switches that don't belong to the network resulting in malicious users launching distributed denial of service (DDoS) attacks.
- Vulnerability of the communications: In the southbound interfaces the protocol used for guaranteeing the communications in the data plane is the TLS protocol, but sometimes this protocol is disabled administratively, which can contribute to man-in-the middle attacks.
- Control plane threats: SDN controller is the responsible for the management and control of the underlying network. Therefore, if there is a problem in the control plane, the network elements will fail too.
- Confidentiality: Unauthorized users must not be permitted to access network information and use it.
- Availability: if there is a problem in the control plane (for example in the SDN controller) authorized users will not be able to access the data, devices and services.

The most common cryptographic algorithms used today rely on computationally hard security assumptions, which means that their resilience against brute force attacks relies on the fact that it would require an unreasonable amount of computing power or time to break. Due to the nature of quantum computers, they are able to attack the underlying mathematical principles used especially in asymmetric cryptography algorithms very efficiently, so many cryptographic algorithms are considered broken by quantum computers. This is especially impactful for key agreement/exchange protocols used to establish the initial secret used in securing a communication channel, as these key agreements use asymmetric cryptography.

However, QKD satisfies this very same need for secure initial secret sharing, with the difference that it doesn't base its security on computational complexity, but instead on physical principles on one hand, and on Information Theoretic Secure (ITS) algorithms that probably can't be broken even by an attacker with the highest possible computational power.

Section 3.4 details different security solutions that we will use along the project.

3.2. SDN in Military Scenarios

3.2.1. Overview

Software Defined Networking (SDN) has emerged from the civil world as a new and innovative paradigm targeting to revolutionize the way communication networks are built and operate, bringing them flexibility and manageability. It has long been identified as a promising approach for defence networks. In the strategic/operational communications fields, these are quite similar to civil networks, with added requirements to make them even more reliable (security, robustness, resilience). This similarity enables the application of technologies that are proven on the civil field, that - enriched with new requirements - become successful solutions also in the defence area.

Tactical networks, however, are extremely challenging, and unlike civil networks (except, in part, PPDR networks), have to work in an environment where they are "weaponized". These may be the most challenging network environments, as networks have to provide communications in a highly dynamic, unpredictable, and ultimately hostile environment.

Having this in mind, the flexibility and easier way to change any parameter in the network, the concept of an SDN can be quite interesting in dealing with some tactical or operational, single-domain, coalition or federated networks. Nowadays, there are already some pioneer works in implementing these concepts to tactical networks [1]. SDN can also bring some benefits when used to provide enforcement of policies in a federated military network [2].

Analysing first the case of tactical networks, most of the times they have to support various heterogeneous, intermittent, and ad-hoc communications with diverse traffic patterns and security requirements. Thus, these inherent constraints can trigger some questions to be addressed to exploit the combined benefits of SDN in military scenarios. In particular, how can SDN be employed for securely and dynamically managing traffic with multiple security classifications, to handle traffic of different sensitivities and access policies, in an environment that includes legacy applications. There are several types of classified information that can be shared during any military communications. However, the security class of information can change dynamically according to the context of the physical environment. Thus, it is essential, to handle the traffic of such classified information with compatible security features and access flexibility. For instance, a consistent inspection of the data packets and environmental context shall be required. But this approach can also expose sensitive traffic data to various untrusted SDN controllers. An SDN-based approach for tactical ad hoc networks will also bring new challenges.

Another important question in networking is how to forward traffic between different network domains (or segments) of a larger network. SDN is mainly an intra domain network technology and would not work straight out of the box in a federated network with autonomous network segments, each with its own controller. Then, another challenge when implementing an SDN in the military scenarios is the problem with inter-domain state distribution, that is how state information is shared between interconnected domains. The problem is that SDN controllers, in their current form, do not have any knowledge of the network outside of its own domain, unless this information is given. This can be an issue when using SDN in a federated network. Federated networks are not specific to the military but could be used to describe any network where several network segments from different operators are interconnected to create one common service or resource. The purpose of these networks is twofold: Firstly, they serve information sharing between different governmental entities in a country or different nations in a coalition. Secondly, they provide a unified mesh so that the different nations or different national entities can share network resources when needed. Typically, these federated networks (or military networks in general) are built up by a multitude of different network bearers. The access network close to end-hosts are usually built up from high-capacity fibre links, that is core elements between nations and access networks can usually be interconnected through fibre, however often it is based on radio links (radio relays and SATCOM). Lower capacity radio links are also used between access networks and remote end-hosts, such as forward operating bases and individual troops. The problem that can be posed is how SDN can be used to provide enforcement of policies in a federated military network that crosses the domains of strategic, operational and tactical communications. The biggest challenge here is that, while providing policy enforcement for one segment of such a network is already difficult, providing coherent policy enforcement for all flows traversing all segments of a federated network is even more complex. In this sense, the first obstacle with policy enforcement in federated networks is a high-level agreement of the actual policy rules, which is above all a political problem. Then all nations need to trust each other, so that the different operators can be sure that their traffic gets treated correctly, are allocated the proper resources, and is sufficiently secured in other segments of the network than the one(s) they operate and own. In an SDN enabled network, management of network policies can be placed in a policy engine application at the centralized controller (controller or application plane). However, this raises a question for federated networks of where the controller should be placed. In [2] some solutions are presented like distributed controllers, one for each segment, or hierarchical tree structure with a central top-level controller, or even one central controller for all segments.

While some of these challenges could be regarded as purely political, like in the case of federated networks, and mostly related with creating good SLAs and high-level policy rules, there are some areas where SDN can add value, in particular the flexibility in network management that SDN can provide to implement novel management methods and policy rules that fit specific network scenarios. However, it is clear that there are some weaknesses with using SDN at the current time, like inter-domain state distribution and east/west communication protocols (e.g. SDNi), which are still scarcely researched. And, of course, the problem of security, since the SDN approach of separating Data and Control plane exposes a much larger surface to be attacked and new attack vectors to be explored. In this sense security improvements through QKD or cipher machines should be leveraged to keep the SDN as secure as possible.

3.2.2. SDN for Tactical networks

The use of SDN in tactical scenarios is promising. As a principle, the programmability of the network brings with it the flexibility to tackle with a highly heterogenous and dynamically changing environment. Nevertheless, an overall SDN approach is difficult to envision, mostly because of the same heterogeneity and dynamicity and a notoriously challenging principle of SDN: Centralized control. Centralized Control is attained by shifting the control functions from the data forwarding devices to a conceptually centralized network entity, the controller. This controller has a global view of the network and makes decisions about how the data packets should flow through the network. This is a *Logical* centralization, though, which does not imply a single-point-of-failure when paired with a resiliency focused distributed architecture.

Various mechanisms exist to guarantee the resiliency of the control plane (described in section 6 of this document).

The DISCRETION framework for a tactical SDN is described in section 9, and is composed of a set of controllable radio (SDR) equipment under the control of a CCC (Command and Control Center), featuring two major scenarios: one before a mission, where the radios are still attached to a fixed network, and hence can interact with the whole SDN and benefit from QKD and general connectivity; and another, during a mission, where tactical radios are controlled only by the tactical SDN controller on the CCC.

On "real-life" scenarios some of these tactical SDNs may coexist within the same authority (e.g., different forces within the same national army), but typically a hierarchical control structure is created so these can still be considered as a single SDN domain**.**

On a coalition scenario this is not as clear: while all missions usually develop under a single command, the communications network may not share this hierarchy (either because of technical incompatibilities or specific policies), and multiple SDN domains may hence be involved, not under common control, but relying on "East-West" interactions between the various domains involved.

3.3. SDN-QKD networks

The flexibility provided by Software Defined Networking is what allows the introduction of disruptive technologies such as Quantum Key Distribution in modern networks.

While the old paradigms required a complete retooling of the whole software and hardware stack to allow the integration of classical and quantum networking in the same infrastructure, SDN achieves the same results essentially through the SDN controller. Obviously, this is not totally free of requirements and some programmability of the devices in the data forwarding plane together with a QKD-aware controller are needed. This, however, can be done incrementally and does not require a complete network overhaul or the construction of a separate, ad-hoc, network for the transport of quantum signals.

The resulting infrastructure when mixing QKD and SDN is much more convenient in terms of management, deployment and upgradability than using standard schemes although, obviously, we are limited by the laws of physics: the network can be optimized in many ways, but the compatibility is finally limited by physical restrictions.

The first proposal to mix SDN and QKD was published in QCRYPT [31] in 2016. The proposal was to install QKD devices in the forwarding plane (see Figure 3-4) and having these devices communicate with the control plane using standard modelling languages such as YANG.

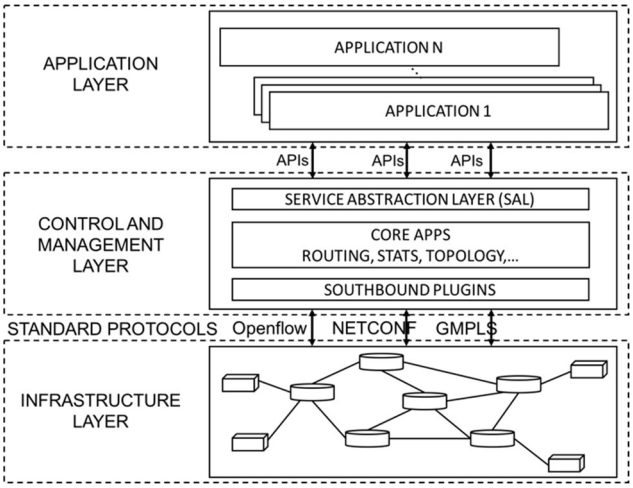

Figure 3-4: Initial scheme of a SDN-QKD network. QKD devices are installed in the infrastructure layer and export their capabilities using common methods to the control layer.

The control plane would then know the capabilities and requirements of the devices. Logic can be built then in the SDN controller to reserve clean optical paths to transport the quantum signals, know the performance and status of the devices, route the connections, etc. such that end-to-end secret keys can be created. The first real-world demonstration of these ideas was done in Madrid in 2018 [32]. Three nodes installed in production facilities were installed (see Figure 3-5), a full network stack was developed and several demonstrations to secure the data plane, the control plane, etc. were performed. Quantum/Classical compatibility was also demonstrated and 17 100Gbps classical channels were transmitted together with a quantum channel using the same infrastructure, including the optical fibre.

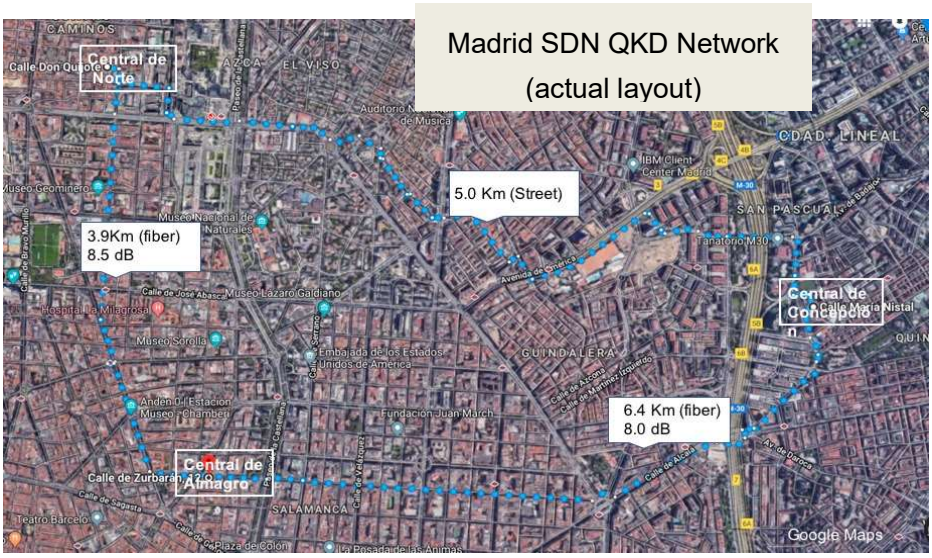

Figure 3-5. Madrid SDN QKD network in 2018. Three nodes in a the production facilities of Telefónica. This was the first SDN-QKD deployment in real production facilities. [32]

This was an important proof of concept that has allowed the creation of the current Madrid Quantum Communications Infrastructure within the H2020 project OpenQKD (see Figure 3-6). This project is considered as the ramp up phase of the European Quantum Communications Infrastructure, the 10 years long program that intends to communicate all Europe using a quantum network.

The current MadQCI is the largest quantum network ever built in Europe, with 28 QKD devices installed in 11 nodes, linking 10 sites in production facilities, and all the attendant optical transport and cyphering infrastructure. It also sports advanced features like optical switching of the quantum channel, allowing for over 50 possible different quantum-level connections among the 11 nodes. The infrastructure is heterogeneous, integrating QKD, optical and cyphering devices from many manufacturers under the control of a single, logically centralized albeit physically distributed, SDN-QKD controller. The optical fibre network that is the substrate of the MadQCI also belong to two different providers, allowing for testing of advanced characteristics such as border nodes. It is important to realize, that MadQCi has been built to be a real-world network and that, together with the QKD services it is simultaneously serving classical communications to many customers in the Madrid area, and is able to produce keys from any to any node

Finally, it is interesting to note that SDN is the architecture that is currently considered for the future EuroQCI.

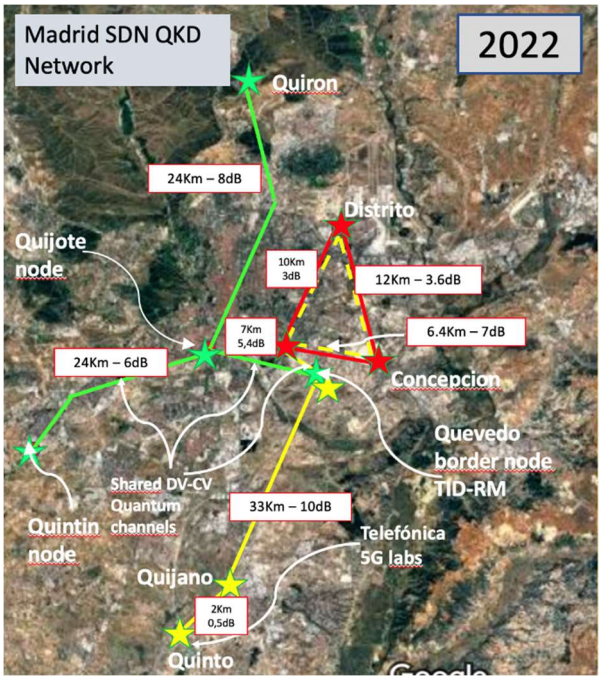

Figure 3-6: Madrid QCI network in 2022.

3.4. SDN Security Considerations

When compared to traditional networks, the defining characteristics of the SDN paradigm and its common architectures carry a new set of relevant security considerations, both in terms of benefits and risks. The centralization of network management paired with the broader network view allows for more efficient management and monitoring, which in turn strengthens the network's incident response capabilities. However, this very centralization poses new security challenges to both resilience and authenticity, as it condenses a large set of critical network functionality on a reduced surface.

To address the first point of resiliency, approaches like the adoption of a distributed controller SDN architecture arise, even though these approaches further exacerbate the challenges of assuring the authenticity and authorization of control plane communications by increasing the number of involved equipment and respective communication.

For this second point concerning the authentication and authorization of control plane communication, a solution could be the adoption of security protocols that ensure not only communication confidentiality and integrity, but also replay protection and mutual authentication, such as TLS or IPsec, bootstrapped using security certificates or pre-shared keys. It should be added that one of the goals of DISCRETION is to ensure resistance against quantum computing threats which could be accomplished by using post quantum cryptography with the previously mentioned protocols. It is also important to include some kind of

functional security policy enforcement system such as role-based access control (RBAC), or equivalent, to restrict functionality based on assigned roles following the least privilege principle. Further analysis of SDN robustness, resiliency and reliability is presented in section 6.

Also important is the major trend towards virtualization, especially in the Control Plane of the SDN. With Data/Control plane separation, Control Plane Functions not only cease to be physically contained in network appliances but are also disaggregated and typically deployed in data centres, as "common" workloads. There, all security aspects that are considered for software running under those conditions will have to be applied.

Another relevant characteristic of SDN networks is the possibility of multi-tenancy, in which multiple tenants can make use of the same physical network for hosting their own virtual networks. In such a scenario, it is necessary to ensure an adequate level of segregation to provide an acceptable level of security. This segregation needs to happen on both control and data planes, as well as at a resource level. In part, isolation can be achieved via tagging mechanisms or other types of encapsulations (MPLS, IPsec, etc), which can be further strengthened though the use of encryption, minimizing damages in case of isolation failures. Additionally, resource isolation is required to prevent tenant specific problems from overloading the underlying hardware and thus causing a negative impact on other tenants.

One of the key SDN features is its programmability, which is the ability to safely, efficiently, and dynamically reconfigure the network and underlying network equipment to reorganize data flows, comply with high level policies or even to comply with SDN application requests. This empowers the network with a large amount of flexibility and agility which can be leveraged for greatly improved incident response and mitigation capabilities, at the cost of having to ensure persistent network coherence against intentional and unintentional threats. Some ways of ensuring coherence are through the use of regression testing upon reconfiguration and persistent monitoring and sanity checking. Conflict resolution mechanisms might also be necessary in architectures exposed to reconfiguration conflicts.

3.4.1. Protected Core Networking

One of the main challenges in military networks is to ensure end-to-end Quality of Service objectives in a changing environment where it is only possible to deploy narrowband-pipe connections and where the conditions are so unpredictable [33]. This is where Protected Core Networking (PCN) appears. The main idea of this network architecture is to create an interconnection that allows a flexible and efficient information exchange between different classified information domains.

These domains may be owned by different countries and may have different levels of protection, so the shared network architecture will be based on an unclassified network infrastructure, orientated to provide connectivity and robustness paths. On the other side, the information transported will also count with its own protection, making both layers independent.

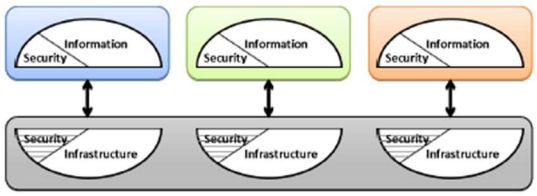

Figure 3-7. Infrastructure and Data Disentanglement [33]

The collection of information nodes that are connected to the Protected Core is named coloured cloud (CC), and this CC will be the one in charge to provide confidentiality to the communication. This security goal will be helped by the Multiple Independent Levels of Security (MILS) concept, that allows the creation of specifics compartments (being each compartment a different CC) with different security domains and security measures, e.g., cryptography units, information tags, hypervisor, etc., with the idea of realising a secured exchange between CCs [34].

In the architecture layer, a group of nodes working together and forming a connected topology is called a Protected Core Segment (PCS). Each PCS works also with other PCSs through a Federation System approach creating a Protected Core. If this PC can provide transport services in dynamic environments warranting availability, we are talking about a PCN. This Federation System also allows the simplification of the authentication implementation, using the different federate identities as trusted Certification

Authorities [35]. This need of disentanglement of the physical and logical layers makes SDN the perfect networking architecture to implement PCN, and its applicability to DISCRETION will be analysed during the project.

3.4.2. Red-Black Network Separation

The Red-Back Architecture, as defined in [36], refers to the careful segregation of signals that contain sensitive or classified plaintext information from signals containing encrypted information. Those borders are physically separated and shielded to prevent unintended information leakage using the TEMPEST standard [37]. The main idea is that you have two networks: one (Red) which is under your control, and you can guarantee the security, so it is not necessary to encrypt communication. The other one (Black) is an external network that you may use to interconnect Red network segments and, as you do not control the security of this network, the information must be ciphered when it is transmitted by this network. This method has been used since the early 1970s in military contexts, but now, we have to be able to mix it with SDN, Protected Core Networking architectures and quantum key distribution, going one step further than the previous implementations.

In a Red/Black paradigm, every communication that passes by the black network must be protected by at least one encryption layer, e.g., IPsec VPN tunnel, TLS or SRTP encryption, in addition to the usage of mechanisms for security administration of the borders of the Red Network with the Black one. Current asymmetric algorithms employed by the identified cryptographic protocols to establish a shared secret for the protection of a secure channel are not quantum secure, and thus the only viable option is the use of quantum secure mechanisms such as QKD or Post Quantum Cryptography (PQC) algorithms and schemes, the later not being standardised or validated yet.

In our scenario, it will be the cipher machine the mechanism that ensures the information is encrypted with the corresponding key before being routed to the outer firewall with the encryption requirements for each known network.

Some architectures also contemplate a third security domain (grey) by specifying that a second encryption layer is required to cross the border, and each domain crossing refers to a different encryption unwrap. [38]

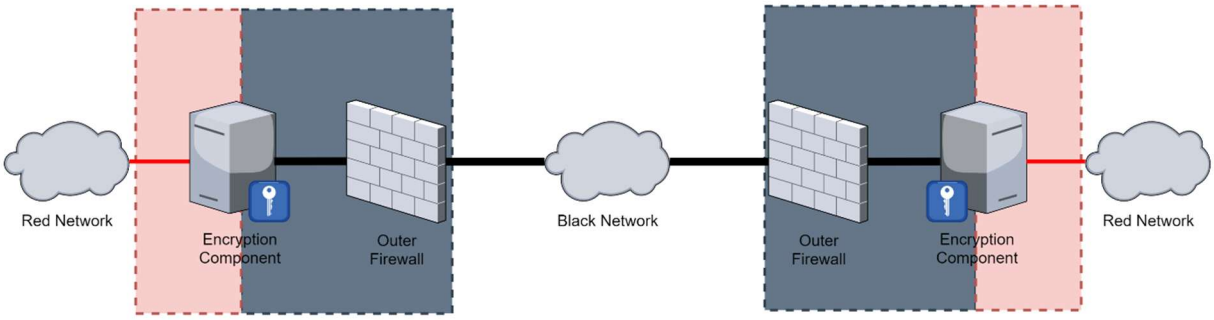

Figure 3-8 - Red-Black Architecture

With the introduction of QKD distribution, there's no consensual agreed upon architecture, with different entities holding different opinions on how to implement the QKD channel (as a red, grey, or black channel).

To retain border isolation we are evaluating several different scenarios including the following three:

1. QKD data is considered confidential or dangerous if manipulated and thus has to be encrypted to traverse through the black network – Figure 3-9;
2. QKD distillation data is considered safe to handle even if compromised, and thus can traverse through the BLACK network unencrypted (using a secondary black-red channel) – Figure 3-10;
3. QKD data is considered to be a part of another boundary (grey), which is isolated from the red network but is not exposed to the same threats as a direct red to black network connection. In this scenario, QKD distillation data is never encrypted but there is a border between the QKD node and

the red network. This custom definition of the grey network represents an additional boundary with its own border security mechanisms which doesn't fit in with classical Red/Black definitions but is an attempt at accommodating the physical security guarantees and constraints presented by QKD – Figure 3-11.

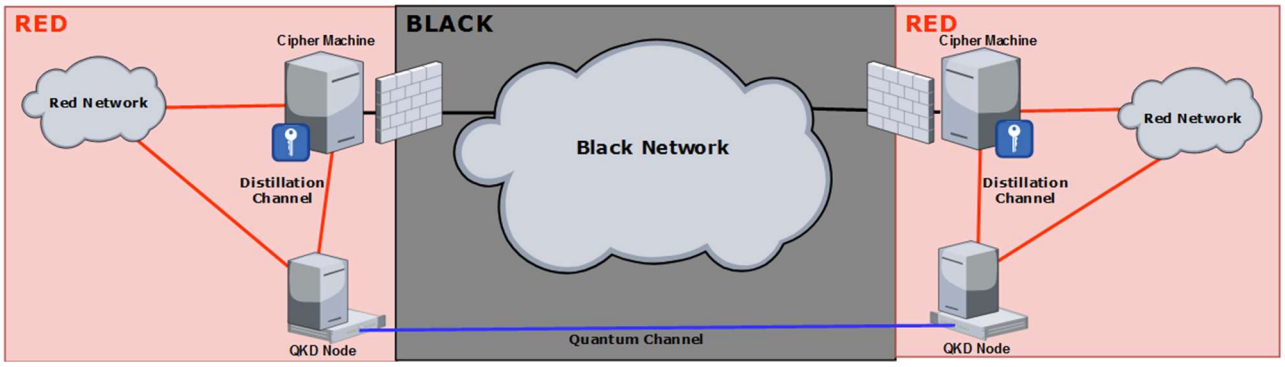

Figure 3-9 - Red-Black QKD architecture with encrypted distillation channel

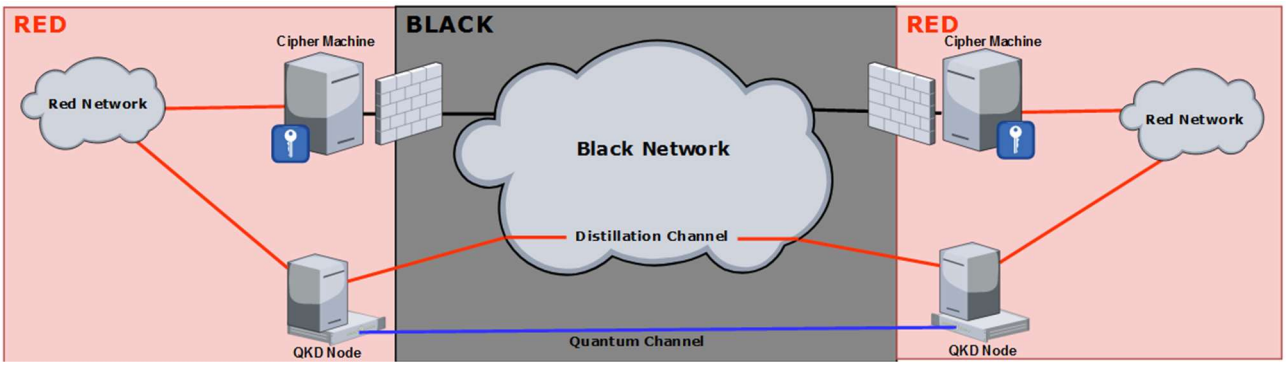

Figure 3-10 - Red-Black QKD architecture with unencrypted distillation channel

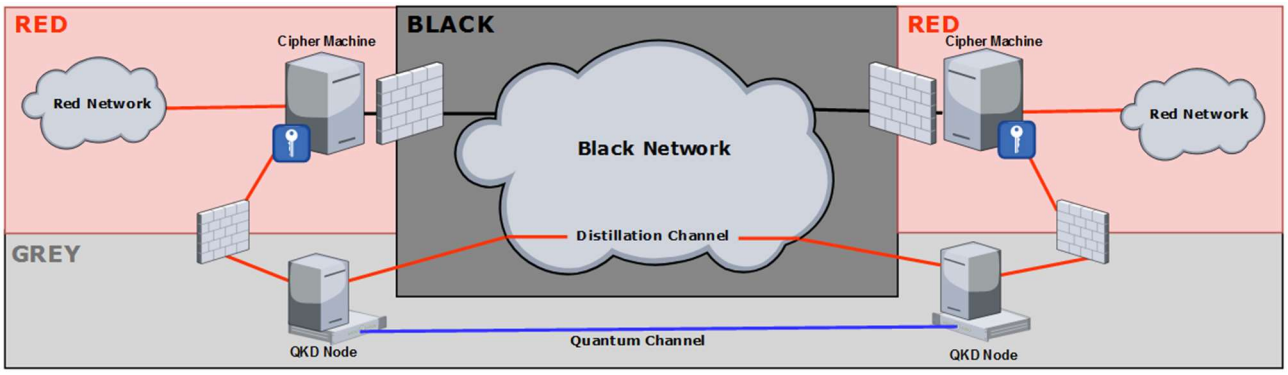

Figure 3-11 - Red-Gray-Black QKD architecture with separate QKD boundary

The red channels communicate in plaintext, the black channels are encrypted, and the blue channels are exceptional channels where the physical security properties of the QKD protocol and its constraints take effect.

	Threats and limitations	Opportunities

1	Huge throughput requirements that can be few orders of magnitude larger than the available throughput depending on key rate, post-processing efficiency, etc. High cost of encryption and resource consumption. Can become manipulated as a denial of service, overloading the cipher machine and effectively blocking all communication to and from the red network.	This would enforce authentication and encryption inside an already established classical channel. Compliance with traditional Red/Black paradigm.
2	By allowing the traversal of unchecked information across the border, we expose another attack surface by allowing communications using the QKD distillation fibre to reach the outside and allow outside information to pierce into the internal networks, violating the traditional Red/Black paradigm.	This model allows the QKD Node and KMS to reside inside the red network. A lot more throughput available for red network usage.
3	Could mean the SDN controllers and/or KMS would need to reside in the grey border, exposing a wider, although controlled attack surface. Might require additional mechanisms and logic to provide similar functionality as the previous scenarios across a new border.	Adds a secondary security boundary behind the QKD node's exposed surface in case it needs to be directly exposed to the black network (for throughput or protocolar reasons). Allows for easier SD-QKD communication with the black network if necessary.

These scenarios significantly change the overall architecture and the features the SDN might be able to provide, and thus will require further evaluating with adequate security impact consideration.

3.4.3. Application Layer Transport Optimization

In military contexts, it may be necessary to provide resiliency, availability and differentiated QoS to some services. Therefore, exposure of network capabilities in the black network can help red SDN controllers to make better decisions in order to improve service delivery. To achieve this goal, it is necessary the implementation of mechanisms that may provide these capabilities. One suitable candidate is ALTO protocol, which has been proposed in IETF as the Network Exposure Function of underlay network capabilities for multiple overlays on the top of the network) [39].

The SDN controller is the management and control element still responsible to calculate optimal paths, using IETF functional component like Path Computation Element (PCE). In contrast, ALTO acts as "one-stop-shop" for retrieving information of the network, leveraging the capabilities of the SDN controller or PCE, and providing certain level of isolation between red and black networks at the same time.

In addition, ALTO has been included in projects like ABNO [40] which let the optimization of traffic flows between applications, information of the topology stored in the ALTO server, The Network Capabilities Exposure can be consulted in Section 4.2.1.

Current implementations of ALTO provide an update of network capabilities upon request. In a scenario where exchange of information is limited, an extension providing periodic and dynamic actualization of the ALTO service could be proposed, avoiding the need of an explicit request from the red network

Additionally, current ALTO standard offers a limited set of multi-cost metrics. To extend these capabilities (e.g. for security applications) it has been proposed to extend the amount of information provided to ALTO clients using multi-cost metrics [41], helping to solve the problems stated in [42]. The goal is to offer to ALTO clients the possibility of obtaining various performance metrics via a pub-sub channel, where each type of information is a different topic to consume.

Figure 3-12 shows an example of how ALTO could serve as an interlocutor between Red and Black SDN controllers. In this secure network, the SDN controller will use this information received to determinate the new routing politics and, if it is necessary, determinate the black nodes to use to deliver the traffic.

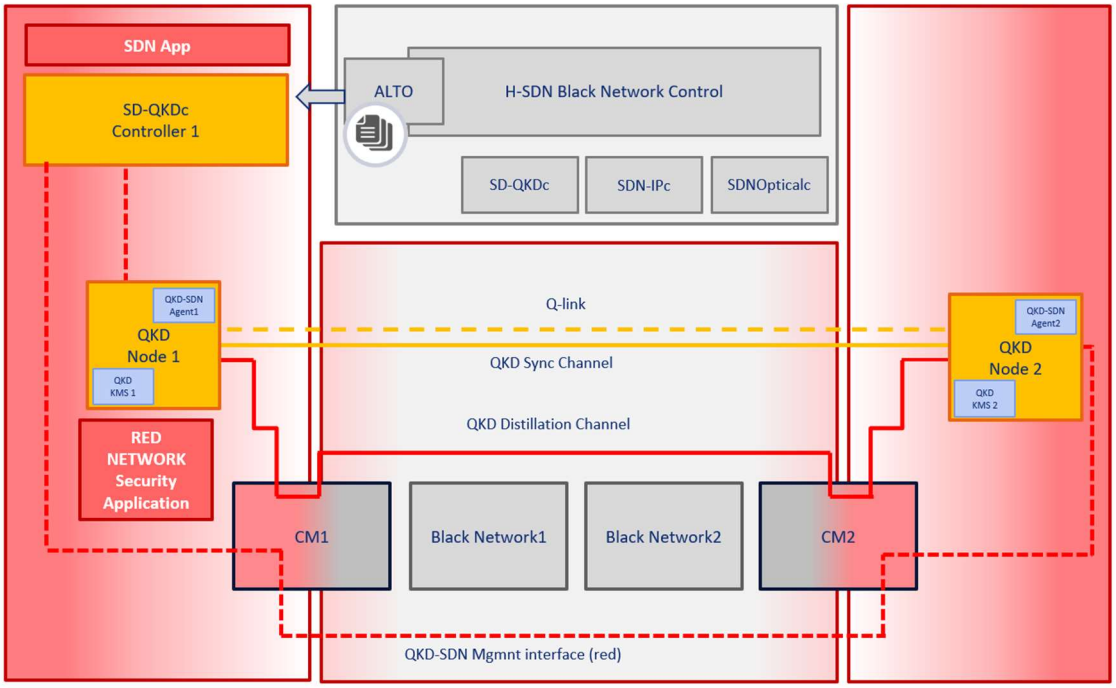

Figure 3-12. SDN architecture in red-black network deployments working with ALTO.

3.5. Reference Platform, Models and Standards

In this section there are discussed some standards related to fundamental elements belonging to an SDN architecture that are used as reference in the optical SDN controller design for DISCRETION. In particular, in the following subsection, there are described 2 top players in the field of open-source SDN controllers, along with TAPI, NETCONF and YANG, a modelling language that provides an abstracted view of the controlled devices.

3.5.1. SDN controller

In the context of software defined networking, the open-source ecosystem offers a set of SDN controllers largely accepted by the telco industry and operators. OpenDaylight and ONOS are part of such a set.

OpenDaylight

OpenDaylight (ODL) is an open-source SDN controller supported by a large community and maintained by the Linux Foundation[1], characterized by a high-level of customizability that allow the covering of significant number of scenarios in terms of both protocols and network devices supported. In Figure 3-13 is depicted the ODL general architecture.

[1] https://www.linuxfoundation.org/

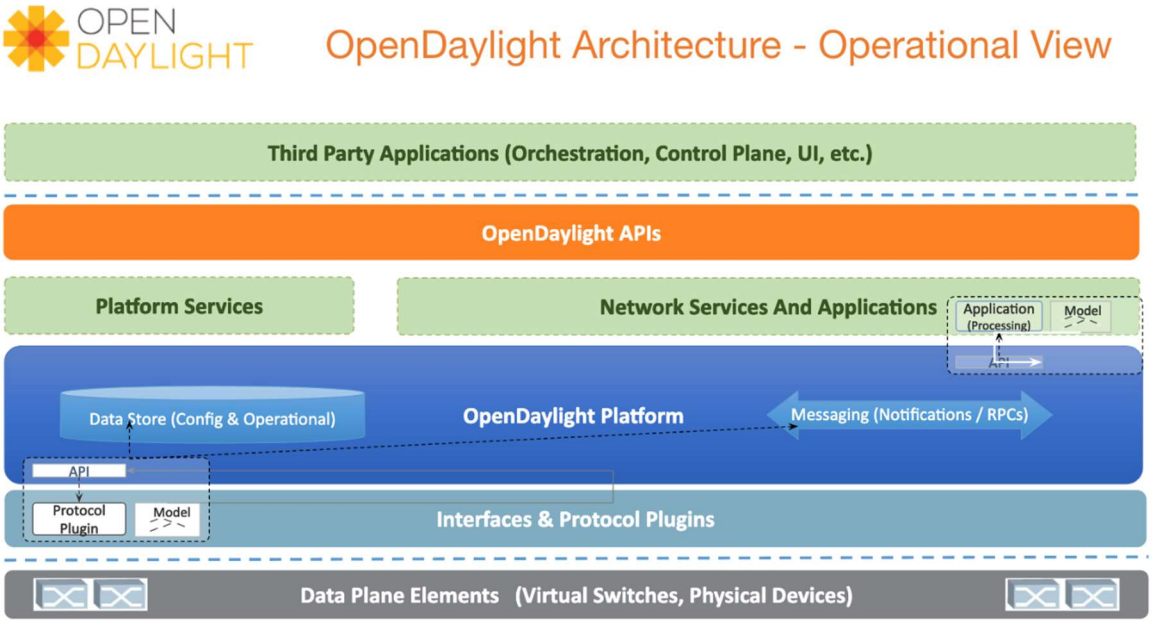

Figure 3-13: OpenDaylight Sodium Architecture [43]

The elements building ODL are the ones bounded by the two horizontal dashed lines. On the very top and bottom levels are placed the third-party applications and the controlled devices respectively. The orange element in the architecture, OpenDaylight APIs, represent the controller's NBI sitting on top of a layer of services and applications. Platform services refers to all those services allows internal and third-party applications to exploit resource belonging to ODL Platform such as internal Data stores and message bus, but also AAA (Authentication, Authorisation and Accounting), JSON-RPC extension etc. The Network Service and Application Layer contains all the element that will communicate directly with external elements (e.g., Network Devices, lower layer SDN controllers, etc.) through the Interface & Protocol Plugins Layer, which represents ODL SBI. Such a layer includes, among the others, a NETCONF plugin provided by ODL itself while the Network Services and Applications is the layer where the Provisioning, Topology and PCE applications, customized for DISCRETION purposes, are placed. Both layers are indeed extensible with custom applications and plugins. Figure also simply describes how the communication between an application a plugin happens: an application can invoke the APIs exposed by a plugin by exploiting a messaging mechanism build in the platform layer that allows to execute RPCs (Remote Procedure Calls) and receive Notifications. It is important to note that in the representations of both plugin and application there is an element called "model". ODL, indeed, requires that an application/plugin must be modelled using YANG that specify both the interface exposed (APIs) and its information model: ODL automatically generates the source code (in Java) on the basis of the model. For that reason, the ODL Platform Layer is called MD-SAL: Model Driven Service Access Layer. The MD-SAL exploits YANG models to create an internal representation of plugins and applications (model-driven) and offers them a mechanism for interacting each other without care about their specific implementations. This mechanism significantly reduces the issues related to inter-module communication and data representation leaving at the same time the possibility to the developers to implement the internal application logic as they prefer. This similarly happens with NETCONF, that exploit YANG to build an abstract representation of the devices to be controlled, as discussed in the next section.

ONOS

Open Network Operating System (ONOS) is one of top player in the context of Open-Source Programmable SDN platforms, maintained by the Open Networking Foundation[2] (ONF) as well as the TAPI. It shares with ODL some characteristic such as the fact that it is a Java-based software that exploits Karaf OSGi container [44], is highly modular and charactered by a high TRL.

[2] https://opennetworking.org/

ONOS is a multi-layer SDN platform. Each layer, called *Tier*, encapsulates several modules that implement certain pieces functionalities related to the control of a network. A unit of functionality is called *service* and encompasses a vertical set of components traversing different tiers (each service component belongs to one of the main tiers) and for that reason they are also called *subsystems*.

Figure 3-14 summarize the ONOS concept of multi-layer architecture.

