



SDN-μSense

Project No. 833955

Project acronym: SDN-microSENSE

Project title:

SDN - microgrid reSilient Electrical eNergy SystEm

Deliverable D4.2

Network management processes

Programme: H2020-SU-DS-2018

Start date of project: 01.05.2019

Duration: 36 months

Editor: IREC

Due date of deliverable: 31/12/2020

Actual submission date: 23/12/2020



DELIVERABLE DESCRIPTION:

Deliverable Name	Network Management Processes
Deliverable Number	D4.2
Work Package	WP 4
Associated Task	T4-2
Covered Period	M4-M20
Due Date	M18 moved to M20 in amendment
Completion Date	M20
Submission Date	23/12/2020
Deliverable Lead Partner	IREC
Deliverable Author(s)	OINF, ALKYONIS, AYESA, CERTH, ENERGYNAUTICS, IEIT, IREC, REAL, PPC, IPTO and UOWM.
Version	1.0

Dissemination Level		
PU	Public	X
PP	Restricted to other programme participants (including the Commission Services)	
RE	Restricted to a group specified by the consortium (including the Commission Services)	
CO	Confidential, only for members of the consortium (including the Commission Services)	

CHANGE CONTROL

DOCUMENT HISTORY

Version	Date	Change History	Author(s)	Organisation
0.1	31/08/2020	ToC Defined. All the involved partners start contributing	Representatives of all the partners involved	All the partners
0.2	23/11/2020	Draft version ready for review	Representatives of all the partners involved	All the partners
0.3	9/12/2020	Second version ready for review	Representatives of all the partners involved	IREC
1.0	22/12/2020	Final version. Included a separate annex for API details (considered CO)	IREC, CERTH	IREC

DELIVERABLE AUTHORS

AUTHOR	INSTITUTION
YANNIS SPYRIDIS	OINF
ZHONGLIN SUN	OINF
ACHILLEAS SESIS	OINF
GEORGIOS EFSTATHOPOULOS	OINF
NIKOS SIAXABANIS	ALKYONIS
ANGEL JAVIER JIMENEZ PEREZ	AYESA
CRISTINA MARTÍN TORRES	AYESA
JOSE MANUEL GARCIA CAMPOS	AYESA
ACHILLEAS PASIAS	CERTH
NIKOS VAKAKIS	CERTH
GEORGE LAZARIDIS	CERTH
ATHANASIOS KOTSIPOULOS	CERTH
KOSTAS PAPACHRISTOU	CERTH
ASTERIOS MPATZIAKAS	CERTH
NIS MARTENSEN	ENERGYNAUTICS
MARIA ATANASOVA	IEIT
MAGDA ZAFEIROPOULOU	IEIT
PANAGIOTIS FAMELIS	IPTO
ALBA COLET SUBIRACHS	IREC
TONI CANTERO GUBERT	IREC
JORGE ALEJANDRO TORRES	IREC
POL PARADELL SOLA	IREC
CHRISTOS DALAMAGKAS	PPC
SOLON ATHANASOPOULOS	PPC
ANTONIOS KARNEMIDIS	PPC
ATHANASIOS KOURAPAS	PPC

IOANNIS ZOIS	PPC
NIKOLA PAUVONIC	REAL
DIMITRIS PAPAMARTZIVANOS	UBITECH
THOMAS LAGKAS	UOWM
PANAGIOTIS RADOGLLOU-GRAMMATIKIS	UOWM
PANAGIOTIS SARIGIANNIDIS	UOWM
ANTONIS PROTOPSALTIS	UOWM
ANNA TRIANTAFYLLOU	UOWM
STAMATIA BIBI	UOWM
GEORGE KARAGIANNIDIS	UOWM
MILTADIS PARCHARIDIS	UOWM

SAB APPROVAL

NAME	INSTITUTION	DATE
DR. MARC STAUCH ON BEHALF OF PROF. DR. TINA KRÜGEL	LUH	23/12/2020
MR. DAVID PAMPLIEGA	SCHN ES	23/12/2020
DR. DAVE RAGGETT	ERCIM	23/12/2020

ACADEMIC AND INDUSTRIAL PARTNER REVISION

NAME	INSTITUTION	DATE
Panagiotis Radoglou	Academic partner: UOWM	22/12/2021
Magda Zafeiropoulou	Industrial partner: IEIT	26/11/2021

QUALITY MANAGER REVISION

NAME	INSTITUTION	DATE
Anastasios Drosou on behalf of Dimosthenis Ioannidis	CERTH	23/12/2020

DISTRIBUTION LIST

Date	Issue	Group
23/11/2020	Revision	WP4 involved partners, UOWM(Academic reviewer), IEIT (Industrial reviewer), SAB, Quality Manager and Technical Manager
23/12/2020	Acceptance	WP4 involved partners, UOWM(Academic reviewer), IEIT (Industrial reviewer), SAB, Quality Manager and Technical Manager
23/12/2020	Submission	Ayesa Advanced Technologies

Table of contents

Table of contents	6
List of acronyms	11
List of figures	13
List of tables	16
1 Introduction	19
1.1 Purpose of the deliverable	19
1.2 Relation with other WPs	19
1.3 Structure of the document	20
2 Self-healing techniques in Grid applications	21
2.1 Objective of self-healing	21
2.1.1 Safe systems	21
2.1.2 Fault-tolerant systems	22
2.1.3 Resilient Systems	24
2.2 Self-healing methods	25
2.2.1 General/Traditional Self-healing Methods in Power Systems	25
2.2.1.1 Reserves	25
2.2.1.2 Automatic Generation Control (AGC)	26
2.2.1.3 Low Frequency Demand Disconnection	27
2.2.1.4 Automatic Voltage Control (AVC)	27
2.2.2 Self-healing methods in smart grids	28
2.2.2.1 Variable Renewable Energy (VRE) and storage control	28
2.2.2.2 Demand response and demand side management (DR/DSM)	29
2.2.2.3 Controlled islanding (microgrids)	30
2.2.2.4 Optimized grid reconfiguration	31
2.2.2.5 SDN-Based Self-Healing Communication Network	31
2.2.3 Summary and Conclusions on Self-Healing Methods	33
3 Analysis of requirements for EDAE, SDN Controller and Northbound Interfaces	34
3.1 D2.2 Functional Requirements	34
3.2 SDN-microSENSE platform specifications (related to the WP4)	36
3.3 Technical constraints	39
3.4 Functional and non-functional requirements coverage	39
3.4.1 EDAE requirements	39



4	<i>Architecture and detailed design</i>	42
4.1	Architecture overview	42
4.2	Components view	43
4.2.1	Detailed information flow of the inputs and outputs	44
4.3	Interfaces Model	46
4.3.1	AIDB – EDAE	48
4.3.2	AIDB – EDAE Dashboard	49
4.3.3	AIDB – S-RAF	49
4.3.4	AIDB – SDN-C	51
4.3.5	EDAE – EDAE-Dashboard	51
4.3.6	EDAE – S-RAF	52
4.3.7	EDAE – SDN-C	53
4.3.8	EDAE-Dashboard – SDN-C	54
5	<i>SDN Controller design and implementation</i>	56
5.1	Interfaces Model - Northbound Interfaces	56
5.1.1	Rest_topology API	56
5.1.2	Ofctl_rest API	57
5.1.3	Components Model	57
5.1.3.1	SDN dashboard architecture	57
5.1.3.2	Database schema	58
5.1.3.3	User roles and privileges	59
5.1.4	User Interfaces	60
5.1.4.1	Users	60
5.1.4.2	Homepage	64
5.1.4.3	Flows	64
5.1.4.4	Topology	66
5.1.4.5	Settings	67
5.1.5	SDN dashboard prototype deployment	68
5.1.5.1	Prerequisites and Installation	70
5.1.5.2	Source code repository	72
6	<i>Electric data Analysis engine (EDAE) design and implementation</i>	74
6.1	SDN-based System Model	74
6.1.1	Network	75
6.1.2	Power Grid	75

6.2	EDAE core: Architecture.....	75
6.2.1	Graph Construction Module	76
6.2.2	Control Module	76
6.2.3	Solvers Module.....	76
6.3	EDAE Algorithm formulation.....	77
6.3.1	Related methods from the literature	77
6.3.2	Problem formulation	79
6.3.2.1	Variables and Symbols	79
6.3.2.2	Search for optimal path	80
6.3.2.3	PMU-PDC allocation for EPES observability	86
6.3.3	Problem Definition and Use Cases	88
6.4	EDAE Dashboard	89
6.4.1	Architecture and functionality	91
6.4.2	Interfaces.....	93
6.4.2.1	EDAE-Dashboard – EDAE: Network topology changes	94
6.4.2.2	EDAE-Dashboard – EDAE: Network topology proposal acceptance	94
6.4.2.3	EDAE-Dashboard – AIDB	95
6.4.2.4	EDAE-Dashboard – SDN Controller	95
6.5	Component Model	95
6.6	Interfaces Model	98
7	EDAE Core Engine Evaluation.....	102
7.1	Evaluation framework of re-routing functionality	102
7.1.1	Structure of the network topologies.....	102
7.1.1.1	One Ring Bottleneck Topology (ORB)	102
7.1.1.2	Two Ring Bottleneck Topology (TRB)	103
7.1.2	Scale of the network topologies	104
7.1.3	QoS requirements	104
7.1.3.1	Definition of QoS requirements	104
7.1.3.2	Modelling of QoS requirements.....	105
7.1.4	Modelling the status of the network	106
7.1.5	Comparison of EDAE’s modelling with something.....	107
7.2	Results over the evaluation framework.....	107
7.2.1	Evaluation details	107
7.2.2	Computational time	108