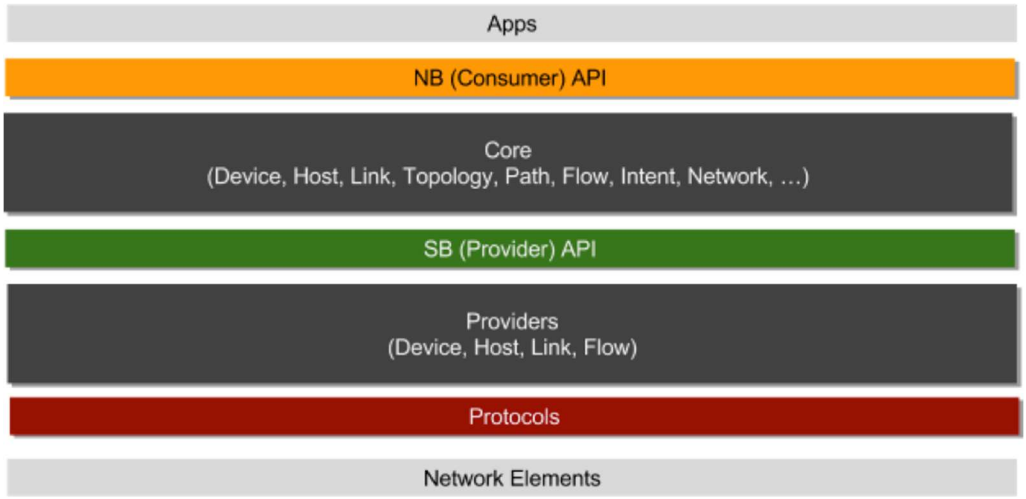

Figure 3-14: ONOS Multi-tier architecture [45]

The *Providers* tier is the lower layer tier and encapsulated modules interfacing directly the network. Such tier exposes an interface (the green rectangle in the figure) towards the upper one, the Core and it is considered the ONOS SBI. Components belonging to a service and residing in the *Core Tier* are also called Managers: it interacts with providers consuming their interfaces and serving information to applications or other services. The Core includes a Store element that manages the information required by the Managers ensuring consistency and persistency. Since ONOS can work also in a distributed manner, i.e., multiple instances of ONOS interacting together, the Store also offers mechanism for the remote synchronization with its own homologues running in other ONOS instances. *Application* is the upper layer tier that manipulate the set of information received by the Managers. As per ODL, applications cover a wide range of functionalities such as manipulate network topology, provision path, visualize information, etc. It is important to note that not all the subsystems consist of components residing on all the 3 tiers. One example is the *TopologyProvider* that relies on an abstract representation of the network provided by the Core tier and never interacts with the devices, avoiding the usage of any element in the Providers tier.

TeraFlowSDN

As a result of TeraFlow project, ETSI group decided to create a secure open-source cloud-native SDN controller, trying to realize an integration between actual NFV and MEC frameworks and allowing the integration for traffic management and network devices. During the current development stage, scalability and control resilience are the main goals to archive for this project. TeraFlowSDN is also working to be able to offer extensions for OpenConfig network elements and an extension of the ONF T-API for optical SDN controllers. [22]

The controller designed here is orientated to provide the capabilities and services expected for beyond fifth-generation networks. The main areas that this solution deal with are: (i) the definition of a cloud-native network operating system (NOS) to serve high performance control plane operations, (ii) native support for IP, optical and microwave traffic transport technologies, (iii) automated zero touch provisioning (ZTP) of network services and operations and (iv) multi-tenant network slicing as a service and SLA requirements. Teraflow also includes ML modules to provide security. This project uses Netconf [8] and Restconf [9] as the protocols to communicate between APIs, being YANG the data model used.

Some of the features included in this project are machine learning based security module, automatic resource administration component, Inter-domain manager, centralized and distributed attack detectors, etc. In the next figure it is possible to see a more complete features view.

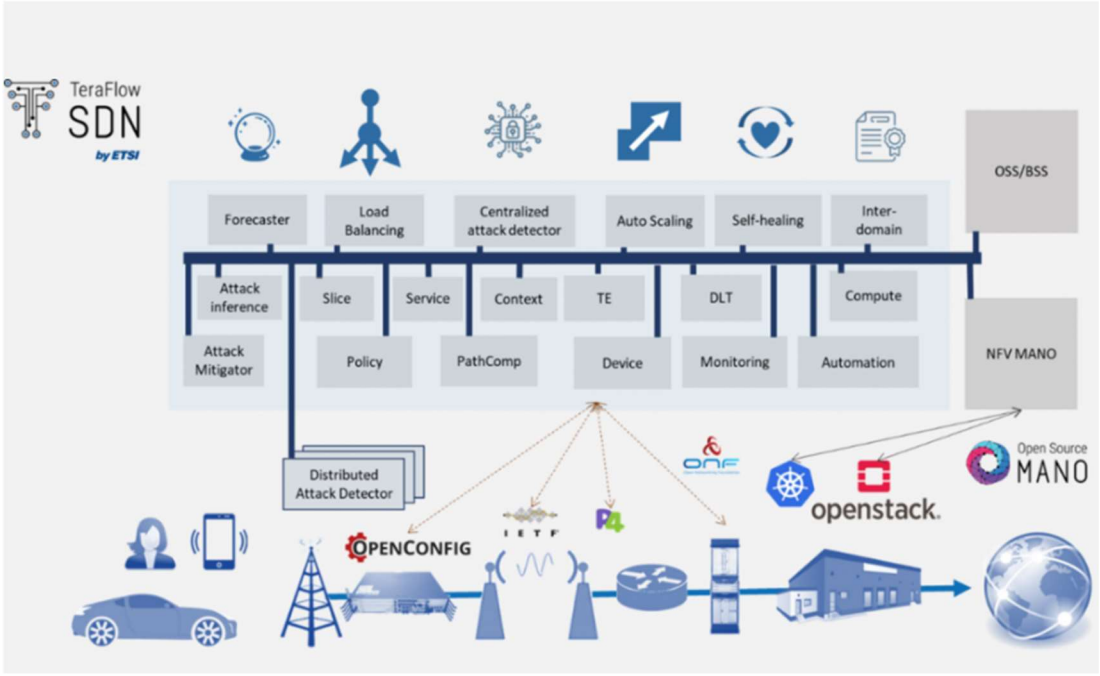

Figure 3-15. TeraFlowSDN components [22]

3.5.2. Interface and Network Modelling

In this subsection are described 3 standards/large accepted solutions for in the context of SDN environment in terms for the SDN communication protocol, the definition of the NBI and the modelling of the devices: NETCONF, Transport API and YANG modelling language.

NETCONF and Yang

The NETCONF protocol is an SDN Network Management protocol standardized by IETF in a set of RFCs, that include the base definition of the protocol (RFC6241), the secure connectivity establishment (RFC6242, RFC7589) and other capabilities.

The protocol is in charge to manage the lifecycle of a configuration in a network device, allowing all the 4 classical CRUD operation: Create (Install), Read, Update, Delete configuration.

NETCONF is defined and implemented as a 4-layers protocol, as shown in Figure 3-16:

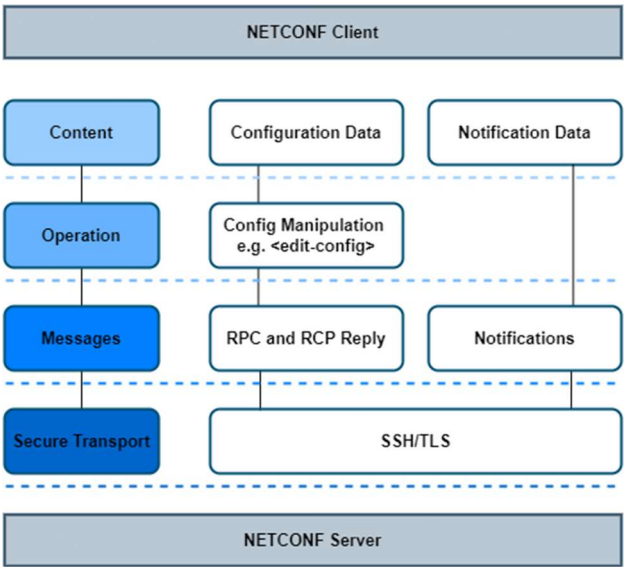

Figure 3-16: NETCONF Protocol stack

- **Secure Transport Layer:** it allows the establishment of secure connection (NETCONF Session) between NECONF client and server. As per RFCs, both SSH and TLS are supported as Secure Transport mechanisms. Even though, these protocols nowadays don't have key exchange algorithms that support Quantum cryptography, so it will be necessary to apply future versions that support this characteristic (e.g., pre-shared keys, quantum keys, …).
- **Messages Layer:** Encodes RPCs (invocation and results) and Notification. RPCs transports protocol messages related on protocol operations while notifications are special messages sent by device to inform the controller about an event (e.g., high temperature, high BER, etc).
- **Operation Layer:** implement the base set of protocol operation e.g., `<edit-config>`, `<get-config>`, `<close-session>`, etc
- **Content Layer:** contains config and notification data. Both XML and JSON are supported.

It is important to note that the NETCONF Client is the invoker of NETCONF operation: this means that from an SDN point of view, the NETCONF Driver (hence the SDN Controller) is the NETCONF client while the SDN agent of the controller device, is the NETCONF server. The NETCONF Client request the NETCONF session establishment to the NETCONF Server.

With the exception of the Secure Transport, the NETCONF standards allow the definition of custom data, operation and messages on the basis of proper specifications modelled using the YANG language (model-driven protocol).

This high-flexible mechanism is exploited to model abstractions of the controlled devices. The YANG language provides indeed a set of constructs to model:

- ***Custom RPCs***. This feature is used to model the control interface of a network device: the SDN agent exposes custom RPCs in addition to the base operations allowed by NETCONF
- ***Data.*** This allows to specify the custom configuration data models for the specific devices
- ***Notification.*** Custom notification message that can be triggered on events (even-driven)

The customizations are still part of the protocol so any NETCONF peer (including the SDN Controller) must be able to deal with the under the condition that the YANG model is known. To fulfil such a condition, NETCONF implements a special exchanging mechanism used during the creation of a NETCONF Session, so that each NETCONF peer can acquire information regarding the other. In the specific case of an SDN Controller and a network device, The SDN Controller establishes a NETCONF Session with the SDN Agent by sending the standard NETCONF opening message <*HELLO*>, that transport several information such as the protocol version supported, the capabilities, etc. The Agent will reply with another <*HELLO*> message specifying its own capabilities. At the end of the exchange each NETCONF peer knows the other and also its protocol extensions modelled in YANG. From the controller's point of view, the custom YANG model exchanged by the SDN agent represent the way to communicate with the agent itself and an abstraction

of the device, whose hardware complexity is completely hidden to the SDN controller. For the SDN Controller, the SDN agent is the device.

Transport API

One of the main issues to be faced in the SDN context is the heterogeneity of the NBI interfaces exposed by devices provided by different vendors, that is in turn reflected in the NBI interface of the SDN controllers. Despite the standardization of the protocols for the NBIs (e.g., RESTConf) the set of APIs exposed by the different device may significantly vary, forcing the consumers of such interfaces to build specific logic to deal with them.

Transport API (TAPI) is a project from the Open Networking Foundation aims to provide a standard interface across multiple domains (network segments) and vendor's devices, meeting at the same time the requirements to be both protocol and interface to be exposed by an SDN controller towards users, 3rd party application, other SDN controllers and orchestrators.

TAPI provide an abstraction of several control plane functions that includes the management of Connectivity, Topology, Path Computation and also OAM functions and Network virtualization. Such kind of abstractions are defined to be generic but include support to OTN, Optical Transport Networks and other technology-specific interfaces. All of such features are modelled as service offered by the TAPI in YANG, as depicted in Figure 3-17

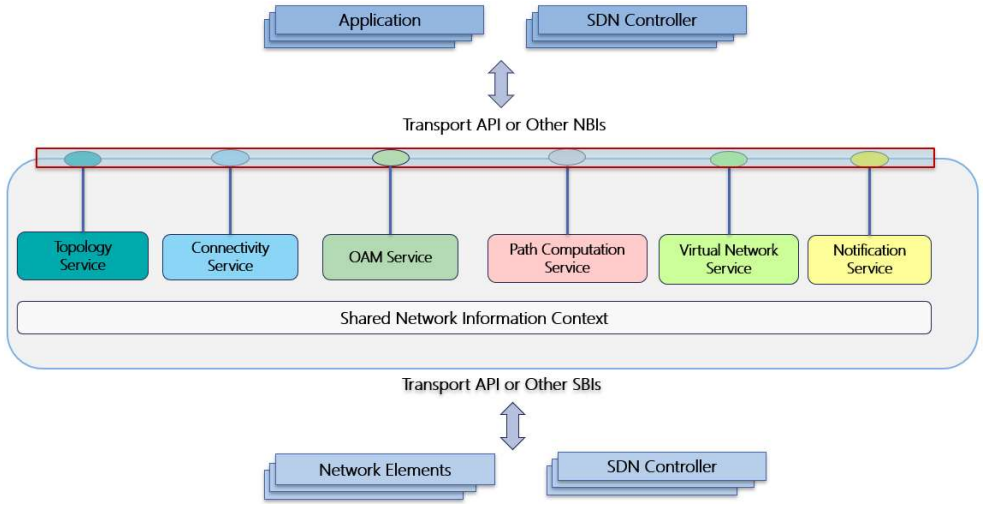

Figure 3-17: Transport API function architecture in terms of services (Yang modelled)

TAPI defines a complete information model that abstracts the peculiarity of the underlying network (or networks) and the set of services that can be provisioned on it. The **TAPI Context** is the main container of such an information that are shared between the *TAPI Server* (the SDN controller implementing the TAPI) and the *TAPI Client* (any of the SDN consumer: users, 3rd party application, high-level SDN Controllers).

TAPI allows TAPI clients to provisioning services through specific abstract interfaces called **SIP** (**Service Interface Points**) that represent the access points for requesting network services with the API. In particular, a TAPI client requests a (one or more) **TAPI Connectivity-Service** to the TAPI server, which is a request intent-like for connectivity between SIPs. Other than the SIPs, other parameters can be specified such as the throughput or path constraints (the path calculation is charge of the TAPI server that takes into account such constrains if specified). TAPI NBI uses RESTCONF as transport protocol and support CRUD HTTP operations.

The underling network is represented as a set of network resources in turn abstracted as a set of Node and Links. The former, is an abstract representation of the forwarding capabilities of a give network resource, the latter is the abstract representation of an adjacency between nodes in the topology. Each node includes a set of ports that belong to the associated network resource and called NEPs (Node-Edge-Point) which are also the termination points of the Links. SIPs are It is important to note that a Node can be itself an abstraction representing a set of Nodes so that allowing the existence of different level of topology abstraction.

4. GLOBAL ARCHITECTURE

This section provides a general view of the global architecture selected for DISCRETION, proving some details of the different domains, building blocks and of the interfaces between them.

DISCRETION incorporates four domains (Quantum, IP, Optical, and SDR), structured in a Hierarchical SDN topology, that is considering the red/black network isolation paradigm specific to the DISCRETION application scenarios.

The Core of a SDN architecture is the SDN Controller, which is the one in charge to manage the different SDN Domains in a multi-domain environment through the SDN Domain Controller NBI (in our approach, the different domains will be: IP, Optical, Quantum and SDR).

This module has the End to End (E2E) visibility of all network resources and transport networks segments and the ability to export an abstracted topology view of the network.

The internal hierarchical SDN Controller architecture can be saw as two conceptual blocks [46]:

- E2E Transport Network Control. Which provides E2E control by coordinating the different technologies through its corresponding SDN Domain controllers.
- E2E Transport Service Abstraction. Which provides the service interface towards client applications and its translation to network service configurations.

When the transport network is divided in a technological multi-domain, we will also need SDN Domain level controllers. This configuration matches in a better way with a hierarchical network structure, being the SDN Domain Level Controller the one that controls the network elements, communicating with them through its SBI. The different SDN Domain Level Controllers that we will use are:

- Optical SDN Controller. These controllers are the ones that provide optical network programmability and interoperability towards H-SDN (in a hierarchical layer structure) and between vendors through its NBIs. There are some solutions to obtain legacy compatibility with GMPLS devices, using control protocols as PCEP or BGP-LS, but there are also more projects that are trying to realize a standardization using the ONF Transport API (TAPI) and IETF models. Current commercial solutions are vendor specific, although in the medium term, the goal is to archive optical disaggregation, in order to obtain open management of the line system and devices, controlling every system (no matter the vendor) with a single optical SDN Controller. Some projects that are progressing in this field are OpenROADM and OpenConfig (for the transponders).

- IP SDN Controller. In this level, it seems achievable to use a single vendor agnostic controller to manage IP network elements, although it cannot be assumed that devices support natively the data models, being possible the implementation of a mediation layer to realize the translation. In IP layer, it is also possible to realize a subdivision due to scalability reasons, where each IP node is classified depending on the performed functions.
  This controller communicates with the hierarchical controller through its NBI, which implements RESTCONF protocol and uses YANG data models. This controller provides to the upper levels' information about: Device inventory, layered topology, LSPs provisioning and path computation, a device abstraction for network services and network state and performance.

- Quantum SDN Controller. In this controller there are an extra goal, and it's the management and allocation of the keys of each link. The connection between SD-QKD controller and the Quantum network interface is defined to be done by standard protocols specificized by ETSI documents. The SD-QKD controller is able to realize connections with their neighbours and with other controllers to create E2E points, and if it's possible, create this associations over quantum links. The adaptation of this paradigm should not imply any security risk to the QKD network or the trust over it.

- SDR SDN Controllers. The main goal of these controllers will be the possibility to realize wireless communications in tactical scenarios. For a mission, a SDR SDN controller would be deployed to implement the SDN that will support it. Multiple tactical SDNs may exist at the same time, hence the set of SDR SDN controllers can be considered a SDN Domain. In the scope of Discretion, a single tactical network (hence a single controller) will be assumed. The scenario for the deployment of SDR SDN controllers is defined in Section 9.

4.1. Federated Hierarchical SDN architecture

In this approach each federation will structure its own network using a hierarchical structure and each top-level controller will communicate with the others by Eastbound/Westbound interfaces and SLA mechanisms. The benefit of this architectural paradigm is the is that it permits to combine the scalability and adaptability of hierarchical architectures and the control that provides federated schemes to each federation over their own network. In the next figure we can see a representation of these communications.

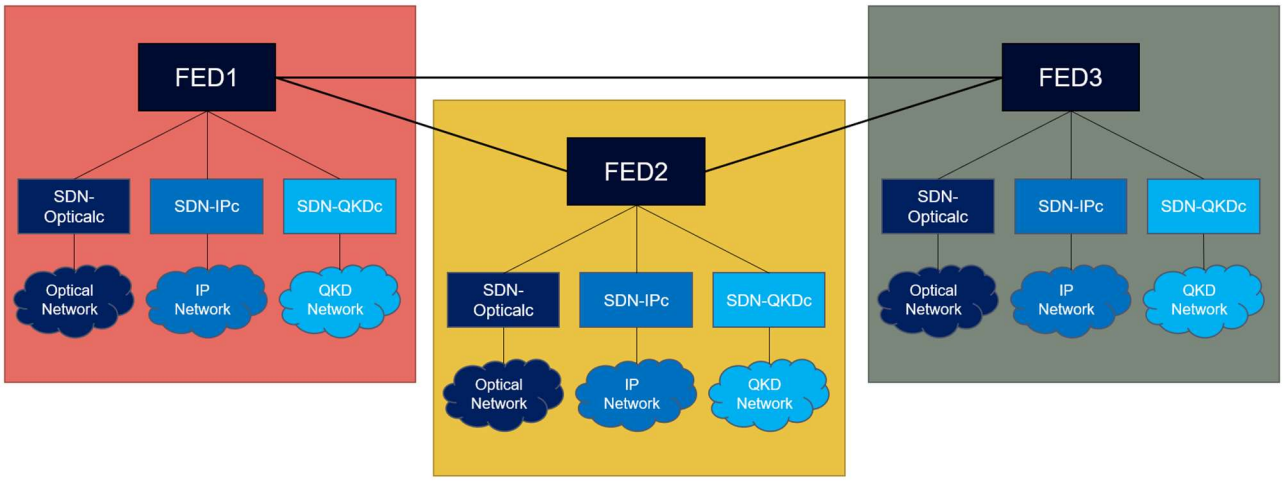

Figure 4-1. Federated Hierarchical SDN architecture

4.1.1. Hierarchical SDN architecture

As mentioned in the previous paragraph, each inner network will use a Hierarchical architecture, where the definition and management of the multidomain environment will depend on the Hierarchical Controller and the management of the different Domain networks will be delegated to the respective Domain Level Controllers, which will report to the Hierarchical Controller through its NBIs.

Section 6.2 of this deliverable presents a detailed analysis of the different SDN architectures, with their pros and cons, although it is worth to advance some of the reasons for the selection of the distributed hierarchical paradigm:

- This approach is more flexible and scalable, being two great characteristics when we are talking about military contexts.
- Other advantages of this approach are an easier control of each domain when it is provided by a specialized controller, a greater adaptability in multi-provider scenarios, more scalable solution, and a higher availability as we can count with several distributed Domain Level Controllers for the same domain.

In this architecture we will also keep in mind the Red/Black Isolation Requirements, so we will need to realize a secure information exchange when a Red SDN Controller communicates with a Black one, and also when it is done at a Data Layer level. As the main goal of this network-split is to protect the information domains, we will isolate and reduce the points of contact, communicating Red/Black SDN Controllers by just one module: ALTO. This communication will be asynchronized and should be only writable for one of the sides in order to reduce even more the communications between zones (this last thing can change with the appearance of a grey zone). This technology is not defined yet, so, the usage of this kind of mechanism to achieve the necessary isolation will be one major result to be produced for this project.

To provide isolation at a Data Layer level, we will count with Cypher Machines at the Red Network Border that encrypt all the outcoming information decrypt all the incoming. Cypher Machine definitions and a deeper explanation about Red/Black Network isolation was provided in Section 3.4.2.

Even though, Hierarchical architectures have a main problem when more than one organization is working together: the management and decisions is centralized so all the members lose the control over the network ("my network" turns into "our network"). In real scenarios where more than one company or country is working together, if no superior entity is accorded by all the parts, this control relinquish is at least very difficult to happen.

4.1.2. Federated SDN communications

A federated network scheme gives the possibility to maintain network sovereignty on each administrative domain (e.g. country), establishing communications according to SLAs previously agreed allowing the different members of a coalition to share information and network resources without losing control over their own networks.

A federation is based on the application of coherent policy enforcement of traffic flows in all the segments. To achieve this, nations must agree on high level policy rules, though bilateral o multiparty SLA (Service Level Agreement) mechanisms to implement it.

In this architecture, three main classes of traffic [2] may be distinguished:

1. Local traffic: Traffic generated on one entity that passes only between nodes of that entity. This traffic will always reside in a red network, meaning that the information will not go through black nodes or nodes from other entities.
2. Federation traffic: Traffic generated inside the federation network that passes between nodes of different entities of the federation but not though external to the federation nodes. The classification of this traffic depends on the level of trust agreed between the different entities, but it can be considered Grey traffic as it has to cross security boundaries but there are also a certain trust level.
3. Transit traffic: Traffic that passes across the federated network but also across external nodes. This type of traffic corresponds with a Red-Black communication traffic.

One of the main topics to agree on in a federation scheme is how much topology information is shared with other segments and the mechanisms used to exchange that information. ALTO protocol is a good candidate to expose inter-federation information, maintaining a tight control of the set of (minimal) information exchanged, but depending on the needs and the level of trust between organizations, information may be increased to enhance connectivity (e.g. in terms of performance, resiliency, or flexibility). The details of what information share and how to do it will be defined more deeply in the WP5.

This architecture has also some hurdles. First, all entities must agree on the policy rules to implement and trust the other entities without losing control over their own nodes when network resources are shared.

Despite this, the first contra will exist in every inter-domain communication and in the next steps of DISCRETION we will work to minimize the possible impact of the second one.

4.1.3. Previous exchange Environments

Internet Exchange mechanisms are a good example of federated architectures implementation. Some of relevant approaches used on IP environments are listed as follows:

- IPX. IPX is a network protocol that provides a communication structure for both local area networks (LANs) and wide area networks (WANs). It is a connectionless packet-based protocol and supports the routing protocols RIP and SAP what allows it to route packages between different networks (even different type of networks as Ethernet, Token Ring, and FDDI). Even this network protocol has been large replaced by TCP/IP, it's very helpful to connect multiple, separate networks together into a larger, federated network.

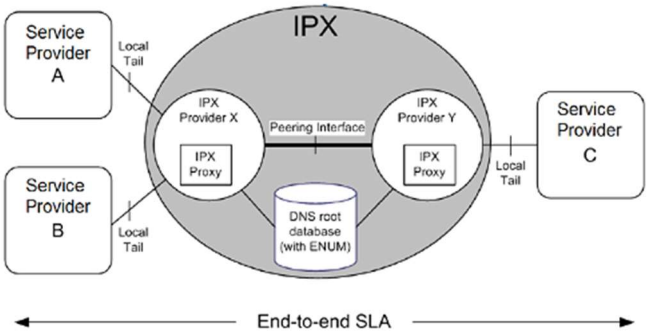

Figure 4-2. IPX SLA model.

- AMX-IX. A distributed exchange currently available in several different independent colocation facilities throughout Amsterdam. The AMS-IX infrastructure can be connected to at least one access device at each site. The MPLS/VPLS infrastructure is used in the present AMS-IX peering platform architecture. While maintaining the common shared Layer 2 Ethernet platform as the interface to the members and customers, this configuration enables the durable and highly scalable infrastructure inherent to MPLS. This can be considered as one variation of the IPX paradigm. [3]

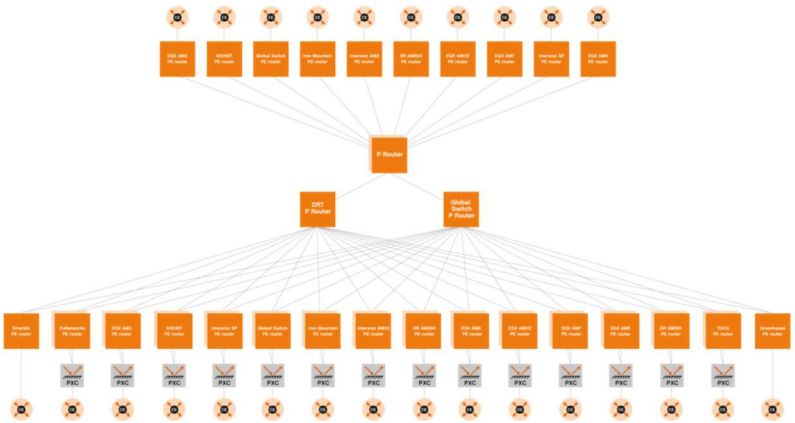

Figure 4-3. AMS-IX MPLS/VPLS platform for Layer 1. Source: [3]

- SDX[4]. On the same line, another application of the classical IPX is the SDX or SDN exchange points. This approach intends to deploy SDN-capable switches at IPX. The participating IPs can execute a wide range of network applications on their packets passing through the IXP because SDN enables direct expression of flexible policies. Examples include inbound traffic engineering, traffic redirection to middleboxes, wide-area server load balancing, and blocking of unwanted traffic. The key characteristics of SDX include giving each operator a virtual topology, ensuring that participants' policies don't conflict or interfere with one another, and scalability. [47]
- CDNi. Federations have been envisioned for the content delivery industry, namely for CDNs. The concept behind CDNi and its various forms is that CDNs combine their infrastructure and services to aggregate content and users or lease infrastructure and presence at specific geographic regions strictly on-demand, as suggested by vendors like Cisco [48], standardization bodies like IETF [49], and research projects [50]. Additionally, CDNi provides

[3] https://www.ams-ix.net/ams/documentation/ams-ix-topology
[4] https://noise-lab.net/projects/software-defined-networking/sdx/

intriguing use cases and concepts for member engagement, such as the bilateral agreement and the exchange model.

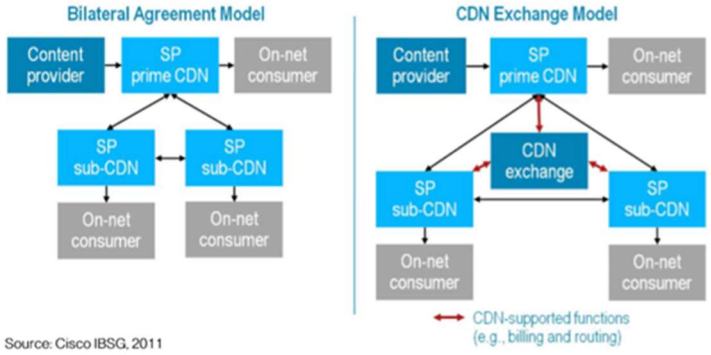

Figure 4-4. CDNi Federation coordination models: Bilateral (left) and exchange-based (right).

4.1.4. Consensus algorithms for Federated Networks

In general, we can find a wide range of methods to reach consensus. Some of these methods are based on traditional consensus algorithms ad in newer technologies as Blockchain or machine learning:

- Federated voting: Using this technique, each network node votes on a proposed course of action. The action can only be carried out with a majority of votes.
- Federated Byzantine Agreement (FBA) was created for use in federated networks. To come to an agreement, it combines consensus messages and digital signatures.
- Federated Proof of Stake (FPoS): This consensus method allows network nodes to "stake" their tokens to improve their chances of being chosen to approve a transaction.
- Federated Proof of Work (FPoW): This consensus method selects network nodes to validate transactions based on their computing capacity.
- Federated Multi-Party Computation (FMPC): This technique encrypts data and distributes it among several parties, who then work on the data without having access to it. This enables them to agree on the computation's findings without endangering the data's privacy.
- Federated Machine Learning (FML): This is a technique that allows several parties to share their data and work together on a machine learning model without disclosing any of their individual data. On the basis of its own data, each party develops a local model, which are then combined to enhance performance.

In further stages of the project, we will decide which approach have the best fit on the purpose of DISCRETION.

4.2. DISCRETION Building Blocks

Once we have defined the network architecture, let's see how it would be. In Figure 4-5, we have a representation of the main blocks defined. In grey colour we have the modules that conforms the Black Network, having the hierarchical scheme defined before and some special modules: ALTO and Topology DB, both modules will be necessary for communications with the Red zone. The other main blocks represented are the four SDN Domain Level Controllers that realize the management tasks for each of the four Domains treated in DISCRETION (i.e., Quantum, IP, Optical and Radio). Also, they exchange information with the Hierarchical Controller to provide to it a more completed and actualized vision of the network situation.

On the other side, we have an abstraction of a Red Module, where the main blocks are the two SDN components (SDN App and SDN Controller) and there are also modules destinated to provide security: a Red Network Security Application, a barrier cypher module and a QKD Node. These modules will manage the security aspects about the communication between zones at a data layer level.

The Red zones are interconnected across the black network segments, using Cypher machines to encrypt/decrypt the information in each frontier.

There are five different QKD related channels: Quantum link, QKD Synchronization Channel, QKD Management channel, QKD Distillation Channel, and Key Synchronization Interface. Only the quantum link is quantum. The others are classical. The first two channels will go unencrypted due to its characteristics (more information in Section 8) but the other three will need to pass this security barrier to provided security to the communication. The different network decisions in this level will be adopted by the QKD nodes, having each of them a QKD-SDN agent and a QKD KMS module.

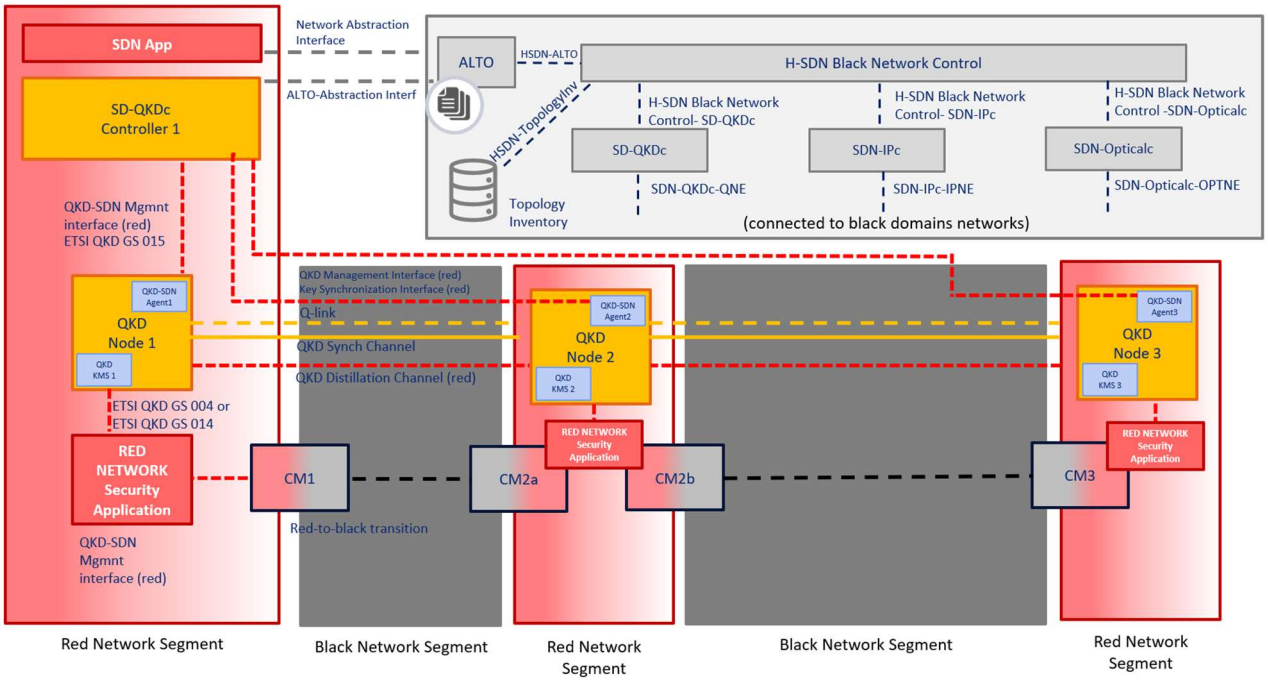

Figure 4-5. Block Diagram for SDN domain interconnections.

SDR SDN is not shown in this architecture because, for the moment, it is not included as a domain in the SDN multidomain hierarchy, since the scenarios currently under consideration for the SDR SDNdo not justify this approach. On Section 0, a basic architecture is unveiled for the SDR SDN, which already assumes the existence of this federated hierarchical SDN to provide a secure mechanism for key distribution (QKD), as well as connectivity between tactically relevant sites

Current definition of the architecture must be considered preliminary, part of the ongoing work within the project, and therefore partial, incomplete, and subject to changes that may occur as a result of the different analysis being carried out in the project.

Initial effort has focused on defining the SDN building blocks of the individual network planes considered in DISCRETION, leaving the hierarchical orchestration details for the next phase. Thus, those components are not described in this document, with the exception of the network exposure function and the red/black isolation modules, the necessity of which is drafted in the following paragraphs.

4.2.1. Network Capabilities Exposure module (ALTO)

Selecting the most appropriate network resources is a complex task and sometimes topology information is not enough to obtain a complete view of the network. There are some nodes that might have useful services for some routes and in other cases could be requirements that can be only provided temporally by some nodes. The Application-Layer Traffic Optimization (ALTO) service provides on-demand enriched

network information with additional cost metrics that can help to improve network performance by, for example, adapting to network resource consumption patterns. In other words, the main goal is to allow other SDN controllers to have a more accurate and dynamic view of the network to predict the best network resources to use (e.g. routes) with a higher accuracy and to adapt when context changes [39]

One of the advantages of using this module in DISCRETION is the possibility to share some characteristics that would be very attractive in a military context, e.g., nodes with high security mechanisms or with URLLC services. Also, the red network would have more information about the black one, helping them to take the best decisions depending on how critical the information is.

The data will be exchange over HTTPS using JSON objects as it's defined in [51]. In case the communication includes a Red SDC Controller, this will be realized with two one-directional channels in order to create an isolation between zones (this would be a physical implementation, so it will be transparent to the module).

The functionalities that will be provided by this module are:

- Update existing information of traffic routes.
- Provide application-level network-related information of the endpoints.

Provide a TLS secured connection to exchange network context information.

4.2.2. Red-Black Network Isolation modules

One alternative to provide isolation between red and black network management planes (i.e., SDN applications to orchestrate service between service (red) layer and black optical, QKD and data layers is to add a data diode to create a boundary between trusted (red) and untrusted (black) management networks.

Black network can expose topology characteristics that the red network can access to optimize its performance over the black network and provide a limited cross-layer optimization based on the information that the red network can access related to black network characteristics. As such, data diode mission is to protect the destination network. The data diode will be deployed at the security edge of the destination (red) management network. Data from the black network may be transferred into the red network, but no information will be allowed to flow from red to black management systems or leak back though the one-way path.

In case of a use case also requiring data to be provided to the black network (e.g., configuration commands) a data diode may be also deployed in the in the red to black network flow. In this case, data from the red network (source) remains available to the black network (destination), but red network itself is inaccessible to external attacks, with the objective of virtually eliminating (in practice, severely limiting) any available attack surface for the red network.

Also, for use cases requiring bidirectional comms, two separate data diodes over authenticated nodes, over specific IPs and restricted paths could be used, implementing in practice two separate one-way data transfer mechanisms.

Communication between black and red network must be also encrypted, augmenting the inherent security implemented by network segmentation with the privacy of the encryption mechanisms.

Once the required information is one-way safely transfer though a network diode, it may be safely accessed at both ends without compromising the integrity of the red network management system.

4.3. DISCRETION Interfaces

As it is shown in the Figure before, the SDN Controller will count with some important interfaces. The main ones will be those that connects the Hierarchical Controller with Domain Level Controllers and these with the Domain Networks. However, we will also count with other interfaces that are not standardized yet and which we will need to define from almost zero. All these interfaces are:

1. **H-SDN Black Network Control- SD-QKDc**: NorthBound interface that connects the Hierarchical SDN Controller with the QKD SDN Controller. The main purpose is to obtain an abstraction of the network elements and services information. The protocol used for the communications is RESTCONF and events defined by the YANG data models using ETSI GS QKD 018 Standard [52].

2. **SDN-QKDc-QNE**: SouthBound Interfaces that connects the QKD SDN Controller with the Quantum Data Layer Network Elements (QNE). Then main purpose is to realize and disaggregated management of the QNEs. The protocol used for the configuration of the QNEs is NETCONF events defined by the YANG data models using ETSI GS QKD 015 Standard [53].
3. **H-SDN Black Network Control- SDN-IPc**: NorthBound interface that connects the Hierarchical SDN Controller with the IP SDN Controller. The main purpose is to obtain an abstraction of the quantum nodes and services information. The protocols and interfaces used are defined in MUST IP SDN Controller NBI Technical Requirements [54].
4. **HSDN-TopologyInv**. This is the interface that connects the general Topology Database with the hierarchical SDN controller. Tis interface is also ruled by the [54] standard.
5. **SDN-IPc-IPNE**: SouthBound Interfaces that connects the IP SDN Controller with the IP Data Layer Network Elements (INE). Then main purpose is to realize and disaggregated management of the INEs. The protocols and interfaces used are defined in MUST IP – SDN Controller SBI / Router NBI Technical Requirements [55]
6. **H-SDN Black Network Control -SDN-Opticalc**: Northbound Interface that connects the Hierarchical SDN Controller with the Optical SDN Controller. The main purpose is to obtain an abstraction of the optical network elements and services information The protocols and interfaces used are defined in MUST Optical SDN Controller NBI Technical Requirements OOPT [56].
7. **SDN-Opticalc-OPTNE**: Southbound Interface that connects the Optical SDN Controller with the Optical Network Elements (OPTNE). Then main purpose is to realize and disaggregated management of the OPTNEs. The protocol used for the configuration is gRPC [57]. This is usually a proprietary interface due to the specific optimizations (i.e., physical impairment compensations across the whole optical system) that every vendor implements for their specific products in optical networks.
8. **Network Abstraction Interfaces.** East-west interfaces for network exposure capabilities and network programming from red to black network. No standard defined. Currently under investigation within the project.
9. **Key Synchronization Channel (red):** KMS instances on different nodes use this channel to synchronize data. This channel uses a common TCP/IP based communication. The communication follows application specific proprietary protocols. The data send includes forwarding and establishing keys and keeping databases as well as processes synchronized. Depending on the type of data sent the KMS will encrypt and authenticate this data beyond the encryption and authentication the CM might offer.
10. **Q-link**: The quantum link transports the quantum states used as key material to generate the secret keys. After preparing the quantum states, the transmitting node sends this information to the receiver through this link. This interface connects the physical layer of the QKD modules to the quantum communication channel.
11. **QKD Synch Channel**: This channel transports a reference signal that synchronizes the physical layer of the QKD nodes at the hardware level. The timing reference (or clock) used at the transmitter and receiver modules is derived from this channel.
12. **QKD Distillation Channel**: Authenticated classical channel used by the QKD nodes to exchange information resulting from the post-processing of the secret keys. The exchanged information depends on how the different stages of the CV-QKD protocol are implemented.
13. **QKD-SDN Management Interface**: This interface is dedicated to managing and coordinating the control events in the node with the network control.
14. **Key Provision Interface:** This interface is located between the KMS and any Security Application (SA) on the same node. It exists to provide the Keys from the KMS to the SA upon request by the SA. The Interface follows one of the QKD key provision standards by ETSI, which is either ETSI 004 [58] or ETSI 014 [59].
15. **Cipher Machine Interface**: This interface is located between the Security Application (SA) and the Cipher Machine (CM). It is mainly used to push the keys to be used by the CM, though other control and management data is also transmitted.

5. INTERFACES SECURITY ANALYSIS

There is a wide range of interfaces being studied within the scope of DISCRETION, ranging from local management interfaces for QKD nodes to remote key provisioning interfaces for security applications or even SDR interfaces. However, aligned with the scope of this document, this section will focus on the SDN interfaces pertinent to the architecture discussed throughout, except for subsection 5.2.2 where an exception is made for exterior-facing QKD interfaces.

One of the big challenges when it comes to SDN interfaces is standardization. Within this project, added emphasis was put into adhering to existing standards whenever possible and suitable, taking as much advantage as possible of the pre-existing body of knowledge regarding the chosen solutions. Regarding security, this pre-existing work translates into a more robust final product, reached in a more efficient manner. Nonetheless, due to the disruptive nature of the goals of this project, some bespoke solutions and adaptations are to be expected.

Two important base assumptions to consider for this section are that any interfaces exposed within the Red network are within a controlled environment, meaning there is a much lower risk of physical compromise, and also that any privileged user with management access is operating under good faith. Additional security mechanisms can be added to minimize the risks represented in these assumptions, but that is considered beyond the scope of DISCRETION.

A security analysis of the current state of the DISCRETION interfaces will be performed in order to identify and address the relevant security considerations in preparation for the design and implementation stages of the project.

1. **Network Abstraction Interface.**
    - **Description:** East-west red to black interface for network exposure capabilities and network programming from red to black network. No standard defined. Currently under investigation within the project.
    - **Type:** Bidirectional
    - **Protocol:** RESTCONF (HTTPS requests over TCP/IP).
    - **Transported Info** Bidirectional. The red network receives a json with the editable configuration of the black network and the model defined on a YANG data model (also available) and sends it back with the modification needed (on the same format) via a PUT or POST message. The information available to be modified is currently under investigation within the project but it should include the ALTO maps available to be acceded by the ALTO-Abstraction Interface. The Black network must not know any information of the Red network: the programmability must be unidirectional even the communication is bidirectional.
    - **Criticity Level:** High
    - **Security Assessment:** This interface represents a breach within the Red/Black separation model. However, it is considered necessary in order to leverage the programmability attributes of the black SDN. As such, additional security assurances will be necessary.
    - **Protection Measures to be adopted:** Security measures to ensure an adequate level of protection are still being studied but some examples are Authentication of both ends and encryption of the data. As the communications are over https, it's possible the use of user IDs or Sessions tokens (with the correspondent security measures) plus the use of Digital Certs. The data communicated through this whitelist shall also be very strictly monitored and whitelisted.
2. **ALTO-Abstraction Interface**:
    - **Description:**
      East-west red to black interfaces for network exposure capabilities No standard defined. Currently under investigation within the project.
    - **Type:** Unidirectional
    - **Protocol:** ALTO over UDP. In order to avoid the possibility of a bidirectional communication, the information will be exposed using a Pub/Sub event transmission

platform, e.g. Apache Kafka [60] (Message-oriented middleware can be other option [61]). The information will be distributed on queues available by UDP requests.

- **Transported Info:** Edge-nodes properties from the Optical and IP topology. The black network will send the maps requested through the Network Abstraction Interface, removing from the maps the unneeded nodes (e.g. routers or network devices that communicate other network zones and do not connect CDNs or users networks) and sending the IDs codified, to make almost impossible the identification of individual nodes. The information to send nowadays is the number of jumps between nodes, but it is currently under investigation within the project the possibility of exposure topology information related with the Data Privacy Enforcement.
- **Criticity Level:** Medium
- **Protection Measures:** Data integrity and authentication (e.g. hashes). The access information to the topic's queues can be managed through the Network Abstraction interface, using this just as the mailbox where the black network asynchronously upgrades the information. As the nodes are already codded at the application level, the use of encryption can be studied in future steps. The information communicated through this interface shall be very strictly monitored and whitelisted. **Security Assessment:** This interface represents a breach within the Red/Black separation model. However, it is considered necessary in order to leverage the programmability attributes of the black SDN. As such, additional security assurances will be necessary.

3. **Key Synchronization Channel (red):**
   - **Description:**
     KMS instances on different nodes use this channel to synchronize data. This channel uses CoAP as communication protocol which is based on UDP/IP. The communication follows application specific proprietary protocols. The data send includes meta-data for key synchronization and keeping databases as well as processes synchronized. The channel is also used to perform QKDN key forwarding tasks. Depending on the type of data sent the KMS will encrypt and authenticate this data beyond the encryption and authentication the CM might offer.
   - **Type**: Bidirectional
   - **Protocol:** Proprietary protocol on top of CoAP as base communication protocol
   - **Transported Info:** KMS synchronization protocol and QKDN key forwarding.
   - **Criticity Level:** High
   - **Protection Measures:** This interface uses its own layer of encryption, with different implementations for information of different criticality, providing confidentiality and authenticity. Specifically, for QKDN key forwarding, keys are required to be encrypted whereas for all other synchronization messages exchanged by the protocol, authenticity of the exchanged information is sufficient.
   - **Security Assessment:** The information transmitted through these interfaces, which will often be key material, is of the highest criticality. As such, this interface does not delegate confidentiality and integrity assurances to other components but rather implements an additional security layer on top of the transport layer encryption and authentication ensured by the Red/Black separation model and CM.

4. **Q-link**:
   - **Description:**
     The quantum link transports the quantum states used as key material to generate the secret keys. After preparing the quantum states, the transmitting node sends this information to the receiver through this link. This interface connects the physical layer of the QKD modules to the quantum communication channel.
   - **Type**: Unidirectional
   - **Protocol**: CV-QKD physical layer.
   - **Transported Info:** Quantum states with the key material.
   - **Criticity Level:** High.
   - **Protection Measures:** The CV-QKD protocol is designed to ensure the security of this interface. In an attack scenario, an eavesdropper cannot gain information without being detected.

- **Security Assessment:** The incorrect design or implementation of the CV-QKD protocol may allow the eavesdropper to obtain information about the secret keys.

5. **QKD Synch Channel**:
   - **Description:**
     This channel transports a reference signal that synchronizes the physical layer of the QKD nodes at the hardware level. The timing reference (or clock) used at the transmitter and receiver modules is derived from this channel.
   - **Transported Info:** Clock signal, timing reference.
   - **Criticity Level:** Low.
   - **Protection Measures:** None.
   - **Security Assessment:** This interface should pose no confidentiality or integrity risks, but the disruption of the sync channel might compromise availability through denial of service.

6. **QKD Distillation Channel**:
   - **Description:** Authenticated classical channel used by the QKD nodes to exchange information resulting from the post-processing of the secret keys. The exchanged information depends on how the different stages of the CV-QKD protocol are implemented
   - **Transported Info:** Classical information resulting from the post-processing of the raw keys.
   - **Criticity Level:** High
   - **Protection Measures:** The CV-QKD protocol only requires that the distillation channel be authenticated; however, to ensure the proper red-black separation in the DISCRETION architecture, this channel should be encrypted by the CM.
   - **Security Assessment:** An authentication failure of the distillation channel may allow an attacker to manipulate the post-processing algorithms, thus breaking the security of the CV-QKD protocol.

5.1. SDN interfaces

5.1.1. Southbound interfaces

Southbound interfaces (SBIs) connect the forwarding plane (e.g. networking elements) to the control plane (SDNc). By separating the forwarding plane from the control plane, SDN introduces a new border that needs to be secured. For instance, this new channel makes the SDNc a very valuable target for denial-of-service attacks, with varying degrees of impact based on architecture specificities. A detailed look at architectural mitigations for this threat is presented in the next section.

Another important aspect that concerns SBIs is topology discovery, which happens via link discovery and host tracking. This topology information is then used to better manage network resources and adjust paths as required. It is, however, possible to poison this process, exploiting the lack of security mechanisms in protocols like LLDP (Link Layer Discovery Protocol), leveraging the injected malicious node for Man-in-the-Middle attacks, denial-of-service and other threats. Thankfully, there is already a wide range of mitigations and security mechanisms that can be applied here.

Some of the existing work for mitigating the identified threats includes SGuard [60] which leverages a data plane level cache, a custom ACL and a machine learning powered traffic classification module to improve the network's resiliency against denial-of-service or resource exhaustion attacks; and DAISY [61] which performs threat detection via statistical analysis and applies adequate mitigation. Additionally, regarding topology poisoning attacks, the idea is to avoid using OpenFlow and instead focus on leveraging NETCONF with YANG to help defend against these types of attack [62].

5.1.2. Northbound interfaces

Northbound interfaces (NBIs) connect the control plane (SDNc) to the application layer's applications and services. By exposing an API from such a critical component as the control layer for application consumption, there is an incurred risk of misuse or even malicious exploitation of said API. Even when ensuring the usage of trusted applications, there is still a risk of application compromise or even application impersonation. There are a few solutions that attempt to mitigate these risks. One of them is Indago [63]

which leverages machine learning and behaviour graphs to detect malicious activity and deploy appropriate countermeasures. Another alternative is Shield [64] which employs control-flow graphs to analyse malicious activity.

Another problem that might arise is having different applications competing for resources, intentionally or unintentionally disrupting each other. A few approaches to mitigate this problem exist. One such approach is SAIDE [65], which is based on the usage of mathematical models for reducing inter-app interference.

One of the paths to tackle the previously identified issues is with a more robust access control implementation. This can be done using mechanisms such as Role-Based Access Control (RBAC) or Mandatory Access Control (MAC). One of the different solutions to tackle this challenge are MD-UCON [66] which is a multidomain approach to UCON that seeks to offer RBAC for cross-domain access control. Another solution could come from the AEGIS framework [67] which uses real time API monitoring and hooking in order to review input/output against existing AC rules.

It is important to ensure that the security mechanisms used are robust and do not introduce new attack vectors, namely denial-of-service attacks.

5.1.3. Westbound/Eastbound interfaces

Westbound interfaces and eastbound interfaces define inter-controller communication. To the east, distributed SNDcs communicate with other distributed SDNcs in a different administrative domain. To the west, the SDNcs communicate with traditional networks.

A single controller architecture is very vulnerable to denial-of-service attacks, as it presents a singular point of failure. This can be fixed with a distributed controller architecture, using a wide array of solutions. These options within the context of DISCRETION will be expanded upon in the next subsection.

5.2. Interfaces within DISCRETION

5.2.1. Red/Black separation

One important aspect of DISCRETION is the Red/Black architecture, which places heavy restraints on information sharing between Red and Black domains. However, in order to leverage some of the benefits of SDN, it might be very valuable to be able to feed some black network information to the SDNc inside the Red network. For this, a specialized interface and communication protocol might be required, using, for example, a publish subscribe structure where the middleware sits in a Gray area of sorts, under the Red network's control and where data sanitization can be performed.

5.2.2. QKD interfaces and trust

In DISCRETION there is an additional set of interfaces which is of extra importance, and those are the QKD interfaces. These interfaces can, depending on the architecture adopted, establish the boundary between the internal, secured network and the external, unsecured network directly, without going through the cipher machine. In total, three different channels exist here for quantum key distribution. An analogue quantum link channel, a classical quantum sync channel and a classical quantum distillation channel. Even though the protocols operating in these three channels are theoretically secure, with no loss of security derived from eavesdropping threats, it is still paramount to ensure the necessary robustness against denial-of-service threats and, more importantly, any kind of injection threats that could be used to compromise a border node and gain access to the Red network. For these, particularly this last set of threats, the identified outwards-facing interfaces alongside the cipher machine's external interface need to be considered the critical points for border security and the assurance of the necessary isolation.

6. ROBUSTNESS, RESILIENCY AND RELIABILITY

6.1. Problem statement

The SDN architecture provides multiple advantages for network management and control, namely: global topology view and network awareness, operation flexibility, reduced total cost of ownership (COTS hardware), simpler data plane (hence, more efficient), easier and more consolidated management of the network equipment. SDN is thus the enabler for a lot of new use cases in the context of cyber-security and network resilience (faster reaction and global awareness to network topology changes and failures. However, simplest design of SDN (and in multiple practical implementations) the control plane only uses a single controller node for a given network, as shown in Figure 6-1.

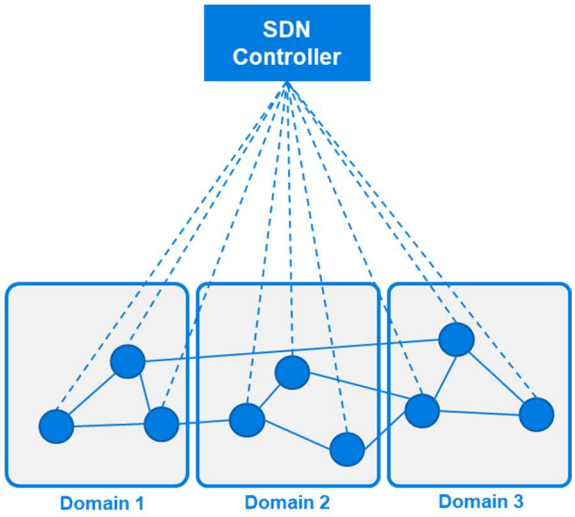

Figure 6-1 - SDN network controlled by a single controller node

A single controller may strike as an obvious *Single-Point-of-Failure* for the Control Plane. Nevertheless, this logical centralization does not mean that the controller is running as a single instance, on a single location.

Even if this makes a centralized view over the global network topology possible, it also introduces several limitations to the architecture. Again, due to the increase of interconnected devices and day-to-day network demand, using a single controller instance introduces a high bottleneck in terms of performance at the controller node, i.e. the number of events the controller can handle in a short time frame. Even in the "classical" case of OpenFlow, not used in DISCRETION, with proactive flow rule instantiation, having all the masterships of the network equipment associated to the same controller node also affects the control plane performance: all messages generated for flow table synchronization or counter updates end up reaching the same node. In addition, the centralized controller maybe significantly far from controlled devices, making the control unstable and decreasing the overall network performance (high control network latency).

Moreover, the crucial factor limiting the usage of a single controller architecture is the fact that, when in a single instance deployment, it represents a single point of failure on the overall SDN network. A controller failure will cause the disconnection between the controller and the equipment under control, and this may end up causing the disruption of connectivity between network hosts (or at least, the incapability to monitor and change the network behavior as is the case of in DISCRETION, where SDN is used mainly for configuration and monitoring). The likelihood of a failure on a controller is somehow high if we consider that in addition to possible software failures in the controller software, the commodity hardware where it runs may also be subject to hardware failures.

Finally, in today's generation of SDN controllers (e.g. ONF's ONOS), software architectures are somehow monolithic with a high degree of inter-dependency between the controller core services. This increases the likelihood that a hypothetical failure in a specific controller module may affect other core modules causing a cascading effect to the point that the whole control plane is brought down.

6.2. SDN Distributed Architectures

Due to the limitations of single-controller-based SDN architectures, a lot of effort has been placed into redesigning the architecture to consider either domain segregation (and delegation of control responsibility to individual control planes) or clustered control planes made of redundant controller nodes.

Distinct topologies for SDN-based networks exist, based on the segregation of the control plane per network domain in opposition to the classical centralized architecture of Figure 6-1. These illustrate how a control plane failure may ultimately be restricted to a single isolated domain. The description of the said architectures is presented below [68]:

Hierarchical: SDN-based architecture composed by several layers of controllers. Most implementations present a two-level tree consisting of local controllers and one "root controller". The local controllers handle local operations (e.g. intra-domain routing) while the root controller has a global control (and global topology view) over all the local domains. While this approach improves the overall architecture scalability, it only partially solves the robustness problem: the root control is still a centralized entity (and a single point of failure). Figure 6-2 illustrates this distribution.

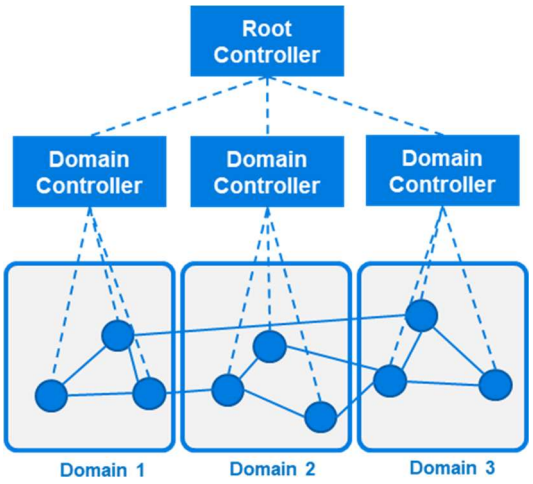

Figure 6-2 - Hierarchical Distribution

Distributed Logically Centralized: Contrarily to the hierarchical architecture where local controllers only control a single domain topology, in this architecture there is one controller per domain but managing both intra and inter-domain operations. Each time a controller creates or updates a network resource, it broadcasts this modification to all other controllers. Thus, all controllers maintain an up-to-date copy of the global knowledge, behaving just like a single logical entity. Figure 6-3 illustrates this distribution.

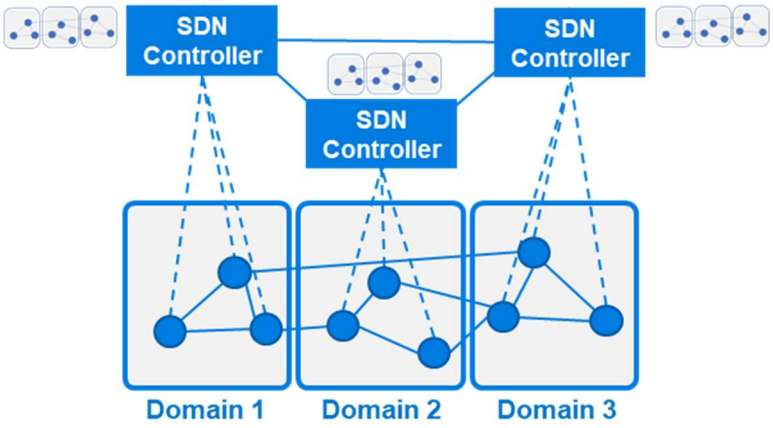

Figure 6-3 - Distributed Logically Centralized Distribution

Fully Distributed (or logically distributed): This architecture is similar to the previous architecture with the exception that network creations and/or modifications are not transferred to other controllers. Local-specific data remains in the instance where it has been created and is shared with the other instances only when needed. In such architectures, controllers only exchange information to set up inter-domain-services. Controller failures in these architectures, only impact a single domain. Figure 6-4 illustrates this distribution.

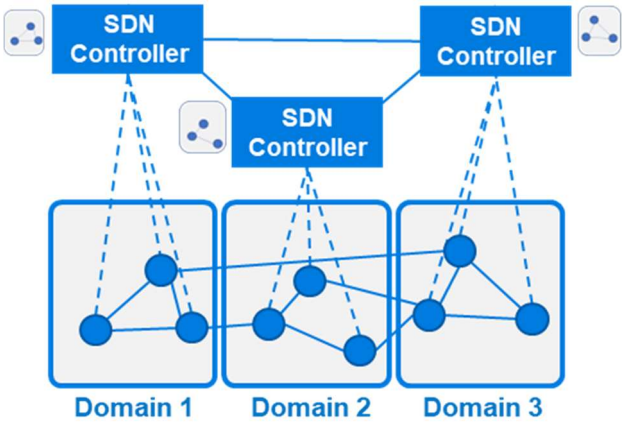

Figure 6-4 - Distributed Logically Distributed Distribution

Hybrid: Two-layer architecture that mixes both the distributed and hierarchical architectures. The control plane consists of several controllers (seen as root controllers) at the top layer in which each root controller manages multiple local controllers in charge of a specific domain. These root controllers are also organized in a distributed manner, distributing global network state information among them. Figure 6-5 illustrates this distribution.

The goal of all these distributed architectures is to ensure the creation of a federation of control domains while ensuring that possible failures stay as localized as possible.

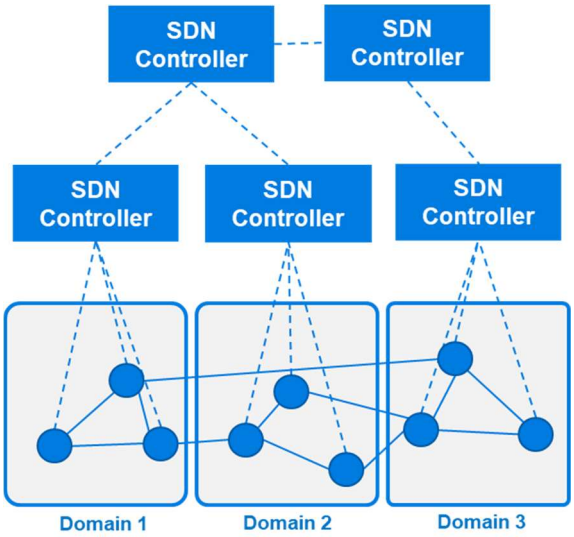

Figure 6-5 - Hybrid Distribution

When considering a distributed SDN architecture, the following aspects shall be taken into consideration [69]:

- **Controller placement** – Placing multiple controllers in a distributed architecture is an effective method to tackle the challenge of scalability. Current state-of-the art in the scope of multi-controller placement consider network parameters such as the delay, traffic and distance to identify the number and optimal location of the controllers in the network [70]. Controller placement algorithmic solutions usually consider game models ([70], [71]), optimization mathematical models ([72], [73], [74]) or hybrid controller role distribution.
- **Domain partitioning** – In multi-controller architectures, scalability can also be improved through the division of a network into multiple domains. Examples of work in this field include [75] where they abstracted domains as nodes and formulated a multi-domain partition problem as a Greedy Sub-Graph Cover Problem (GSGCP). In [76], on the other hand, the matrix perturbation theory was used to determine the topology of domains and its optimal number by using a test framework based on the Beacon SDN controller.
- **Controller consistency** – Multi-controller architectures must make decisions based on consistent and coherent information belonging to different domains. Furthermore, they need to interact with each-other to ensure a consistent view over the network. Approaches to data partitioning and state consistency include the publish-subscribe model [77], network information base (NIB) [78], fast consensus algorithms [79], cross-domain control state snapshots [80] or load variance-based synchronization (LVS) [81]. In what concerns control strategy consistency, AMQP was proposed and used for neighbour control discovery and to establish a publish-subscribe channel by [82]. [83] and [84] propose customizable consistency generators.
- **Controller reliability** – In the unfortunate event of a node failure, multi-controller SDN architectures should be able to be map or migrate instances to another node. On an SDN-based architecture, a stateful controller is responsible for traffic management within a single domain and therefore cannot be easily transferred. As such, research on the field of controller node reliability and controller placement in respect to controlled network equipment is required. Work done on this field include [85] who has designed a K-critical algorithm for controller node reliability together with optimal location placement and load balance between controllers. [86] and [87] formulated mathematical models to reduce switch-to-controller latency (with optimal controller placement) although only feasible in small to medium control networks.
- **Controller clustering** – Clustering in multi-domain SDN architectures is proposed as a means to achieve load-balancing. Clustering approaches are transversal to the employed global SDN architecture, proposals including both hierarchical-based architectures or fully distributed architectures. BalanceFlow [88] is an example of a clustering implementation based on a

hierarchical deployment. Other references include cooperative load balance [89], cluster vectors for controller dependability reduction [90] or dormant mechanism models [91].

- **Switch migration** – The main idea of switch migration is to dynamically change the relationships between switches and controllers by migrating the network equipment control from the overloaded controller node to another controller which does not have substantial load. Examples of work on this field is the Elasticon switch migration framework [92], the game-theoretic approach applied to switch migration proposed by [93], the BalCon balanced controller [94] which considers the communication patterns of SDN switches and the load balancing mechanisms described by [95].

6.3. Clustering and distributed architectures in production-grade SDN controllers

Despite the work that has taken place in academia regarding SDN controller robustness and resilience, implementations usually target SDN controllers that were either not designed for production environments or are not actively maintained (e.g. DISCO [82], Ryu or Beacon). Furthermore, only a small minority of SDN controller software is in fact designed and developed for distributed implementation scenarios. When reviewing the state of the art in the scope of SDN controller resilience and robustness, it is of uttermost importance to analyze the two main open-source distributed SDN controllers for production environments, both members of the Linux Foundation[5]:

1. ONOS (developed by the Open Networking Foundation) and
2. OpenDaylight (ODL).

In a later version of this document, this may be complemented with the analysis of other options, like TeraFlowSDN, provided that they are eligible as distributed SDN controllers.

6.3.1. Clustering

In both ONOS and ODL, strong consistent models for clustering are employed using the Raft consensus algorithm (Figure 6-6).

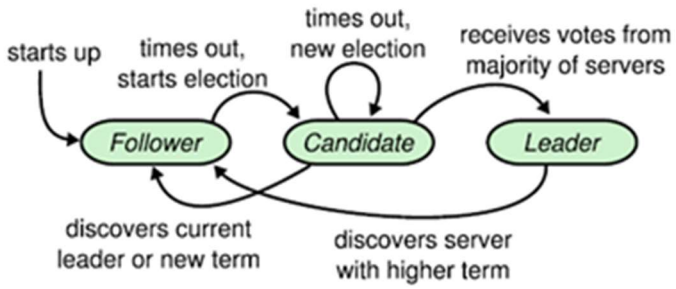

Figure 6-6 -ONOS RAFT algorithm and state synchronization between nodes

Raft consists of a cluster of nodes, each having a log to maintain. This log is fully replicated in all nodes achieving consensus through replicated state machines. These state machines process identical sequences of commands to produce the same output. To keep consistency, elections occur between nodes – a distinguished leader (responsible for replicating the log to other nodes) is elected within the cluster. Other nodes request the network state from the leader, which serves and guides them through log changes to avoid inconsistencies. In case of a leader failure, any other node is eligible to become leader -the election starts when a given follower could not reach the distinguished leader for a period known as the election timeout. The leader selection is achieved through elections between nodes in randomized time intervals. All nodes in the cluster move to candidate state and vote for themselves as leaders. As the election time is

[5] At the time of writing this deliverable there is still little information about TeraFlowSDN ETSI controller, targeting high-scalability (a Tera of flows)

randomized, the likelihood of having split-votes is quite low - in such situations, a new election takes place. When a node is elected as leader, all the others move to follower state.

ONOS takes advantage of the Atomix [96] framework for clustering (and distributed primitives) while OpenDaylight relies on the Akka [97] platform. Figure 6-7 shows a cluster of ONOS instances logically associated to three Atomix nodes, in charge of replicating the log (and ensure consistency) and partitions state across the cluster.

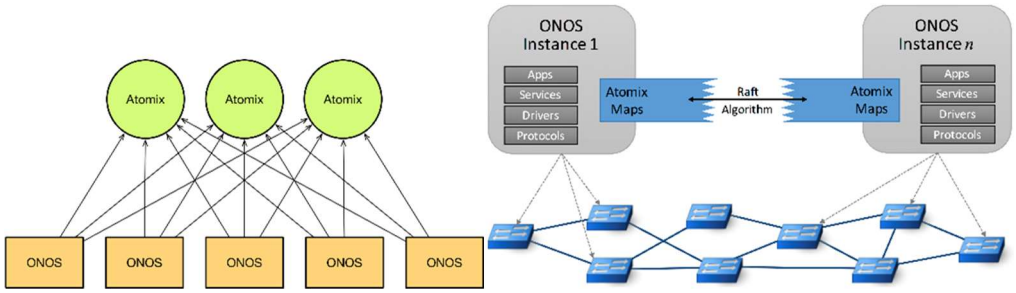

Figure 6-7 . Consistent data-grids in the Atomix framework

Although the Raft implementation is somehow similar in both controllers, the fact both rely on different frameworks also guarantees some fundamental differences between them. For example, the Atomix framework allows ONOS to take advantage of more relaxed distributed primitives such as eventually consistent maps (using the gossip protocol) for data that is not considered critical – thus achieving a performance level that would not be attainable if it relied solely on strong consistency models like Raft. Therefore, ONOS is often identified as being more performant than ODL [98]. On the other hand, the use of the Akka framework in OpenDaylight, as an actor-based software architecture, enables the implementation of more fine-grained use cases.

Both ONOS and ODL RAFT clustering require time sensitive operations (for voting/elections between nodes) which delegates the controller cluster placement closer to the devices being controlled - essentially within the same datacenter/domain. OpenDaylight, unlike ONOS, supports hot-standby cluster implementations by configuring a subset of the cluster nodes as non-voting [36]. This implementation allows for a multi-node cluster to failover to other nodes placed on a closer location with respect to the main "active" cluster nodes. Nevertheless, the transition between controller roles upon the active cluster complete failure still requires manual intervention [99]. The Atomix framework seems to also support a similar approach using consistent data-grids with a subset of cluster nodes belonging to a management group (see Figure 6-8) [96]. However, at the present date, this is not supported in the ONOS Atomix client implementation [100].

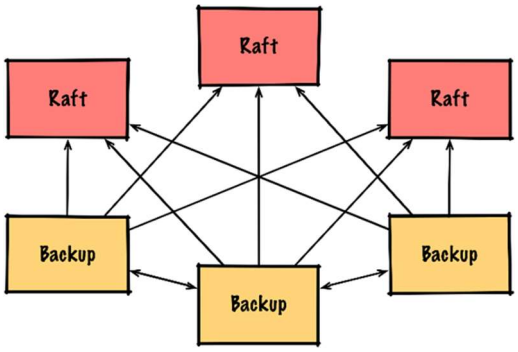

Figure 6-8 - Consistent data-grids in the Atomix framework

A close look should also be taken to the way the network device mastership is assigned in a clustered controller environment like the one provided by ONOS and ODL and how it inherently ensures the resilience of the SDN architecture. Network equipment is configured with a primary connection to one of the controller nodes but also with multiple backup/redundant connections to other nodes in the cluster. When the

controller node that is the master of the switch fails, the switch detects the connection failure and automatically fallbacks to one of the backup connections. The ONOS controller also includes a Mastership Load balancer application with the goal of distributing (and balancing) the switch mastership across all the available nodes in the cluster. This balance ensures that the controller connections are evenly distributed within the cluster increasing its stability and promoting an optimal number of open connections per node. According to the official ONOS documentation, a clustered environment of three nodes can failover the mastership of a switch to another node in just 3 milliseconds [101].

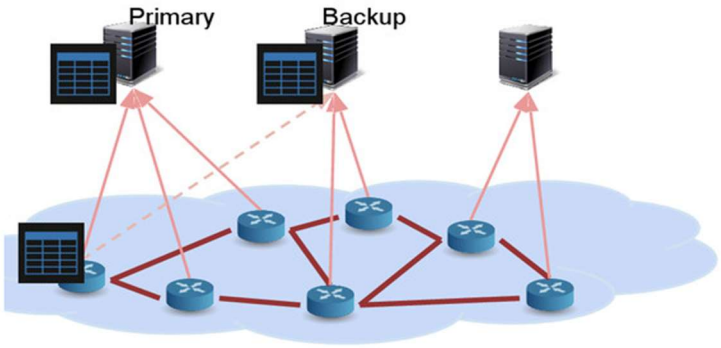

Figure 6-9 - Switch mastership with active and backup connections in a clustered SDN controller environment [101]

Modern controllers also support advanced mastership balancing policies. For example, ONOS introduces the concept of regions, where the network administrator can assign a number of controller instances and devices to specific geographical regions, and setup policies for mastership failover events. By exploiting this kind of mechanisms, SDN may assure that controlled devices are always assigned to closest controller nodes, only falling back to a different region if no controllers are available in the primary region [102].

In the same line, other resiliency advantages are also ONOS specific, one of them being the Intent framework. These high-level intents are compiled and translated to a set of flow rules on the underlying switch fabric and are continuously monitored by ONOS. A loss of throughput or connectivity between topology nodes may impact the viability of a previously successfully compiled and installed intent. In such cases, the controller will automatically try to recompile the intent and, if an alternate approach (e.g. another redundant path between network devices) is available, its installation will be attempted [103]. The workflow for intent compilation and installation in ONOS is illustrated in Figure 6-10. The intent framework thus ensures the installed policy is always enforced on the network.

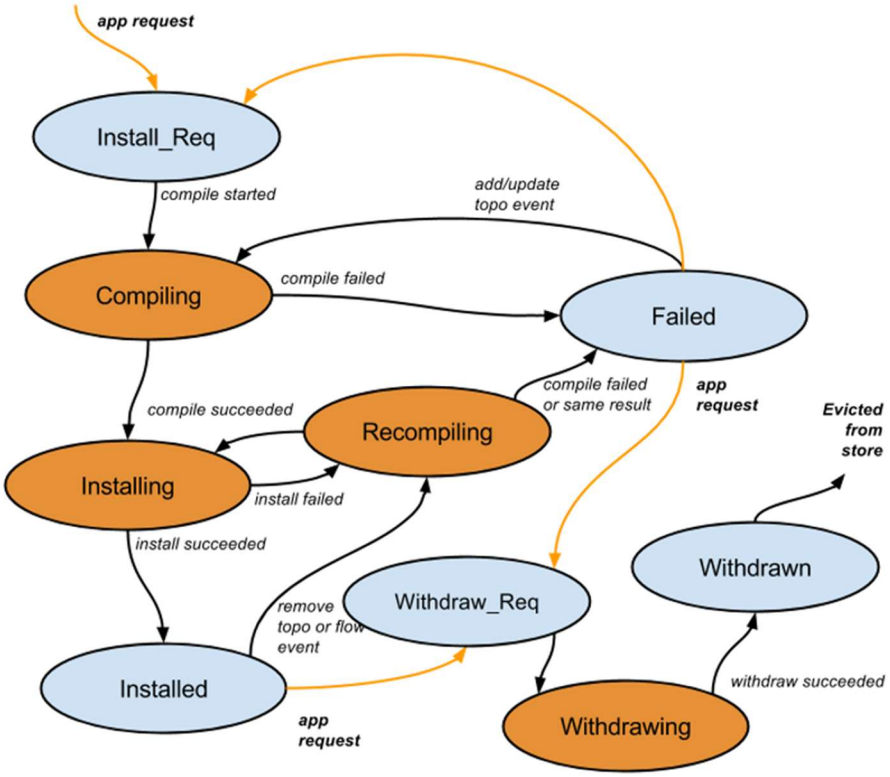

Figure 6-10 - The intent lifecycle in the ONOS controller [103]

6.3.2. Distribution

As seen in the first section, besides single domain clustering, one of the approaches to guarantee robustness and resilience in an SDN network is the application of distributed architectures composed of multiple controlled domains. As such, it is also important to understand what mechanisms are available in ONOS and ODL to transfer network information across different SDN domains. At the best of our knowledge there is not a single de-facto standard for east-west interface definition for SDN controllers.

ODL, for example, supports the federation of SDN domains via the AMQP (Advanced Message Queuing Protocol [83]) protocol using the RabbitMQ broker. In this federation, each site can basically have two roles: producer or consumer of messages. A subscription in the federation plugin is defined as a request between a consumer-site to a producer-site. Hence, in order to connect site A with site B, site A will have to subscribe to site B and site B will have to subscribe to Site A. The plugin declares which entities from the core (MD-SAL in ODL) it wants the federation infrastructure to subscribe and gets notified when the state of any of those entities' changes [104].

ONOS, on the other hand has some degree of inter-site communication using a native Kafka application. While being used mainly to export metrics from the controller cluster or the underlying network domain, it can also be used to publish network events to any interested party. The application sets up listeners for multiple controllers generated events (e.g., DeviceEvents, ClusterEvents) and sends it to a previously configured Kafka broker address. Other entities, including other ONOS clusters, can be configured (or programmed) to be consumers of those events.

Also worth mentioning is the ICONA project (c.f. Figure 6-11) developed for ONOS. Although providing only a subset of its original intended functionality and nowadays somehow unmaintained, the application shows how two independent ONOS clusters may be interconnected in a peer-to-peer model to share topology information. The ICONA application registers itself as a protocol provider at the southbound interface of the controller (just like a "real" protocol such as OpenFlow) and subscribes to topology and device events from the network domain being controlled by the respective cluster. Upon the reception of those events, the ICONA application computes the overall domain topology in the form of a single "big switch" (with the

domain hosts connected to it) and invokes the external interfaces of its peer controllers to add that switch (and hosts) as part of that controller topology.

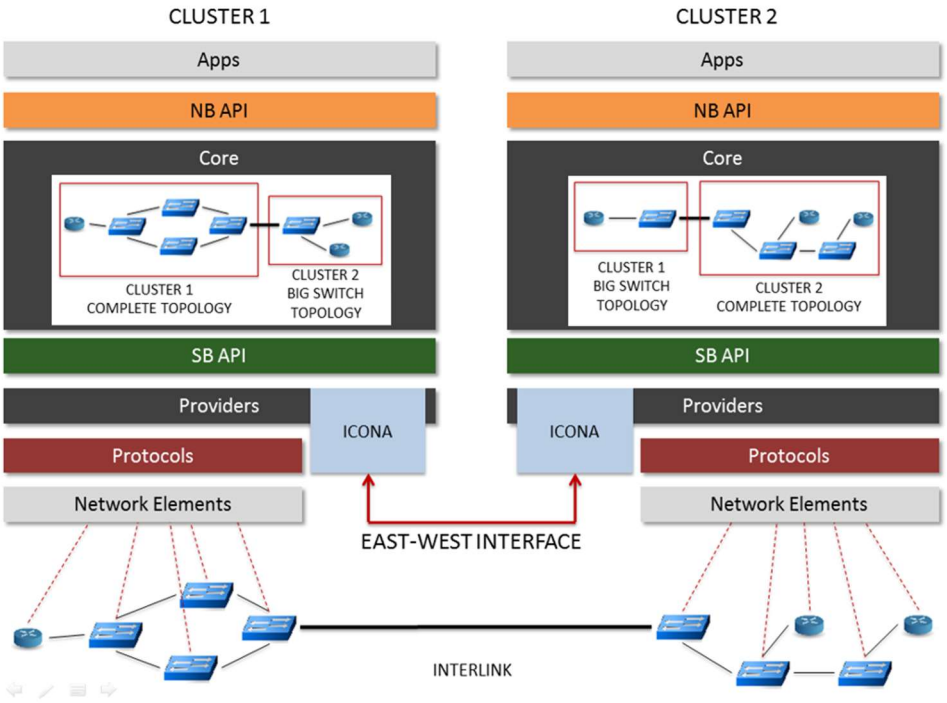

Figure 6-11 - ONOS ICONA application for cluster topology sharing [103]

Hence, all clusters will see the complete topology of their respective domain but also an abstracted view of the topology provided by another cluster. This allows multi-domain connectivity to be programmed within each individual domain without being completely dependent on a centralized control plane. Moreover, it also allows each domain to take advantage of other resilient SDN approaches such as those provided by the intent framework. Furthermore, the ICONA application is also policy based allowing each domain to specify which hosts and devices to make visible to an external domain. Despite the potentiality of this approach, the current version of the application only shares network topology and lacks any mechanism for specifying inter-domain flow rules or connectivity intents [105].

6.4. The future of SDN controller architectures towards resilience and distributiveness

Despite the existing clustering and resiliency mechanisms in the current generation of distributed SDN controllers, their implicit architecture also limits the overall capacity of the controller to sustain failures and scale. In fact, despite the modularity of ONOS and ODL (both implementing the OSGi specification), they can be seen as monolithic based architectures where multiple essential core services depend completely on each other. Applications developed on top of both controllers have a strong dependency on the provided core services and, as such, have a high risk of affecting the controller stability. In the era of cloud computing and cloud native (CN) applications, a monolithic controller implementation with a static core does not scale. As a result, there is currently an effort to redesign the previous generation of SDN controllers into micro-service-based architectures (Figure 6-12).