7.2.3	Objective success ratio and resource management	110
7.2.3.1	Delay objective.....	111
7.2.3.2	Jitter objective.....	113
7.2.3.3	Bandwidth objective	115
7.2.3.4	Packet loss objective.....	117
7.2.3.5	Security objective.....	119
7.3	Evaluation framework for the maximization of the EPES observability	121
8	<i>Assets inventory database design and implementation</i>	123
8.1	Interface model	124
8.2	AIDB Information	125
8.2.1	SDN asset modelling.....	125
8.2.1.1	Switch	127
8.2.1.2	Host	128
8.2.1.3	SDN Controller.....	129
8.2.2	SDN Topology.....	129
8.2.2.1	SDN Topology Asset node	130
8.2.2.2	SDN Topology Relationship	130
8.2.3	Grid asset modelling	131
8.2.4	Grid topology.....	132
8.2.5	SDN and Grid asset relationships.....	132
8.2.6	Vulnerability information	133
8.3	Asset inventory database Implementation	135
8.3.1	Architecture	135
8.3.2	LDAP	135
8.4	Asset inventory API	135
8.4.1	Code lists	137
8.4.2	Data partitions	138
8.4.3	<i>AssetQuery()</i>	138
8.4.4	<i>Asset Create/Update/Delete()</i>	139
8.4.5	<i>TopologicQuery()</i>	140
8.4.6	<i>GridModelQuery()</i>	141
8.4.7	<i>GridModelUpdate()</i>	141
8.4.8	<i>AssetRiskQuery()</i>	141
9	<i>Unit Testing</i>	143

9.1	Northbound Interface unit tests	143
9.1.1	Technical environment	143
9.1.2	Unit tests	144
9.2	SDN Dashboard	153
9.2.1	Technical environment	153
9.2.2	Unit tests	154
9.3	EDAE core	161
9.3.1	Technical environment	161
9.3.2	Unit Tests	161
9.4	EDAE workflow	173
9.4.1	Technical environment	173
9.4.2	Unit tests	173
9.5	EDAE-Dashboard	177
9.5.1	Technical environment	177
9.5.2	Unit tests	177
9.6	Assets inventory database	180
9.6.1	Technical environment	180
9.6.2	Unit tests	180
10	<i>Innovation Summary</i>.....	185
11	<i>Conclusions</i>	187
12	<i>References</i>.....	188
13	<i>Annexes</i>	194
13.1	EDAE Interface details.....	194
13.2	Grid model example A.....	204
13.3	AIDB unit test results	206
13.4	EDAE unit testing details.....	217
13.4.1	Inputs for the workflow	217
13.4.2	Output of the workflow	227
13.5	SDN-C API details.....	228

List of acronyms

AGC	Automatic Generation Control
AIDB	Asset Inventory DataBase
AVC	Automatic Voltage Control
AVR	Automatic Voltage Regulation
API	Applications Programming Interface
ASGI	Asynchronous Server Gateway Interface
DER	Distributed Energy Resources
DDoS	distributed Denial-of-Service
DDPG	Deep Deterministic Policy Gradients
DoS	Denial-of-Service
DRL	Deep Reinforcement Learning
EDAE	Electrical Data Analysis Engine
EPES	Electrical Power and Energy Systems
EU	European Union
EVEM	Electric Vehicle Energy management
FCR	Frequency Containment Reserves
FPGA	Field Programmable Gate Array
FRR	Frequency Restoration Reserves
IED	Intelligent Electronic Devices
JSON	JavaScript Object Notation
M2M	Machine-to-Machine
MILP	Mixed-Integer Linear Programming
MITM	Man-In-The-Middle
MTD	Moving Target Defence
NBI	Northbound Interface
ONF	Open Network Foundation
PDC	Phasor Data Concentrator

PMU	Phasor Measurement Unit
RDBMS	Relational DataBase Management System
RTU	Remote Terminal Unit
SCADA	Supervisory Control and Data Acquisition
SAP	Shuffle Assignment Problem
SCS	Synchronisation and Coordination Service
SDN	Software-Defined Networking
SDN-C	SDN Controller
S-RAF	SDN-microSENSE Risk Assessment Framework
SUN	Smart Utility Networks
UDCC	Utility Data and Control Centre
VCS	Version Control System
VRE	Variable Renewable Energy
WAC	Wide-Area Control System
WAMS	Wide-Area Monitoring System
WAPS	Wide-Area Protection System
WAMPAC	Wide Area Monitoring, Protection and Control

List of figures

Figure 1: Types of reserves by the time and duration of activation (Source: ENTSO-E)	26
Figure 2: Example of controlled islanding in response to faults [39]	30
Figure 3: WAMS structure [42]	32
Figure 4: SDN-microSENSE Architecture Structural View	42
Figure 5: SDN-SELF in the application plane	42
Figure 6: Detail of the architecture overview related to task 4.2.....	43
Figure 7: Interaction between components	44
Figure 8 Detailed information flow of the Input and outputs	45
Figure 9: Architecture of the SDN dashboard	57
Figure 10: The ER diagram of the SDN dashboard	58
Figure 11: Login page of the SDN dashboard.....	60
Figure 12: Reset password page of the SDN dashboard.....	61
Figure 13: Reset password notification of the SDN dashboard	61
Figure 14: The user management system of the SDN dashboard	62
Figure 15: Profile editing menu of the SDN dashboard	62
Figure 16: New user menu of the SDN dashboard	63
Figure 17: Homepage of the SDN dashboard	64
Figure 18: Flows menu of the SDN dashboard.....	65
Figure 19: The Flow Control menu of the SDN dashboard	65
Figure 20: The topology menu of the SDN dashboard	67
Figure 21: The Settings menu of the SDN dashboard	68
Figure 22: Deployment architecture of the SDN dashboard	69
Figure 23: Import Appliance via Oracle VirtualBox.....	70
Figure 24: Locating the SDN dashboard OVA file.....	70
Figure 25: Import options for the SDN dashboard	71
Figure 26: Wait VirtualBox to finish importing the SDN dashboard OVA.....	71
Figure 27: Start the SDN dashboard image.....	72
Figure 28: SDN dashboard VM credentials	72
Figure 29: Aruba 2930F as a network switch.....	75
Figure 30: Schneider Electric RTU	75
Figure 31: EDAE Core Architecture	76
Figure 32: Illustration of the Naive (top) and EDAE's (Bottom) chromosome representation	86
Figure 33: EDAE-Dashboard user interface schema.	90
Figure 34: Architecture of the data acquisition and storage	92
Figure 35: Architecture of the front-end tool	92
Figure 36: Architecture of the EDAE proposal representation	93
Figure 37: Architecture of the EDAE proposal acceptance.....	93
Figure 38: Airflow workflow	96
Figure 39: Docker containers and APIs	96
Figure 40: Different machine/server in EDAE product environment	97
Figure 41: Relevant information for EDAE operation from S-RAF	98
Figure 42: Network topology selected to test EDAE.....	99
Figure 43: One Ring Bottleneck Topology.....	103
Figure 44: Two Ring Bottleneck Topology	104

Figure 45: Execution time to find a path for a pair of hosts for a specific topology for EDAE (ORB) ..	108
Figure 46: Execution time to find a path for a pair of hosts for a specific topology for [63] (ORB) ...	109
Figure 47: Execution time to find a path for a pair of hosts for a specific topology for EDAE (TRB) ..	109
Figure 48: Execution time to find a path for a pair of hosts for a specific topology for [63] (TRB)....	110
Figure 49: Statistics for delay objective for each topology [EDAE - (ORB)]	111
Figure 50: Statistics for delay objective for each topology [[63] - (ORB)].....	112
Figure 51: Statistics for delay objective for each topology [EDAE - (TRB)].....	112
Figure 52: Statistics for delay objective for each topology [[63] - (TRB)].....	113
Figure 53: Statistics for jitter objective for each topology [EDAE - (ORB)].....	113
Figure 54: Statistics for jitter objective for each topology [[63] - (ORB)].....	114
Figure 55: Statistics for jitter objective for each topology [EDAE - (TRB)].....	114
Figure 56: Statistics for jitter objective for each topology [[63] - (TRB)].....	115
Figure 57: Statistics for bandwidth objective for each topology [EDAE - (ORB)]	115
Figure 58: Statistics for bandwidth objective for each topology [[63] - (ORB)].....	116
Figure 59: Statistics for bandwidth objective for each topology [EDAE - (TRB)].....	116
Figure 60: Statistics for bandwidth objective for each topology [[63] - (TRB)]	117
Figure 61: Statistics for packet loss objective for each topology [EDAE - (ORB)].....	117
Figure 62: Statistics for packet loss objective for each topology [[63] - (ORB)]	118
Figure 63: Statistics for packet loss objective for each topology [EDAE - (TRB)].....	118
Figure 64: Statistics for packet loss objective for each topology [[63] - (TRB)].....	119
Figure 65: Statistics for security objective for each topology [EDAE - (ORB)].....	119
Figure 66: Statistics for security objective for each topology [[63] - (ORB)]	120
Figure 67: Statistics for security objective for each topology [EDAE - (ORB)].....	120
Figure 68: Statistics for security objective for each topology [[63] - (TRB)].....	121
Figure 69: AIDB - Asset Inventory update	123
Figure 70: AIDB internal components	124
Figure 71: Interfaces with AIDB	124
Figure 72: AIDB - Information areas.....	125
Figure 73: AIDB - SDN asset inventory example	126
Figure 74: Vulnerability info.....	133
Figure 75: AIDB/Vulnerability manager - Vulnerability life cycle states.....	134
Figure 76: Testing environment of the NBI.....	143
Figure 77: TLS traffic of NBI - Unit tests	148
Figure 78: Testing environment of the SDN dashboard	154
Figure 79: Initial communication paths	162
Figure 80: Resultant communication paths after security event	163
Figure 81: Initial communication paths	164
Figure 82: Resultant communication paths	165
Figure 83: Topology	166
Figure 84: Execution results	166
Figure 85: Example of a preliminary version of SRAF.json	199
Figure 86: AIDB Grid model example (single line)	204
Figure 87: AIDB - Grid Bus model.....	204
Figure 88: AIDB - Grid Trafo data model.....	204
Figure 89: AIDB - Grid Line data model.....	205

Figure 90: AIDB - Grid Load data model.....	205
Figure 91: AIDB - Grid external grid data model.....	205
Figure 92: AIDB - Grid Generator data model.....	205
Figure 93: AIDB - Grid Sgenerator data model	205

List of tables

Table 1: Self-healing methods and their applicability to electrical power distribution and transmission grids.....	33
Table 2: Extracted and summarised requirements for each component.....	35
Table 3: Specifications coverage of each component	37
Table 4 Functional and Non-functional requirements for EDAE operation.....	39
Table 5: AIDB-1 interface for EDAE component	48
Table 6: AIDB-2 interface for EDAE component	48
Table 7: AIDB-1 interface for EDAE-Dashboard component	49
Table 8: AIDB-1 interface for S-RAF component.....	50
Table 9: S-RAF-AIDB interface: Risk asset summary	50
Table 10: AIDB-3 interface for SDN-C component.....	51
Table 11: EDAE/EDAE-Dashboard interface for EDAE-Dashboard component	51
Table 12: EDAE/EDAE-Dashboard interface for EDAE component.....	52
Table 13: S-RAF/SDN-SELF-01 interface for EDAE component	52
Table 14: XL-EPDS_NBI-1 interface for EDAE component	53
Table 15: XL-EPDS_NBI-1 interface for EDAE-Dashboard component	54
Table 16: User Management Views	63
Table 17: Flows Views - SDN Dashboard.....	66
Table 18: Topology Views - SDN Dashboard	67
Table 19: Settings Views - SDN Dashboard	68
Table 20: Objective coverage per work displayed	77
Table 21: Notations and description of variables and symbols used in EDAE algorithm formulation ..	79
Table 22: Network topology changes message data	94
Table 23: Network topology proposal acceptance message data	94
Table 24 QoS requirements in applications related to an EPES	105
Table 25 Filtered applications and the required assets	105
Table 26 Ratio of each type of asset with respect to the total number of assets	106
Table 27: Evaluation results of MILP formulations	122
Table 28: AIDB – Switch table definition.....	127
Table 29: AIDB – Host table definition	128
Table 30: AIDB – SDN Controller table definition	129
Table 31: AIDB – SDN topology asset node table definition.....	130
Table 32: AIDB – SDN Controller table definition	130
Table 33: AIDB – SDN – Grid relationship table definition	133
Table 34: Vulnerability risk level ranges	133
Table 35: AIDB – vulnerability information table definition	134
Table 36: List of components used for the AIDB	135
Table 37: AIDB - Asset type code list.....	137
Table 38: AIDB - Asset level code list	137
Table 39: AIDB - Asset status code list.....	137
Table 40: AIDB - Asset Attribute code list	137
Table 41: AIDB – Asset query method	138
Table 42: AIDB - Create/Update/Delete asset operation methods.....	139
Table 43: AIDB – Topologic Query method.....	140

Table 44: AIDB – Asset risk query method	141
Table 45: NBI_01 unit test description	144
Table 46: NBI_02 unit test description	145
Table 47: NBI_03 unit test description	147
Table 48: NBI_04 test unit description	148
Table 49: NBI_05 test unit description	149
Table 50: NBI-06 test unit description	150
Table 51: NBI_07 test unit description	152
Table 52: NBI_08 test unit description	153
Table 53: SDNGUI_01 unit test description	154
Table 54: SDNGUI_02 unit test description	155
Table 55: SDNGUI_03 unit test description	156
Table 56: SDNGUI_04 unit test description	157
Table 57: SDNGUI_05 unit test description	158
Table 58: SDNGUI_06 unit test description	160
Table 59: List of software and version used for EDAE unit testing	161
Table 60: EDAE_GENETIC_01 unit test description	161
Table 61: EDAE_GENETIC_02 unit test description	163
Table 62: EDAE_GENETIC_3 unit test description	165
Table 63: EDAE_receiver_transmitter_1 unit test	167
Table 64: EDAE_receiver_transmitter_2 unit test	168
Table 65: EDAE_receiver_transmitter_3 unit test	170
Table 66: EDAE_receiver_transmitter_4.....	171
Table 67: List of software and version used for EDAE workflow implementation and testing	173
Table 68: EDAE_workflow1 unit test description	173
Table 69: EDAE_workflow2 unit test description	174
Table 70: EDAE_workflow3 unit test description	175
Table 71: EDAE_workflow4 unit test description	176
Table 72: EDAE_workflow5 unit test description	176
Table 73: List of software used for implementation and testing	177
Table 74: EDAE-Dashboard Unit tests summary	177
Table 75: EDAE_DASH_001 - Current SDN network state representation	177
Table 76: EDAE_DASH_002 – EDAE proposal representation	179
Table 77: Base software and version used for AIDB unit testing.....	180
Table 78: AIDB Unit test summary	181
Table 79: AIDB_001 SDN Asset creation - Unit Test	181
Table 80: AIDB_002 SDN Asset update - Unit Test	181
Table 81: AIDB_003 SDN topology attribute update - Unit Test	182
Table 82: AIDB_004 SDN topology update - Unit Test.....	182
Table 83: AIDB_005 Grid definition registration - Unit Test.....	183
Table 84: AIDB_006 SDN-Grid relationship creation - Unit Test.....	183

Executive Summary

Self-healing capabilities refer to resilience against all kinds of faults, including faults in both the communication system and in the electrical power grid. SDN applications as part of the self-healing concept architecture enable important self-healing capabilities especially on the communication layer.

In this document, a specific application called EDAE was developed as an innovative method for self-healing applied to smart grids. EDAE component is the brain of the network management process and it has been designed to maximize the observability and the QoS of the communication network. EDAE is not only rearranging logical connection in the communication network but also deciding the data paths based on the risks of the assets that are present in the electrical grid and information of the SDN switches.

The content of this deliverable is focused on the results of task T4.2, describing the network management processes of the EPES when it is compromised, either by accident or through cyber-attack. It is described in great detail the interfaces of EDAE and each of the components of SDN-uSENSE architecture: SDN-C, AIDB and S-RAF, the requirements achieved and the unit testing of those components.

1 Introduction

1.1 Purpose of the deliverable

This deliverable is the second document of Work Package 4 (WP4) of SDN-microSENSE. WP4 focuses on Cyber-secured & resilient SDN-based Energy Ecosystem: It includes components for the monitoring and control of the equipment in the EPES infrastructures, for the analysis of the current situation in EPES domain and issue mitigation through islanding or energy management mechanisms and also for the recovery of the grid observability and the improvement of its QoS when the communication network is under attack.

The content of this deliverable is focused on the results of task T4.2, describing the main EPES Self-Healing Network Management processes and defining the mechanisms of the EPES under attack, either by accident or through cyber-attack. These processes are establishing the corresponding network connections that are rearranged when these are under fault or under attack. This is achieved by establishing new communications paths: the SDN controller uses high-level decisions for guiding underlying switches to handle data flows throughout the communication network. The network control capabilities are separated from the switches that are supervised by the SDN controller. The SDN controller is made in such a way that it is protected from malfunction or failures and therefore the functionalities of the communication network can always be preserved. The approach and the way how it works is described within this document.

More specifically, this document is structured in three parts. The first part (Section 2) describes the state of the art and methodology on the self-healing techniques in grid applications. The second part of the document is related to the overview of the architecture model, related behaviour in the system architecture, requirements and specifications. Here are presented in a structured way (through Sections 3 and 4) overall specifications and requirements to be used as roadmap to development and implementation of all tools and their functionalities. The third part of this document (which includes Sections 5, 6, 7, 8 and 9) is related to the tools' specifications. These specifications includes tools' design and implementation description. This third part of the document is actually the description of the three main tools: SDN Controller, Electric Data Analysis Engine (EDAE), and Asset Inventory Database (AIDB).

The purpose of the deliverable is to give a detailed overview of the software components and toolsets that are involved in Task 4.2. EDAE, AIDB and SDN-C are the main components that are specified in this document.