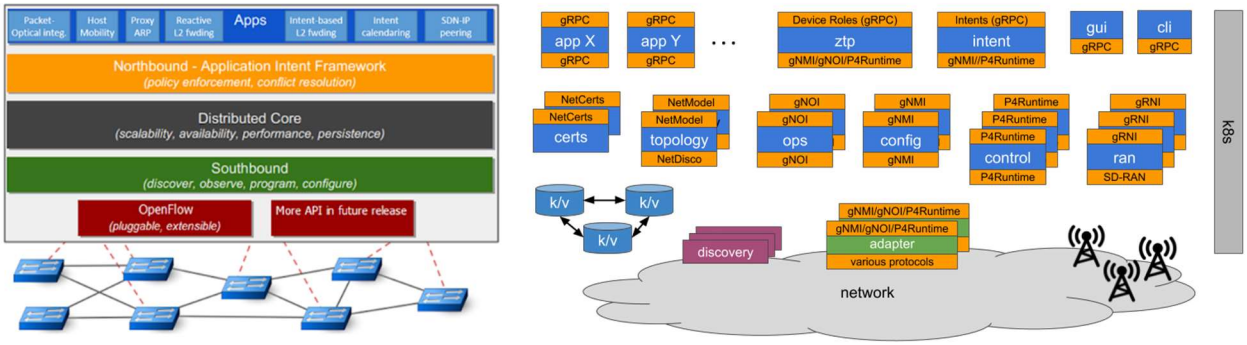

Figure 6-12 - ONOS classic (left) architecture vs. the next generation of ONOS (µONOS) [106]

One of such examples is **µONOS**, the under-development evolution version of the ONOS SDN controller platform. µONOS plans to support a unified control plane, micro-service-based architecture and common control and data plane orchestration [107]. Taking advantage of open interfaces (such as gRPC and protocol buffers [106]) between micro-services, µONOS is expected to bootstrap the controller services to the minimum viable set required for the developed SDN applications to operate. Moreover, those interfaces do not enforce a single programming language to be used for SDN applications which may also help with controller performance. Furthermore, it will also allow independent scalability and the elasticity of the stateless core micro-services (in an individual basis) as well as their placement in desired (and specific) locations. Combined with container orchestration frameworks like Kubernetes, controller micro-service replicas are expected to reach the specified configuration value across a given cluster.

It is worth mentioning unikernels as a promising reduced sandboxed environment for securely running applicational workloads (single address space, no shared memory). Recent work has addressed the usage of Kubernetes to deploy of unikernels. Although there are still no references of application of this approach to SDN controllers, this appears to be a feasible path towards more efficient and secure distributed SDN controllers.

6.5. Options for DISCRETION

The considerations above apply generally to the various SDNs that are considered on DISCRETION scope, like the Optical SDN, QKD SDN or Data SDN.

Based on the studies above, three basic alternatives were considered for DISCRETION, where four clearly distinct network domains exist (Optical, KQD, Data, and SDN)[6]:

1. **Multiple SDN domains without a multidomain Control Plane:** a controller is deployed for each network domain. There isn't a hierarchical controller, so each domain is addressed directly at the application level. **Error! Reference source not found.** shows the distribution of the architecture for the IP plane.

[6] Note: The figures will represent only the Data Layer (IP) plane

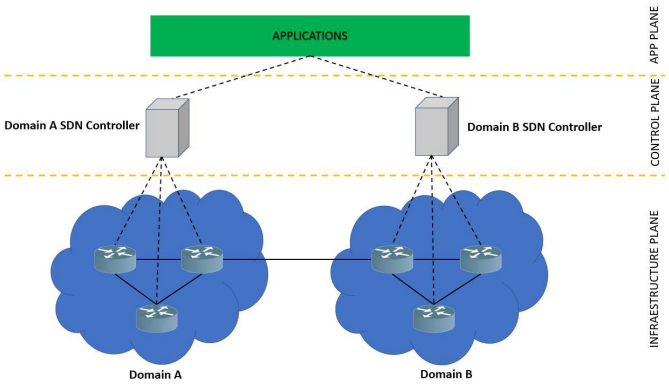

Figure 6-13 - Multiple SDN domains without a multidomain Control Plane

There are per-domain controllers, each one must control and manage its domains independently. The limitations of this architecture are:

- Limited support for multi-domain SDN. It is up to the Applications to orchestrate across domains all aspects that are multidomain.
- SNR will decrease because of using more devices (in this case there are more controllers).
- This architecture is suitable for single-provider scenarios, but not for multi-providers.

2. **Logically centralized SDN deployment** (see 6.2)**:** A centralized SDN architecture with a single SDN controller responsible for the management and control of the different network elements. The problems or disadvantages that presents this architecture are:

    - Scalability: using one controller can cause the saturation of the network and the performance may decrease significantly.
    - Availability use only one controller can cause that the network goes down if the controller fails, although High Availability (e.g., 1+1) redundant schemes may be deployed in this case.

3. **Distributed Hierarchical SDN architecture** (see 6.2)**:** The hierarchically superior controller is responsible for the management and control of the multidomain environment. Southbound domain controllers are responsible for their domain and report hierarchically to their superior via their Northbound interfaces.

At an Organizational level, DISCRETION is currently considering the third alternative (distributed Hierarchical SDN) because it is perceived as flexible and scalable, as well as one that supports resilience-oriented configurations:

- One controller for each domain provides a better response to failures due to the distributed structure.
- Suitable for multi-provider heterogeneous scenarios.
- Each plane provider can deploy its own software control logic.
- Scalable solution. This architecture can be expanded horizontally by adding new nodes and resources to meet demand needs.
- Greater tolerance to faults, since when one node falls, the information will be found in others.

With this architecture, each organization in the federation retains full control of its own network, independently of its internal implementation (e.g. a hierarchical multi-level scheme). Even though, other architectures, also described in section 6.2, like Distributed Non-hierarchical or Hybrid may be considered at a later stage in the project.

7. OPTICAL SDN

In the following subsections it is discussed the DISCRETION SDN solution for the control of the Optical Domain. For that reason, this kind of control is placed, and controls devices placed in the black part of the network, while the red part, usually placed at the edge of the network, is out of the authority of the optical SDN controller. It is important to highlight that the concept of SDN control in the transport optical network has been and is still widely explored so that, several key elements in terms of protocol, control platforms and interfaces, are already available and usable to fulfil the purpose of DISCRETION, as discussed in Section 3.5.

7.1. Features

Given the flexibility provided by the SDN paradigm, it is possible to envisage a number of possible features to implement spanning from full manual to complete automation, from the coarsest to the finest degree of control. Nevertheless, it is important to note that the actual realization of SDN functionalities always implies a certain amount of effort so that, a prioritization of some features, with respect to others is required. Optical connections (lightpaths) lifecycle management features that includes Provisioning, Updating and Decommissioning, fall on the set of high-priority features, along with the possibilities to read some important information related to the established lightpaths. DISCRETION aims to implement such priority features for the software control of the Optical networks as part of a wider set of features that be implemented according to the architectural requirements reported in DISCRETION D2.1 deliverable. In particular, the set of features foreseen are:

Lightpath lifecycle

- **Provision of a lightpath**: the Optical SDN Controller offers the possibility to provision a lightpath between two endpoints. This also includes the automatic calculation of the optimal path in the optical network, given certain constraints.
- **Decommission of a lightpath:** given an established lightpath, the optical SDN control platform offers the possibility of terminating the lighpath with the consequent automatic release of the resources allocated in the involved optical devices.
- **Update of a lightpath:** an established lightpath can be updated at runtime in terms of QoS constraints and optical path.
- **Lightpath information retrieval:** the optical SDN platform maintains meaningful information on the established lightpaths that can be retrieved on demand.

Topology Management

- ADD/REMOVE optical devices: an optical device can be added to and/or removed from the optical network topology maintained in the SDN control platform internal datastore.
- Topology information retrieval: the optical network topology can be retrieved on demand in the format defined by the Transport API (TAPI, see Section 3.5.2)

7.2. Functional Architecture

One of the key elements in an SDN architecture is the SDN Controller, a specific piece of software that exposes the control features, incorporate the control logic, and interacts with the network devices (optical devices, in this specific case), as depicted in Figure 7-1.

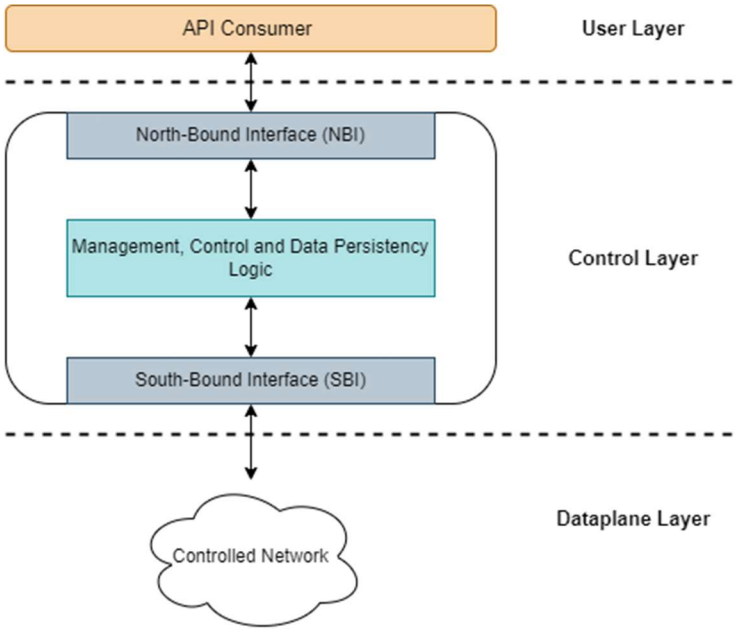

Figure 7-1: SDN Control general architecture

An SDN Controller consists of 3 main elements:

- North-bound interface (NBI), in charge of exposing the SDN control APIs towards user, 3rd party applications or even other SDN controllers.
- Control Logic, that implements network control and internal data storages to maintain network information.
- South-bound interface (SBI): implements specific protocols (SDN protocols) to interact with the controlled devices.

While NBI and SBI are usually implemented following certain standards (see Section 3.5), the Control Logic is highly dependent from the SDN controller capabilities. It is important to highlight that the implementation of an Optical SDN Controller is out of the scope of the project: several stable and well-known solutions are available, even with a high TRL (production-ready). In this regard, the focus is on open-source solutions (see Section 3.5.1), more suitable for a research project, rather than commercial/proprietary solutions provided by big vendors like Cisco, Juniper etc. Independently from those examples, in this section it is taken into account the capability of the controller itself to be programmed in order to properly customize the SDN control solution.

Figure 7-2 illustrates the functional architecture of the DISCRETION Optical SDN controller, with the details of the functional blocks (FBs) envisaged to cover the set of features described in the previous section.

As for the previous figure, the Controller is characterized by a NBI and an SBI and an internal logic depicted in a more detailed way. In particular, the internal logic consists of 3 functional blocks that implements the logic for lightpath lifecycle management (Provisioning), network topology management (Topology) and network path computation (PCE, Path Computation Element). For the sake of simplicity, the figure does not show any datastore, considering it to be embedded in the related functional blocks.

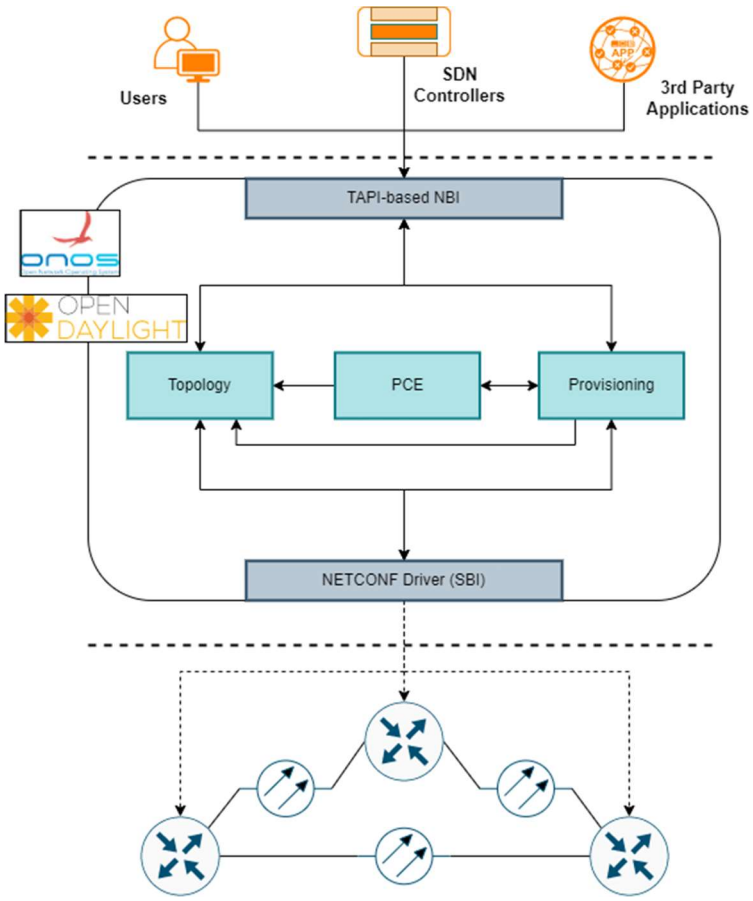

Figure 7-2: DISCRETION Optical SDN Controller Functional architecture

The Provisioning FB (FB) is in charge to cover all the features related to the management of lightpaths' lifecycle, including maintaining the related information retrievable on demand. In a similar way, the Topology FB maintains the topology information and provides the logic to ADD/REMOVE network items (optical devices e.g., ROADMs). PCE FB calculates the network end-to-end path between to endpoints that is used to provision the lightpath i.e., properly configure all the optical device belonging the path to enable the optical transmission. The arrows in figure link the 3 FBs to highlight that they need to cooperate to proper control the underlying optical network. As an example, during the provision of a lightpath, the Provision FB requests the PCE FB to calculate the path between two endpoints, the PCE FB in turn asks the Topology FB for the network topology (graph) to execute the path computation.

The internal FBs are also linked with the NBI and SBI. In particular, the NBI exposes REST-based APIs that represent the command interface for the controller, enabling the consumer to access the features defined by the FBs. As mentioned, NBI FB is follows the large-recognized reference model called Transport API or TAPI, defined by the Open Networking Foundation. TAPI provides all the required functions to enable the features described in Section 7.1, i.e., APIs for Lightpath lifecycle management, Topology management and Path computation.

The requests executed through the NBI are elaborated by the internal FBs, and when required, translated in commands to be applied to the optical device through the SBI. The SBI implements one or more communication protocols to interact with the controlled device, usually called SDN protocols. Several standard SDN protocol have been defined over time; among them, NETCONF can be considered one of the most suitable for the configuration of optical devices. The devices, in turns, implement the same SDN protocol to enable the communication with the SDN Controller, through a special piece of software running on top of the device itself and called SDN agent.

7.3. Operational Workflows

In the following subsection is describe a set of workflows representing the base lightpath lifecycle management operations (provisioning, decommissioning, updating), detailing the set of modules, belonging to the DISCRETION optical SDN Controller, that are involved. The **Lightpath information retrieval** feature offered by the SDN Controller, mainly related to the retrieval of information are here omitted due to the simplicity of their workflow that consists of a request on controller's NBI to get information from internal datastores.

7.3.1. Optical Path Provisioning

In Figure 7-3 are shown all the steps required to configure a lightpath between to endpoints (a, b) belonging to the controlled optical network.

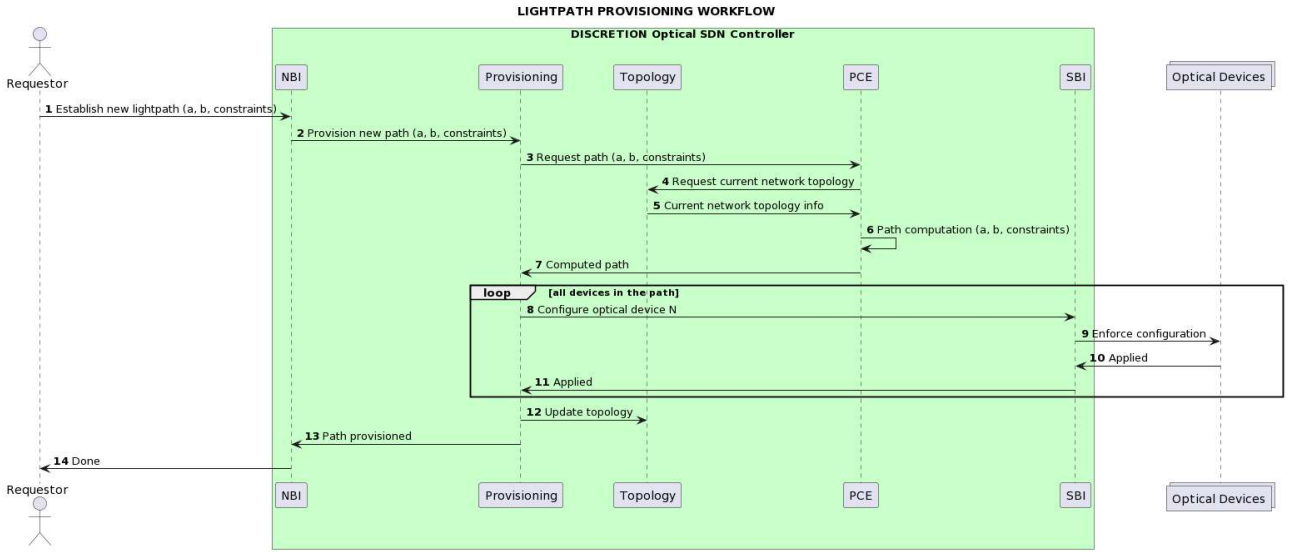

Figure 7-3: Lightpath provisioning workflow

Steps 1-2: A requestor (User, 3rd party application, SDN controller) requests towards the Optical SDN Controller NBI the establishment of a new lightpath between to optical node in the network, a and b, specifying also lightpath characteristic (QoS specifications), used as constraints for the calculation of the path.

Step 3: The *Provisioning APP* receives the provisioning request and invokes a path computation through the interface of the *PCE FB*

Steps 4-5: The *PCE FB* asks the *Topology App* to provide network topology with the current status of nodes and links in order to be able to compute the new path, if possible.

Steps 6-7: Path is computed and forwarded to the *Provisioning*

Steps 8-11: The *Provisioning FB* extract the list of the network devices from the received path and configure them through *SBI.*

Steps 12-14: Once all the devices are properly configured the Lightpath is created. The Topology is updated i.e., resource used by the new lightpath are no longer available in the devices and the requestor is notified

7.3.2. Optical path Decommissioning

Figure 7-4 shows the sequence of steps required to the decommissioning of an existing lightpath. At the end of the workflow all the resources employed in the lightpath are released and available again.

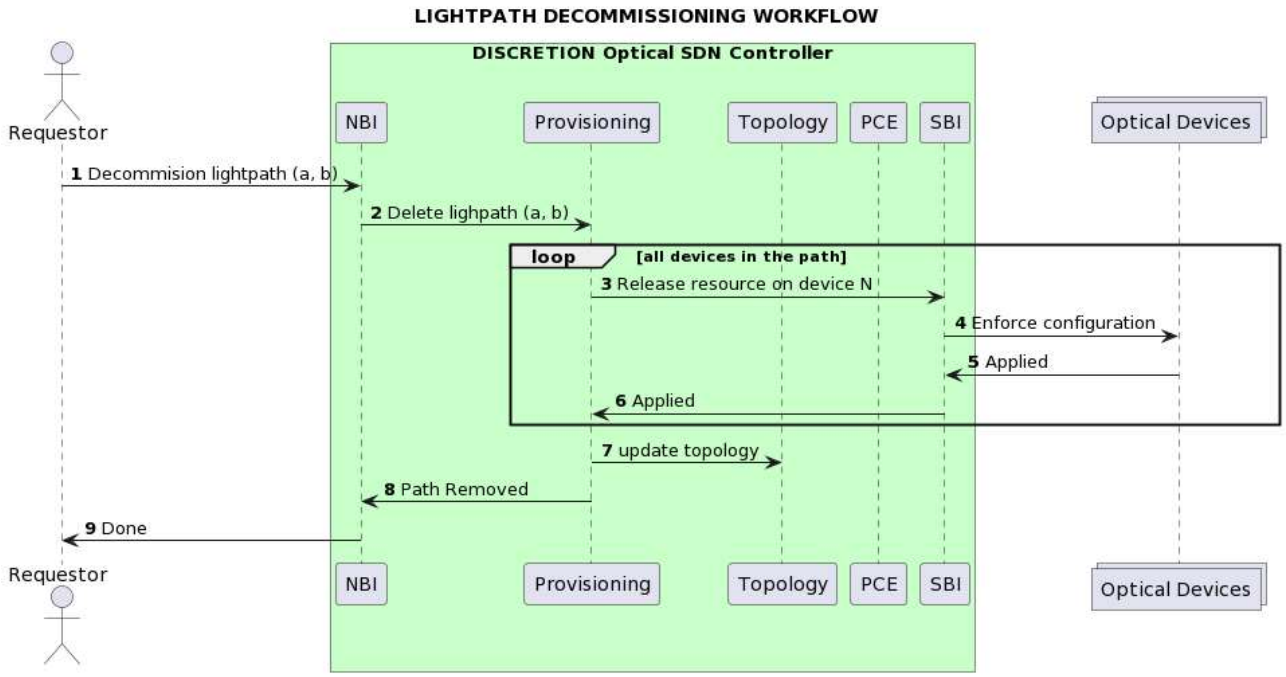

Figure 7-4: Lightpath decommissioning workflow

Steps 1-2: A requestor (User, 3rd party application, SDN controller) requests towards the Optical SDN Controller NBI the decommissioning of an existing lightpath.

Steps 3-6: The *Provisioning FB* enter in a loop and remove the lightpath configuration from all the device belonging to the path

Steps 7-9: Topology is updated with new information regarding the resource availability and the requestor is notified

7.3.3. Optical Path Reconfiguration

Figure 7-5 show the steps required to update an existing lightpath i.e., a change in the path characteristics (such as QoS) without modifying the original endpoints. It is important to note that such an operation could be performed in two steps by exploiting the two previous workflows of lightpath decommissioning and provisioning: the old path is decommissioned and a new one is created with characteristics and same endpoints of the old one. The following workflow aims to provide a single operation for the update of the lightpath where the new path is created before the hold is removed, in order to try to avoid outages in the optical communication.

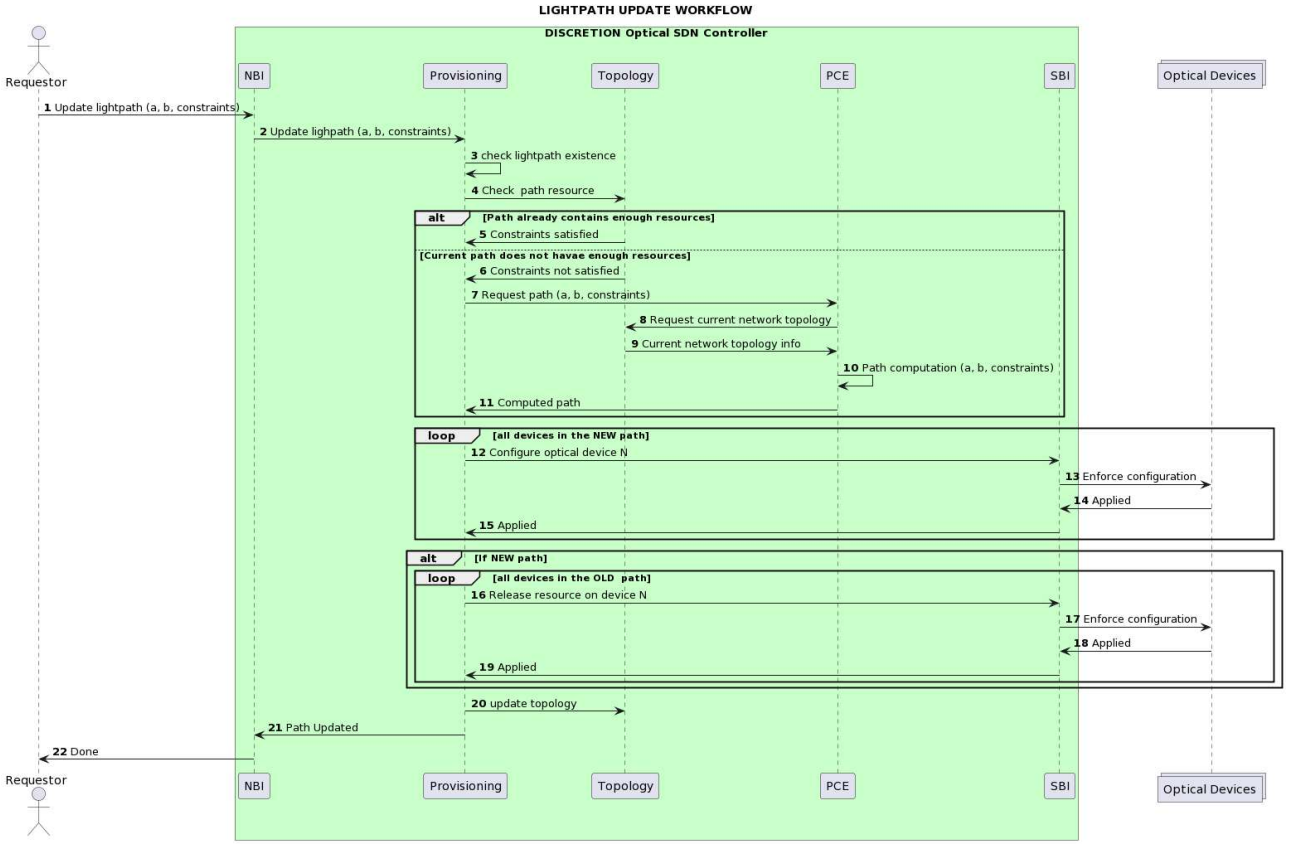

Figure 7-5: Lightpath updating (reconfiguration) workflow

Steps 1-2: A requestor (User, 3rd party application, SDN controller) requests towards the Optical SDN Controller NBI the reconfiguration of an existing lightpath, specifying its new characteristics.

Steps 3-4: The *Provisioning FB* first check the existence of the lightpath then verify with the *Topology App* the resource availability in all the node belonging to the existing lightpath

Steps 5-11: If the new set of constraints are already satisfied all the node in the existing lightpath, the same path can be reused as-is, and the workflow continues from Step 12. If the new constraints cannot be satisfied, a new path computation must be requested (Steps 6-11).

Step 12-15: All the Optical devices in the path are configured on the basis of the new constraints provided

Step 16-19: In case of new path, the resources in the OLD one is released

Step 20-22: Topology is updated with new information regarding the resource availability and the requestor is notified

7.3.4. Operations in an SDN multi-domain environment

The SDN Optical controller in DISCRETION is part of the set of "Network Domain" Controllers that includes Data SDN Controller and QKD SDN Controller. Such controllers may need to exchange information to coordinate they control actions in the respective network segments, as discussed in Section 0. A particular case that requires such an interaction is the establishment of quantum key exchanging sessions (QKD links) in the QKD network: QKD SDN Controller must coordinate with the Optical SDN controller for the establishment of a secure lightpath. More detailed information is provided in Section 8.4, where the QKD and Optical Coordination is discussed.

8. QUANTUM KEY DISTRIBUTION SDN

The success of QKD networks radically depends on the degree in which they can be adopted to the existing infrastructure, and it can only happen via standard protocols and interfaces. Based on this principle, the flexibility of SDN allows a faster integration of quantum communications in the telecommunications network than following previous schemes, where the different network devices should be modified, one by one, to create a quantum channel. The integration of QKD on the current military networks requires the adaptation of this technology to the premises and restrictions adopted on these types of networks. This attempt pushes forward this technology on military environment, counting on SDN for making such integration possible.

The control plane design will envision an architecture with strong and resilient controllers and a possible network orchestrator to manage a multi-level architecture. Also, there are different architectural possibilities to adapt to different security constraints and restrictions along the project, for example, the segregation of the red and black network. Nevertheless, on any SDN environment, civil or military, the use of well-known standards, interfaces and protocols is necessary for the deployment. The SDN based design for DISCRETION take advantage of this concept and is highly based on the ETSI Standards.

8.1. SDN Design Architecture

The design decisions on how classical and quantum functions are distributed on military networks and how they relate to each other constitute an important architectural milestone for the DISCRETION project. For flexibility, adaptability and ability to incorporate all hardware and functionalities developed in the project, the paradigm adopted for QKD integration is the software-defined networking (SDN). In that framework, the functions and components are wrapped with software techniques, such as exposing its functionality with software interfaces. This enables functional plane decoupling, especially the control and data functional planes, component disaggregation and additional software enabled techniques such as virtualization. As a result, software-defined networks' design assumes the logical centralization of the control functions into a specific software component called SDN controller, which responsibilities are defined on the control plane of the network.

8.2. SDN Control plane

The SDN control plane architecture assumes the control interfaces defined by ETSI GS QKD 015 [108], applicable to a general network model built by the connection of nodes suitable to be controlled by SDN mechanisms. This standard defines an open interface in YANG [109] [110] language to simplify the management of all QKD resources through the implementation of an abstraction layer. This optimizes the creation and usage of Keys by introducing a logically centralized element to orchestrate all the resources, the SDN controller. The integration has the added benefit of using well-known mechanisms and tools in the classical community, which will facilitate its adoption and deployment by general telecommunications networks.

The SDN central controller organizes the resources, optimizes the key allocation per link based on demands and automate the creation of either direct (physically connected through an uninterrupted quantum channel) or virtual (multi-hop-based) links, where the keys are relayed from one hop (direct link) to the next in the chain connecting the initial with the final points. However, ETSI GS QKD 015 does not define the specific protocol, data structures or specific implementation chosen to carry the YANG-structured information it defines. Therefore, the standard still permits some flexibility during the system design and implementation phases. The YANG data model for a node is divided in four main groupings:

- **Interfaces**: Interfaces represents an abstraction of the Quantum Forwarding Plane (QFP) elements contained within a secure location, allowing the SDN controller to identify all these elements and aggregate them as a single network element with multiple interfaces.
- **Applications**: Defined as any entity requesting Keys from the key manager within the node. These applications might be external or internal.

- **Capabilities**: Set of parameters and features to support certain functionalities.
- **Key Association Links**: Logical key associations between two remote nodes. These links associations can be of two different types: physical (also called directs), if there is a direct quantum channel through, or virtual if keys are forwarded (key relay) through several trusted nodes to form an end-to-end key association.

8.2.1. Architectural Design

In ETSI GS QKD 015 standard, the control unit is the *Software Defined Quantum Key Distribution Node* (SDQKD Node) which have been deeply explained on the "Deliverable 4.1: QKD Nodes Preliminary Design Report", defining an aggregation of multiple elements that interacts through a set of interfaces with a SDN controller using standard SDN protocols. The Figure 8-1shows this design. Through this open interface, the SDN controller orchestrates the resources to optimize the key allocation per link. This is done by automating the creation of physical Quantum Forwarding Plane (QFP) channels and of virtual (multi-hop-based) QFP links, and keeping a network wide visibility. The connection between SDQKD Node and controller allows retrieving information from QFP and dynamically and remotely configure the behaviour of the QFP elements to create, remove or update key associations between secure locations. The SDQKD NODE abstraction follows a disaggregated approach, considering a set of interconnected components with well-defined tasks and interfaces, as described in the following sections.

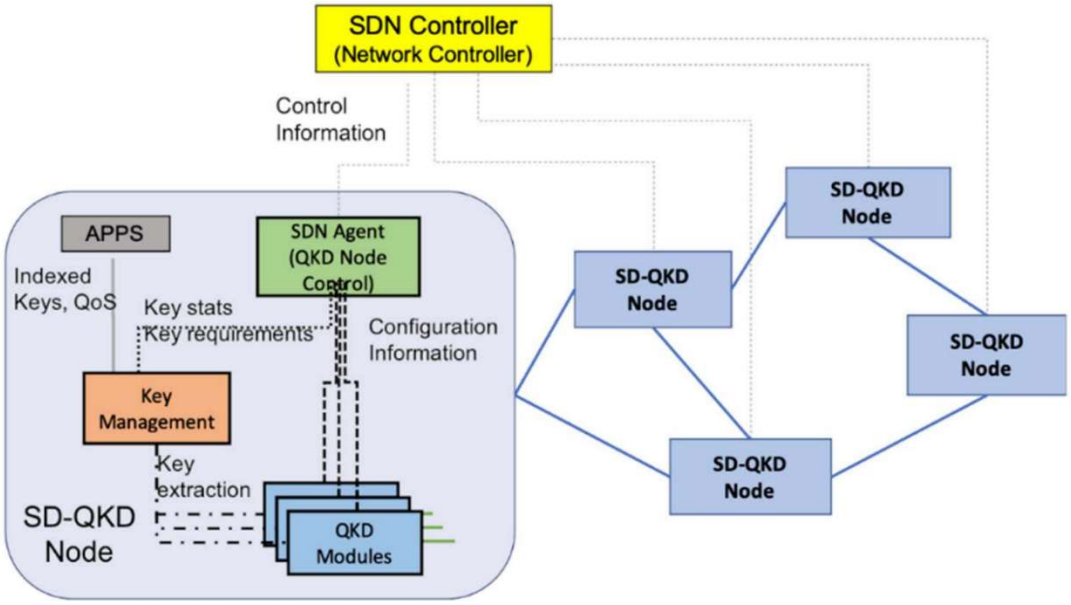

Figure 8-1 General structure of a Software Defined QKD Network based on ETSI QKD 015. The SDQKD Nodes are connected to the controller and also among them. The top left node shows the internal set of components and the main functionality of the interfaces that connect these components themselves

8.3. Quantum Forwarding Plane (QFP)

As a basis of any quantum communications network, there is a set of functionalities dedicated to the generation, manipulation, transport and measurement of quantum signals and the associated classical processing done on behalf of the quantum-level protocols. These functionalities enable executions of general QKD protocols to distribute end-to-end key, or some classes of direct point-to-point secure, secret or authenticated communication. This set of functionalities can be seen as a logical layer. Here we refer to this layer as the quantum forwarding plane. This plane is being designed as an essential add-on to

communication networks extended by QKD technology taking advantage from the current network infrastructures.

The new segments based on QKD are the only ones that needs quantum expertise. The remaining network segments are fundamentally adapted variations of traditional communication technology. The full QKD Network requires this expertise plus the ability to interface the quantum plane.

The main functionality of any quantum-based network is the transport of quantum signals, even the ones based on beyond QKD protocols. Thus, the quantum plane (not necessarily including the forwarding mechanism) is a suitable concept also in these cases, that needs to be addressed in the context of the project.

Any technology capable of the basic functions of generating/transmitting and then receiving/measuring quantum signals could be employed in principle to generate a quantum communication. The practical choices for selecting quantum communications transport media are restricted to either optical fibre or free space. This is a consequence of the physical properties of communication channels and available technology (now and in the mid-term future). In general quantum communication networks, even those that will prospectively employ technologies that are not yet available, each network node, whether trusted or not, would need to perform operations (quantum and classical) to ensure the forwarding of the quantum signals or enable the utilization of persistent quantum resources such as distributed entanglement. In any case, a construction as the proposed QFP will be needed in any generation of quantum key distribution technology. The QFP objective is to encapsulate all the functionality required to obtain an end-to-end secret key. This implies the transmission of the quantum signals and the execution of the protocols. In this sense, also the forwarding of the keys at intermediate nodes in the example of a multi-hop chain, is also part of the QFP, since it is done exclusively on behalf of the QKD functionality.

At the same time, any general key management functionality that just imports the keys and manages them for other applications, or the network controllers are considered outside of the QFP, but depending on the operation to achieve these components, they could interact with the QFP. All network functionalities beyond the QFP are conventional (or partially modified) ones. The QFP concept can also accommodate future networks in which the objective is not the generation of a secret bit stream, but also entanglement distribution for alternative purposes.

The QFP constitutes a common abstract foundation for quantum communications networks, regardless of their final purpose. This abstract foundation constitutes a perfect element for the management of all the forwarding elements (classical and quantum) through the SDN infrastructure developed in this project.

8.3.1. QKD Physical Link

The integration of existing QKD systems in regular communications networks is currently a complex task. New QKD systems require to be manually installed and configured, requiring sometimes exclusive dark fibre connecting between two sites and thus forming a fixed QKD link. Network operators have already started to analyze different integration strategies for the QKD resources. Some of these strategies directly study the costs of deploying a completely parallel and isolated QKD network. On the other hand, the advent of disaggregation for optical networks and the already well-studied (and partially adopted by operators) software-defined optical network architectures are key technologies that enable high configurability in the optical domain. These technologies are the base to bring coexistence of classical and quantum signals and dynamically manage the creation of classical and quantum channels integrated into the same network and even sharing the same physical media, reducing time and cost of deployments. The Figure 8-2shows the steps of the physical link creation:

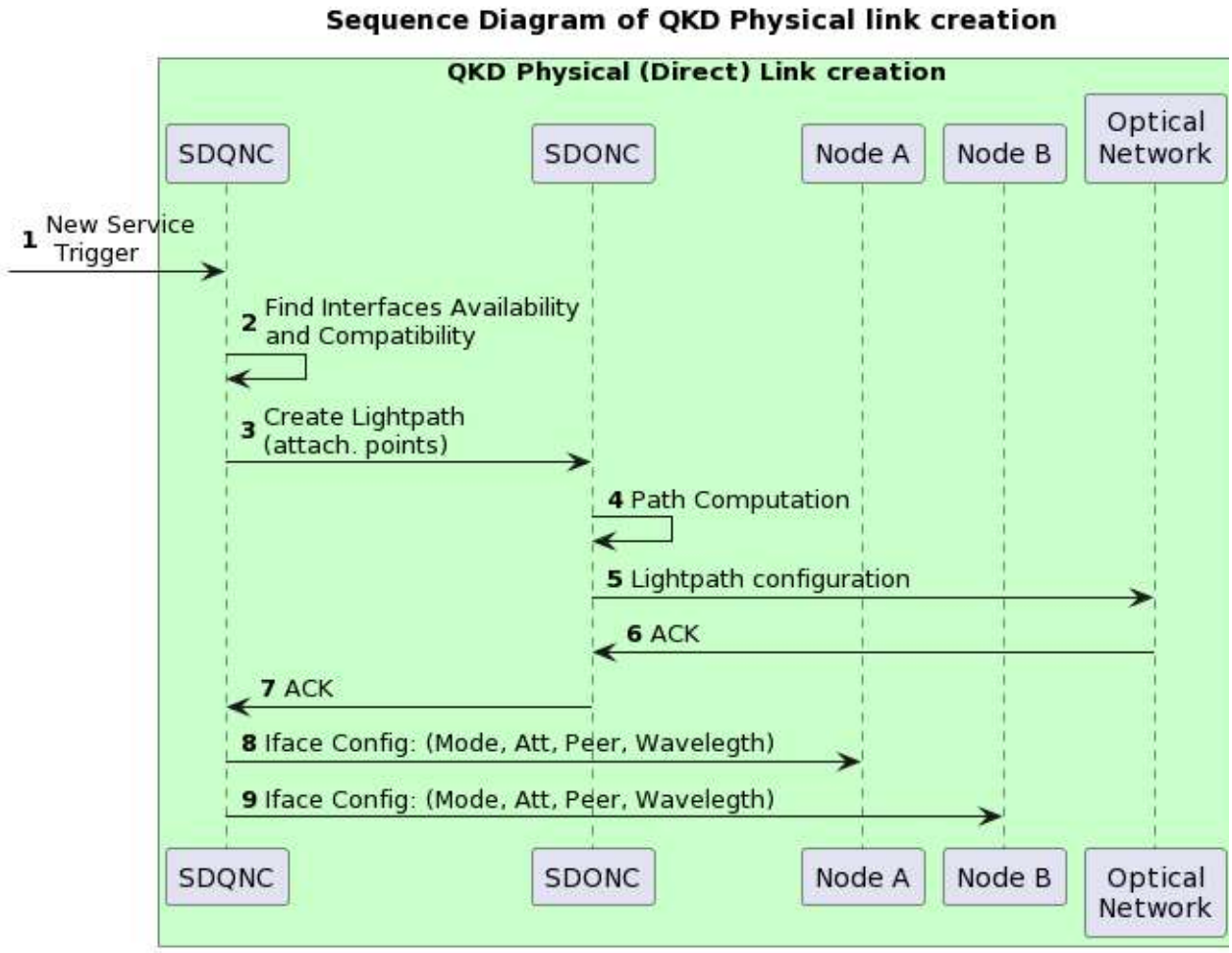

Figure 8-2: Sequence diagram for the dynamic creation of a physical QKD link

Physical QKD link creation sequence diagram:

• The logically centralized Software-Defined QKD Network Controller (SDQNC) receives a key request either from detecting key demands from the nodes or through the controller's North Bound Interface (NBI) in case of a key request directly done from the Orchestrator.