1.2 Relation with other WPs

The following tasks and deliverables are related to the current report:

- D2.2, where the requirements of the SDN-microSENSE platform are elicited;
- D2.3 describes the SDN-microSENSE architecture and specifications;
- D2.4 [SDN24] describes the validation methodology and the list of threats and attacks associated to every pilot and use case;
- D3.5 describes the SDN-microSENSE Risk Assessment Framework and its very important part about sending the incidents towards the SDN-SELF framework. In this particular case the incidents are sent to the EDAE component;

- D4.1, where the SDN-based Measurement and Control Unit Configuration models are presented, which is used to support SDN-enabled RTU prototype. Also, this deliverable describes how all necessary interfaces and APIs are deployed, between the measurement and control units, the OpenFlow switches and the SDN controller.

1.3 Structure of the document

This document is structured as follows:

- Section 2 is focused on the self-healing techniques in grid applications. Extensive research was conducted in order to present detailed description on SOTA related to SDN relation to the self-healing techniques.
- Section 3 covers the analysis of requirements and specifications. These are agreed among project participants and they are presented within two deliverables from WP2, i.e. deliverables D2.2 and D2.3. Requirements and specifications are extracted from previously mentioned deliverables and presented only for the tools that are described in this document.
- Section 4 provides the WP4 architecture overview and describes relevant architecture and interfaces model. This model gives further information on the mutual interaction of the components.
- Section 5 describes the design and implementation of the SDN controller. Here are presented detailed information on how this component interact with other components and how the implementation is achieved.
- Section 6 describes the design and implementation of the component Electric Data Analysis Engine (EDAE). Here are also presented SDN-based system model and used algorithms that are used to implement mechanisms of the self-healing.
- Section 7 describes the EDAE core engine evaluation, a quantitative evaluation of the proposed component.
- Section 8 describes the Asset Inventory Database and the ways of its implementation.
- Section 9 details the unit testing to validate the mechanisms described in all previous sections.

2 Self-healing techniques in Grid applications

2.1 Objective of self-healing

The world's electrical power systems are undergoing major transformations with serious challenges, namely the large-scale integration of variable generation based on renewable energy; increasing need for energy efficiency; and fast increase of complexity due to new generation and transmission technologies and new control systems. While the power systems have always been designed to withstand some level of faults, the transformation also introduces new vulnerabilities to new types of faults and threats (such as cyber-attacks) that need to be adequately addressed. Self-healing has been coined as a term describing increased resilience in smart grid environments and the corresponding system capabilities in particular.

Self-healing capabilities refer to resilience against all kinds of faults, including faults in both the communication system and in the electrical power grid. Self-healing capabilities are thus an important prerequisite to achieving a reliable smart grid, enabling secure system operation through protecting against propagation of faults within and between the underlying systems, and contributing to speedy post-disturbance system recovery. SDN applications as part of the self-healing concept architecture enable important self-healing capabilities especially on the communication layer, but nevertheless also significant impact on the electrical power grid. A brief overview of this approach is presented in the following chapters, identifying the importance of SDN for dealing with grid faults and minimizing their impact on the security of electricity supply.

2.1.1 Safe systems

The SDN technology offers significant benefits regarding the safety of the involved systems in terms of network automation and management, programmability, global visibility and interoperability. Several academic and industrial works have already identified the importance of SDN regarding safety and security purposes. In [1] the authors introduce a detailed study, where they present how SDN can enhance the safety of a smart grid environment. In particular, E. Leal et al. in [2] provide an SDN-based architecture for the monitoring and automation of a substation. It is composed of three layers: (a) infrastructure layer, (b) virtualisation layer and (c) functionality layer. The infrastructure layer and the virtualisation layer are responsible for handling the physical and virtual resources, respectively. On the other side, the functionality layer encloses the various operations. Furthermore, the proposed architecture consists of four modules, namely (a) S3N-PROTECT, (b) S3N-MANAGE, (c) S3N-MEASURE and (d) S3N-CONNECT. S3N-PROTECT protects and control the operational characteristics of the substation. The second module and the third module offer management and measurement operations, respectively. Finally, S3N-Connect focuses on the connectivity of the substation assets. In a similar work in [3], S. Cahn et al. propose an SDN solution for auto-configuring a substation environment. The proposed architecture focuses on the future characteristics of the smart substations and relies on the Ryu controller. The authors give special attention to how to isolate the relevant network traffic via Ryu without forming several VLANs. In order to evaluate their implementation, the Mininet simulator is utilised.

Moreover, SDN participates actively in utility Machine-to-Machine (M2M) applications. In focusing on the smart grid environment, various smart grid devices, such as smart meters, Phasor Measurement Units (PMUs) and Intelligent Electronic Devices (IEDs) can communicate directly with each other as well as with the Utility Data and Control Centre (UDCC). Through SDN, the appropriate data and

communications can be controlled and isolated. More specifically, taking full advantage of the Field Programmable Gate Array (FPGA) advances, software-defined meters are available now in the market [4]. These meters constitute significant building blocks for Smart Utility Networks (SUNs). A characteristic case of SDN-enabled M2M is provided by [5]. In particular, in [5], the authors investigate an SDN-enabled Electric Vehicle Energy management (EVEM) consisting of 100 Electrical Vehicles (EVs), one gas generator and four wind turbines. The SDN controller undertakes to keep the information related to the functional characteristics of EVs, such as charging time, location, charging status and battery. In this case, the main goal of SDN is to optimise the mobility management of EVs as well as to assist in performing resource allocation. Supposing that the number of EVs increases and simultaneously the probability of a collision increases, the SDN controller can activate the resource allocation block based on the particular requirement, thus reducing the number of the competing EVs. Finally, a remarkable paper is also provided in [6] where the authors combine SDN and cloud computing in order to virtualise functional SG hardware devices.

SDN also benefits the entire security status of an infrastructure. In [7], P. Manso et al. provide an SDN-based intrusion detection system which can mitigate timely potential Denial of Service (DoS) and Distributed DoS (DDoS) attacks that are usual threats against SG. The proposed intrusion detection system relies on Ryu and Suricata. In particular, Suricata is responsible for detecting indications related to DoS and DDoS, while Ryu is adopted in order to mitigate these indications. The authors utilised Mininet in order to evaluate their implementation. Based on the experimental results, it seems that their intrusion detection system can mitigate timely DoS and DDoS activities. On the other side, in [8], the authors focus their attention on mitigating Distributed Network Protocol 3 (DNP3) cyberattacks and anomalies again based on Ryu. More detailed, an autoencoder called DIDEROT autoencoder was developed in order to recognise six DNP3 cyberattacks and anomalies, namely (a) masquerading, (b) replay, (c) DNP3 reconnaissance, (d) flooding, (f) injection and (e) DNP3 anomalies. Next, based on the outcome of the DIDEROT autoencoder, the Ryu controller transmits the necessary OpenFlow commands in the SDN switches in order to isolate the malicious DNP3 flows.

2.1.2 Fault-tolerant systems

The traditional Software Defined Network (SDN) architecture is usually based on single controller that is placed in the control plane. Therefore, network functioning become highly dependent on the performance of the single controller in the Control Plane, which is undesirable for any reliable application. According to many opinions, and despite many advantages of SDN, its deployment in the practical field is restricted since reliability and fault-tolerance capabilities of the system are not satisfactory [9]. It is possible to overcome these difficulties and there are many proposed architectures that are consisting of one or multiple controllers. Commercial SDN controller solutions incorporate fault tolerance, but there has been little discussion in the SDN community on the design of such systems and the trade-offs involved [10]. Fault tolerant SDN should have implemented fault-tolerance mechanisms in order to (for example) periodically update the controller's state so that in case of fault the SDN has ability to select another controller that is in operational state. Fault-tolerance mechanisms are required to ensure high availability and high reliability in systems. SDN, as a newly developed concept, has presented new challenges and provided opportunity to develop new strategies, architectures, and standards to support fault-tolerance.

Several fault-tolerance techniques are being used to avoid service failure in the presence of faults [11]. Fault-tolerance is carried out through error detection and system recovery, or simply detection and recovery mechanisms. Error detection identifies the presence of an error, while "recovery transforms a system state that contains one or more errors and (possibly) faults into a state without detected errors and faults that can be activated again " [11]. Recovery techniques can be further classified into two main categories [11]: i) recovery with error handling; which eliminates errors from the system state; and ii) recovery with fault-handling; which prevents faults from being activated again. The choice of error detection and recovery techniques are being adopted based upon the underlying fault assumption.

SDN architecture is usually comprised through several abstraction layers. The three most important layers are Data Plane, Control Plane and Application plane. In the following text will be presented the overview of fault-tolerance support of all three mentioned planes.

SDN DATA PLANE. SDN data plane fault-tolerance is related to the issues already present in traditional architectures (e.g. Multiprotocol Label Switching technology) [12]. Due to the static nature of traditional networks, these approaches can achieve good performance upon link and node failures. However, failure detection and recovery approaches in dynamic networks such as SDN must be re-designed to adapt to the dynamics of the rapidly changing networks. Traditionally, reactive and proactive approaches were used to provide Fault-tolerance [13]. In the reactive approach, an alternative path is calculated after the fault becomes active. In proactive techniques, the resources and backup paths are pre-programmed before the occurrence of a fault (when a fault is dormant). If the fault becomes active, the pre-programmed logic starts to defend immediately and recover the system from faults [13].

SDN CONTROL PLANE. Control plane fault tolerance represents very important requirement for normal operation in networks: the controller is vital and with that said the controller must be able to process all required traffic commands in all situations. There are few approaches to implement SDN control plane fault-tolerance. The first approach is to replicate a controller on a different control network. If failure occur, the replicated controller takes over the traffic. In another approach, the controller must be embedded with mechanisms (build-in module) to self-heal from targeted attacks such as Denial of Service (DoS), flooding and fake traffic routing and other network- related targeted attacks. However, the control plane time to recover from such attacks is critical, and ideally, recovery mechanisms must be developed to mitigate failures within the set network requirements. In addition to these, the recovery process must be efficient and must be able to selfheal during a failure event with minimum overhead. In-band and out-of-band signalling solutions have been adopted to offer SDN control plane reliability [14]. In practice, most SDN deployments use out-of-band control, where control packets are managed by a dedicated management network [14]. In SDN architectures, the controller is a logically centralized entity. It is responsible for translating the SDN applications requirement, via a Northbound interfaces, down to the SDN data layer [12]. Furthermore, it is also responsible for providing SDN applications an abstracted view of the network (including statistics and events). Currently, OpenFlow is a default standard for the communication in southbound APIs. It is important to mention that OpenFlow also has fault-tolerance capabilities. In SDN networks, operations rely on the proper functioning of the controller. The control plane in SDN manages the control logic of switches. The control logic is critical in SDN based networks. This problem is minimized in the latest version of the OpenFlow protocol by a master-slave configuration at the control layer: to increase resiliency.

SDN APPLICATION PLANE. The application layer allows business applications to modify and influence the way the network behaves in order to provide services to customers. This requires the definition of an API, to allow third-party developers to build and sell network applications to the network operator. The development of such an API has not yet properly addressed by the Open Network Foundation (ONF) but is required in order to guarantee interoperability between a business application and network controllers from different suppliers [12]. Existing SDN programming languages offer several features such as flow installation, policy definition, programming paradigm and abstraction for developing and enabling network and application fault-tolerance in SDN.

2.1.3 Resilient Systems

In recent years, the notion of resilient operation has become prominent in the context of self-healing electrical power and energy systems (EPES). The profound impact of challenging events in such systems has raised the requirement of increasing the network resilience and providing rapid-response restoration capabilities in critical situations. Resilience refers to the self-healing aspect of power systems that allows their prompt reconfiguration in case of emergency in order to endure critical events and manage disturbances in an efficient manner, aiming to minimize the supply interruption and maintain active services. With the ever-growing increase of distribution networks in capacity and complexity, there is a growing need to include smart self-healing features that guarantee the system's resilience and enhance the grid's stability when faults occur.

Introducing SDN-enabled architectures provides enhanced functionalities with regard to the reliability of the involved system, as showcased in a number of related studies in literature [15], [16]. By dividing the architecture into application, data, control, and management planes, the decision-making logic is transferred to an SDN controller, which communicates with the switches through OpenFlow commands. As a result, the introduced SDN features facilitate the process of monitoring the network infrastructure and analysing the traffic, enabling the deployment of procedures that greatly support the system resilience. In addition, SDN can suggest reconfigurations that complement the current resilience strategy, or in some cases propose entirely new sophisticated strategies, enabling the adoption of more adequate self-healing solutions [17].

In order to effectively address the issue of resiliency, the concept of self-sufficient microgrids is utilised [18]. In this context, microgrids are small portions of the power grid that include distributed generators and load centres and are able to operate both connected to the wider utility power grid and in islanded mode, operating autonomously as independent components of the EPES. Utilising microgrids allows the decentralized control of distributed energy resources (DERs), leading to the improvement of power delivery and increasing the stability in unreliable distribution systems, while facilitating energy conservation and resulting in reduced operational costs [19]. Efficient control of microgrids in islanded mode provides flexibility to the EPES, by allowing the local management of the DERs, thus offering an appealing approach for providing unhindered operations and establishing resilient self-healing power systems [20].

The required functionality that offers these capabilities is often performed under an emergency scenario, increasing the problem complexity due to existing operational and time constraints. In addition, the solution has to incorporate mechanisms that minimise the imbalance between load and generation [21]. As a result, addressing this problem efficiently is an arduous task. Several techniques to achieve the desired resilient features in power systems have been proposed in literature. Common

approaches involve multi-agent systems, mixed-integer linear programming (MILP), fuzzy logic and heuristic search. The most prominent methods are described in the next subsection.

The essential reconfigurations towards self-healing usually include breaker manipulation, generation start-up or shutdown, and load shedding or pickup. These actions aim to alter the system's operational status and improve the overall condition following a fault occurrence. In general, to achieve robust and resilient self-healing, an EPES should incorporate a number of crucial characteristics: the capacity to comprehend emergency situations, the ability to respond rapidly to critical events, and a control procedure that restores the functionality and leads the system to stable operation. The key performance indicators used to evaluate self-healing in power systems typically involve the speed at which a safe configuration is achieved and the number of components that remain active during the self-healing process.

2.2 Self-healing methods

Electrical power systems have always been designed with fault tolerance and resilience in mind, hence the self-healing concept is not a new concept for power system architects and operators. When discussing self-healing approaches, it is useful to distinguish between general self-healing methods applied in traditional power systems, and new self-healing possibilities that become both possible and necessary with the power system transformation towards a renewables-based smart grid. Both categories of self-healing approaches and methods are discussed in this chapter.

The smart grid is the background to which SDN applications enable an entirely new facet of the self-healing concept. Only a brief overview of this approach is presented in this chapter since discussing this new aspect in detail is in fact the focus of this document (and indeed of the entire project).

2.2.1 General/Traditional Self-healing Methods in Power Systems

Traditional self-healing methods are well-established and have been used for many years to ensure the reliability of electrical power systems. A selection of these methods is presented in the sections below. These approaches deal with maintaining the power balance (equivalent to maintaining the frequency) or with maintaining the voltage by providing reactive power where and when necessary. These aspects are a required component of the overall self-healing concept since disturbances in power systems can happen at any time and for many different reasons.

2.2.1.1 *Reserves*

When the active power in the grid becomes imbalanced, the activation of reserves is an automatic process that restores the balance and thereby ensures frequency stability. There are different types of reserves, in particular Frequency Containment Reserves (FCR) and Frequency Restoration Reserves (FRR), which are distinguished according to the time and duration of their activation as illustrated in Figure 1.

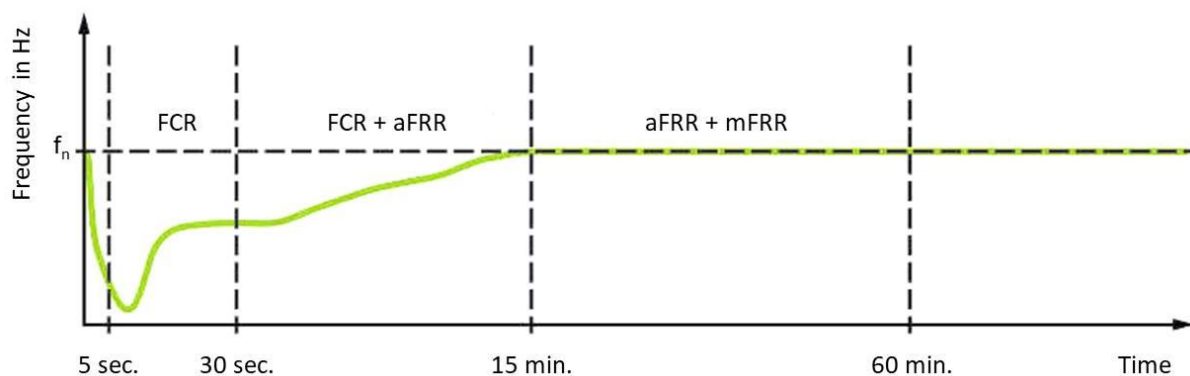


Figure 1: Types of reserves by the time and duration of activation (Source: ENTSO-E)

Pursuant to [22] FCR have to be fully activated within 30 seconds after the disturbance to stabilize the frequency. This is done by droop controllers of generators in the entire synchronous zone. After maximum 15 minutes, the automatic Frequency Restoration Reserves (aFRR) in the area of the TSO where the imbalance has occurred are fully activated to lead the frequency back to its nominal value. The aFRR are replaced by the manual Frequency Restoration Reserves (mFRR) within 60 minutes after the disturbance has occurred, in order to make the aFRR available again while keeping the frequency at its set point. As the name suggests, mFRR are typically ordered manually by phone or through a web portal, but in Germany this is now also retrieved automatically from a Merit-Order-List-Server [23].

The reserves described above are traded on dedicated energy markets, and their allocation needs to be planned and controlled so that there is sufficient capacity available to be activated when needed. This process of self-healing in terms of power balancing is only used in the transmission grid, since distribution grids and the grid users connected there are not responsible for real-time balancing.

2.2.1.2 Automatic Generation Control (AGC)

Automatic Generation Control (AGC) is a mechanism used to adjust the power output of interconnected generators in response to changes in the load demand, or more generally to changes in the power balance in a balancing area. Therefore, it can be defined as an automatic control that matches the generated output with the load demand. A control in the generated active power (P) is able to regulate the frequency (f) and adjust it to the demand. On the other side, a control in the generated reactive power (Q) is able to regulate the voltage (V) accordingly.

AGC has traditionally relied on hydro-powered generators mainly because of the following two reasons: these units have the capability of responding quickly to changes in power reference set points, and they do not have restrictive ramp limitations, other than the ones imposed by the maximum generation capacity. Other units that are suitable for this control are turbine-controlled thermal units, although the ramp limitations in these units are more restrictive.