• The SDQNC finds two available and compatible interfaces from the two endpoints of the quantum channel.

• Then, the SDQNC sends a request for lightpath creation to the Software-Defined Optical Network Controller (SDONC) if the optical path is not provisioned, with the specific constraints of the quantum channel.

• After the evaluation of the path computation, if it is satisfactory, the SDONC configure the optical devices and creates the lightpath.

• After the lightpath is successfully created, the SDQNC connects both SDQKD Node to configure each involved interface in the communication, including the characteristics of the optical channel.

8.3.2. QKD Virtual Link

A virtual link is defined as a key association between two SDQKD nodes, which are connected through multiple SDQKD nodes in relay mode, using multiple physical QKD links (multi-hop). This key association consumes Keys for the end-to-end key transport operations, as well as a new link providing keys for the two remote endpoints. This is a very complex distributed process that needs to be performed carefully. Virtual links could potentially end up causing congestion over other physical connections if it is not treated adequately. The SDN controller collects performance metric of the QKD network to improve decision making and help on mitigating this issue, as it has a view of the QKD network as a whole, keeping track of consumption demands, key store availability and generation rates. When a new key request is received by the SDN controller, it is under its responsibility to allocate the new optimal path over existing links to reduce the impact over other running applications and key associations. That means that the SDN Controller, and therefore all the QKD network, needs to deal with multiple key requests simultaneously (Applications) and allocate all the necessary resources to each key request independently and without interfering between them. The Figure 8-3shows the sequence diagram virtual link creation:

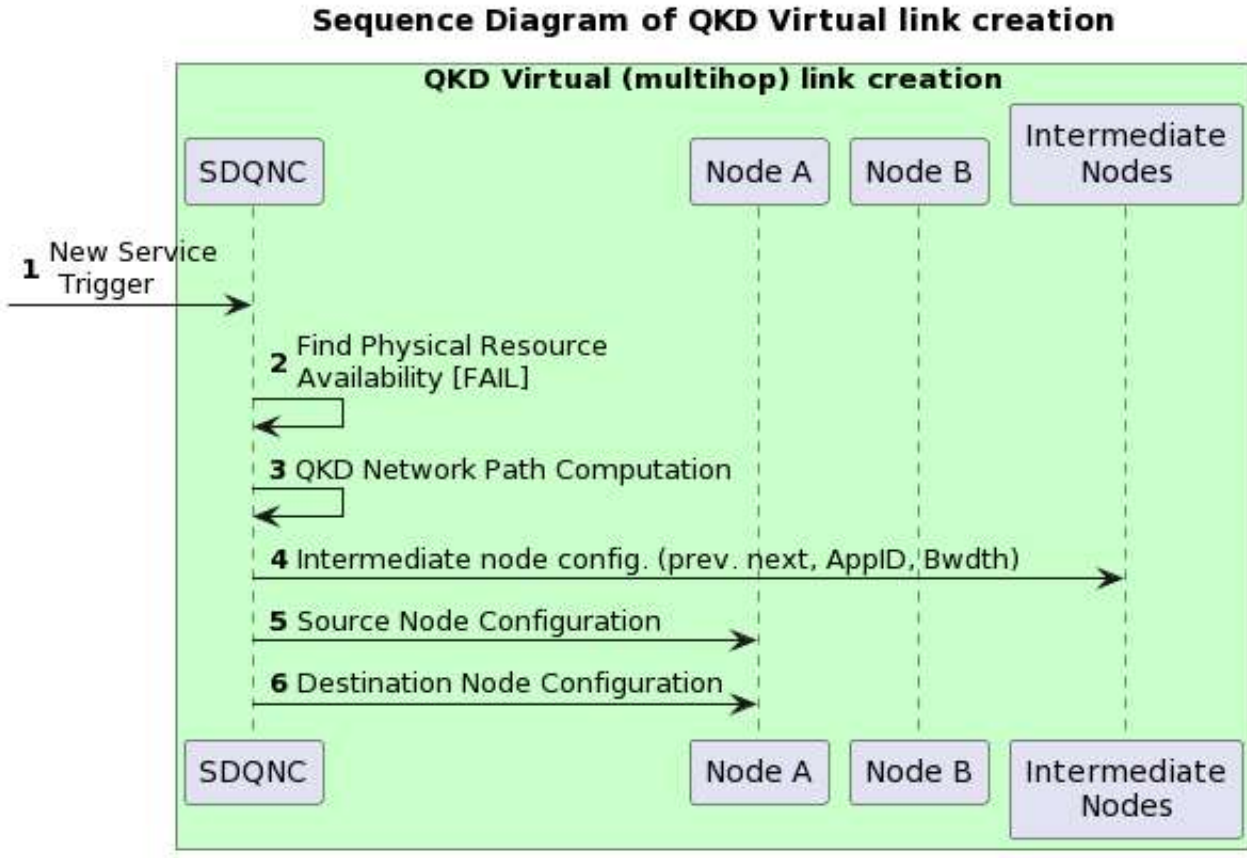

Figure 8-3: Sequence diagram for the dynamic creation of a virtual (multi-hop) link

Virtual QKD link sequence diagram:

• The SDN controller will receive a key request, exactly in the same way as the physical link (by the nodes or by one operator).

• The SDQNC fails on finding available resources, either from the SDQKD Node side or from the optical network side.

• The SDQNC ask to the Path Computational Element to find a path over the existing QKD network, e.g. based on a weighted graph built using the key availability, key generation rates and consumption for each link.

• If succeeded, the SDQNC needs to configure each node involved in the key transport path. To do that, the SDQNC sends the configuration information to each intermediate node in key relay mode, informing of the previous and next nodes in the path. If there are multiple links, the controller could also specify the link for the relay.

• The controller finally configures the endpoints of the virtual link/key association, notifying the previous or next hop, the bandwidth to be provided and other optional information.

8.4. SDN Orchestration Plane

An SDN orchestration can be defined as the continuing process of automatically coordinating the available resources according to optimization criteria, to establish and release the end-to-end service provisioning through different network domains controlled by each SDN controller. This section presents the interface between an SDN orchestrator and an SDN controller of a QKD network, describing the flow of information between both entities where the SDN orchestrator acts as a client of the SDN Controller (server). Like ETSI GS QKD 015, the ETSI GS QKD 018 proposes an information model given as a YANG model, which makes this approach implementation agnostic by any vendors. This information model enables the SDN orchestrator to manage and configure the SDN controller of a QKD network, permitting it to orchestrate the QKD network and a classical optical transport network.

When a QKD key based service is required, the Keys need to be supplied to a secure application through an Optical Transport Network (OTN). It is necessary to know which QKD nodes in the QKD network can supply Keys to secure application entities in an OTN. Each SDN controller has only local information about the network it controls. The introduction of an SDN orchestrator interface to the SDN controller of the QKD network, ease the maintenance a QKD network in terms of network topology, configuration, management policy and performance management.

The information model for the interface between an SDN orchestrator and an SDN controller of QKD network has been proposed to simplify the management and configuration of QKD networks through the North Bound Interface (NBI) of the SDN controllers of the QKD networks.

8.4.1. Architectural design

The deployment of QKD network to secure data in a communication network needs to be done considering the classical communication network for the correct key delivery to secure application in a classical communication network. These applications can reside on different network domains, where they can be managed and configured independently via domain-specific SDN controllers. As a consequence, a multi-domain SDN orchestrator to orchestrate the whole network system is required.

For QKD key delivery process to the secure application entities in an Optical Transport Network (OTN), it is common to use a separate QKD network from a classical OTN and to operate and manage each network separately. That separation allows to optimize the performance of QKD links, secure isolation, separation of responsibilities for management, etc. Both network domains need to be aware of nodes that belong to both network domains for delivering Keys in a secure way. The SDN orchestrator can perform this role with the information from the SDN controllers of the QKD network and the OTN. The coordination between network domains, an interface between the SDN orchestrator and the QKD network SDN controller needs to be defined to describe the flow of information between the SDN Orchestrator and each SDN Controller.

Before provisioning an E2E QKD service, the SDN orchestrator should collect the topology and inventory information of the QKD network from the SDN controller of the QKD network as a previous step to check the status of QKD network. After collecting this information, the SDN orchestrator requests a service link and the candidate paths for the service link within a QKD network to transport Keys for end-to-end QKD service provisioning. After the SDN orchestrator receives the candidate paths from the SDN controller of the QKD network, the SDN orchestrator decides which candidate path in the QKD network will use for the service and requests to deploy a specific path from the SDN controller of the QKD network. In the YANG models proposed by ETSI GS QKD 018, the topology and inventory YANG models are separated from the

connectivity YANG model. With this configuration, an SDN orchestrator can orchestrate multi-QKD network domains via each SDN controller of each QKD network as well as both QKD and classical network domains, as shown in Figure 8-4.

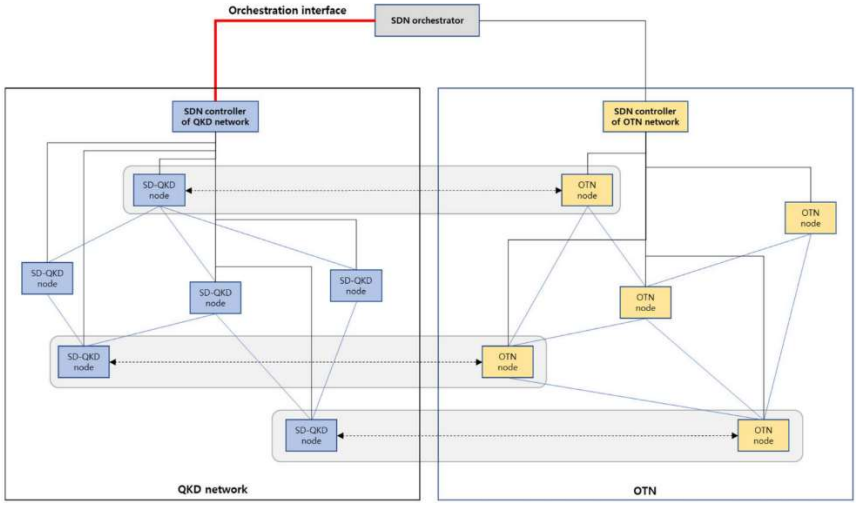

Figure 8-4: SDN orchestrator orchestrating the SDN controller of QKD network and the SDN controller of the OTN network

In the context of DISCRETION, to enable end-to-end service provisioning through different network domains, the SDN orchestrator has the following functions:

- Configuration of the different network domains through each SDN controller and the allocation of secure application entities for service provisioning according to the end-to-end QKD service requests
- Multi-domain path calculation across the different network domains.
- Reserve, configuration and release requests of the end-to-end QKD service provisioning with the inter-domain connections between secure application through orchestration interfaces.
- Requests to change the established path calculation with constraints from a network operator.
- Auto-discovering new QKD nodes and QKD links under each SDN controller in the QKD network managing an abstracted view of the QKD network topology.
- Inventory monitoring of QKD resources available, like QKD key provisioning from the key management system in each QKD node and each QKD link.

The orchestration interface between the SDN orchestrator and the SDN controller of the QKD network needs to be defined to address the outlined functions of the SDN orchestrator. The SDN controller is a server through the orchestration interface, and the SDN orchestrator is a client in terms of communication between them. This orchestration interface covers the following operations:

- **Discovery of QKD network topology:** The model allows to compose the overall multi-domain network topology with the information from the underlying SDN controllers.
- **Monitoring of QKD network status and resource inventory:** It includes the possibility of monitoring QKD network status and even Key resources available from the key management system in each QKD node.
- **Monitoring of end-to-end QKD service status:** the SDN orchestrator needs information about external applications, local QKD node, remote QKD node, and QKD service links between local QKD node and remote QKD node for inter-domain end-to-end QKD service.
- **End-to-end QKD service provisioning with path calculation:** The SDN orchestrator can request path calculations with constraints to the QKD network SDN controller. The SDN controller repeatedly provides a candidate paths list that satisfies the constraints received from the SDN orchestrator until the SDN orchestrator finally chooses a path and decides to deploy it for the QKD service link in the QKD network.

- **Notifications:** The SDN orchestrator can receive notifications from each domain SDN Controller.

Now it is provided the base use case in which the SDN orchestrator and the SDN controller in a QKD network and the SDN controller in an OTN interact to perform the end-to-end QKD service provisioning across the QKD network and OTN. The workflow in Figure 8-5, as a sequence diagram, shows the interactions and the exchange of information among the SDN orchestrator, the SDN controller in a QKD network and the SDN controller in an OTN.

In this scenario, the SDN controller in a QKD network and the SDN controller in an OTN do not directly communicate with each other, and the SDN orchestrator requests the SDN controller in a QKD network and the SDN controller in an OTN respectively to execute the necessary tasks in their domains for the end-to-end QKD service provisioning.

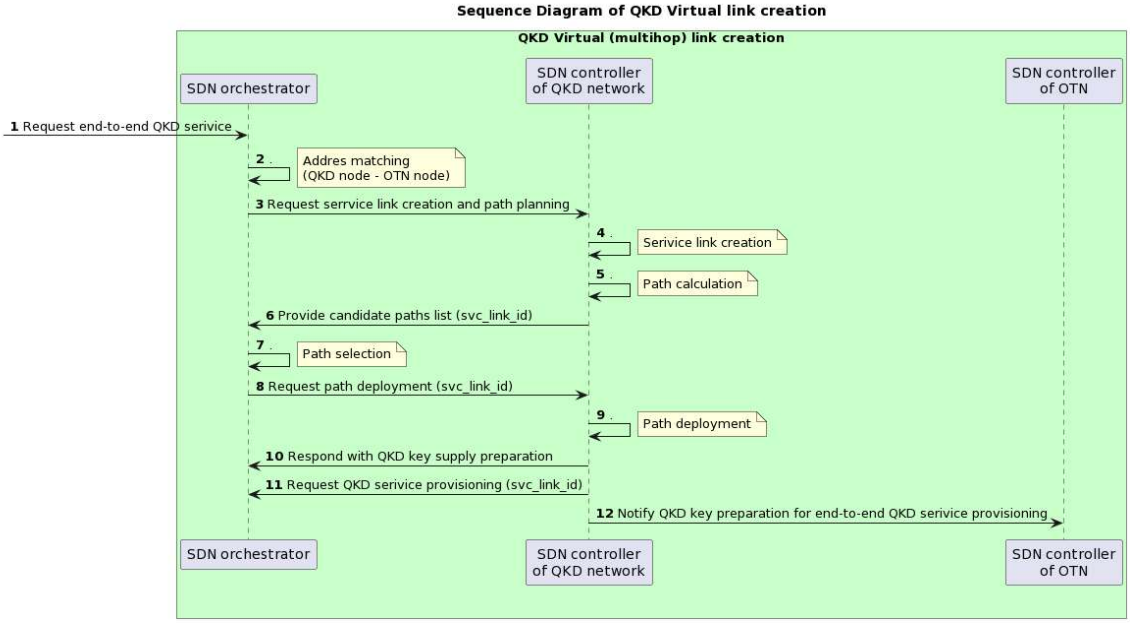

Figure 8-5: Sequence diagram for end-to-end QKD service provisioning by an SDN orchestrator

E2E QKD service provisioning by an SDN orchestrator:

- An E2E key request is done as a QKD service from the SDN orchestrator with information about the secure application and their QoS requirements.
- The SDN orchestrator finds the appropriate SDN controller in a QKD network and the SDN controller in an OTN. Then the SDN Orchestrator matches the address of the OTN node under the SDN controller in an OTN to receive Keys from the appropriate QKD node and the address of the QKD node under the SDN controller in the QKD network to provide Keys for the secure application entities in the OTN node. The SDN orchestrator has the address of all QKD nodes in the QKD network as well as the QKD network topology and inventory information.
- The SDN orchestrator requests for a QKD service link creation and path calculation for Key transport from the SDN controller in a QKD network.
- The SDN controller in a QKD network issues QKD service link id, calculates a candidate paths list to fulfil the path calculation constraints from the SDN orchestrator, and responds with the calculated candidate paths list to the SDN orchestrator. The SDN orchestrator can request for a new candidate path until the SDN controller for the QKD network satisfied the request of the SDN orchestrator.
- When the SDN controller receive information about the path for deployment from the SDN orchestrator, the SDN controller in a QKD network deploys the path in the QKD network. Then, QKD nodes generate QKD derived keys and transport them along the deployed path through the

QKD network until two QKD nodes in the QKD service link are ready to provide Keys for the secure application entities in OTN nodes.

- When the SDN orchestrator receives a response that Keys are ready for supply, the SDN orchestrator requests E2E QKD service provisioning from the SDN controller in a QKD network. After that, it notifies to the SDN controller in an OTN the preparation of Keys in the QKD network. The OTN nodes can start to receive Keys from the designated QKD nodes in the QKD network based on address matching information.

9. SDN BASED SOFTWARE DEFINED RADIO

9.1. Consolidated Requirements, Scenarios and Use Cases

During WP3 of the DISCRETION project, research was conducted to better understand how the interoperability between the SDN and SDR-based devices could be achieved. During this research, particularly considering military applications for mission based tactical networks, one thing stood out: This kind of network is typically organized as standalone hierarchical structure, with a Mission Command and Control Center (CCC) at its top.

Considering that the CCC is typically deployed to the field of operations, and that it may be (and usually is) physically disconnected of the secure operational/strategic networks, like the various integrated SDNs described in the above sections, an approach involving a full integration of SDNs seemed unrealistic. Instead, a hybrid approach was chosen, with two distinct scenarios, corresponding to two phases of a typical mission:

- A **Setup Phase**, where the CC is still connected to the fixed infrastructure and can benefit from the strategic/operational network services for preparing the mission (e.g., key distribution, initial comms equipment configuration);
- A **Mission Phase**, where both the CC and the radio devices (SDR) are deployed to the field, on a tactical environment, and disconnected from the Operational Network.

While the first scenario still explores the synergies with the DISCRETION integrated SDN by using it as an infrastructure, the second scenario has its focus on the tactical aspects of the mission, and does not necessarily rely on the operational/strategic networks.

The Use Cases that are explored on these scenarios, along with deeper view into the architecture, are described on Deliverable D3.2.

9.1.1. Scenarios

A key aspect of DISCRETION is the usage of QKD for securing key distribution in an optical environment. Nevertheless, at the present stage, QKD technology is not mature enough to be considered within a tactical scope, where mobility is a key aspect. Nevertheless, the mission-based nature of tactical networks provides a workaround that can overcome (or at least mitigate) this limitation: If the keys that are to be used throughout the mission are distributed to their users before the mission starts, then this can be done in a fixed, physically protected environment, and won't interfere with the mission itself. The figure below illustrates these two moments:

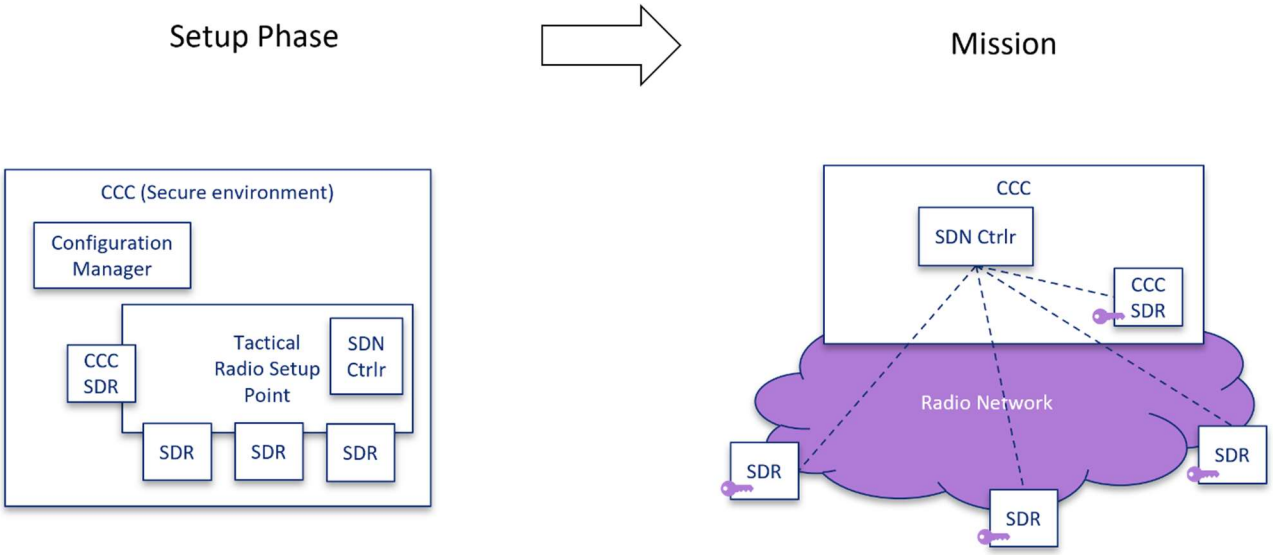

Figure 9-1 - Setup and Mission phases

At Setup Phase, all tactical equipment that will carry cryptographic material is gathered at a secure environment, where keys can be securely delivered by the Key Management System: The Tactical Radio Setup Point (TRSP). The TRSP provides a framework for distributing mission keys to the tactical equipment attached to it. Also, all initial configuration for the involved equipment is performed at this moment.

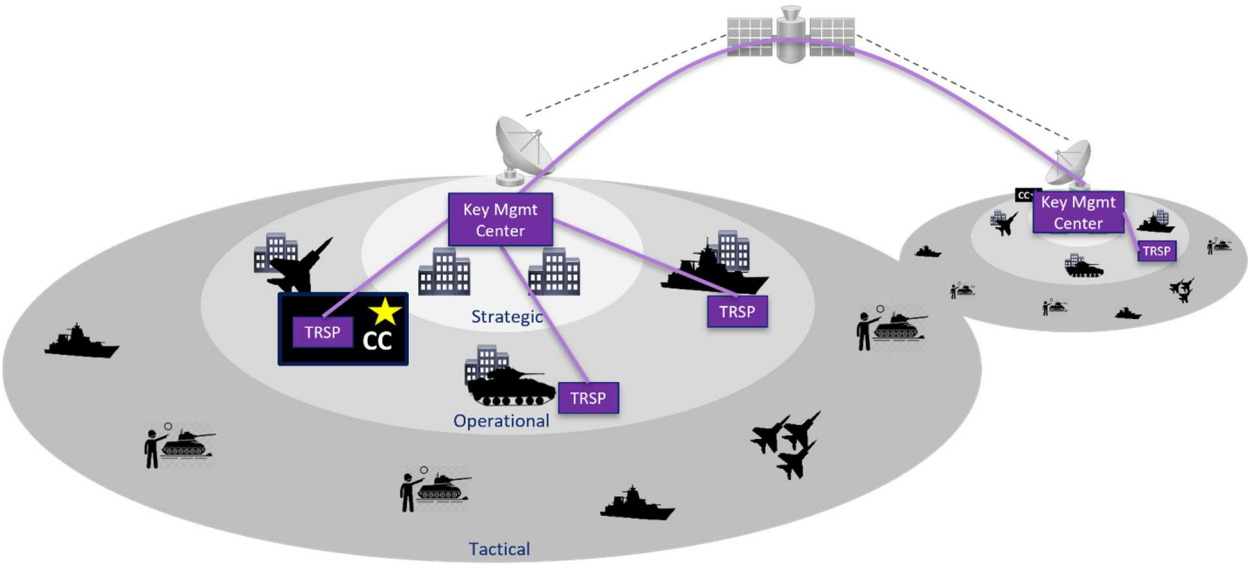

Figure 9-2 - Setup Scenario

For the same mission, more than one TRSP may be involved in key delivery, thus enabling scenarios where missions have more than on starting point, or even coalition scenarios where keys may be distributed across different military authorities, depending on the capabilities for key distribution of the whole system. The figure below illustrates this.

During the Mission, forces are deployed, and all fixed connections are (or at least may be) severed. The CCC becomes the master of all communication in the mission scenario. Communication is kept secure using the keys that were pre-shared, and that are rotated according to a pre-established policy.

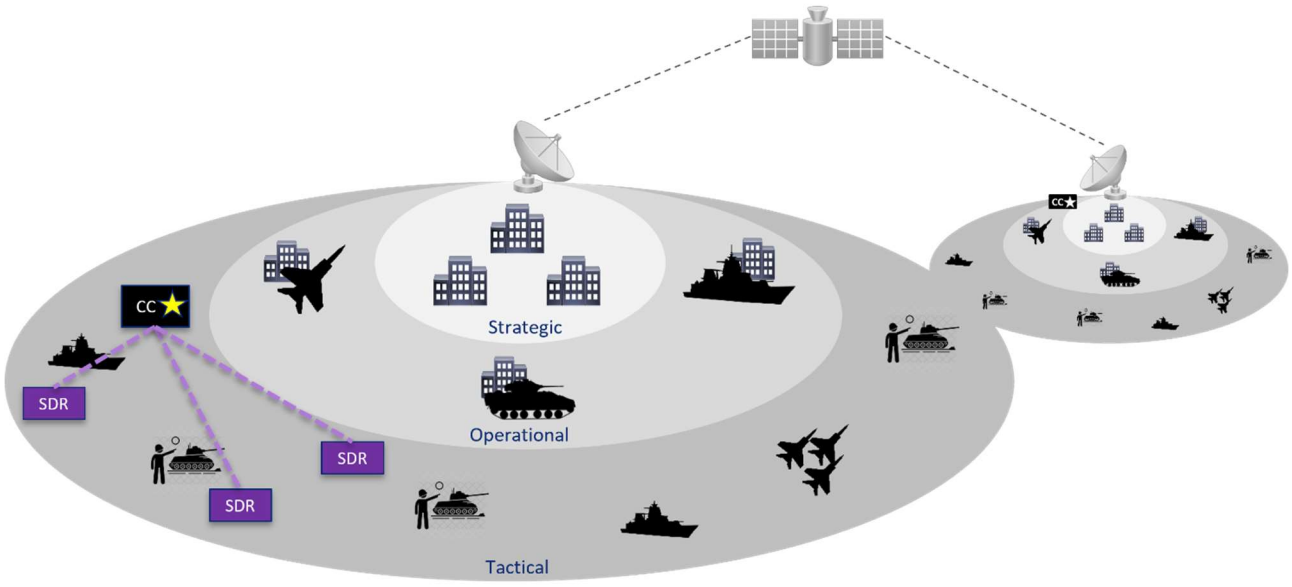

Figure 9-3 - Mission Scenario

The network that is established at a tactical level is an SDN where the controller and the applications that run and manage the tactical network are placed at the CCC. There, the situational awareness obtained from data collected on the field (from sensors, the SDR equipment, or other sources) drives decisions that are enforced to the network, eventually changing SDR equipment behavior and associations, in order to optimize their field performance or respond to specific events.

During the mission, the relation between this SDN and the Operational/strategic communications SDN is very limited or nonexistent.

9.2. Functional Architecture

For what is relevant, the basic architecture is one and the same for both scenarios. It is that of a typical SDN setup, featuring a number of control applications, a SDN controller and a number of controlled communication equipment – The SDRs. Still, other functions are mentioned for context at the particular scope of each scenario, especially the first – Setup.

As mentioned in the previous subsection, the proposed scenarios describe phases, namely, Setup and Mission. The functional architecture is explained taking this into account, as well as the perceived/assumed features of the tactical environment to be addressed. Namely:

1. A tactical scenario is always defined within a time bound mission;
2. There is a command and control structure with one CCC in charge (either at the top of a hierarchy or as a single instance. A single instance is considered in the project scope);
3. Before the mission starts (or forces are deployed), the CCC is connected to the Operational infrastructure in a secure way;
4. Before the mission starts (or forces are deployed), the SDR equipment that will participate will be directly connected to a Tactical Radio Setup Point (TRSP), which is connected to the Operational infrastructure in a secure way;
5. The TRSP may exist within the CCC or in a remote location.
6. When the mission starts, the CCC becomes the master of the tactical network. All connections that it had to the Operational infrastructure may be broken

9.2.1. Scenario flows

The setup phase takes place when the tactical devices are being prepared for the mission. Here, the devices are configured, and the keys are loaded into the devices. These keys will allow the devices to communicate securely during the mission phase. The setup phase occurs within a secure infrastructure, in which the SDR devices' management interfaces are connected to the Tactical SDN controller. This controller "represents" the SDR devices in the setup process described below.

The Setup process is initiated by the **Configuration Manager**, an application that would exist in the CCC environment, responsible for setting up the tactical environment (this manager is not in the scope of the project):

1. The Configuration Manager requests the distribution of a set of keys to a specific security application that exposes such a service in the Key Management System (this application is not in the scope of DISCRETION);
2. The App interprets the request and initiates the distribution of the key material by acting as Security Application with respect to the KMS;
3. The KMS distributes the keysets to all involved TRSP;
4. At each TRSP, on receiving the keys from the KMS, the keys are delivered to the SDR devices
5. The configuration Manager delivers the initial configurations for the SDR devices to a Configuration Agent, which acts as an SDN Application to enforce configuration delivery to the SDR devices through the Tactical SDN controller.

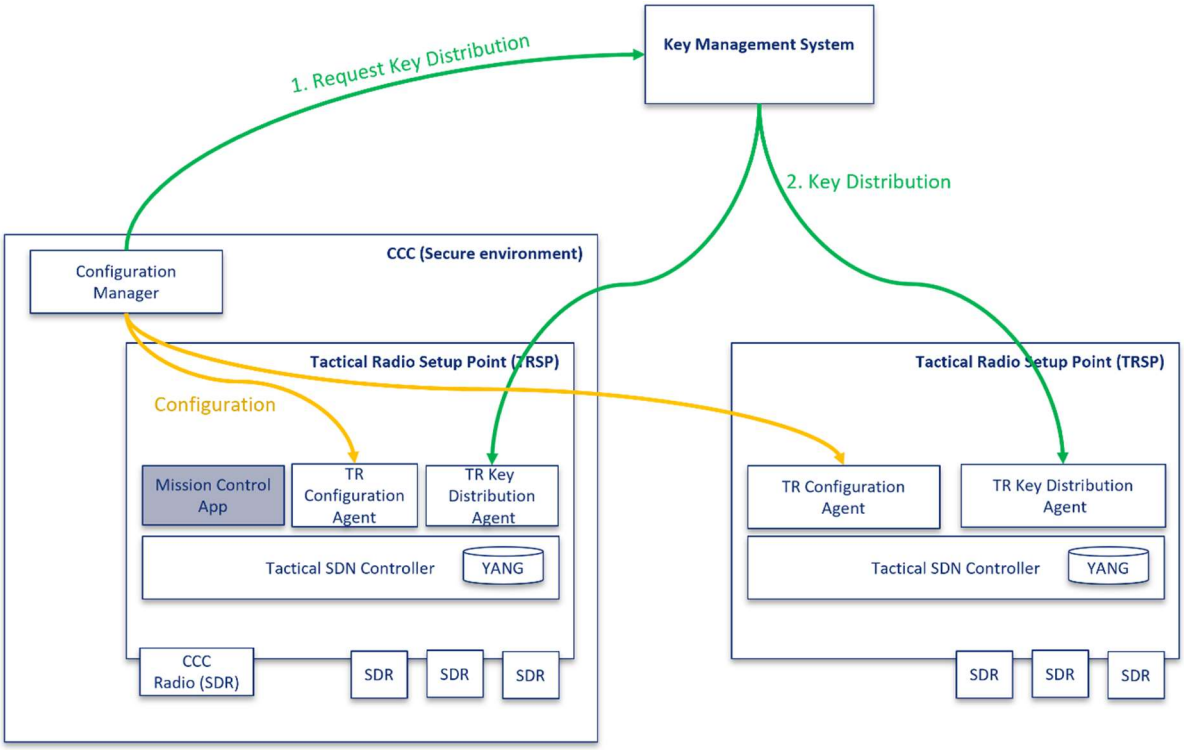

Figure 9-4 - Architecture for Setup Scenario

Once the SDR devices are configured and contain a set of pre-shared keys, the setup phase is completed and they are ready to be deployed in a tactical environment. The CCC itself is also deployed, and its fixed connections to the operational network may be severed. From this moment on, the Key Distribution Agent and the Configuration Agent take different roles, mostly providing services to **Mission Control Apps**, which take over the control of the tactical network. Figure 9-5illustrates this scenario. These apps

are not in the scope of DISCRETION, as they will be strongly specific to military applications featuring tactical behavior, but a simple demo application may be developed for demonstration purposes.

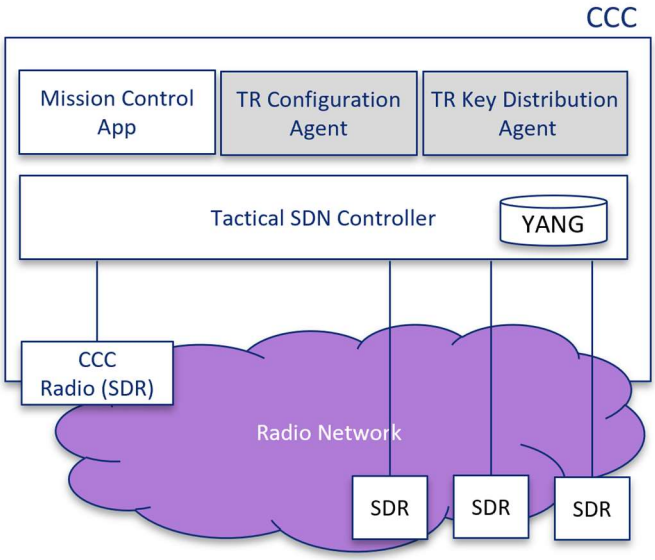

Figure 9-5 - Architecture for Mission Scenario

More than one Mission Control Apps may exist, taking care of distinct aspects of network control during the mission.

9.2.2. Tactical SDN Controller

The **Tactical SDN Controller**, which, together with the Mission control Apps, the TR Configuration Manager, the TR Key Distribution Manager, and the SDRs, compose the tactical architecture, illustrated in Figure 9-6, is composed of three main blocks:

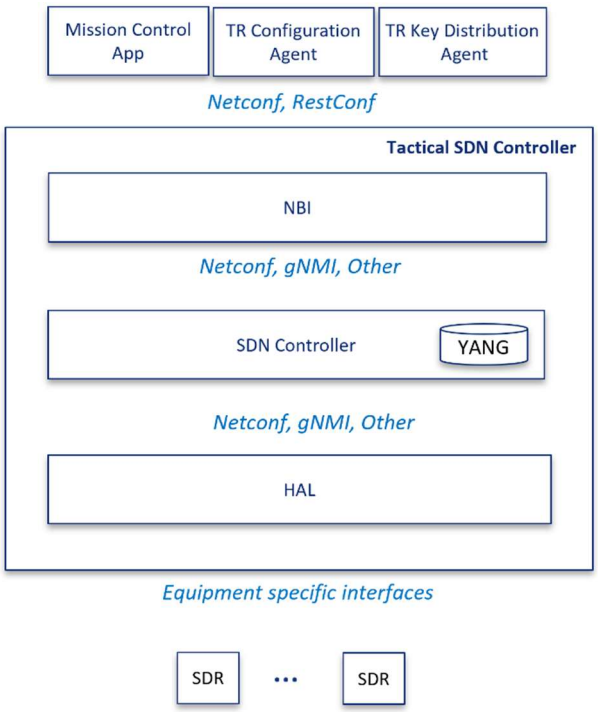

Figure 9-6 - Detailed Architecture for the Tactical SDN Controller

- **SDN Controller**: Either a commercial or purpose-built controller that exposes the SDR resources to be used/controlled/managed by software applications. YANG is used to model the network/resources, which are typically reached via southbound interfaces using protocols like Netconf or gNMI. Northbound interfaces are typically Netconf/gNMI or Restconf.
- **NBI**: Northbound interface adapter, to align the SDN controller interfaces to the Southbound interfaces used by applications.
- **Hardware Abstraction Layer (HAL)** is responsible for interfacing between the SDN Controller and the KMS, and the SDR devices. The HAL module has two main components, the northbound component is ideally independent of the SDR devices in use. It should interact with the SDN controller, for instance to obtain the configurations and keys to be loaded into the devices while having little knowledge about their architecture. On the other hand, the southbound component of the HAL module is intended to apply the configurations and load the keys into each specific device while it is docked in the TRSP, but also to reach the device and receive information from it when it is on the field, limited by the features that are provided by the equipment. This component uses the device specific standards to communicate with the SDR devices, based on the abstract information received by the HAL's northbound component.
  The HAL southbound interfaces are heavily dependent on the SDR device in use. Notably, the military devices might use classified standards for this purpose, or simply proprietary interfaces/APIs. The HAL northbound interface is aligned with the SDN Controller, typically Netconf/gNMI. The specific interface to be used in DISCRETION will be defined in design phase. It will depend on whether the HAL will expose a YANG model of the SDRs to the SDN controller and how it will be reached.

A number of applications consume the resources exposed by the SDN Controller and take care of the control flows on the SDR SDN:

- **TR Configuration Agent** (Tactical Radio Configuration Agent): is the application that exposes the SDR resources under the SDN Controller to be configured, either by a northbound application (via a NBI) or by the other SDN applications.
- **TR Key Management Agent** (Tactical Radio Key management Agent): is the application in charge of managing the keysets that are to be used by the SDRs. It can request the provision of keysets to an external Key Management System or internally generate its own keysets.

- **Mission Control Apps:** these are the applications that implement the rules that define the tactical behavior of the network.

9.3. Operation

This subsection briefly describes how the interoperability between the SDN and SDR-devices might be demonstrated, using the scenarios described above to illustrate what can be achieved for the preparation of a tactical environment, as well as for the tactical scenario. To do this, a hypothetical use case is described: A mission, comprising its preparation and the evolution of the tactical scenario, featuring a simple OODA loop for the SDR, featuring a mechanism for optimizing transmission parameters according to environmental conditions.

Initially, when the mission is ordered, the tactical resources that will be involved are all attached to an operational network. The preparation of the mission for the Tactical Radio Resources is assumed by the Configuration Manager, which will coordinate all the actions. Then...

1. Either a human operator or an API creates a request for the configuration of a tactical scenario on the Configuration Manager. The request includes
   - Identification of all the Tactical Radio equipment to be involved
   - Where the equipment is (which TRSP it is connected to)
   - Mission duration
   - Key usage policy (kind of keys to be used, rotation frequency, etc)
   - Radio configuration profiles to be used, according to roles and models of the radio equipment
   - User Groups, Slices, Routing, etc... according to the capabilities to be exposed by the mission control apps
   - Other, to be specified
2. Configuration Manager takes care of the distribution of Keysets and configuration of the involved terminals, as described above, in the Functional Architecture.
3. When the mission starts (e.g., forces are deployed), the CC is moved to the operations theatre and disconnected from the secure infrastructure, as well as Radios that were on other TRSP
4. From this moment on, all SDR equipment involved is under the control of the SDN controller, running a number of Mission Control Apps
5. For this description, we can consider that one of these apps is called RadioTweak and is in charge of analysing data collected from the radios on the field and, based on that analysis, determining the best transmission/reception parameters for the radios and enforcing the right configurations. We assume here that in this tactical scenario this kind of metrics are available and can be collected, and that radio setup reconfiguration is possible (At the moment of issuing this deliverable this is not guaranteed in DISCRETION)
6. Transmission/reception metrics and relevant SDR events are collected by the SDN Controller and made available to RadioTweak
7. RadioTweak is aware of overall network conditions and probably will have knowledge of local policies. This may de determine that some radio parameters should be changed. Then RadioTweak uses the SDN Controller to enforce the change of configuration (eventually this is done by selecting one from several pre-set configurations, in the simplest case)

Other operations could be described, namely those involving the management of keys in the tactical environment, like Key Revocation or Forced Key Rotation, but this are left for a more thorough specification considering what can be achieved, given the SDR equipment features and limitations. Namely, Deliverable D3.2 will provide a deeper view into the SDN SDR architecture and a number of Use Cases, detailed according to this architecture.