AGC is not usually an ancillary service itself. It refers to the automatic mechanisms through which resources are activated in the economic dispatch and aFRR processes. Besides the response to frequency deviations, these mechanisms also address minimizing the area control error (ACE) of balancing areas within large interconnected systems. [24]

2.2.1.3 Low Frequency Demand Disconnection

Low Frequency Demand Disconnection (“Load Shedding”) is the technique of temporarily reducing the supply of electricity to an area to avoid overloading the generators. In line with the previously described self-healing techniques, it is used when there is a frequency drop in the power system due to an imbalance of the generated power and the load demand.

Many systems can be protected from frequency collapse by importing large blocks of power from neighbouring systems to make up for the lost generation. However, in an islanded system, or in an interconnected system with a shortage of tie-line capacity, this might not be possible and the only way to prevent a frequency collapse following a large disturbance can be to employ automatic load shedding. Automatic load shedding is implemented using under frequency relays. Each relay is configured to trigger at a specific low frequency threshold at which it opens the breaker without delay. As an ensemble, these relays detect the onset of decay in the system frequency and shed appropriate amounts of system load until the generation and load are once again in balance and the power system can return to its normal operating frequency. Load shedding relays are normally installed in distribution and subtransmission substations, as it is from here that the feeder loads can be controlled.

Load shedding is an emergency measure to prevent blackout in case of large disturbances in the power balance. As such it is only used as a last resort, when all other methods to restore the balance have been exhausted, including reserves, adjusting the output of any electricity storage units, and loads contracted to provide flexibility during contingencies. Therefore the frequency thresholds configured at the relays are all well outside the normal frequency operation ranges. In order to disconnect only as much demand as necessary, system operators in continental Europe split the demand to be disconnected into at least 6 groups of similar size that get disconnected at decreasing frequency thresholds. European transmission system operators publish their high-level load shedding schemes as part of their system defence plans according to the Commission Regulation (EU) 2017/2196. [25]

Load shedding used to be implemented by disconnecting entire distribution systems. This concept is being replaced to account for the fact that distribution systems (and even feeders) no longer represent pure load, but with connected DER can also feed power back into higher voltage levels. A more fine-grained approach is now implemented that takes into account the current state of each distribution system by blocking the relay in case of reverse power flow, and by moving the relays from the HV/MV transformers to the MV feeders. [26] [27]

Not all load will be shed during load shedding as it aims to disconnect only the minimum amount required; also system operators usually take care that critical loads remain with the demand segment for which automatic load shedding is never activated. In particular in small-scale setups with different reliability requirements (such as microgrids), a related approach can be to assign different priority levels to individual loads, and integrate load disconnection into the power balancing concept even during “normal” island operation.

2.2.1.4 Automatic Voltage Control (AVC)

Controlling voltages on the power system allows for the efficient transmission of power whilst respecting equipment limitations. One of the most efficient ways to control voltages on the power system is to place generators in voltage control mode. This requires generating equipment to measure a voltage, compare the measurement to a reference and increase/decrease the reactive power flow

out of the generating equipment. This can be accomplished by increasing/decreasing the excitation in the field winding for a synchronous machine or changing the firing angle of inverter based equipment.

Automatic Voltage Regulation (AVR) is a device used in generators with the purpose of automatically regulating voltage, which means that it will turn fluctuating voltage levels into constant voltage levels. AVRs work by stabilizing the output voltage of generators at variable loads, but can also divide the reactive load between generators that are running in parallel (voltage droop), and helps the generators respond to voltage drops caused by short circuits in the grid [28] [29].

Beyond the asset-level control of the AVR, automatic voltage control (AVC) includes an automated coordination of reactive power supply from multiple generators in the system, which allows an optimized level of voltage in the system – and distribution of reactive power contributions – to help minimize losses and ensure a sufficient margin against voltage instability and consequential voltage collapse. This contributes to the self-healing capabilities of the system by making it more robust against disturbances that affect the reactive power balance, such as outages of transmission lines or some power plants.

2.2.2 Self-healing methods in smart grids

According to [30], the definition of smart grid given by the European Commission is as follows:

A Smart Grid is an electricity network that can cost efficiently integrate the behaviour and actions of all users connected to it – generators, consumers and those that do both – in order to ensure economically efficient, sustainable power system with low losses and high levels of quality and security of supply and safety.

This description indicates that a smart grid needs to be highly observable. Also, in order to facilitate efficient integration of the various users, bidirectional communication between relevant system actors needs to be enabled. Especially at the distribution network level, this is not given in the traditional grid. Smart grids implement more possibilities for the automation of processes and accordingly a greater potential for self-healing processes.

Self-healing methods in smart grids are presented in the following subsections.

2.2.2.1 Variable Renewable Energy (VRE) and storage control

The transition from conventional thermal generation technology towards renewable energy sources is implemented in many countries through increased capacities of wind and solar power generation, together referred to as variable renewable energy (VRE). All modern VRE generators are based on power electronic converter technology. Small-scale generator systems connected to the distribution systems represent a significant share of installed VRE capacity.

Due to the primary energy being available at no cost, it is desirable in principle to have VRE generators feed their power into the grid whenever it is available, thereby displacing conventional thermal generation and reducing fossil fuel consumption and carbon dioxide emissions. However, this priority dispatch approach comes into conflict with the power system's needs for power balancing when the installed VRE capacity reaches significant shares. This problem is addressed through two different but

complementary approaches: Firstly, VRE generators are equipped with smart inverter technology that not only responds intelligently to variations of voltage and frequency, but also implements remote control interfaces to enable real-time monitoring as well as remote operator access to setpoints, operational constraints, and control modes and configurable characteristics. [31] Smart inverters therefore provide additional flexibility to system operators. Secondly, VRE generators can be combined with storage facilities such as battery storage. This option not only allows storing excess power instead of just curtailing it, but of course also allows providing the stored power when there is not sufficient primary renewable power available. In addition, battery storage is also inverter-based and can provide extremely fast and intelligent response to voltage and frequency and to remote control signals. [32] [33] [34] Flexibility from both VRE and storage has been increasingly used by system operators in the past decade already, and will be used more in the future.

Due to VRE generation and storage often being connected to medium voltage and low voltage grids, these technologies are introducing new flexibility and controllability capabilities to distribution systems. This new flexibility is not a self-healing method in itself, but it is an enabler for improved existing self-healing capabilities such as contingency management and for new self-healing methods such as intentional islanding capabilities of distribution grid sections (microgrids). Efficient use of these capabilities requires suitable communication infrastructure and control structures, hence it relies on Smart Grid infrastructure to provide its flexibility benefits.

2.2.2.2 Demand response and demand side management (DR/DSM)

Since implementing and using flexibility from VRE and storage can be quite expensive, it is a good idea in general to identify demand-side flexibility and make it accessible to system operators. This approach is not limited to load shedding in emergency situations (reducing reliability of supply for affected grid users); instead, the idea is to differentiate between required energy and the time when it has to be supplied. Shifting required demand in time has the potential of significantly improving the power balancing task involved in frequency control and contingency management.

Common approaches to accessing demand flexibility are referred to under the terms of demand response (DR) and/or demand side management (DSM). DR usually means giving incentives to consumers, but not necessarily controlling their response directly. This can be implemented through time-variable power tariffs or even real-time prices, or offering other benefits coupled to a response to some signal indicating the needs of the power system. In contrast, DSM is less clearly defined in that multiple competing definitions are used in the literature. According to one of these definitions, it means direct user access to defined flexibility of controllable loads, providing a specific and reliable response to the DSM user (who can be a flexibility aggregator, or a system operator) as a contracted service. [35] Another definition sees DSM as encompassing all sorts of demand modification including both DR and energy efficiency measures. [36]

DR and DSM are already used in some countries today, although still to a limited extent (typically with suitable industrial consumers). [37]

Similar to VRE and storage control, new flexibility from demand control is not a self-healing method in itself, but enables improved contingency management and can support intentional islanding capabilities for microgrids. This method also relies on Smart Grid infrastructure.

2.2.2.3 Controlled islanding (microgrids)

With conventional distribution system design and operation, there is only one feeding point, and the electrical power flows through the system from the upstream substation to the downstream customers [38]. When a fault occurs and is isolated, all customers downstream of the fault location are disconnected.

Microgrids are an approach to minimize the unavoidable disconnection by making it possible for sections of the distribution systems to operate in island mode (i.e., without any connection to the upstream grid). Microgrids rely on the availability of local generation resources with sufficient capacity to supply the load that otherwise would get disconnected. A microgrid can act as a reliable source to supply customers in the area downstream of a fault, and help the system recover properly and quickly. By forming island networks intentionally and systematically, the reliability and continuity of supply can be increased for the loads in the islanded areas.

Once the disturbance has been eliminated, the microgrid is usually reconnected to the main grid. For a coordinated reconnection, without intermediate interruption of supply, it is important to measure the status of the main power grid and of the microgrid during the whole process, and implement a resynchronization procedure.

The concept of controlled islanding in response to isolation of faults is illustrated in Figure 2 [39].

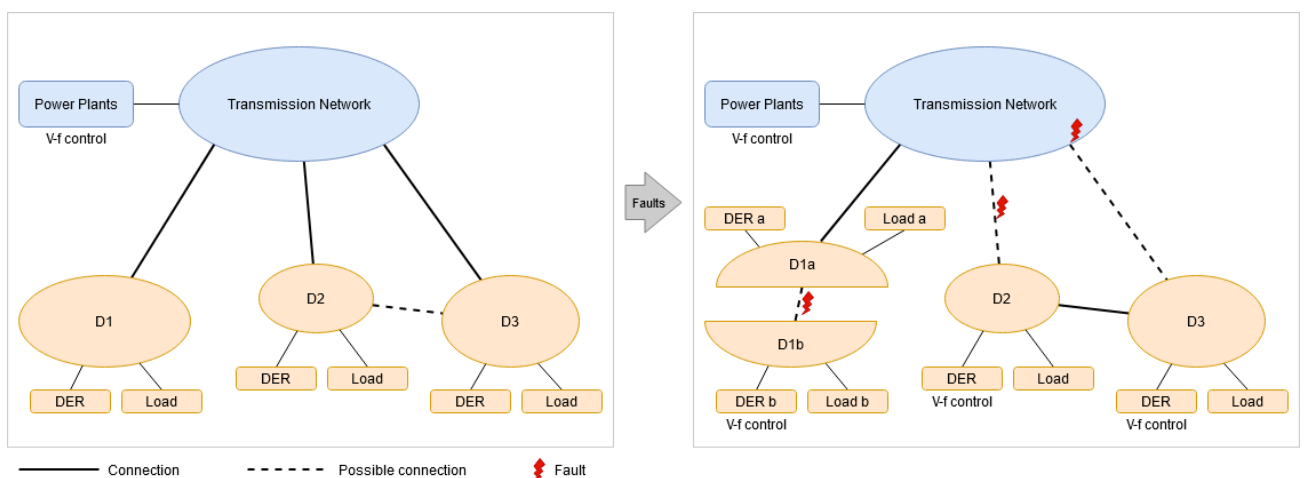


Figure 2: Example of controlled islanding in response to faults [39]

The example shows the connections between one transmission grid and three distribution grids (D) before and after three different faults occur [39]. Distributed Energy Resources (DER)¹ and loads are connected to each distribution grid. Distribution grid D1 is split into two parts in response to an internal fault. Part D1a remains coupled to the transmission grid and thus remains in grid-connected operation, while part D1b operates as a microgrid in island mode and controls the local DERs to maintain stability of voltage and frequency in the islanded grid section. Two additional faults occur, in the interconnection between the transmission grid and D3 and in the transmission grid itself close to the

¹ The following technologies are referred to under the term DER: Distributed generation (e.g., rooftop solar photovoltaics), flexible load (e.g. electric vehicles), and energy storage.

interconnection to D2. To prevent an outage of D2 and D3, both distribution grids are transferred into island operation after the faults and use the local DERs for voltage and frequency control. D2 and D3 are then connected to form a single island in order to improve the balancing of supply and demand. Microgrids with black-start-capable DER may be able to restore themselves if a blackout occurs.

The SDN-microSENSE includes intentionally forming island networks in its holistic approach to improving self-healing capabilities and resiliency. The decisions on where the boundaries of the islands should be are taken care of by the IIM module developed in task 4.3; the processes related to balancing the power within the islands and restoring consumer power supply are addressed by the EMO tool, which is developed in task 4.4.

2.2.2.4 Optimized grid reconfiguration

Most distribution grids are operated in a radial configuration, although there are usually tie lines and other connections that would allow meshed or ring configurations, since radial configurations are simpler and easier to operate [40]. The radial feeders are configured through appropriate breaker settings at the interconnection points with other feeders. Feeder configurations for normal operation are typically selected for minimizing the losses and reducing the loading of grid equipment, thereby increasing the economic efficiency of the distribution system.

This distribution grid design also allows reconfiguring the feeders after faults have forced the disconnection of grid sections. If only the faulty grid segment (e.g., a line or a transformer) is isolated, then it is usually possible to restore electricity supply to the downstream grid users by reconfiguring the feeders through adjusting one or more circuit breakers or switches.

In today's medium voltage and low voltage distribution grids this optimized grid reconfiguration is usually carried out through human intervention. Fully automating this process can be useful not only to further minimize losses in the presence of variable generation in the distribution grid, but also to minimize power supply interruption times after faults, thereby implementing improved self-healing capabilities in a smart distribution grid.

2.2.2.5 SDN-Based Self-Healing Communication Network

Any self-healing capability relies on measurements and signals that enable a state assessment of the grid and automatic detection of any faults. For this purpose, digital and real-time devices are used to measure and communicate the status at critical locations in the network at any time.

One of these devices is the Phasor Measurement Unit (PMU), which provides synchronized measurements that accurately capture the real-time wide-range dynamics. The microsecond accuracy of time synchronization required for this is facilitated via the Global Positioning System (GPS). The interoperability of PMUs from different manufacturers is ensured through IEEE Standard C37.118.2-2011. By placing PMUs in strategic locations, a wide-area visibility and near real-time observability of dynamic phenomena in the power system is created. This allows grid operators to better identify and diagnose upcoming grid problems and improves the implementation and evaluation of corrective actions for maintaining system stability.

The measured data from several PMUs is delivered to a phasor data concentrator (PDC) [41]. This device bundles the measurements and transfers them to the next-level PDC or a control centre. Since PDCs provide the communication link between the PMUs and the control centre, they are a predestined target for cyber-attacks. The physical and logical connection structure between PMUs,

PDCs and the control centre is illustrated in Figure 3. A switching network situated between PMUs, PDCs, and the control centre, enables the physical connection between these components to be changed, and PMUs to be assigned to other PDCs if a PDC is out of service due to a cyber-attack or failure. The entire system that integrates PMU and PDC data into the control centre processes is called Wide-Area Monitoring System (WAMS).

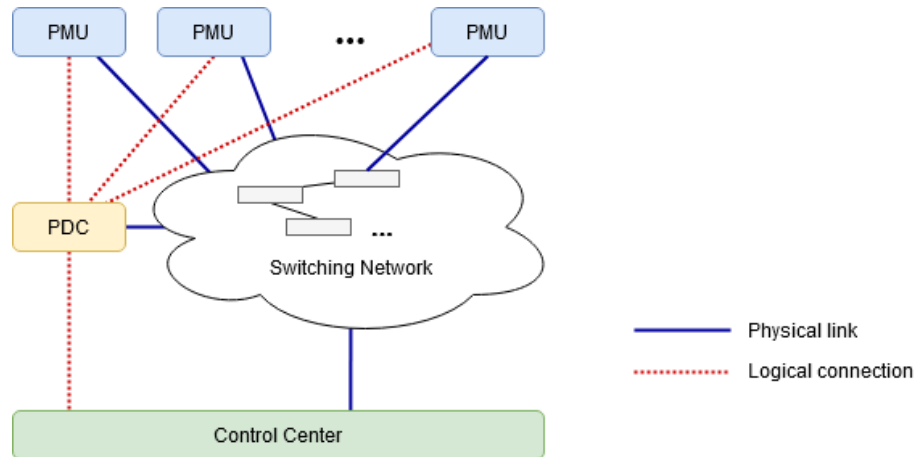


Figure 3: WAMS structure [42]

Together with the Wide-Area Protection System (WAPS) and the Wide-Area Control System (WACS), the Wide-Area Monitoring System (WAMS) builds the Wide Area Monitoring, Protection and Control System (WAMPAC) [38]. The WAMPAC combines the following capabilities:

- Dynamic measurement and depiction of events
- Wide-area system view
- Coordinated and optimized stabilizing actions
- Adaptive relaying in coordination with local protection devices
- Treatment of cascaded failures.

Thus WAMPACs provide improved possibilities for self-healing compared to conventional Supervisory Control and Data Acquisition Systems (SCADA).

Not only the data collection hierarchy from the control centre through the PDCs to the PMUs, and the control hierarchy from the control centre through the station controllers down to the asset controls, but also the underlying communication infrastructure with routers and switches is usually set up to operate in a tree-like structure where the path of the information flow is preconfigured. However, similar to power distribution grids, there exist alternative connection paths that can be used if needed.

In SDN-enabled communication networks, the reconfiguration of information flows can be changed dynamically based on the needs of the system; e.g., to optimize data transmission latency or bandwidth. This also allows isolation of faulty elements in the communication network, for instance a malfunctioning switch, or a PDC that has been detected to be compromised in a cyber-attack. Disconnecting one or more PDCs or other critical elements in the data flow path results in constrained observability and/or controllability of the power system. SDN controllers managing this communication infrastructure therefore need to aim for two main goals [41]:

- Stage 1: Restoring power system observability and controllability by reconfiguring the communication network as quickly as possible.
- Stage 2: Maximizing the system observability and controllability by recovering as many communication endpoints as possible (e.g., all disconnected PMUs).

By being able to quickly and efficiently contribute to restoring power system observability and controllability, SDN-enabled communication technology contributes to the self-healing of the energy ecosystem in the case of communication malfunction or new threats such as cyber-attacks.

2.2.3 Summary and Conclusions on Self-Healing Methods

The following table illustrates the discussed self-healing methods and their applicability depending on the grid level (transmission or distribution). Essentially only transmission grids have been equipped with a high degree of self-healing capabilities in the past; today, modern smart grid technologies enable new self-healing capabilities both in the transmission and distribution system.