10. DATA LAYER SDN

According to the considerations in 6.5, DISCRETION is considering a Distributed Hierarchical SDN architecture in its black network, counting with two levels: Hierarchical level controller (H-SDN-C) and Domain SDN Controllers. The first level, H-SDN-C, will be the one to be in communication with the OSS and SDN Applications layer through its NBI, and with the IP-SDN-C (in the case of the IP data layer) with its SBI.

IP-SDN-C will implement some functionalities also present in H-SDN-C, that will apply to its domain, and abstract as required to H-SDN-C, although there are some functionalities that may be hosted just in H-SDN-C.

Inventory Module, PCE, Configuration Module, or Performance Monitoring Modules are expected to be implemented both in H-SDN-C and IP-SDN-C, the functions implemented in H-SDN-C with a multi-layer, multi-technology scope.

It the case of the ALTO module, although there may be different approaches, it is foreseen that it resides just in H-SDN-C, since the proposal is to use it as a network exposure function to enable the Red Network Control Plane to select the most appropriate network resources available. Current definition of ALTO needs to be augmented several functionalities such as topology abstraction, or support for other than IP technologies.

This service will have its own interface exposed, being only reachable by other Hierarchical level controllers (the main purpose is to have a high-level – yet complete - vision of the topology, not too focused in each sub-domain situation).

In the Figure 10-1 we can see how features are replicated and structured in modules. In the H-SDN-C we will control how IP-SDN-Cs are displayed and it will communicate with the different instances by its SBI. ALTO service will be deployed over a one-directional connection to provide a safe way to communicate with the (private) Red Network. To achieve this goal, the communication should count with two one-directional channels (using Data Diodes to provide isolation between zones) and the service will be offered using the PUB-SUB pattern through a Middleware which will control a list of buses dedicated for topics, being each topic a different type of the ALTO maps available.

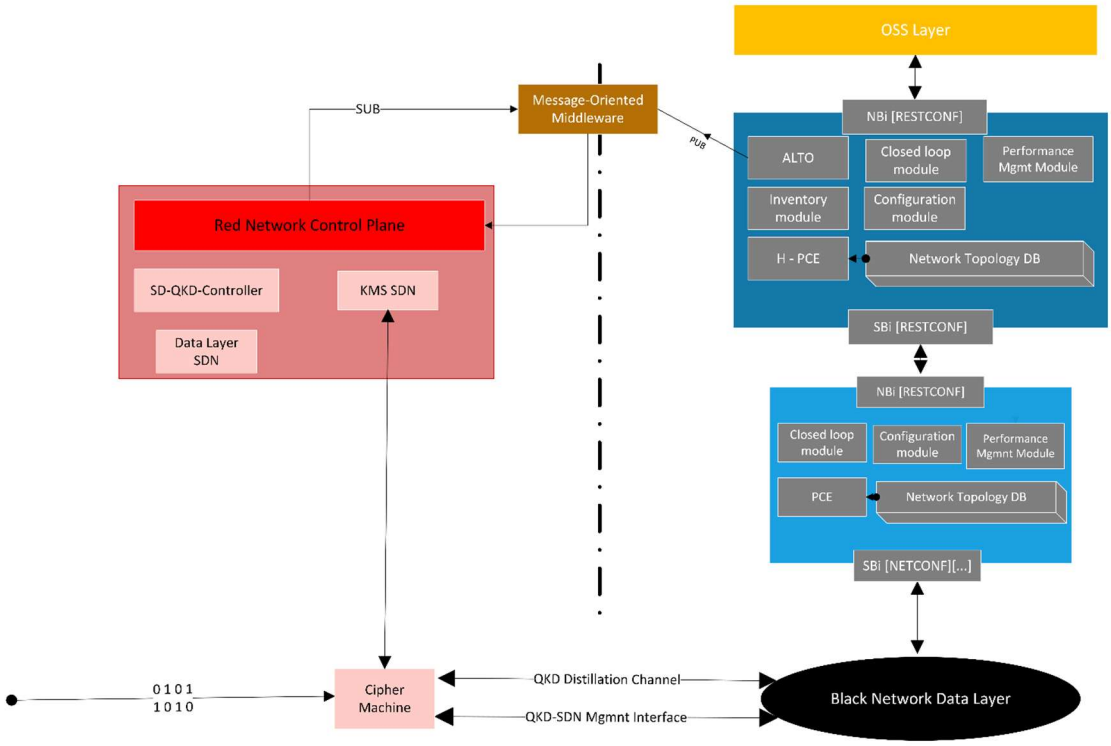

Figure 10-1. Hierarchical SDN architecture block model. Data layer Detail. Source: own elaboration

Next sub-sections will detail the features offered by Data Layer SDN framework, and the functional building blocks that compose it.

10.1. Features

The use of Software Defined Networks permits the introduction of concepts like network programmability and automation in IP and optical transport networks. This technology provides the ability of initialize, manage, control, and change the behaviour of networks dynamically and, when implemented through open interfaces, allows for the interoperability of different network technologies and domains. Data Layer SDN DISCRETION components address the following set of categories and use cases:

1. **Service Provisioning**: Service provisioning includes the setup of a service and the management of the data of the service. Some of the services that DISCRETION could provision are: L2VPN, L3VPN between network elements, topology update.

2. **Resiliency**: Due to the applied technology, it can provide and maintain an acceptable level of service in the face of failures that challenge normal operations. SDN may be used in conjunction with network planning tools and real time topology information to pre-provision a set of disjoint paths from the working LSP according to predefined SLAs, or to enable dynamic restoration based on path computation capabilities

3. **Inventory Support**: The persistent inventory database resides in the global orchestrator. This functionality facilitates network resource catalogue management for multi-vendor and multi-domain networks. This module will be used in SDN to upload of the Hardware inventory of network elements.

4. **Topology**: Construction, visualization, and export of the IP topology to upper layers. This module is used to know the topology of the networks at every time. This module will be inside SDN controllers in a Database that will be updated automatically using real time protocols to provide SDN applications in upper layers with a real-time topology map.

5. **PCE (Path Computation Element):** It is the module used in a network path or route based on a network graph. The PCE is a module located inside the SDN controller and PCE will use the Real Time Database Topology to the update information of the topology. It can be applicable in intra-domain, inter-domain, and inter-layer contexts:
    a. Intra-domain path computation to determine the different paths that can be in a domain.
    b. Inter-domain path computation involves the topology, routing, and policy information from multiple domains.
    c. Inter-layer path computation is the use of PCE where multiple layers are involved and when the goal is to compute the path of one or multiple layers taking into account the topology and resources of the layers. Thus, this module can be applied to the infrastructure layer and SDN control layer. For example, a channel for Quantum Key Distribution must be set up, and to realize this, it needs the computation of the path to send it. Irrespectively of multi-layer PCE deployment architecture, data layer PCE must support multi-layer path computation process.

    On the other hand, there are diverse ways to apply PCE in a network taking into account the distribution of the PCEs and the possible paths computation:

    a) In a single PCE path computation, when there is only one domain, to compute the given path only must use a single PCE. It can be multiple PCEs in a domain, but only one PCE per domain if there is only a single path computation
    b) In a multiple PCE path computation, is obviously that multiple PCEs are used to compute the path in a domain, because in this case there are multiple paths possible.

c) One more possibility is to use one Centralized PCE to compute all the possible paths in a domain. This case is called Centralized computation model. This model only can use single PCE path computations.
d) Finally, the distributed computation model is the case in which the computation of paths in a domain is being shared among multiple PCEs. In addition, this model could use either single PCE path computation or multiple PCE path computations.

At the time of writing this document there is not final decision on the best approach for DISCRETION although the hierarchical distributed cascaded approach (option d) seems to be a suitable one.

6. **Traffic Engineering:** This functionality allows traffic to be directed using MPLS (Multi-Path Label Switching) tunnels or segment routing paths. This leads to increase the efficiency of the use of the resources of the network mapping the traffic flows to the available resources, and improving the network management, including troubleshooting, to solve difficult failure situations and carry on. Each time, the scenarios which the networks are subjected to are more complex, such as large single domain environments, multi-domain or multi-layer networks, and require the usage of algorithms for efficiently computing end-to-end paths. To achieve this goal (drive the traffic correctly taking into account the network resources) it is necessary to use SDN controllers with the functionality described above (PCE). The objective of the use of traffic engineering is to reduce overall operating costs through more efficient use of the network resources, including link occupation, traffic rerouting, network availability, and others. In addition, traffic engineering avoids situations of network overload, while other parts of the network remain underused, and thus, ensures the optimal path for the network traffic and allows for the implementation of mechanisms to protect the traffic against network failures. Traffic engineering provides topology and LSP info via BGP-LS, initiate LSPs, compute path requests, receive delegations and receive reports.

7. **Performance Management:** It refers to sending performance and telemetry information to the relevant management systems (e.g. to the H-SDN-C, to a telemetry engine, etc.)
The Simple Network Management Protocol (SNMP) has been traditionally used for the Performance Management. SNMP has been activated in the network elements as an SNMP Agent, and some EMS/NMS subsystems as SNMP Managers with Proxy Forwarder capabilities for feeding Performance Management systems. SDN overcomes the traditional direct extraction (directly from the network elements) or proxy based (though a proxy specifically designed for the purpose) methods with two mechanisms to provide the performance and assurance functionality in SDN-ready elements.

   - SDN-based Extraction using the IP Agnostic SDN Controller as a proxy with standard OpenConfig data and information models.
   - Real-time SDN-based Extraction using Telemetry Engines and exposing the information with a standard bus. This approach solves the challenges in terms of scalability that, depending on the nature of the extracted information, a simple proxy agent in the controller could suffer.

8. **Network Creation: I**nitial configuration of a network, concretely, the configuration of the network elements like routers, switches. There are two important points:

   - Day zero configuration: Some of the day 0 configurations tasks include setting up the interfaces, management of the network, stablish static or dynamic IPs, SSH keys, and enable NETCONF. In addition, the configuration of the network elements of the IP plane like routers, switches, etc.
   - Update of routing parameters: establish the different paths, configure the default gateway, etc.

9. **Support to Fault Management:** is one of the most relevant functional areas in terms of impact to customer experience / service quality. Fault Management application provides the functionality needed to manage failures in the network and service infrastructure.
Network management means understanding the details of a network or service failure.

ISO (International Organization for Standardization) [111] specifies that a fault management system must contain an FCAPS (Fault management, Configuration, Accounting, Performance, and Security) framework. A fault management system must have at least:

- Alarms collection in real time and synchronization.
- Alarm storm protection.
- Availability monitoring through management heartbeat.
- Alarms reduction through classification (severities), filtering, maintenance mode, top level graphical view.
- Alarms troubleshooting through historical alarms, context-sensitive menus, and photo-realistic equipment views.
- Alarm notifications through streaming, SNMP, e-mail, and SMS.

IP-SDN-C need to cover these functionalities as part of its alarm handling capabilities.

10. **Security:** Besides the security considerations discussed in sections 3.4 and 5, it is worth to mention that, within the scope of DISCRETION, the flexibility of SDN and the availability Quantum Key Distribution (QKD) technology, may be combined to secure the SDN control plane with the use of quantum generated keys, to increase robustness against attacks to DISCRETION control plane.

10.2. Functional Architecture

10.2.1. Functional blocks

10.2.1.1. Configuration Module

DISCRETION IP Data Layer SDN System must expose a set of APIs to allow the programmability of the IP forwarding plane and IP networks. These APIs must be aligned to standards to ease of use for user and third party SDN applications and/or controllers.
The communication between the hierarchical SDN controller and the SDN controllers is made by using RESTCONF protocol, which is used for obtaining and setting the network state in the controller. The communication between the SDN controllers and the network elements will be through NETCONF protocol, for configuring all the network services paths, with measured SLA's.

This module will implement the functionality for network creation and service provisioning, interacting with the rest of the modules when necessary. Among others, Configuration Module will:

- Get the topology of the underlying network from the Real Time Network Topology Database and Path Computation Element optimal paths information to provision a new service in the network
- Interact with element responsible for closed-loop process to reflect the changes that are producing in the network path. Closed-loop module will be also responsible for extraction, mapping and translation of the information from the network monitoring function.

10.2.1.2. Inventory Module

One of the most important things in this architecture is to discover the components between all the domains layer SDN orchestration and there should be a way to store this information. Thus, this module is a datastore that DISCRETION must have to store the neighbour elements (like nodes, cards, ports, transceivers, borders nodes, interior nodes…). This information will be stored separately and used to conform the full inventory representation. In addition, DISCRETION should permit the validation of the components described above (the neighbour elements) connected between the domains/layers. If the components are not compatible, DISCRETION must notify.

When inventory systems are synchronised with the network, costs may be optimized both in terms of CAPEX and OPEX because the operator can use the network information for things like capacity management, accurate service design, service activation…

A very important thing in the DISCRETION architecture is to have a consistent and synchronized Inventory. Whether it will be a single or a domain-based inventory is a matter still to be explored in the Design phase.

10.2.1.3. Real Time Network Topology Database

This is a real time Database that stores the information of the network topology (nodes, border nodes, interior nodes, connections between network devices, connection with SDN controllers, links…). This is a module integrated on the SDNs controllers of the architecture of DISCRETION. The functionalities of this module are:

- Validate traffic routes.
- Overflows.
- Plan network growth.
- Validate failures in the network.
- Perform network maintenance.

In addition, This Database performs the discovery of the L2 and L3 topology using LLDDP and/or BGP-LS. The PCE makes a connection to this database to take the information about the path to follow using the protocol BGP-LS. The database must be able to collect L2 and L3 topology using RESTCONF protocol and must be able to subscribe to the topology updates using gRPC [57]. Furthermore, the database must be able to correlate the services with the network topology. In addition, DISCRETION must supply topology modules to export, in read-only mode information about the network topology of the layers. These modules are defined with IETF Topology Yang models.

10.2.1.4. Path Computation Element

Finally, the distributed computation model is the case which the computation of paths in a domain is being shared among multiple PCEs. DISCRETION will use this model because there will be multiple domains and multiple providers too. In addition, this model could use either single PCE path computation or multiple PCE path computations. PCE needs the information of the underlying networks (IP, Optical and QKD) for compute the best path to take and the answer is the Real Time Database Topology. The PCE module is integrated in the SDN controllers, and this module realizes a connection to the Topology module to take the topology information.

The protocol used for the request/response for accessing the services of a PCE is PCEP (Path Computation Element Communication Protocol) [112]. Is the protocol for the communications between a PCC (Path Computation Client) and a PCE, or between two PCEs. It says, the PCEP is a special set of rules that allows a PCC to request path computations form PCEs and lets the PCEs return responses. This protocol operates over TCP, and the messages are:

- Open And Keepalive Messages: used to initialize and maintain a PCEP session.
- PCReq: a PCEP message sent by a PCC to a PCE to request a path computation.
- PCRep: a PCEP message sent by a PCE to a PCC in reply to a path computation request.
- PCNtf: a PCEP notification message either sent by a PCC to a PCE or vice versa.
- PCErr: a PCEP message in the case of a protocol error condition.
- Close message: a message that indicates that the PCEP session is over.

The phases of the PCEP protocol are:

1. **Initial Phase:** this phase is separately into two different phases:
   a. First of all, is necessary to establish a TCP connection (3-way handshake) between the PCC and the PCE.
   b. After that, is necessary the establishment of a PCEP session.
2. **Session Keepalive:** once a session has been established, a PCE or a PCC has to know that the session PCEP is available to use. Due to this, is necessary the exchange of this message to indicate that the session is open and ready to use.
3. **Path Computation Request Sent by a PCC to a PCE:** The PCC knows that the session is available after the previous message. Thus, if the PCC needs to compute the path to follow, it will send the Path computation request to the selected PCE. This message contains a variety of objects that specify the attributes for the path, like for example: IP address in the format a.b.c.d, destination IP address in the same format, etc.

4. **Path Computation Reply Sent by The PCE to a PCC:** once the PCE received the request Path Computation, having in count the possible paths there are two possible messages to answer to the PCC:
    a. **Positive:** In this case the PCE manages to compute a path that satisfies the attributes and required constraints from the PCC. The PCE returns the set of computed paths that the PCC requested.
    b. **Negative:** In this case there is not a possible path having count the request of the PCC. Thus, the PCE send a negative message to the PCC saying that is not possible the request. After that, the PCC has to send another different request with other appropriate attributes maybe.
5. **Notification:** Sometimes, there are circumstances in which the PCE must notify to a PCC. For example, if suddenly the network is overloaded. Then, the PCE must send a message to the PCC saying that de request has been cancelled because of unexpected problems.
6. **Error:** The PCEP Error message is sent in several situations: when a protocol error condition is met or when the request is not compliant with the PCEP specification (for example: capability not supported…)
7. **Termination of the PCEP Session:** When one of the PCEP peers decides to finish the session, firstly sends a PCEP Close message to the PCC and after that closes the TCP connection.

This module will be in both levels of SDN Controllers to realize a division of the tasks, like a Divide-and-conquer process. The objective is to have a Domain Level module to realize a domain-level path computation and a hierarchical module to coordinate and to realize a high-level path computation.

10.2.1.5. Performance Management Module

The Performance Management Module is responsible for managing monitoring processes in the IP-SDN Controller where external users, agents or components (like SDN controllers) can subscribe to receive information like metrics or key performance indicators (KPIs), coming from various parts of the system depending on their nature. We propose that a KPI can be composed of one or more metrics for one or more subscribers. A KPI can be the outgoing traffic of a switch device of the underlying network. [113]

This module will implement the fault management, telemetry, and monitoring functionalities interacting with the rest of the modules when necessary. Among others, Performance Management Module will:

- Provide multiple metrics and KPIs.
- Manage multiple external requests.
- Generate alarms and notifications to the external requestors (User, 3rd party application, SDN controller).
- Allow requestors to add, modify, and visualize metrics and KPIs

10.2.1.6. Closed loop module

DISCRETION IP Data Layer SDN System needs a module to facilitate the automation of several services instead of using manual network operations.

This module provides:

- Automated network resource discovery (i.e., topology discovery).
- Automated Network configuration through open standards like NETCONF.
- Automated service provisioning to create and management of the network services at a high-level.
- Automated low configuration to the underlying network elements.
- Service Restoration: in case of a connectivity service failure, service paths automatically reroute.
- Service Optimization: automatic administration of Connectivity Services.
- Zero Touch Provisioning: process through which a device can be inserted into the network without the need of a human configuration.

Most of actions taken by this module will be realized at a Domain Level but every action, even the Domain Level ones, will be announced to the hierarchical closed-loop module, in order to have a complete vision of the network situation.

10.3. Operational Workflows

Before depicting the workflows, it is necessary to give a particular name to the controllers and modules for the IP Data layer:

- **Hierarchical SDN Controller:** H-SDN-C will be the name, (being H: Hierarchical)
- **Ip SDN Controller:** IP-SDN CONTROLLER. The name of the modules will be:
  - **Configuration Module:** Configuration Module
  - **Inventory Module:** Inventory Database
  - **Real Time Network Topology Database:** RTN Topology DB
  - **Path Computation Element:** PCE
  - **Performance Management Module:** Management Module.
    - **Monitoring Component.**
    - **Management Database:** MgmtDB
    - **Metrics Database:** MetricsDB
  - **Closed Loop Module:** CL Module

10.3.1. Network Creation

This use case covers all Day 0 operations, with alll required initial configurations for a network element. Next figure shows Network Creation Worfklow to configure all the network elements of the underlying network:

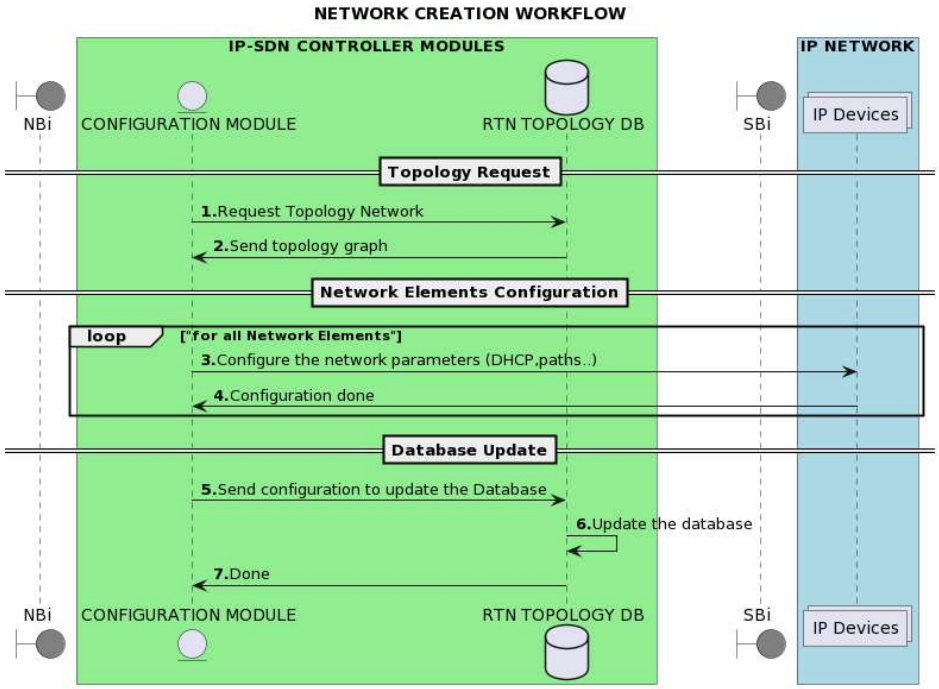

Figure 10-2: Network Creation Workflow.

Steps 1-2: Configuration Module of the IP-SDN-C requests the topology of the underlying network to the Real-Time Network Topology Database to configure the network.

Steps 3-4: Once Configuration Module has the information of the topology, is time to configure the underlying network. First of all, is necessary to assign an IP address to all network elements. This configuration will be done by NETCONF protocol through SBI interfaces. The underlying network will send a message to the Configuration Module indicating that the configuration has been done.

Steps 5-6-7: Topology Database Update. Configuration Module sends the information to the Database, which updates the information. Finally, the Real Time Network Topology Database sends a message to the Configuration Module to acknowledge succesful update of the database.

10.3.2. Service Provisioning

The architecture for DISCRETION data layer SDN domain controller must provide support for the implementation of different use cases for network services provisioning. Although DISCRETION SDN framework should be generic so that they may support different kinds of services (namely services that are specific to the integrated nature of Optical + QKD + Data + SDR SDNs), L3 VPN and L2 VPN are common use cases and may have a relevant role:

- L3 VPN Service (draft-ietf-opwsaw-l3sm-l2nm-03), called L3NM. This is used to manage the Layer 3 VPN service provisioning within IP/MPLS Network. This data model can be used to facilitate communication between the orchestrator and the SDN controller. [114]
- L2 VPN Service (draft-ietf-opsawg-l2nm-00) called L2NM. This data model is similar to the previous but is used for managing L2 VPNs in the network. This module supports additional capabilities like exposing operational parameters, the selection of different transport protocols and it also serves as a multi-domain orchestration interface. [115]

As an example, workflow for L3 VPN Service, in the presence of traffic engineering mechanisms based on a PCE, and segment routing is described.

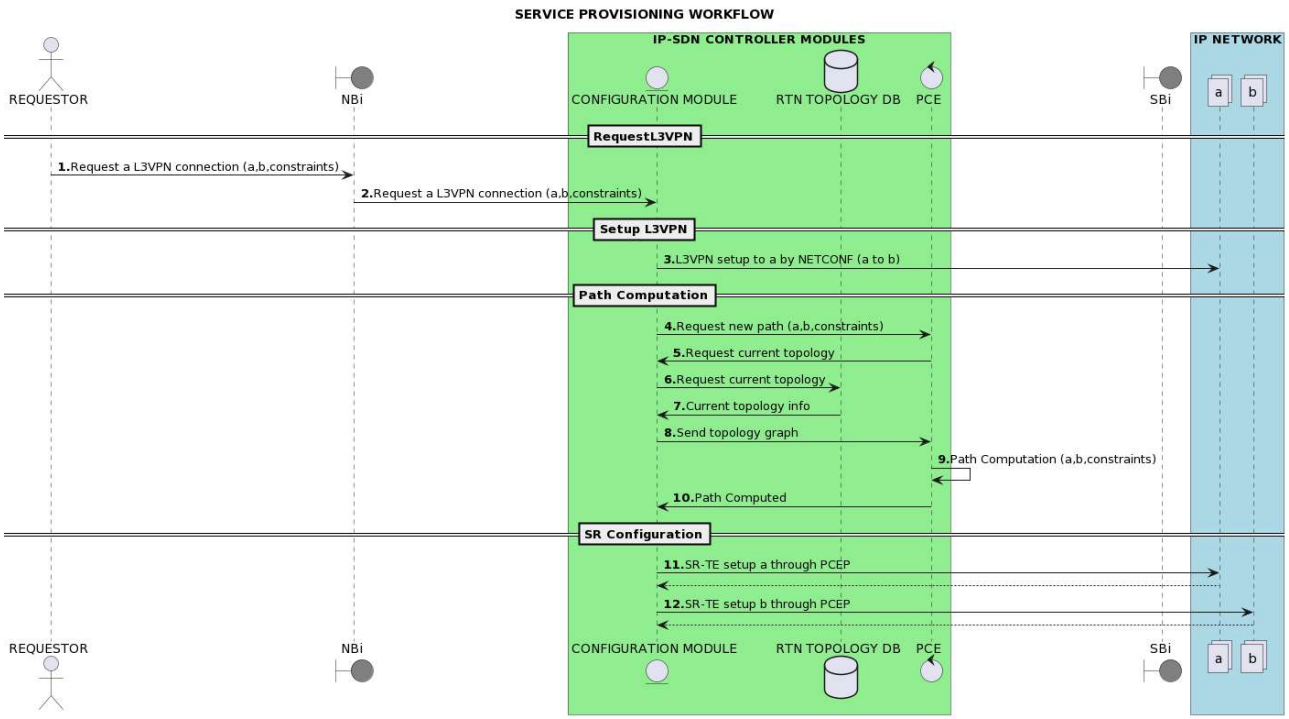

Figure 10-3: LVPN3 Workflow.

Steps 1-2: A requestor (User, 3rd party application, SDN controller) requests towards the IP-SDN Controller NBI the establishment of a new path between two IP nodes in the network, a and b, specifying also path characteristic (QoS specifications), used as constraints for the calculation of the path. The Configuration Module will process this request.

Step 3: the Configuration Module of the IP-SDN Controller configures the L3VPN using the protocol NETCONF (from IP-SDN CONTROLLER to the network elements) to the appropriate endpoints of the

network. Is necessary to configure the VRF (Virtual Routing Forwarding) [116] and RT (Routing Table) values to enable the VPN between two points.

Step 4-10: The Configuration Module of the IP-SDN Controller requests a new path between the two endpoints (a, b) to the PCE module. PCE needs to know the topology of the underlying network to compute the path. Therefore, the PCE requests the current topology, through the Configuration Module, the current topology of the network, stored in the Real-Time Network Topology Database. With the current topology the PCE computes the path between the two endpoints and sends the information of the path to the Configuration Module.

Steps 11-12: After the configuration of the L3VPN, the next step is configuring the SR (Segment Route) path that will be for the connection used for the VPN. Then, the Configuration Module with the path settings computed, setup the SR-TE between both elements via PCEP [112], firstly establishing the SR-TE with "a" and after that with "b", establishing by this way the L3VPN connection between the endpoints.

10.3.3. Inventory Support

In the network is essential to maintain an inventory of all elements (physical and logical elements) and the information of all the Network elements. Figure 10-4 shows an exemplary workflow for inventory support in a SDN network.

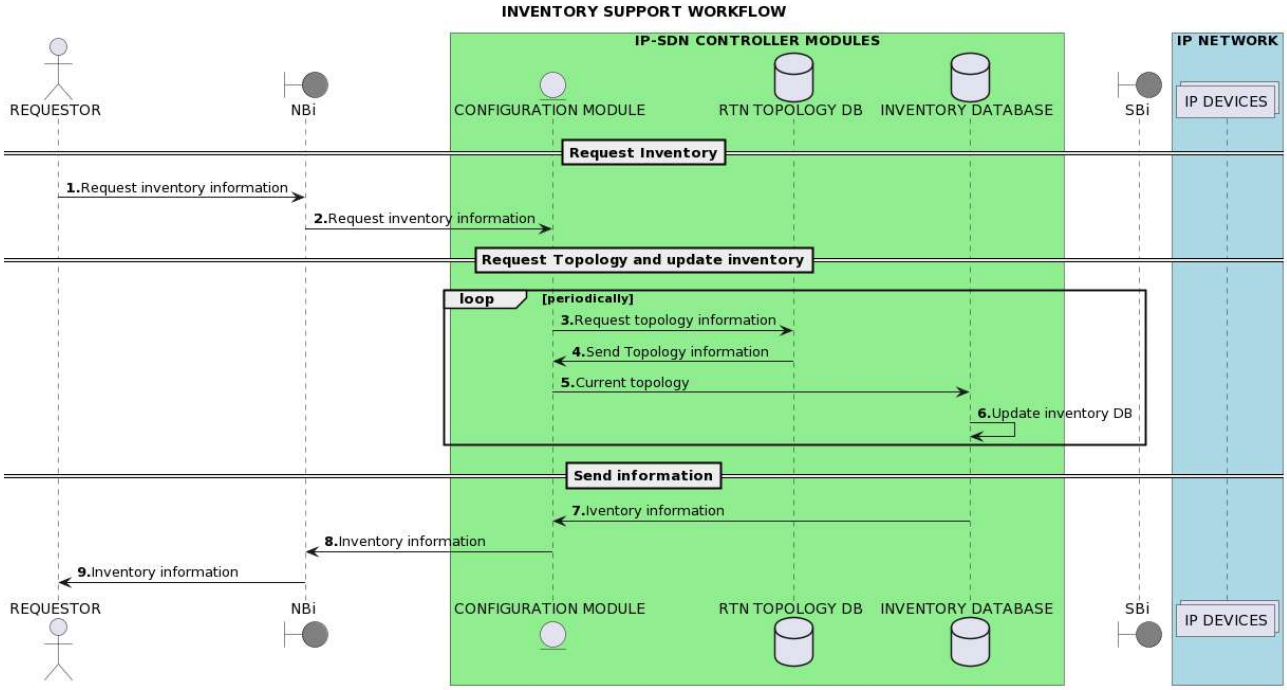

Figure 10-4: Inventory Support Workflow.

Steps 1-2: A requestor (User, 3rd party application, SDN controller) requests towards the IP-SDN Controller NBI, using RESTCONF protocol, the inventory information of the underlying network. In this case the requestor could be for example the Inventory Database located in the application layer.

Steps 3-6: First of all, is necessary to retrieve the topology of the underlying network to verify if there is some logical or physical change. Therefore, the Configuration Module requests the current topology to the Real-Time Network Topology Database, and this module sends the current topology to the Configuration Module. The Configuration Module sends the current topology to the Inventory Database (module of the

IP-SDN Controller) to verify if there are changes. Then, if there is a new network element, logical connection, or something different, the Inventory Module must update it.

Steps 7-9: The Inventory Module must send the inventory information to the Configuration Module and this one reports this information to the Requestor through the NBI Interface of the IP-SDN Controller.

10.3.4. Topology

The controllers of the network must know the topology of the underlying network every time. Thus, is necessary to have a Database which store in real-time the information of the topology of the network. This Database will be in the controllers. Figure 10-5 depicts topology updating workflow:

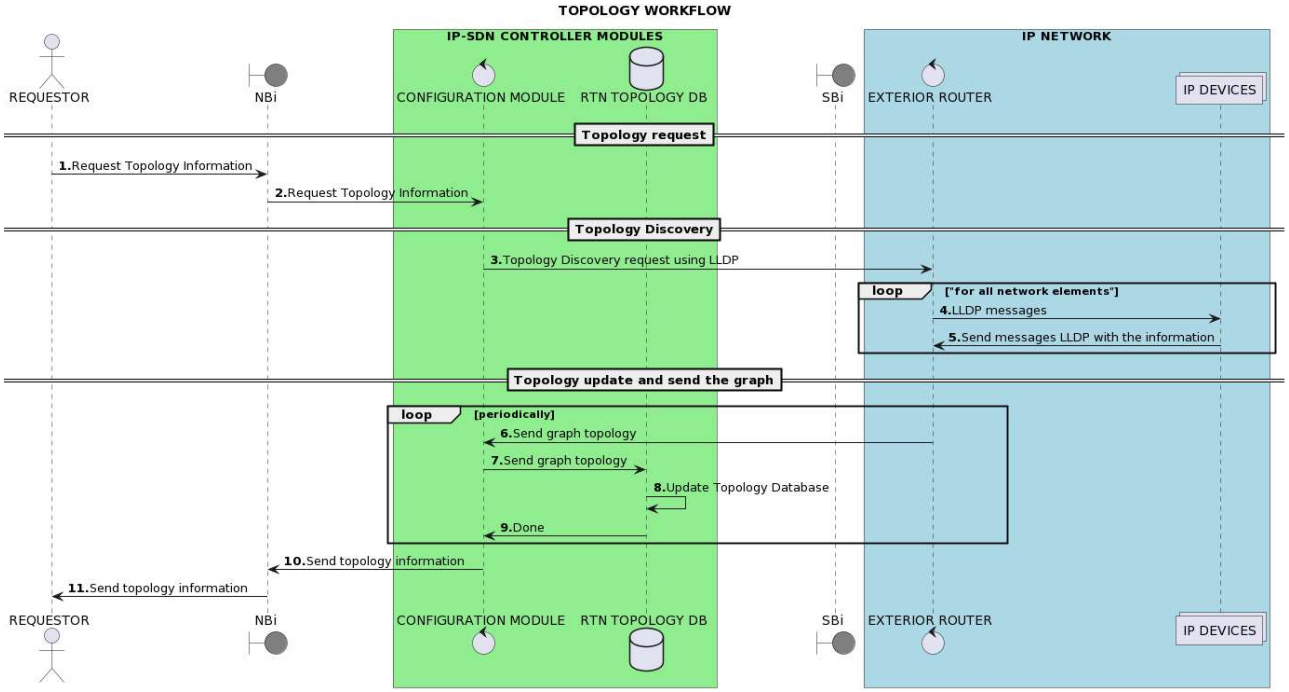

Figure 10-5: Topology Workflow

Steps 1-2: A requestor (User, 3rd party application, SDN controller) requests towards the IP-SDN Controller NBI the topology information of the underlying network. The module responsible for retrieving the topology information is the Configuration Module.

Steps 3-5: Configuration Module of the IP-SDN controller sends a LLDP (Link Layer Discovery Protocol) [117] message to the exterior router of the IP network to discover the underlying network topology. The exterior router will send this LLDP message to all the network elements, and, they must send the information to the exterior router using the same protocol, LLDP.

Steps 6-11: The exterior router sends the graph of the underlying network to the Configuration Module, and this one sends this information to the Real-Time Network Topology Database to update the database information. Finally, the topology information will be sent to the Requestor through the NBI Interface of the IP-SDN Controller using RESTCONF protocol.

10.3.5. Traffic Engineering and Path Computation

Figure 10-6 shows the workflow for traffic engineering using Path Computation Element:

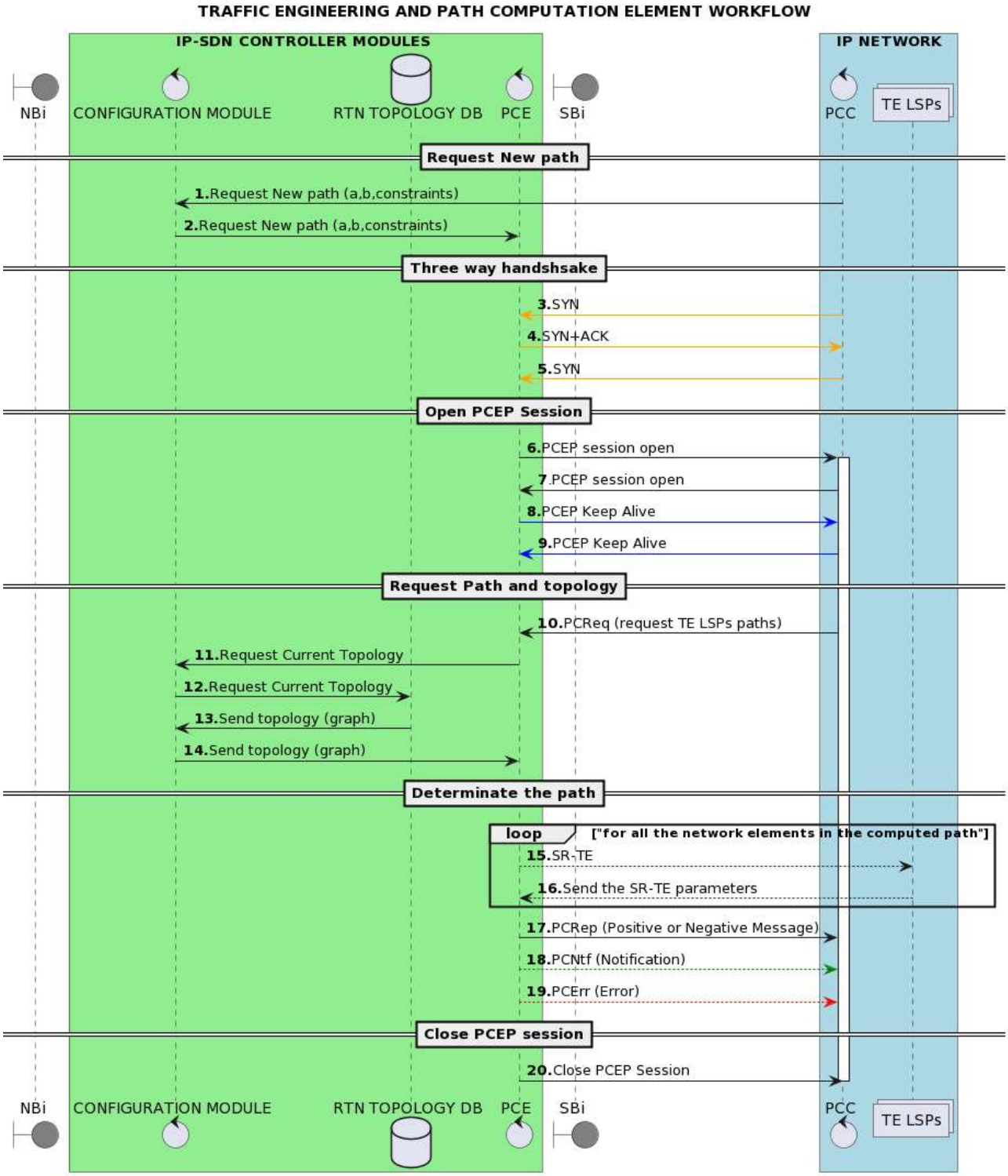

Figure 10-6: Traffic Engineering workflow using PCE

Steps 1-2: PCC requests a new path between two endpoints (a,b) to the IP-SDN controller. The module responsible for the provisioning service is the configuration module, therefore, the request should be processed by this one. Therefore, the configuration module requests a new path to the PCE module of the controller.

Steps 3-5: Before path computation, it is necessary to stablish a TCP connection (3-way handshake) between the PCC and the PCE (Path Computation Element)

Steps 6-9: After the establishment of the TCP connection, it is necessary to establish a PCEP session between both. Therefore, PCE send a message to PCC to open the PCEP session, and PCC to PCE too. After that, both must send the message "*PCEP keep alive*" to notify that the session is ready.

Step 10-14: The PCC knows that the session is available after the previous message. Thus, if the PCC needs to compute the path to follow, it will send the Path computation request to the selected PCE. This message contains a variety of objects that specify the attributes for the path, like for example: IP address in the format a.b.c.d, destination IP address in the same format. PCE cannot determinate the path without knowing the current topology. Therefore, PCE request to the Configuration Module the topology of the network. Then, this module sends a request to the Real Time Network Topology Database, and by this way PCE will know the graph of the underlying network.

Steps 15-16: Once PCE knows the topology is time to determinate the path. Thus, PCE is going to determinate the paths using SR-TE (Segment Routing-Traffic Engineering) [118], for LSP signalling, and using Path Computation for determinate appropriate path.

Steps 17-18-19: once the PCE received the request Path Computation, having in count the possible paths, there are two possible messages to answer to the PCC:

a. **Positive:** In this case the PCE manages to compute a path that satisfies the attributes and required constraints from the PCC. The PCE returns the set of computed paths that the PCC requested.
b. **Negative:** In this case there is not a possible path having count the request of the PCC. Thus, the PCE send a negative message to the PCC saying that is not possible the request. After that, the PCC must send another different request with other appropriate attributes

There are different circumstances in which is necessary that the PCE must notify to the PCC. For example, if the network suddenly if overload, the PCE must notify to the PCC that the request has been cancelled. In addition, PCEP Error message is sent in several situations: when a protocol error condition is met or when the request is not compliant with the PCEP specification (e.g., capability not supported).

Step 20: When one of the PCEP peers decides to finish the session, firstly sends a PCEP Close message to the PCC and after that closes the TCP connection.

10.3.6. Fault Management

In order to meet performance requirements, network management must be efficient and quickly respond to network updates and requests. Therefore, is necessary to implement an efficient Fault Management System, as you can see in the workflow of the Figure 10-7

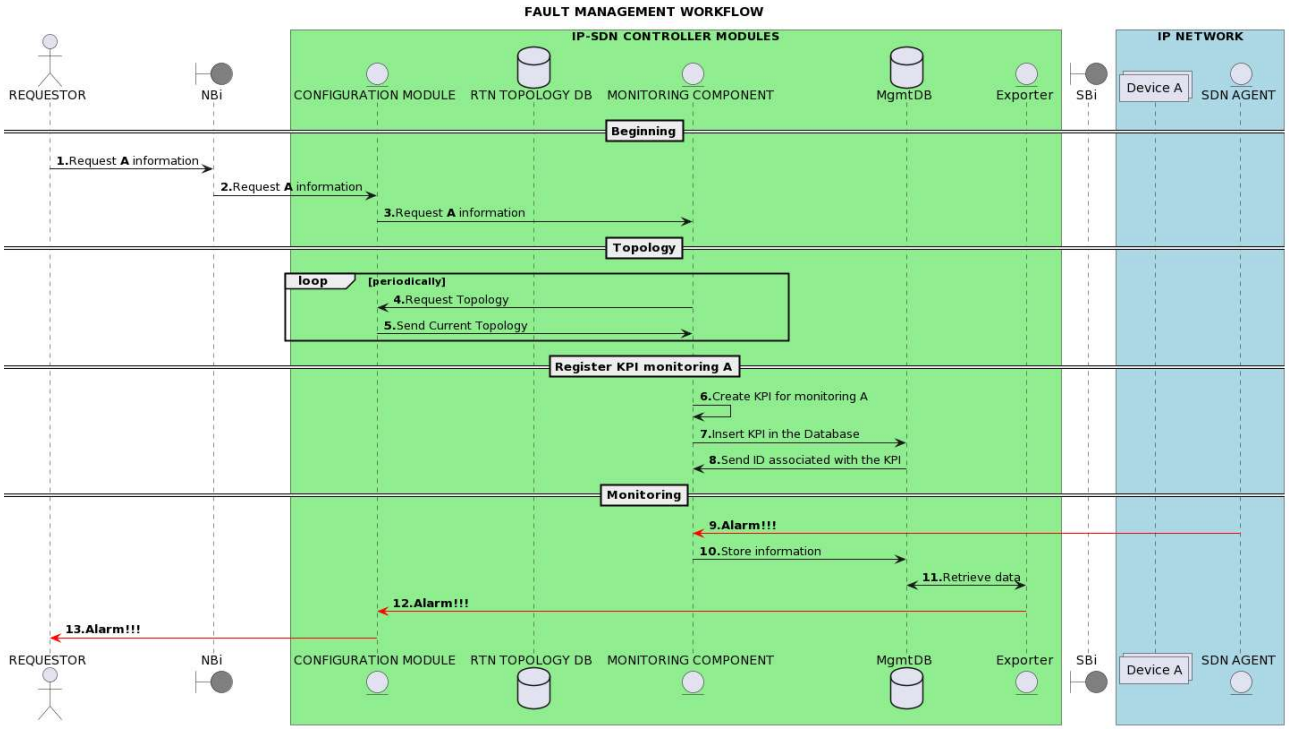

Figure 10-7: Fault Management Workflow.

Steps 1-3: A requestor (User, 3rd party application, SDN controller) requests towards the IP-SDN Controller NBI, using RESTCONF protocol, information of the Device "A" (i.e., state of the network device, metrics, etc.). This request will be processed by the Configuration Module to provide a connection with the Monitoring Component (a block of the Management Module).

Steps 4-5: Monitoring Component requests the Current Topology of the underlying network to obtain the information of the device.

Steps 6-8: Monitoring Component registers the KPI (metrics to obtain from the Device A) in the MgmtDB (Management Database, a block of the Management Module) and this one sends the ID associated with this KPI to difference from the rest. After that, Monitoring Component sends a request to the SDN agent to obtain the metrics.

Steps 9-13: Suddenly, the SDN agent detects that there is a problem with the monitorization and sends the message to the Monitoring Component. Then, this information is stored on the MgmtDB and the Exporter (module of Management Module) retrieves the data from the database to exports the alarms to the requestor.

10.3.7. Performance Monitoring

The monitoring module is responsible for collecting metrics from the underlying network producing valuable data than can be used to improve service performance.

Figure 10-8 depicts an exemplary workflow for monitoring a network element of the underlying IP network:

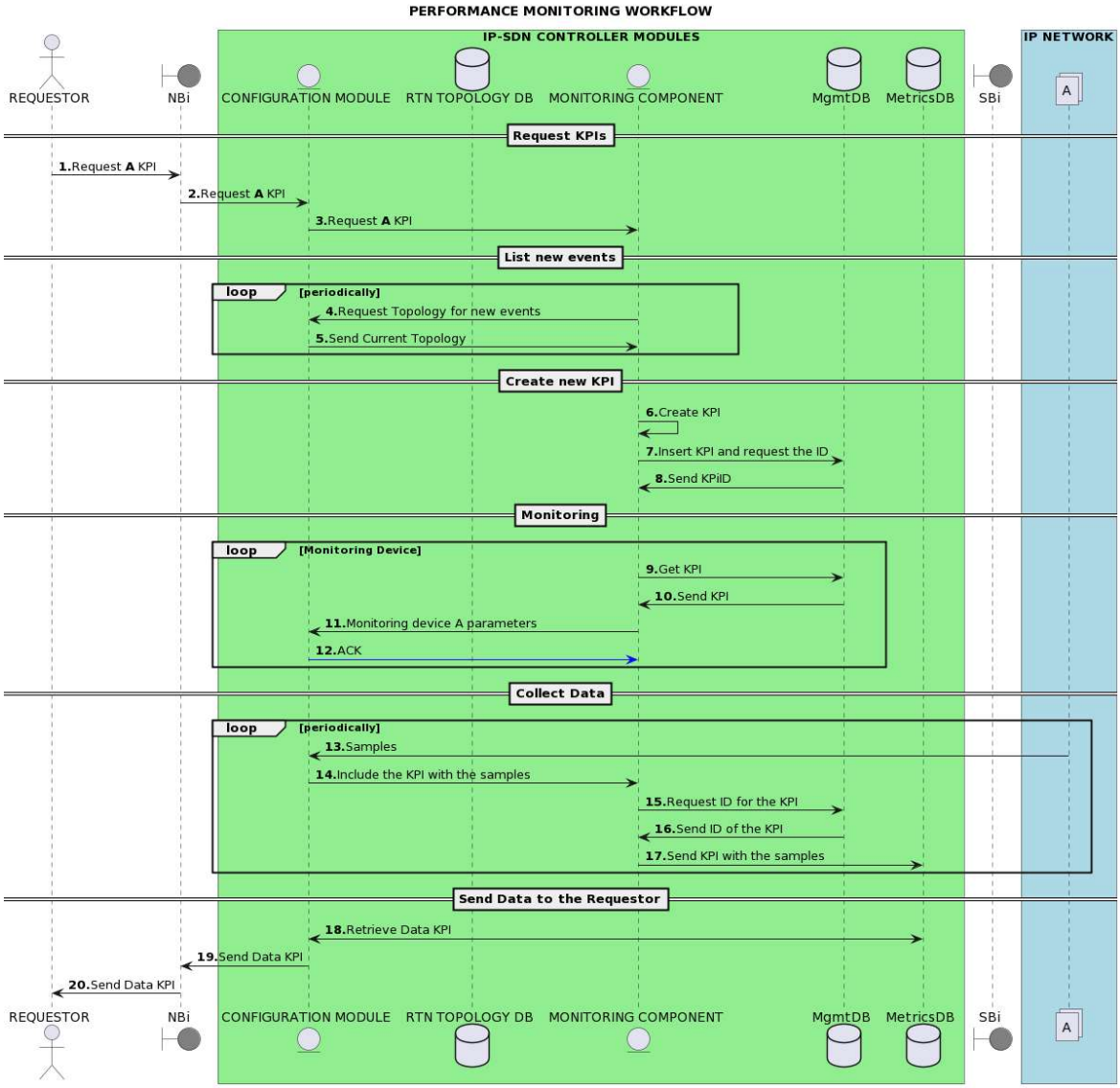

Figure 10-8: Performance Monitoring Workflow.

Steps 1-3: A requestor (User, 3rd party application, SDN controller) requests towards the IP-SDN Controller NBI, using RESTCONF protocol, a KPI of the Network Element (A). Then the Configuration Module sends the request to the Monitoring Component (responsible for establishing the monitorization parameters of the "A" device).

Steps 4-5: Monitoring Component requests Topology of the underlying network for list new events of the devices. With the graph topology Monitoring Component can know if there are new devices or something different in a path/device. Configuration Module requests the current topology to the RTN Topology DB and then sends the information to the Monitoring Component.

Steps 6-8: The Monitoring Component processes and registers the KPI in the Management Database (MgmtDB) and this database assigns an ID to this KPI to differencing from the rest.

Steps 9-12: After that, Monitoring Component requests the KPI to the MgmtDB and sends the monitoring parameters of the device A to the Configuration Module. This one response with an ACK message to the Monitoring Component indicating that all the parameters have been received correctly, and then starts the monitoring process.

Steps 13-17: The SDN agent of the IP network sends the samples to the Configuration Module through the SBI using NETCONF for including the KPI with the samples in the MetricsDB (the Telemetry Database of the Management Module), retrieving the ID associated to this KPI.

Steps 18-20: Finally, the Configuration Module retrieves the data from the MetricsDB to sends it to the Requestor. In this case is necessary to use a Data Viewer to plot the data for the requestor.

10.3.8. Telemetry

Figure 10-9 shows the telemetry workflow of Discretion to retrieve telemetry information of a device (b in this case) of the underlying network.

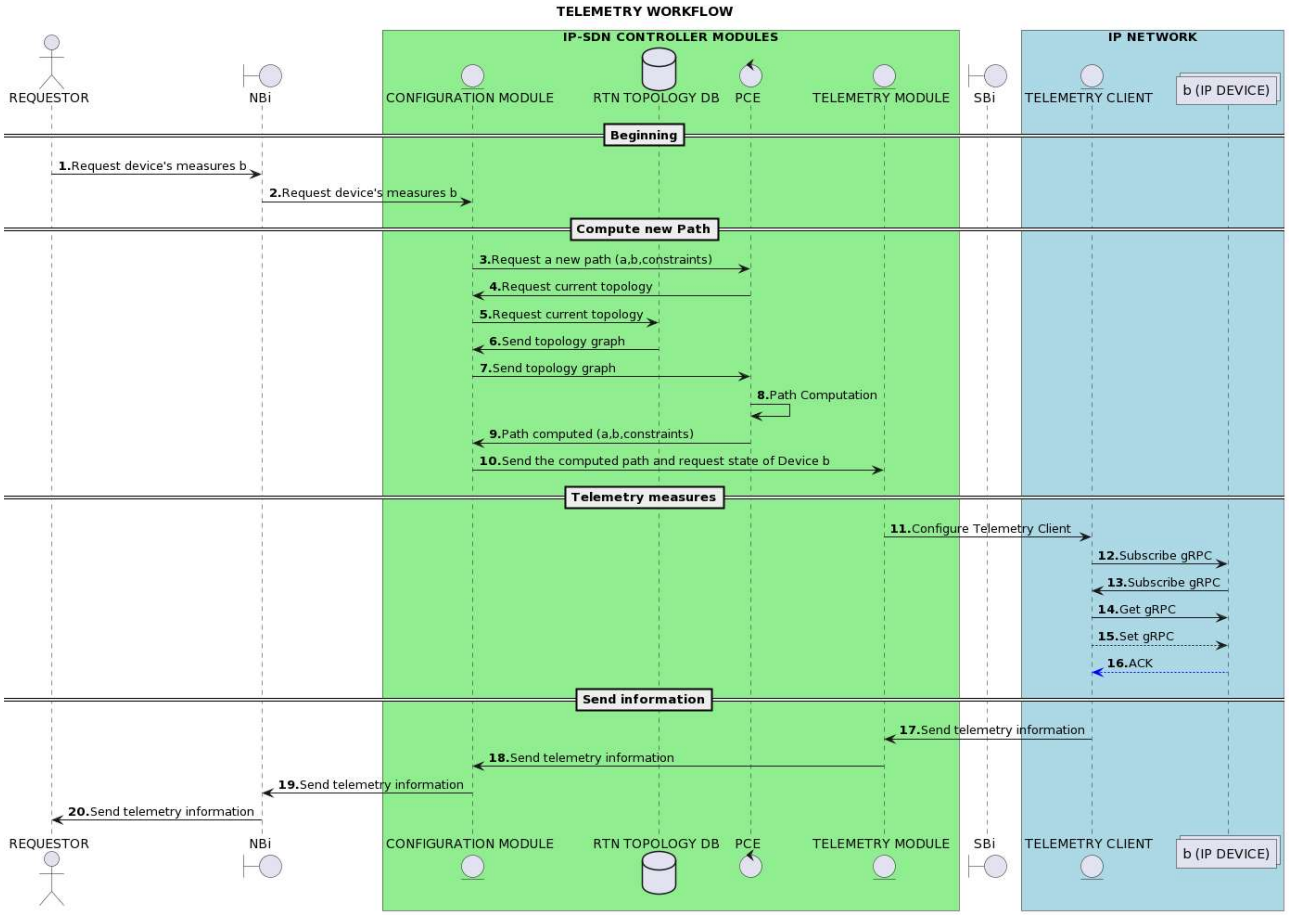

Figure 10-9: Telemetry Workflow.

Steps 1-2: A requestor (User, 3rd party application, SDN controller) requests towards the IP-SDN Controller NBI, using RESTCONF protocol, telemetry information (in this case the state of the network element) of a device located in IP network.

Steps 3-10: First of all, is necessary to determinate the best way to achieve the destination (b) from the source (a). Therefore, Configuration Module requests a new path to the PCE. PCE needs the current topology to compute the optimal path. Hence, PCE requests to the Configuration Module the current topology, available in the Real Time Network Topology Database. After that, Configuration Module sends the topology graph to the PCE to compute the path. PCE, using the current topology, computes the optimal path and

sends the information of the path to the Configuration Module. Then, Configuration Module sends the path information and requests state of Device (b) to the Telemetry Module of the IP-SDN Controller.

Step 11: Telemetry Module (block of the Management Module) configures the Telemetry Client, a network management application located in the IP network responsible for querying/modifying data on the target (device b in this case) or acting as a collector for telemetry data, through IP-SDN Controller SBI to realizes the telemetry measures. The protocol used for collecting telemetry information is gRPC [57]. Subscribe message is a bidirectional streaming RPC that allows Telemetry Client and target (in this case b) to send independent sequences of telemetry messages. Telemetry Client requests to the device the current state and the client collect the information. Telemetry Client can send to the device the message Get that allows to update the state of the target b. This is a message that may or may not occur. If there is an update of the state of the device, the device can send to the telemetry client a confirmation message (ACK) indicating that update has been done correctly.

Steps 17-20: Finally, Telemetry Client sends the telemetry information to the Telemetry Module and this one to the requestor.

11. KEY MANAGEMENT SYSTEM

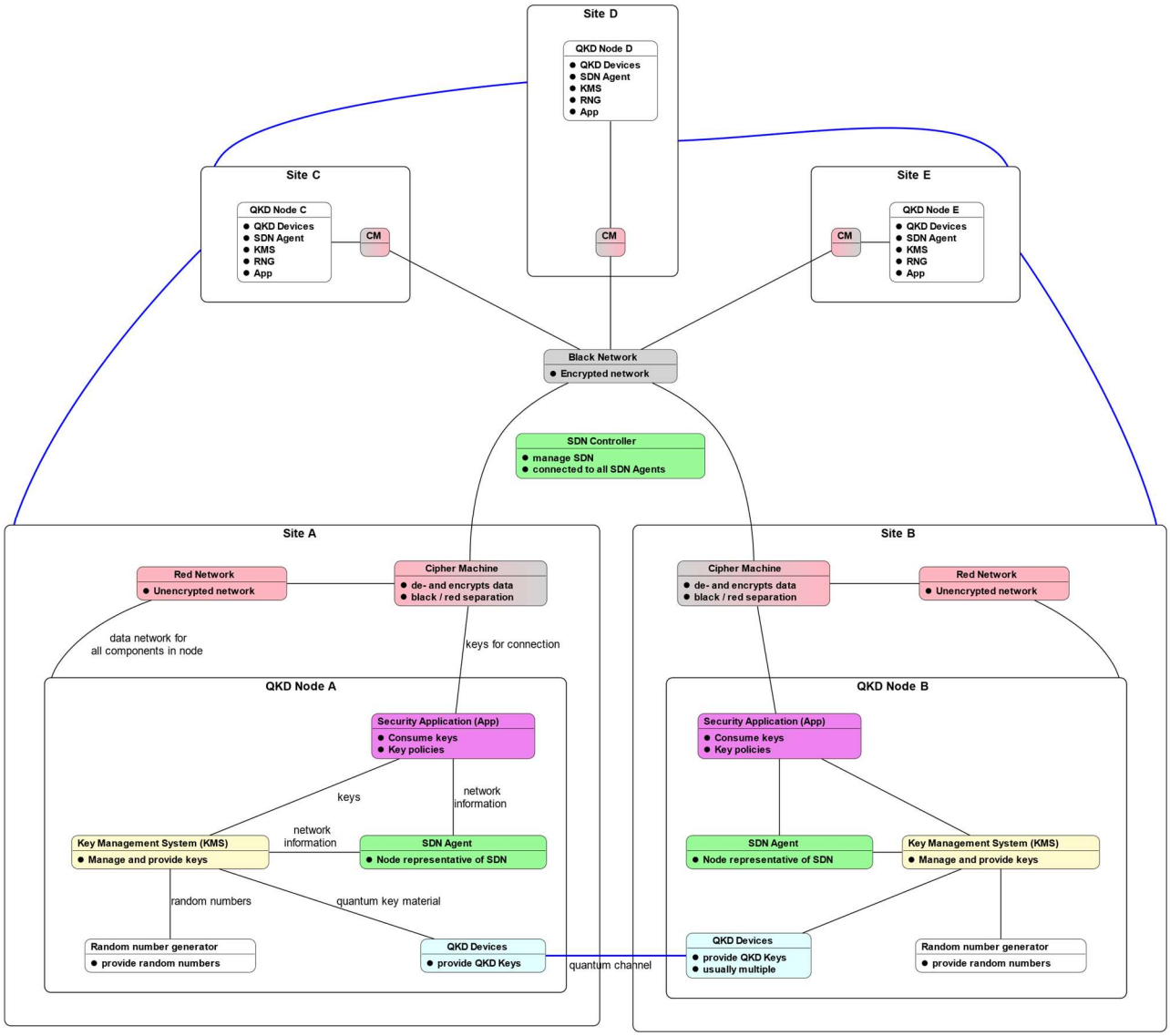

Figure 11-1: Discretion system overview. Blue bold lines are quantum connections. Green SDN Controller is connected to all other SDN Agents on the respective node, connecting lines are not shown for a clearer representation. Sites A and B are shown with all components, Sites C, D and E are simplified.

The Key Management System (KMS) is part of the Software Defined Quantum Key Distribution (SD-QKD) Node. Within DISCRETION it is located in the red network meaning it only has access to that network. The main task of the KMS in the system is to manage the QKD originating key material and provide keys to the application. Another important task of the KMS is to ensure that the same key is available at any two nodes in the network.

11.1. Features

The main tasks of the KMS in a Quantum Key Distribution Network (QKDN) are the following:

- Receive the symmetric key material generated by the QKD modules.
- Store key material, metadata and other keys such as pre-shared keys (PSK).

- Manage the stored key material, which mainly involves monitoring and executing the key material lifecycle.
- Derive keys from the QKD key material.
- Provide keys to the application.
- Forward key material through the QKDN to the remote peer KMS.
- Interact with the SDN to receive information about the network and provide statistics.
- Synchronize with peer KMSs for protocol and event executions.
- Authenticate peer KMSs, applications and other directly connected components.
- Bootstrap the initial KMS communication.
- Consider the border node application.
- Post Quantum Cryptography (PQC) enhanced security.
- General quality of life features.

Due to the prototypical implementation of the KMS it may not fully support all features.

11.1.1. Key material retrieval

The KMS is connected to one or more QKD modules, that provide symmetric key material with the QKD module on the other node connected through the quantum channel. In principle there are two main ways to design this interface, push-based in which the QKD module sends key material as soon as it is generated or pull based where the QKD module holds the generated key material until it is requested by the KMS. Both modes can follow the ETSI QKD GS 004 standard [58].

In broad strokes, the main protocol is outlined in Figure 11-2 and follows an open (message 1-6), read (message 7-10), close (message 12-15) principle. One deviation of that figure from the standard is the naming, since the standard describes the interface between an application and a KMS, however in DISCRETION and in general the standard, due to its flexible nature, is also applicable for the communication between a QKD module and a KMS. The figure at this section replaces the application with the Initiator and the KMS with the Responder. As the names suggest, the Initiator initiates the communication, and the Responder responds to the Initiator.

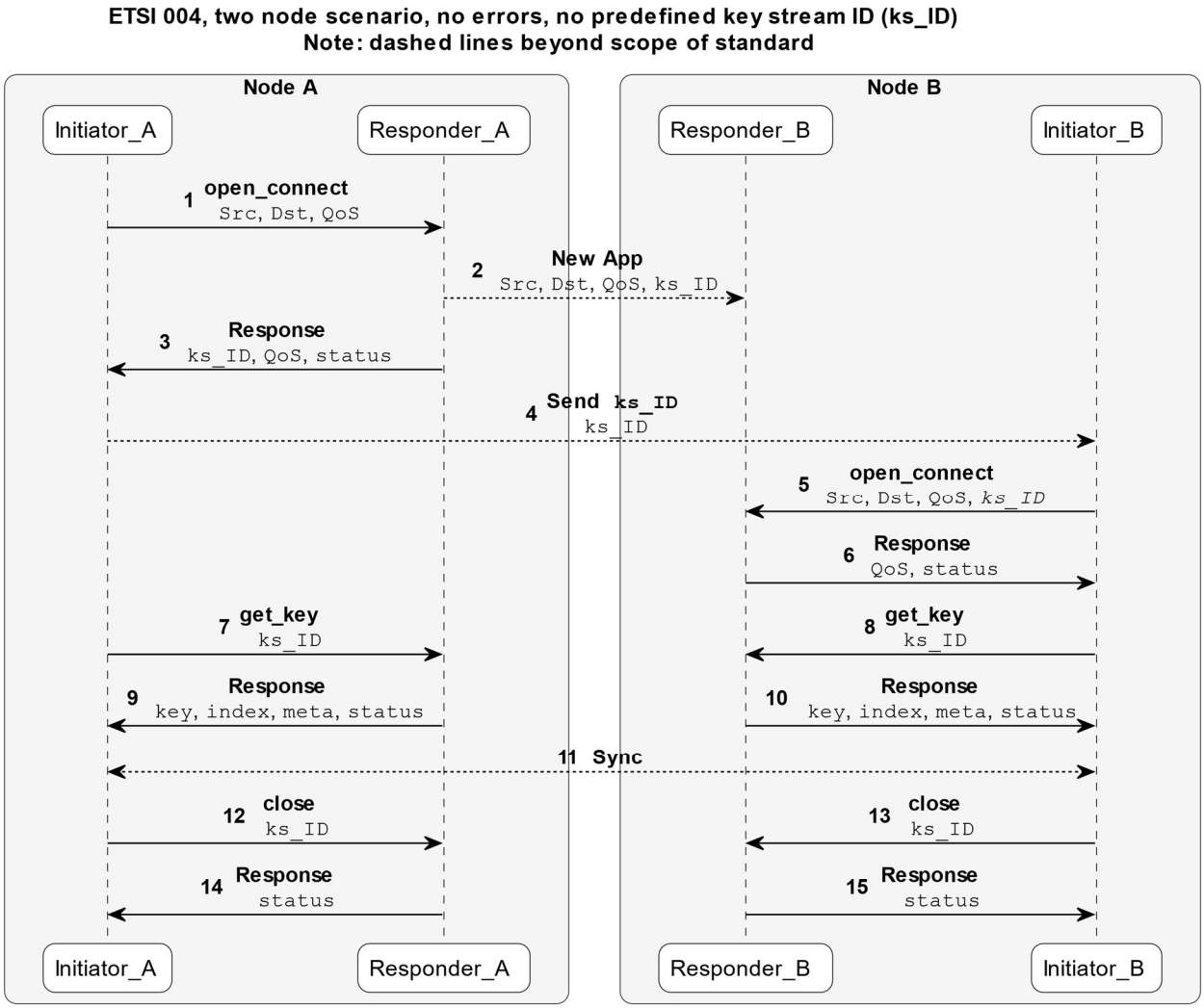

Figure 11-2: Outlined protocol based on ETSI QKD GS 004 standard. Dashed lines (message 2, 4 and 11) are necessary for the correct execution but are beyond the scope of the standard. Figure based on [58]

First the Initiator, for example the KMS, connects to the Responder, for example the QKD module, with the `open_connect` command, to which the Responder responds to. There are different nuances regarding this interaction but after the interaction a key stream ID is negotiated, and each key material transmitted subsequently during this connection is corresponding to that ID. Also, in that phase the length of the key material is negotiated as well as several other Quality of Service (QoS) parameters such as the rate and jitter of the key material provision.

Thereafter the keys are transferred between the communication parties. At this stage there is a slight differentiation in the protocol execution between the pull and push mode. The available commands are `get_key` and the corresponding response. In pull mode, the request (messages 7&8) and response (message 9&10) are executed in a loop, each request has one response and nothing happens until the next request is sent by the Initiator. In push mode, one `get_key` request starts the push delivery of the key material, so whenever new data is available the Responder sends the `get_key` response to the Initiator.

The connection can and should be closed according to implementation needs. For that both peer Initiators send a close command to their connected Responder. Thereafter the key stream ID is invalid resulting for example in any subsequent `get_key` commands specifying that key stream ID to be invalid.

As stated in the Figure 11-2 description, additional messages are necessary for a correct execution, but the standard specifically states, that their characterization is beyond the scope of the standard, they are depicted with dashed lines.

After the first `open_connect` command the key stream ID is set for the Responder. This information also has to be present at the peer Responder, so that they operate on the same key stream ID. This is depicted by message 2 and it is up to the implementation of the Responder on how to transfer this information.

The same is true for the Initiator and its peer, both must have the same key stream ID to work with, which is depicted by message 4.

Message 11 bundles many different synchronization messages, which must be exchanged between the peer Initiators. For example, after key material has been received, they have to make sure it's the same value and give this key material unique identifiers to store them in their respective databases. Another need for synchronization is to initiate the close sequence with the respectively connected Responder instance. The exact details of this synchronization protocol are up to the implementation and are not covered by the standard.

11.1.2. Key material storage

The KMS has a database to store different kinds of keys or key material. The database is designed to be persistent so that even after a power outage or similar faults the operation can continue as normal. Each key or key material has several metadata such as UUID, state, creation time, identification or address of the peers and other metadata. The database stores key material received by the QKD module as well as pre-shared keys (PSK). These are used especially during bootstrap to secure the initial inter KMS communication when no QKD key material is available. This bootstrap is discussed in a later section.

11.1.3. Key material management

Considering the NIST SP 800-57 [119] standard, the KMS implements a key lifecycle management. The main reasoning is that if an unnoticed attacker obtains key material once, the data automatically becomes invalid after a time thus reducing the security impact of the data breach.

Another task of key material management is peer synchronization. It must be assured, that the same key material with the same key material identifier (key ID) is present and in the same state at both peers. This includes making sure the received key material, but also the key provision to the application is synchronized.

11.1.4. Derive keys from key material

The KMS generates keys to provide to the applications from the key material obtained from the QKD devices and other sources such as the Post Quantum Cryptography (PQC) Key Encapsulation Mechanism (KEM) for example. The derivation process is done to enhance security by combining different key material origins, but also to adjust the size of the requested key to the available key material for example concatenate multiple key materials or split them into smaller sizes.

11.1.5. Provide keys to application

The KMS implements an ETSI QKD GS 004 [58] compliant interface where the application requests keys on demand, thus corresponding to the pull mode. An additional ETSI QKD GS 014 [59] interface might be built on top for combability. Apart from the keys, additional data may be transferred, such as metadata and key statistics.

11.1.6. QKDN key forwarding / relay

Since the QKD devices only generate symmetric key material between two directly connected devices, but symmetric keys are required to be present at two arbitrary nodes in the QKDN, the KMS must implement a key forwarding or key relay mechanism. This only affects the QKD generated key material. The KMS forwards key material through multiple KMS instances on the intermediate nodes until the data reaches the destination node.

In more detail, the KMS obtains key material from the random number generator which is then forwarded to the next directly connected node secured with QKD generated key material using Information-Theoretic Security (ITS) algorithms. This forwarding process is executed until the destination is reached.

A simplified forwarding scenario is outlined in Figure 11-3. Application A on Node A requests a key from its KMS A to communicate with Application B on Node B. The KMS communicates with the node local SDN agent A to receive the network information regarding the destination. If KMS B is not the directly connected KMS, it receives a path, in this figure the path goes through the intermediate trusted Node C. The generated key material to be used is forwarded through Node C using ITS algorithms and QKD key material. If the KMS B is the directly connected peer KMS, forwarding through a trusted node is not necessary. As soon as the same key with the corresponding key ID is present at both endpoints, KMS A provides the key ID and Key to Application A. It's the responsibility of Application A to forward the key ID to Application B, which requests the key from its KMS B by providing this key ID. Thus, Application B receives the same key as its communication partner.

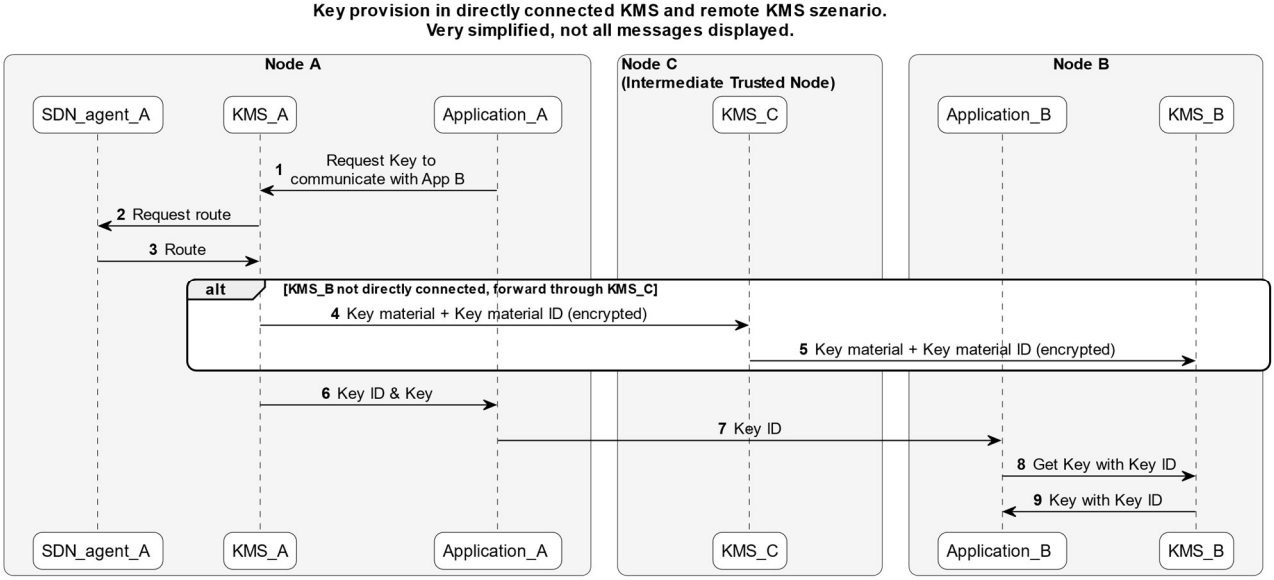

Figure 11-3: Simplified protocol of key provision to an application pair, located on two different nodes. Key forwarding (displayed in the "alt" box) is only necessary if Node A and B are not directly connected, but rely on the trusted intermediate Node C.

11.1.7. SDN Agent interaction

The KMS interacts with the node local representative of the SDN, called SDN Agent. Especially for the afore mentioned key forwarding mechanism the KMS needs the information on how to reach the destination node through directly connected intermediate nodes. This path information alongside other information is requested from the KMS. The SDN requests information regarding key statistics, such as amount of key reserve and other statistical data. This data is required by the SDN for example to compute the optimal paths for communication.

11.1.8. Inter KMS communication

All KMS instances deployed in the QKDN communicate with at least one other KMS in order to fulfil the majority of their tasks. Basically, every KMS operation in the network is a process executed in at least two nodes. For example, receiving key material, providing keys or update the database must be done at both peer KMS on two different nodes. Therefore, a flexible and secure inter KMS protocol is implemented to synchronize the different events and processes.

11.1.9. Authentication

The KMS must authenticate its communication parties. It therefore uses authentication algorithms to verify for example that the QKD devices are legitimate key material sources, peer KMS and SDN are legitimate communication partners, and that the applications are legitimate key consumer.

11.1.10. Secure bootstrap

During normal operation existing QKD generated key material is used for the key material forwarding and authentication processes. During the initial communication no QKD generated key material is available. Pre-Shared Keys (PSK) are used for that. Using PQC KEMs for that purpose is further investigated, though the prototypical implementation of the KMS may not fully support this feature.

11.1.11. Border Node Use-Case

Independent QKD networks might be supposed to be connected with each other. This use-case must be considered. Usually, dedicated border nodes are deployed that connect different QKDNs. The KMS has to consider this operation and bases the border node forwarding mechanism on the existing peer KMS forwarding mechanism. The prototypical implementation of the KMS may not fully support this feature.

11.1.12. Post Quantum Cryptography

Following recommendations of the BSI [120] and other national security agencies the QKD technology alongside ITS algorithms should be used in conjunction with other technologies. PQC provides technology independent means and post quantum secure algorithms to establish symmetric keys and therefore are used to derive keys from QKD and PQC key material. It also has the potential for true end-to-end security without relying on intermediate nodes therefore reducing the trust assumptions on the forwarding nodes in the QKDN.

11.1.13. Other Features

The KMS needs several quality-of-life features such as logging, configuration, and interfaces to be implemented.

11.2. Functional modules

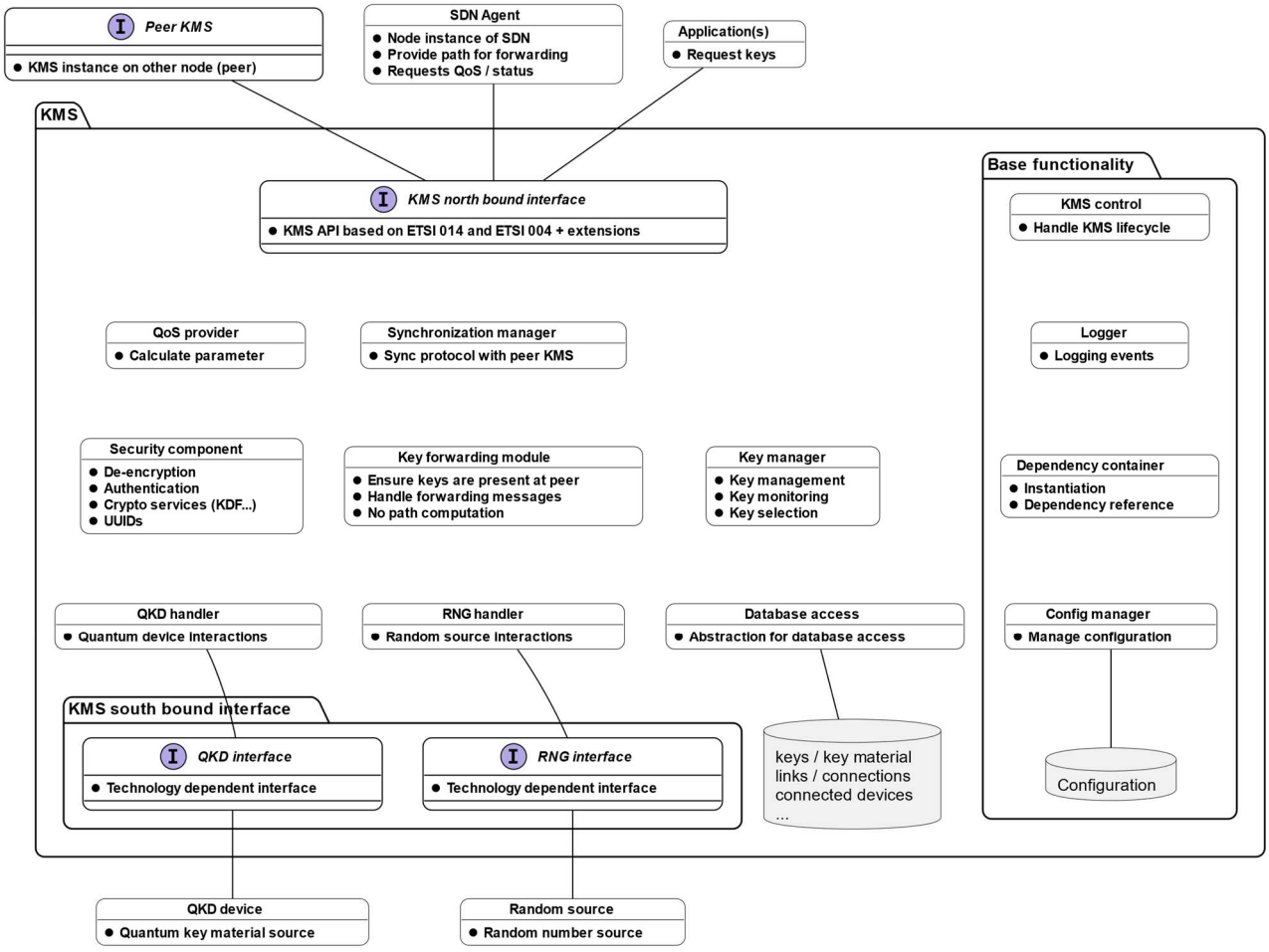

Figure 11-4: High Level modular KMS Software Design. Presented are software defined modules that are dedicated to one specific task or group of tasks. The tasks stated in the modules are only the main task(s) to give a first idea of the module's tasks. Modules in cylindric shape are associated with database technology, modules with an I symbol with interface technology.

The KMS is designed in a modular fashion, meaning each module acts as a service with a defined API. The Figure 11-4 shows this concept in an UML style representation. The biggest package represents all modules from which the KMS is constructed. Two sub-packages are outlined in the form of the Base Functionality package, grouping all basic functionality of a software and the KMS south bound interface package, which bundles the device specific interface technologies. Outside the KMS package, one can identify the external components interacting with one KMS instance. Also displayed is the interface of a peer KMS, which is another instance of the KMS package at another location.

This modular approach allows a certain degree of freedom and adaptability to project changes and a compatibility with agile workflows. Each module will implement certain types of features, for example all cryptographic algorithms required during operation are implemented in the security component. If say a new algorithm needs to be deployed, this module can be changed and ideally affect no other or only a few other components. One module may consist of several sub modules not shown in the overview diagram of Figure 11-4.

In the following sections each module is shortly presented, and the features outlined in the previous section are mapped to the individual modules implementing them.

11.2.1. KMS North Bound Interface

The NBI at the current stage of design consists of an ETSI 004 based interface to the applications. This will use extensions to provide additional features associated with key management. Another interface is the maintenance interface sub-module allowing authorized users to interact with the KMS. The communication interface with the peer KMS instance is also associated with the NBI.

This module is implementing or contributing to the following features outlined in the previous section:

- Provide keys to application
- SDN Agent interaction
- Inter KMS communication
- QKDN key forwarding / relay

11.2.2. Quality of Service provider

This module gathers and calculates the parameter required for the QoS information provided to the NBI. This includes for example querying the database to calculate the key rate or request the number of keys available for certain connections.

This module is implementing or contributing to the following features outlined in in the previous section:

- Provide keys to application
- **Error! Reference source not found.**

11.2.3. Synchronization manager

This module handles the synchronization protocol with the peer KMS. The exchanged synchronization messages are forwarded within the KMS to the corresponding modules. Several events and data must be synchronized, for example the receival of new key material from the QPS. Since most features must be synchronized between at least two KMS instances, the synchronization manager is involved in almost all Features outlined in the previous section.

11.2.4. Security module

The security module implements all cryptographic algorithms used during operation of the KMS. This includes the ITS algorithms such as Key combination and Wegman Carter authentication as well as Post Quantum Cryptography algorithms.

This module is implementing or contributing to the following features outlined in the previous section:

- Derive keys from key material
- QKDN key forwarding / relay
- Inter KMS communication
- Authentication
- Secure bootstrap
- Border Node Use-Case
- Post Quantum Cryptography

11.2.5. Key Forwarding Module

The forwarding module is mainly used to forward keys and other information between the distributed KMS instances.

This module is implementing or contributing to the following features outlined in in the previous section:

- QKDN key forwarding / relay
- Border Node Use-Case

11.2.6. Key manager

The key manager is the module maintaining and handling the keys. The Key Manager can be seen as one of the central modules in the system. While it is mainly used for the Key material retrieval, Key material

storage and Key material management it is involved in any activity related to keys or key material which is basically any operation of a Key Management System. it is involved in any activity related to keys or key material which is basically any operation of a Key Management System.

11.2.7. Device handler

In Figure 11-4 outlined QKD and RNG handler are handlers that interact through the corresponding interfaces with the connected external modules. They can be customized to the individual modules and their ways of working, such as implemented protocols.

Regarding the feature contribution of those modules, it depends on the connected external modules. The QKD key generating devices are used in the key material retrieval process and the RNG in basically all features involving security algorithms.

11.2.8. Database interface

This module handles all direct interactions with the database. This is done to have an abstraction layer on top of the database implementation specifics.

Regarding the feature contribution of the database interface, it is used whenever keys or key material entries are sored, retrieved, modified or deleted. Also, connection information and information about the connected devices, such as their performance parameter are stored in a database. So, all features involving data of the non-key nature involve this module as well.

11.2.9. Base functionality

This groups all basic functionality modules a software needs, such as logging and configuration handling. It mainly bundles all features listed as "other Features".

11.3. Interfaces

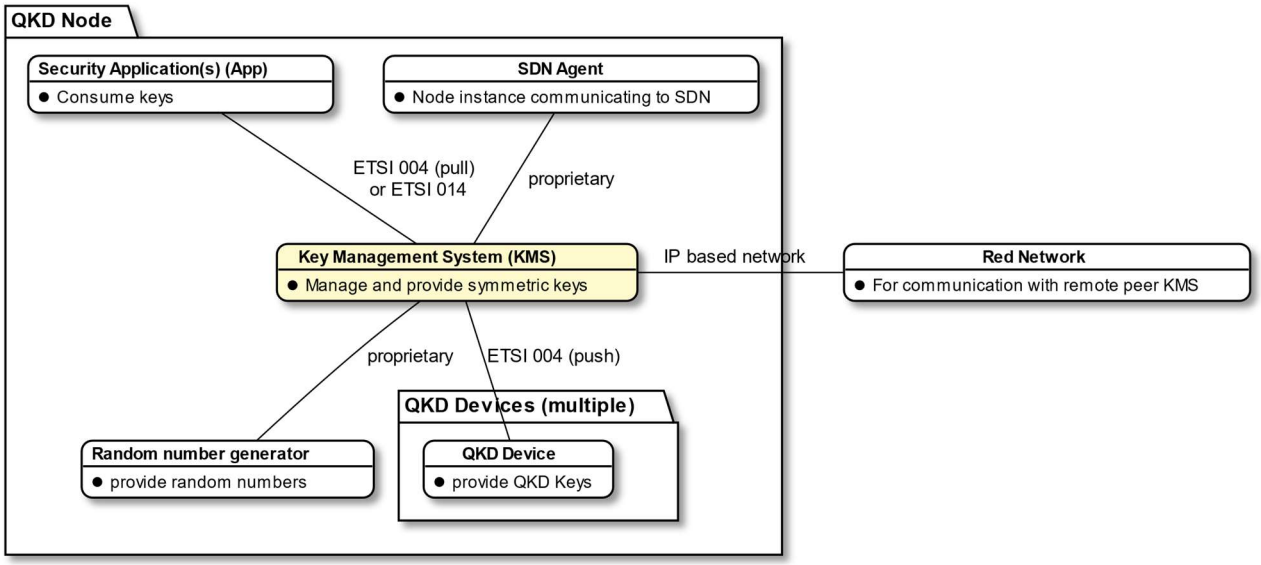

Figure 11-5: A simplified component diagram reduced to all modules directly interfacing with the KMS. The connections state the type and technology of the connection, the connection direction is not shown.

The KMS has several direct connections outlined in Figure 11-5. Since the KMS is a module located in a QKD Node, a more detailed description can be found in the "QKD Nodes Preliminary Design Report" document 4.1. This document gives an overview.

11.3.1. Interface to QKD Module

This interface is used to retrieve key material from the QKD devices.

The KMS in the form of exchangeable QKD handler modules must adapt to the interface provided by the QKD module. As of writing of this document the most feasible protocol for QKD key material transfer is the ETSI 004 standard with the key material push variant.

11.3.2. Interface to Application

This interface is used to deliver keys to applications, but also to provide other information about the KMS and performance parameter.

The KMS provides keys to applications on demand, therefore an ETSI 004 based interface in pull mode is the most feasible option. The interface is likely to be extended. An ETSI 014 interface could be built on top of the ETSI 004 interface depending on the applications needs.

11.3.3. Interface to SDN Controller

This interface is mainly used to provide performance and other data to the SDN controller and to receive path and network information about the QKDN from the SDN Controller.

Due to the lack of applicable standards, the interface will most likely be DISCRETION proprietary.

11.3.4. Interface to other KMS instances

Most tasks require synchronization between at least two KMS instances on different, remote nodes. It is planned to authenticate messages exchanged, to detect illegitimate messages.

Due to the lack of applicable standards, the interface will most likely be DISCRETION proprietary.

11.3.5. Maintenance Interface

For setup and maintenance such as modifying the amount and type of connected devices for example, but also to update pre-shared keys, a maintenance interface is required.

Those are typically tailored to the software and proprietary.

12. SECURITY APPLICATION

12.1. Overview

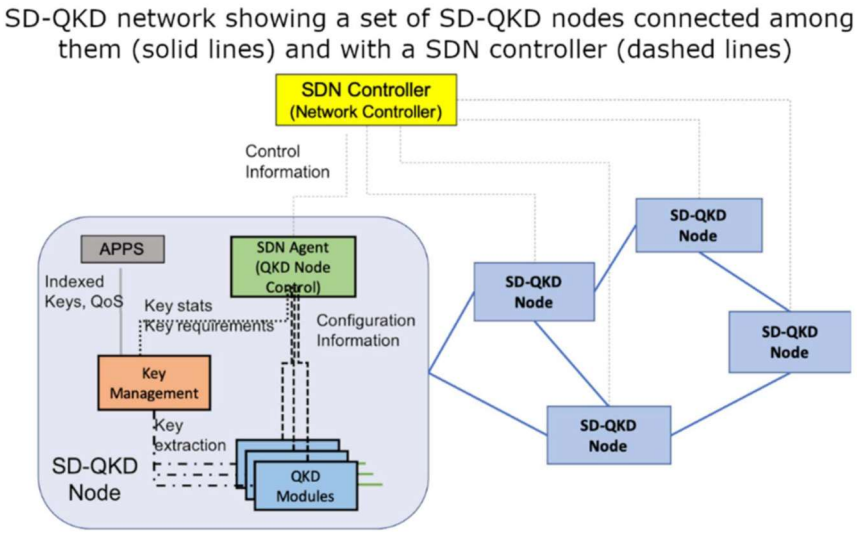

Figure 12-1: SD-QKD network according to ETSI 015. The security applications are depicted in grey as "APPS"

The security application is well defined in the ETSI QKD GS 015: The security application is defined as any entity requesting keys from the key manager within the node. A single instance or process may also require opening different isolated sessions (with a different unique ID) with the SD-QKD node. The SDN controller takes two roles related to application management: the first one is to register any incoming application and, more specifically, their QoS requirements to better optimize the usage of the QKD network based on the real demands; the second is to handle the complexity of finding the remote SD-QKD peer node for QKD applications. The SDN controller acts as a central repository where new applications are registered and where SD-QKD nodes can access to find the peer SD-QKD node for a given application. Any application just knows where its node's key manager is located (IP/port), while the control plane will take care of registering and coordinating new applications [108] p13. The ETSI QKD GS 014 similarly defines it as the Secure Application Entity (SAE) [59].

12.2. Application within DISCRETION

Within DISCRETION the Security Application (SA) can be applied to two scenarios:

1. As an application for the management of the Cipher Machines and used keys.
2. As an application for the SDR Configuration.

Figure 12-2 presents and overview of the interactions between KMS, SA, CM, and Tactical Radio Setup Point (TRSP). It is possible to observe that the interactions between KMS and SA will be based on ETSI GS QKD 004 standard. On the other hand, custom interfaces will be implemented to allow the connections KMS-CM and KMS-TRSP. The ETSI GS QKD 004 standard specifies a high-level API between both QKD Key Managers (KMS) and applications (SA) and QKD modules and KMS. For the former, it proposes a client/server scheme in which the KMS is the server, and the SAs are the clients. According to the standard, the server should provide the following API functions: OPEN_CONNECT, CLOSE, and GET_KEY. The former two methods allow to reserve and terminate a key stream, respectively, the latter might be used to fetch the key material according to the key stream parameters defined in OPEN_CONNECT.

As ETSI GS QKD 004 only provides the logical connection between the entities, it is necessary to define the technology to allow their communication. One solution might the Constrained Application Protocol (CoAP) [123], defined in RFC 7252 [124]. It is an application-layer protocol, designed for point-to-point asynchronous communication. CoAP runs over UDP and supports GET, POST, PUT, and DELETE methods, and the data model of the payload might be selected according to the application. Additionally, there exist open-source CoAP libraries, for instance, in C and Python programming languages.

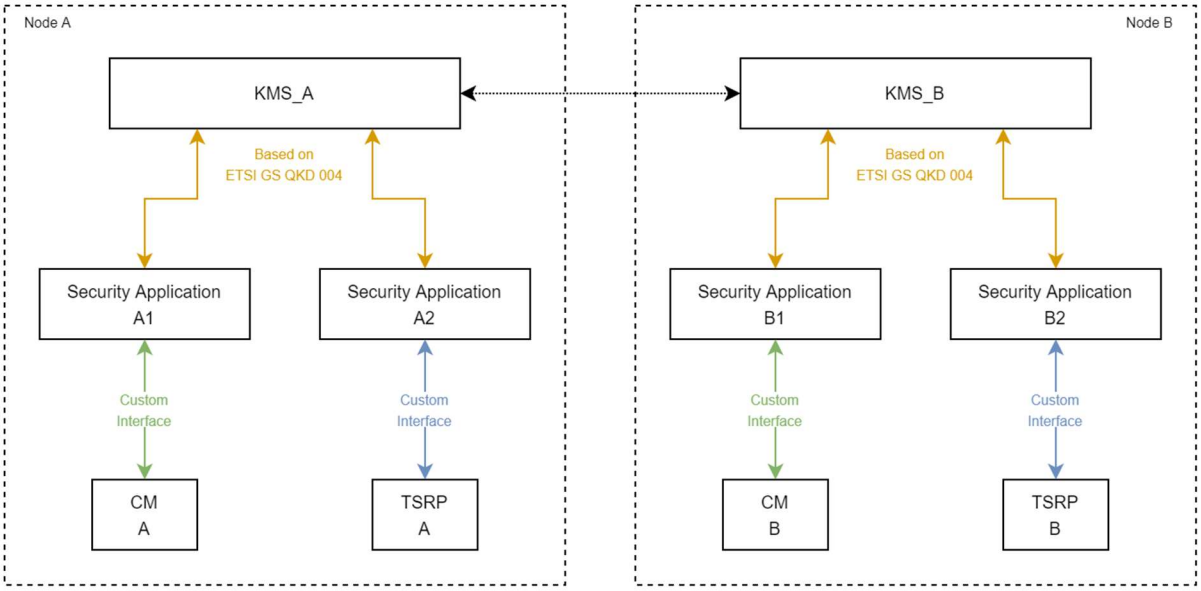

Figure 12-2: Overview of the Security Application Interfaces

12.2.1. Security application for the management of the CMs and used keys

From the definition above a Security Application is defined as the entity that consumes the QKD-derived keys. In the scenario of DISCRETION, this application has a broader role, since it is responsible to manage the loading of both quantum keys and classical keys into the CMs. Due to the design of the CM as a passive component, it cannot actively pull the keys. As such, this application interacts with the KMS at each endpoint of the communication to obtain Keys and with the applications at the other communications endpoints to derive classical keys. The classical keys are derived based on pre-shared key values. Furthermore, advanced programmatic logic is required to select the correct keys in order to consider the mission configuration, security policies and user input, which is beyond the scope of the CM. Therefore, those tasks are handled by the SA for the CM, which retrieves the keys from both the KMS and the derived classical keys to forward them to the CM. The CM is then responsible for merging both classical and quantum-derived keys to protect the exchanged data.

One of the functions of the SA, within the defined security policies, is to manage the CM's internal key exchange and update, such as the periodical change of the session key (obtained from both classical and quantum-derived keys). So, the refresh rate of the quantum-derived keys will be defined at the SA level, this rate will define the frequency at which the SA will fetch keys from the KMS.

It would also be the SA's responsibility to interact with any centralized key management services for the classical keys, if necessary.

The SA might also be required to support the enforcement of any kind of access control and network segregation policies. These could be configured locally or distributed by a central management entity, but details will be implementation specific.

12.2.2. Security Application for the SDR configuration

As outlined in Section 0 during the Setup Phase, the radios are configured with keys according to the mission configuration, security policies and user input.

In this scenario, the Tactical Radio Key Distribution Agent at the TRSP acts as a Security Application, which pulls the required keysets from the KMS to configure the radios.

In this scenario, there exists a Security Manager that has to role to manage all the keys used in tactical environment. It defines security groups and sends them, accordingly, to the TRSPs in each operational site. Each security group contains sets of keysets. The TRSP, in particular the Key Distribution Agent, is in charge of interacting with the KMS either with the local KMS or with a local key generator. The KMS is used when a security group exists in multiple sites, whereas a local key generator is used when a security group only exists in the current site.

Either communicating with the KMS or a local key generator, the workflow will be similar, the Key Distribution Agent will act as a client to pull keys from a server.

13. FINAL REMARKS

This document refines the previous version of the v1.0 edition of the deliverable and covers the preliminary architecture of DISCRETION. Taking as inputs, the requirements and use cases defined in D2.1 it was possible to define the different layers of DISCRETION' SDN.

Several definitions made in this document could already be definitive, but the architecture will only take its final form in Work Package WP5 Design of SDN with QKD. In the following phase of the project, the final design will be presented.

14. REFERENCES

[1] R. e. a. Mahmud, "Software-Defined Multi-domain Tactical Networks: Foundations and Future Directions," in *Mobile Edge Computing*, Springer, 2021, pp. 183-227.

[2] E. Sørensen, *SDN used for policy enforcement in a federated military network,* 2014.

[3] E. Haleplidis et al., "Software-Defined Networking (SDN): Layers and Architecture Terminology. RFC7426," 2015.

[4] O. N. F. (ONF), "P4 Open-Source Programming Language".

[5] opennetworking, "Open Networking Fundation," [Online]. Available: https://opennetworking.org/sdn-definition/.

[6] . A. Hegazy and M. El-Aasser, "Network Security Challenges and Countermeasures in SDN Environments," in *Eighth International Conference on Software Defined Systems (SDS)*, 2021.

[7] A. O. Jefia, . S. I. Popoola, and . A. A. Ata, "Software-Defined Networking: Current Trends, Challenges and Future Directions.," 2019.

[8] R. Enss, M. Schoenwaelder and J. Bierman, "Network Configuration Protocol (NETCONF). RFC6241," 2011.

[9] A. Bierman, M. Bjorklund and K. A. Watsen, "RESTCONF Protocol. RFC8040," 2017.

[10] M. Bjorklund, "The YANG 1.1 Data Modeling Language," 2016.

[11] C. E. R. Results, "Industry-Driven Elastic and Adaptive Lambda Infrastructure for Service and Transport Networks," [Online]. Available: https://cordis.europa.eu/project/id/317999.

[12] C. E. R. Results, "Scalable and efficienT oRchestrAtion of Ethernet services Using Software-defined and flexible optical networkS-EU," [Online]. Available: https://cordis.europa.eu/project/id/608528.

[13] C. EU, "METRO High bandwidth, 5G Application-aware optical network, with edge storage, compUte and low Latency," [Online]. Available: https://cordis.europa.eu/project/id/761727.

[14] R. Casellas, R. Munoz, M. R, R. Vilalta, L. Liu, T. Tsuritani, I. Morita, V. Lopez, O. de Dios and J. Fernandez-Palacios, "SDN Orchestration of OpenFlow and GMPLS Flexi-Grid Networks With a Stateful Hierarchical PCE," 2015.

[15] R. Muñoz, R. Vilalta, M. S. Moreolo, J. Fàbrega, R. Casellas, F. Vílchez, R. Martínez, S. Frigerio and A. Lometti, "Dynamic Differential Delay Aware RMSA for Elastic Multi-path Provisioning in GMPLS Flexi-grid DWDM Networks," *Zenodo OFC,* 2014.

[16] V. Lopez and e. al., "Transport API: A Solution for SDN in Carriers Networks," in *42nd European Conference on Optical Communication*, 2016.

[17] V. Lopez, T. Tsuritani, N. Yoshikane, I. Morita, R. Muñoz, R. Vilalta, R. Casellas and R. Martínez, "End-to-end SDN orchestration in optical multi-technology and multi-domain scenarios," *Zenodo Array,* 2015.

[18] V. Lopez, R. Vilalta, V. Uceda, A. Mayoral, R. Casellas, R. Martínez, R. Muñoz and J. Palacios, "Transport api: A solution for SDN in carriers networks," in *European Conference on Optical Communication, ECOC*, 2016.

[19] J. -J. Pedreno-Manresa, P. S. Khodashenas, M. S. Siddiqui and P. Pavon-Marino, "On the Need of Joint Bandwidth and NFV Resource Orchestration: A Realistic 5G Access Network Use Case," *IEEE Communications Letters,* pp. 145-148, 2018.

[20] L. Gifre, J.-L. Izquierdo-Zaragoza, M. Ruiz and L. Velasco, "Autonomic Disaggregated Multilayer Networking," *Optical Communications and Networking,* pp. 482-492, 2018.

[21] R. Casellas, R. Martínez, R. Vilalta and R. Muñoz, "Control, Management, and Orchestration of Optical Networks: Evolution, Trends, and Challenges," *Lightwave Technology,* pp. 1390-1402, 2018.

[22] R. Vilalta, "Upcoming ETSI TeraFlowSDN release 2 features," 21 Sep 2022. [Online]. Available: https://teraflow-h2020.eu/blog/upcoming-etsi-teraflowsdn-release-2-features.

[23] A. Farrel, J. -P. Vasseur and J. Ash, "A Path Computation Element (PCE)-Based Architecture. RFC4655," 2006.

[24] Y. L. H. Rekhter, " Border Gateway Protocol 4 (BGP4). A Internet Research Task Force," 2006.

[25] M. Fedor, M. Schoffstall and J. Davin, "A Simple Network Management Protocol (SNMP). RFC1157," 1990.

[26] K. Kompella, J. Drake, S. Amante, W. Henderickx and L. Yong, "The Use of Entropy Labels in MPLS Forwarding. RFC6790," 2012.

[27] S. Barguil, O. Gonzalez de Dios, O. Boucadair and M. Mun, "A Layer 3 VPN Network YANG Model draft-ietf-opsawg-13sm-13nm-18.," 2021.

[28] "Next Generaion Mobile Networks Alliance (NGMN)," 2022 Septiembre 2022. [Online]. Available: https://www.ngmn.org/.

[29] L. Contreras and D. López, "A Network Service Provider Perspective on Network Slicing," 2018.

[30] E. T. S. I. (ETSI).

[31] A. Aguado, V. Martin, D. Lopez, M. Peev, J. Martinez-Mateo, J. L. Rosales, F. d. L. Iglesia, M. Gomez, E. Hugues-Salas, A. Lord, R. Nejabati and D. Simeonidou, "Quantum-Aware Software Defined Networks".

[32] A. Aguado, V. Martin, D. López and e. al., "The Engineering of a SDN Quantum Key Distribution Network," *IEEE Comms. Mag.,* no. Special number "The Future of Internet", July 2019.

[33] E. Bolas, G. Capela and L. Bastos, "Protected core networking: communications challenges in tactical environments," in *IEEE Military Communications Conference (MILCOM)*, 2021.

[34] D. Boonstra, T. Hartog, H. Schotanus and C. Verkoelen, "A SECURE NEC-ENABLING ARCHITECTURE DISENTANGLING INFRASTRUCTURE, INFORMATION AND SECURITY," in *TNO*, 2016.

[35] D. McCallam, "Applying Protected Core Networking Concepts to Develop a Protected Trust Based Security Environment," in *Security Northrop Grumman Corporation*, 2009.

[36] NIST, "Computer Security Resource Center - Glossary," 2015. [Online]. Available: https://csrc.nist.gov/glossary/term/RED_BLACK_concept.

[37] NSA, "TEMPEST," [Online]. Available: https://apps.nsa.gov/iaarchive/programs/iad-initiatives/tempest.cfm. [Accessed 29 September 2022].

[38] (NSA), "MOBILE ACCESS CAPABILITY PACKAGE V2.1," *National Security Agency/Central Security Service,* 2018.

[39] L. Contreras, "Considering ALTO as IETF Network Exposure Function - draft-contreras-alto-ietf-nef-01," 2023.

[40] D. King and A. Farrel, "A PCE-Based Architecture for Application-Based Network Operations (RFC7491)," 2015.

[41] S. Randriamasy, W. Roome and N. Schwan, "Multi-Cost Application-Layer Traffic Optimization (ALTO) (RFC8189)," 2017.

[42] Q. Wu, Y. Yang, Y. Lee, D. Dhody, S. Randriamasy and L. Contreras, "ALTO Performance Cost Metrics draft-ietf-alto-performance-metrics-28," 2022.

[43] "OpenDayLight," [Online]. Available: https://www.opendaylight.org/what-we-do/current-release/sodium. [Accessed 27 09 2022].

[44] "Karaf OSGi," [Online]. Available: https://karaf.apache.org/.

[45] "ONF ONOS," [Online]. Available: https://wiki.onosproject.org/display/ONOS/System+Components. [Accessed 27 09 2022].

[46] T. GCTIO, "GCTIO SDN Network Strategy," 2018.

[47] A. Gupta, L. Vanbever, M. Shahbaz, S. P. Donovan, B. Schlinker, N. Feamster, J. Rexford, S. Shenker, R. Clark and E. Katz-Bassett, "SDX: A Software Defined Internet Exchange," in *SIGCOMM'14*, Chicago, IL, USA, 2014.

[48] Cisco, Content Delivery Network (CDN) Federations How SPs Can Win the Battle for Content-Hungry Consumers.

[49] IETF, "IETF CDNi," [Online]. Available: https://datatracker.ietf.org/wg/cdni/charter.

[50] SAIL, "SAIL Project," [Online]. Available: http://www.sail-project.eu/wp-content/uploads/2013/01/SAIL_DA8_Final_Public.pdf.

[51] Alimi, R. Penno, R. Yang, Y. Kiesel, S. Previdi, S. Roome, W. Shalunov and R. S. Woundy, "Application-Layer Traffic Optimization (ALTO) Protocol. RFC7285," 2014.

[52] E. T. S. I. (ETSI), "ETSI GS KD 018. Quantum Key Distribution (QKD); Orchestration Interface for Software Defined Networks," 2022.

[53] E. T. S. I. (ETSI), "ETSI GS KD 015. Quantum Key Distribution (QKD); Control Interface for Software Defined Networks," 2021.

[54] O. González, L. Munoz, M. Vázquez and L. Ndifor, "MUST IP SDN Controller NBI Technical Requirements," Telecom Infra Project, 2014.

[55] O. González, L. Munoz, M. Vázquez, L. Ndifor, L. Mphahlele, R. Díaz, P. Niger, E. Lerouzic, S. Barguil and C. Rodríguez, "MUST IP – SDN Controller SBI / Router NBI Technical Requirements," Telecom Infra Project, 2021.

[56] A. Mayoral and e. al, "MUST Optical SDN Controller NBI Technical Requirements OOPT," Telecom Infra Project, 2021.

[57] A. Kumar, J. Kolhe and L. Ryan, "gRPC Protocol draft-kumar-rtgwg-grpc-protocol-00," 2017.

[58] (ETSI) European Telecommunications Standards Institute, "ETSI GS QKD 004, Quantum Key Distribution (QKD); Application Interface," no. V2.1.1, 2020.

[59] (ETSI) European Telecommunications Standards Institute, "ETSI GS QKD 014, Quantum Key Distribution (QKD); Protocol and data format of REST-based key delivery API," no. V1.1.1, 2019.

[60] T. Wang and H. Chen, "SGuard: A Lightweight SDN Safe-Guard Architecture for DoS Attacks[J].," *China Communications,* pp. 113-125, 2017.

[61] M. Imran, M. H. Durad, F. A. Khan and H. Abbas, "DAISY: A detection and mitigation system against denial-of-service attacks in software-defined networks," *IEEE Syst. J.,* vol. 14, no. 2, pp. 1933-1944, 2020.

[62] S. Popic, M. Vuleta, P. Cvjetkovic and B. M. Todorović, "Secure Topology Detection in Software-Defined Networking with Network Configuration Protocol and Link Layer Discovery Protocol," in *International Symposium on Industrial Electronics and Applications (INDEL)*, 2020.

[63] C. Lee, C. Yoon, S. Shin and S. K. Cha, "INDAGO: A new framework for detecting malicious SDN applications," pp. 220-230, Sep 2018.

[64] C. Lee and S. Shin, "SHIELD: An automated framework for static analysis of SDN applications," in *Proc. ACM Int. Workshop Secur. Softw. Defined Netw. Netw. Function Virtualization*, New York, NY, USA, 2016.

[65] T. Hu, P. Yi, Y. Hu, J. Lan, Z. Zhang and Z. Li, "SAIDE: Efficient application interference detection and elimination in SDN," *Comput. Netw.,* vol. 183, no. 107619, December 2020.

[66] R. Chang, Z. Lin, Y. Sun and J. Xu, "MD-UCON: A multi-domain access control model for SDN northbound interfaces," *J. Phys., Conf. Ser.,* vol. 1187, no. 32091, 2019.

[67] H. Padekar, Y. Park, H. Hu and S.-Y. Chang, "Enabling dynamic access control for controller applications in software-defined networks," in *Proc. 21st ACM Symp. Access Control Models Technol.*, New York, NY, USA, 2016.

[68] INRIA, "Decentralized SDN Control Plane for a Distributed Cloud-Edge Infrastructure: A Survey. Rapport technique," 2020. [Online]. Available: https://hal.inria.fr/hal-02868984. [Accessed September 2022].

[69] Z. G. P. Y. T. B. a. J. L. T. Hu, "Multi-controller Based Software-Defined Networking: A Survey," *IEEE Access,* vol. 6, pp. 15980-15996, 2018.

[70] R. S. a. N. M. B. Heller, "The controller placement problem," *1st Workshop HotSDN,* pp. 7-12, 2012.

[71] V. R. S. M. N. a. A. S. H. K. Rath, "Optimal controller placement in Software Defined Networks (SDN) using a non-zero-sum game," in *Proceeding of IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks 2014*, 2014, pp. 1-6.

[72] M. B. T. T. a. I. B. A. Ksentini, "On using bargaining game for Optimal Placement of SDN controllers," in *2016 IEEE International Conference on Communications (ICC)*, 2016, pp. 1-6.

[73] A. Sallahi and M. St-Hilaire, "Optimal Model for the Controller Placement Problem in Software Defined Networks," *IEEE Communications Letters,* vol. 19, no. 1, pp. 30-33, 2015.

[74] e. a. Y. Fu, "A Hybrid Hierarchical Control Plane for Flow-Based Large-Scale Software-Defined Networks," *IEEE Transactions on Network and Service Management,* vol. 12, no. 2, pp. 117-131, 2015.

[75] Z. Z. J. P. R. L. G. Wang, "An approximate algorithm of controller configuration in multi-domain SDN architecture," in *9th International Conference on Communications and Networking in China*, 2014, pp. 601-605.

[76] X. Z.-y. e. a. Peng, "A K self-adaptive SDN controller placement for wide area networks," *Frontiers of Information Technology & Electronic Engineering,* vol. 17, no. 7, pp. 620-633, 2016.

[77] D. Dotan and R. Y. Pinter, "HyperFlow: an integrated visual query and dataflow language for end-user information analysis," in *2005 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC'05)*, 2005, pp. 27-34.

[78] R. Y. Shtykh and T. Suzuki, "Distributed Data Stream Processing with Onix," in *2014 IEEE Fourth International Conference on Big Data and Cloud Computing*, 2014, pp. 267-268.

[79] C. C. K. W. a. Y. H. H. Ho, "A fast consensus algorithm for multiple controllers in software-defined networks," in *2016 18th International Conference on Advanced Communication Technology (ICACT)*, 2016, pp. 1-1.

[80] C. W. W. G. X. H. M. J. a. S. C. B. Zhou, "Achieving consistence for cross-domain WAN control in Software-Defined Networks," *China Communications,* vol. 12, no. 10, pp. 136-146, 2015.

[81] Z. e. a. Guo, "Improving the performance of load balancing in software-defined networks through load variance-based synchronization," *Computer Networks,* vol. 68, pp. 95-109, 2014.

[82] M. B. a. J. L. K. Phemius, "DISCO: Distributed multidomain SDN controllers," in *2014 IEEE network operations and management symposium (NOMS)*, 2014, pp. 1-4.

[83] X. Xiong and J. Fu, "Active Status Certificate Publish and Subscribe Based on AMQP," in *2011 International Conference on Computational and Information Sciences*, 2011, pp. 725-728.

[84] J. D. C. J. e. a. Zhou W, "Enforcing customizable consistency properties in software-defined networks," in *12th USENIX Symposium on Networked Systems Design and Implementation (NSDI 15)*, 2015, pp. 73-85.

[85] C. C.-P. a. A. J. G. Y. Jiménez, "On the controller placement for designing a distributed SDN control layer," in *2014 IFIP Networking Conference*, 2014, pp. 1-9.

[86] B. P. R. Killi and S. V. Rao, "Optimal Model for Failure Foresight Capacitated Controller Placement in Software-Defined Networks," *IEEE Communications Letters,* vol. 20, no. 6, pp. 1108-1111, 2016.

[87] K. S. e. a. Sahoo, "Optimal controller selection in Software Defined Network using a Greedy-SA algorithm," in *2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom)*, 2016, pp. 2342-2346.

[88] W. W. G. X. e. a. Hu Y, "BalanceFlow: Controller load balancing for OpenFlow networks," in *2012 IEEE 2nd International Conference on Cloud Computing and Intelligence Systems*, vol. 2, 2012, pp. 780-785.

[89] G. G. a. F. A. H. Selvi, "Cooperative load balancing for hierarchical SDN controllers," in *2016 IEEE 17th International Conference on High Performance Switching and Routing (HPSR)*, 2016, pp. 100-105.

[90] H. Sufiev and Y. Haddad, "A dynamic load balancing architecture for SDN," in *2016 IEEE International Conference on the Science of Electrical Engineering (ICSEE)*, 2016, pp. 1-3.

[91] B. J. W. J. C. Z. W. K. a. L. M. F. Yonghong, "A dormant multi-controller model for software defined networking," *China Communications,* vol. 11, no. 3, pp. 45-55, 2014.

[92] F. H. S. M. T. V. L. a. R. R. A. Dixit, "ElastiCon; an elastic distributed SDN controller," in *2014 ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS)*, 2014, pp. 17-27.

[93] G. C. Z. W. Hongchang Chen, "A Game- Theoretic Approach to Elastic Control in Software-Defined Networking," *China Communications,* vol. 13, no. 5, pp. 103-109, 2013.

[94] X. Y. W. A. e. a. Cello M, "BalCon: A Distributed Elastic SDN Control via Efficient Switch Migration," in *2017 IEEE International Conference on Cloud Engineering (IC2E)*, 2017, pp. 40-50.

[95] C. H. W. Z. e. a. Cheng G, "DHA: Distributed decisions on the switch migration toward a scalable SDN control plane," in *2015 IFIP Networking Conference (IFIP Networking)*, 2015, pp. 1-9.

[96] "Atomix architectures," [Online]. Available: https://atomix.io/docs/latest/user-manual/introduction/architectures/. [Accessed November 2020].

[97] "Akka Documentation," [Online]. Available: https://doc.akka.io/docs/akka/current/typed/guide/introduction.html. [Accessed September 2022].

[98] C. W. a. I. H. M. Darianian, "Experimental Evaluation of Two OpenFlow Controllers," in *2017 IEEE 25th International Conference on Network Protocols (ICNP)*, 2017, pp. 1-6.

[99] "Opendaylight geo-distributed active/backup setup," [Online]. Available: https://docs.opendaylight.org/en/stable-carbon/getting-started-guide/common-features/clustering.html#geo-distributed-active-backup-setup. [Accessed September 2022].

[100] H. Jordan, "Distributed Systems in ONOS with Atomix 3," [Online]. Available: https://www.opennetworking.org/wp-content/uploads/2018/12/Distributed-Systems-in-ONOS-with-Atomix-3.pdf. [Accessed September 2022].

[101] "ONOS mastership failover test," [Online]. Available: https://wiki.onosproject.org/display/ONOS/Master%3A+Experiment+M+-+Mastership+Failover+Latency. [Accessed September 2022].

[102] "ONOS Regions," [Online]. Available: https://wiki.onosproject.org/display/ONOS/GUI+Topology+2+View. [Accessed September 2022].

[103] "ONOS intent framework," [Online]. Available: https://wiki.onosproject.org/display/ONOS/Intent+Framework. [Accessed September 2022].

[104] "OpenDaylight Federation with rabbit mq," [Online]. Available: https://docs.opendaylight.org/en/stable-carbon/submodules/federation/docs/developer-guide/federation-developer-guide.html. [Accessed September 2022].

[105] "ONOS ICONA project," [Online]. Available: https://wiki.onosproject.org/pages/viewpage.action?pageId=4162523. [Accessed September 2022].

[106] "grpc," [Online]. [Accessed September 2022].

[107] "Micro-onos," [Online]. [Accessed September 2022].

[108] (ETSI), "ETSI GS KD 015. Quantum Key Distribution (QKD); Control Interface for Software Defined Networks," *European Telecommunications Standards Institute,* 2021.

[109] IETF RFC 6020, "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)," 2010.

[110] IETF RFC 7950, "The YANG 1.1 Data Modeling Language," 2016.

[111] ISO, "International Organization for Standardization," [Online]. Available: https://www.iso.org/home.html.

[112] J. Vasseur and J. Le Roux, "Path Computation Element (PCE) Communication Protocol (PCEP). RFC5440," 2009.

[113] T. Xirofotos, D. Klonidis, R. Vilalta, L. Gifre, R. Martínez, J. Moreno, S. González, S. Pettereti Valiviita, O. Gonzalez de Dios, P. Stritzinger, A. Farrer and D. King, "D3.1: Preliminary Evaluation of Life-cycle Automation and High Performance SDN Components (TeraFLow)," 2021.

[114] B. S., O. Gonzalez de Dios, M. Boucadair, L. Munoz and A. Aguado, "A Layer 3 VPN Network YANG Model draft-ietf-opsawg-13sm-13nm-18.," 2021.

[115] S. Barguil, O. Gonzalez de Dios, M. Boucadair, L. Munoz, L. Jalil and J. A. Ma, "Layer 2 VPN Network YANG Model draft-ietf-opsawg-12nm-00," 2020.

[116] P. Hitchen, N. Leymann, W. Henderickx, A. Gulko and J. Tantsura, "Multipoint Label Distribution Protocol In-Band Signaling in a Virtual Routing and Forwarding (VRF) Table Context. RFC7426," 2014.

[117] A. Lindem, K. Patel, S. Zandi, J. Haas and X. Xu, "BGP Logical Link Discovery Protocol (LLDP) Peer Discovery draft-acee-idr-lldp-peer-discovery-13," 2022.

[118] S. Previdi, B. Decraene, S. Litkowsky and R. Shakir, "Segment Routing Architecture," 2018.

[119] (NIST) National Institute of Standards and Technology, "Recommendation for Key Management: Part 1 – General," no. SP 800-57 Part 1 Rev. 5, 2020.

[120] BSI - Federal Office for Information Security, "Quantum-safe cryptography – fundamentals, current developments and recommendations," 2021.

[121] "Global Environment for Network Innovations (GENI)," [Online]. Available: https://www.geni.net/.

[122] "National Science Foundation (NSF)," [Online]. Available: http://www.nets-find.net/.

[123] M. Lemke, "The European FIRE Future Internet Research and Experimentation Initiative,," 2009 .

[124] "Defense Advanced Research Projects Agency (DARPA)," [Online]. Available: https://www.darpa.mil/.

[125] I. E. T. F. (IETF), "Forwarding and Control Element Separation (FORCES)," [Online]. Available: https://datatracker.ietf.org/wg/forces/about/.

[126] R. Droms, "Dynamic Host Configuration Protocol (DHCP). RFC2131," 1997.

[127] M. Casado and et al, "A protection architecture for enterprise networks," 2006.

[128] M. Casado and et al., "Taking control of the enterprise.," 2007.

[129] "(ETSI), European Telecommunications Standards Institute," [Online]. Available: https://www.etsi.org/.

[130] (ETSI), "ETSI GS KD 018. Quantum Key Distribution (QKD); Orchestration Interface for Software Defined Networks," *European Telecommunications Standards Institute,* 2022.

[131] NSA, "MOBILE ACCESS CAPABILITY PACKAGE V2.1.," 2018.

[132] e. a. P. Layec, "IDEALIST data plane solutions for elastic optical networks," in *European Conference on Networks and Communications (EuCNC)*, 2015.

[133] e. a. P. Layec, "IDEALIST data plane solutions for elastic optical networks," in *European Conference on Networks and Communications (EuCNC)*, 2015.

[134] R. Mahmud, A. N. Toosi, M. A. Rodriguez, S. C. Madanapalli, V. Sivaraman, L. Sciacca, C. Sioutis and R. Buyya, "Software-Defined Multi-domain Tactical Networks: Foundations and Future Directions," in *Mobile Edge Computing*, Springer, 2021, pp. 183-227.

[135] Atomix, "atomix.io," [Online]. Available: https://atomix.io/latest/about/.