Table 1: Self-healing methods and their applicability to electrical power distribution and transmission grids

	Method	Transmission grid	Distribution grid
Traditional Grids	Reserves	✓	
	AGC	✓	
	AVC	✓	
	Load shedding	✓	
Smart Grids	VRE & storage control	✓	✓
	DR/DSM	✓	✓
	Controlled islanding	✓	✓
	Optimized grid reconfiguration	✓	✓
	SDN-based data rerouting	✓	✓

3 Analysis of requirements for EDAE, SDN Controller and Northbound Interfaces

The purpose of this section is to conduct analysis on the requirements and specification that are provided by two deliverables D2.2 and D2.3. Deliverable D2.2 focuses on the requirements of the SDN-microSENSE project. These requirements are there in order to describe what a software system should do (functional requirements), as well as how the system should do it, including constraints and restriction in this regard (non-functional requirements). Second, The Deliverable 2.3 focuses on the overall architecture of the system, system behaviour, system specifications, technical constraints and interaction between components of the architecture (such as interfaces for example). Using these two documents, here is conducted analysis of all requirements and specifications which at the end represent set of rules and guidelines for the development and implementation of all relevant components related to this work package.

3.1 D2.2 Functional Requirements

In Deliverable D2.2 where all requirements are presented, it can be found that requirements are provided without the mapping according to the tools and software components. They represent the guidelines for the development of the tools. Here, in this section, the functional requirements are extracted and presented only for those tools and software components that are the subject of this deliverable. Some of the requirements provided by the D2.2 are covering the functionalities that are covered with the broad number of components or the entire SDN-microSENSE system. Here, the extraction and the summarisation of the requirements was done in such way that specified requirements (provided in tables bellow) are extracted from the D2.2. Depending on the tool the requirements were given through extraction of one or summarisation of more than one functional requirements from D2.2. As addition, the non-functional requirements are here presented also in summarised way and they were provided here to support the implementation of the functional requirements (as by definition of the non-functional requirements, which are commonly used to describe how the functional requirements are implemented).

General user requirements to be considered in the tools developed in task 4.2 are:

- Enhanced protection from attacks against the IT/OT and electrical infrastructure.
- Usable and easy-to-understand graphical user interfaces (GUI) that can be used by personnel who do not have advanced knowledge about cybersecurity.
- To include a tool that operates efficiently in terms of energy consumption and computational resources.

Table 2: Extracted and summarised requirements for each component

Extracted and summarised requirements	Related component	Reference to used D2.2 requirements
<p>The system shall be able to mitigate following cyberattacks in near real time (in seconds) and with high accuracy:</p> <ul style="list-style-type: none"> • DoS (Denial of Services) cyberattacks • MITM (Man in the Middle) cyberattacks • False Data Injection attacks • unauthorised access • Modbus TCP-related cyberattacks • DNP3 cyberattacks • EtherCAT cyberattacks • IEC 61850, IEC 60850-5-101, IEC 60850-5-102 and IEC 60850-5-104 cyberattacks • PROCOME cyberattacks • Ransomware 	EDAE, SDN-C	<p>Functional requirements:</p> <p>R-UR-03, FR-UR-04, FR-UR-05, FR-UR-06, FR-UR-07, FR-UR-08, FR-UR-09, FR-UR-10, FR-UR-11, FR-UR-12, FR-UR-13, FR-UR-14, FR-UR-15</p> <p>Non-Functional requirements: NFR-DPT-27, NFR-SEC-03, NFR-RL-01, NFR-RL-04, NFR-RL-06</p>
<p>The system shall be able to discriminate the various types of cyberattacks with high accuracy.</p>	AIDB	<p>Functional requirements: FR-UR-16</p> <p>Non-Functional requirements: NFR-RL-01, NFR-RL-04, NFR-RL-06</p>
<p>The system shall maintain:</p> <ul style="list-style-type: none"> • An inventory of all the infrastructure elements with their exact location • Status of the inventory (e.g. 'active' or 'not') • Explanation of the status of the inventory (whether the cause is known (e.g. due to maintenance work) or unexpected (e.g., due cyber-attack or failure)). • Notifications system or a polling system should be implemented in order to notify status changes for each single device. • All infrastructure elements shall be identified by a unique ID. This ID is to be shared with different Databases in the system • The processing of historical data shall be implemented (BIG data handling should be implemented on the system level) <p>The requirement applies to all types of devices, i.e. power, IT and OT network devices.</p> <p>The system shall be able to encrypt data when stored.</p>	AIDB	<p>Functional requirements: FR-UR-18</p> <p>General requirements: FR-GR-1</p> <p>Non-Functional requirements: NFR-DPT-09, NFR-DPT-10, NFR-DPT-11, NFR-DPT-15, NFR-DPT-16, NFR-DPT-27, NFR-SEC-03, NFR-SEC-15, NFR-RL-01, NFR-RL-04, NFR-RL-06, NFR-BCK-01, NFR-BCK-02, NFR-BCK-02, NFR-BCK-04</p>
<p>Graphical User interface should be implemented in order to present a dashboard that will enable users to configure the network. Depending on the</p>	SDN-Dashboard,	<p>Functional requirements: FR-UR-19</p>

tool, part of the User Interface shall also be the visual analytics tool and recommendations of the actions for the grid operator.	EDAE-Dashboard,	General requirements: FR-GR-7, FR-GR-11, Non-Functional requirements: NFR-DPT-15, NFR-DPT-16, NFR-DPT-20, NFR-DPT-27, NFR-SEC-03 , NFR-SEC-08, NFR-SEC-15, NFR-RL-01, NFR-RL-04, NFR-RL-06
The software component shall be able to support multiple user roles with distinct privileges.	SDN-Dashboard	Functional requirements: FR-UR-24 Non-Functional requirements: NFR-DPT-20, NFR-DPT-27, NFR-SEC-03, NFR-SEC-08, NFR-SEC-15 , NFR-RL-01, NFR-RL-04, NFR-RL-06
The system shall be able to provide network flow metrics from raw network traffic data.	SDN-C	General requirements: FR-GR-5 Non-Functional requirements: NFR-SDNSEC-01, NFR-SDNSEC-02, NFR-SDNSEC-03, NFR-SDNSEC-04, NFR-SDNSEC-05, NFR-SDNSEC-06, NFR-SDNSEC-07, NFR-SDNSEC-08, NFR-SDNSEC-09, NFR-SDNSEC-10, NFR-SDNSEC-11, NFR-SDNSEC-12, NFR-RL-01, NFR-RL-04, NFR-RL-06
The system shall provide the ability to the security administrator to view, create, read, update and delete network flows on the underlying switches.	SDN-Dashboard	General requirements: FR-GR-13 Non-Functional requirements: NFR-RL-01, NFR-RL-04, NFR-RL-06
The system shall propose countermeasures, including the drop or the redirection of a network flow, in order to tackle ongoing cyber threats.	SDN-Dashboard,	General requirements: FR-GR-14 Non-Functional requirements: NFR-RL-01, NFR-RL-04, NFR-RL-06
The system shall allow the operator to control any automation process.	EDAE-Dashboard	General requirements: FR-GR-23 Non-Functional requirements: NFR-RL-01, NFR-RL-04, NFR-RL-06

3.2 SDN-microSENSE platform specifications (related to the WP4)

The tables from deliverable D2.3 were used as the source for the specification provided in this deliverable. They were used to collect and summarise specifications which are suitable and more

related to the actions that are provided with the descriptions of WP4. Therefore, the following table summarise the specifications that are extracted from the D2.3.

Table 3: Specifications coverage of each component

Specification ID (source D2.3)	Action (Specification)	Related components
SPEC-F6	<p>SDN Control. To provide this functionality the SDN-microSENSE architecture is enabled with an SDN Controller component, which can receive specific instructions (from SDN applications) regarding how the different network flows should be managed in the EPES data network. This SDN Controller will act on the SDN-enabled network devices in the EPES infrastructure by programming the corresponding flow tables in their associated SDN switches. The following functionalities are provided:</p> <ul style="list-style-type: none"> • Editing the settings related to the data network flow control (this is the core functionality). • Provide visibility on the underlying network resources. This includes: <ul style="list-style-type: none"> o Available SDN-enabled network elements (e.g., SDN switches, network links and hosts connected to the SDN switches). o Retrieval of the underlying network topology (how the network elements are connected each other). o Access to network traffic statistics, network state and events. • Provide the necessary interfaces for the previous functionalities: <ul style="list-style-type: none"> o To the software applications requiring access to the controller functionalities (this is called the North-bound Interface; it is commonly implemented by means of REST API interfaces). o To the underlying infrastructure components (this is typically called the South-bound Interface; in SDN-microSENSE we will use the OpenFlow protocol for implementing this interface). <p>To the management components (e.g., the human operator's UI).</p>	SDN Controller and SDN-enabled infrastructure components
SPEC-F7	<p>Updated Assets Inventory. SDN-microSENSE integrates the specific Assets Inventory Database (AIDB) component. The database maintains an updated inventory of all the infrastructure elements with their exact location and their status (e.g., which devices are active or not, whether the cause is known (e.g. due to maintenance work) or unexpected (e.g., due cyber-attack or failure). This applies to all types of devices, i.e. power, IT and OT network devices. All infrastructure elements shall be identified by a unique ID, which would be shared by other components in the system. The database component provides the necessary interfaces for allowing other systems to update their respective information. The data network related information is obtained through the SDN Controller. The grid related information is obtained through a direct interface with the infrastructure components (when available) or even by manual entries (when</p>	AIDB

	automatic updates are not feasible for certain components). This AIDB is constantly updated, and is redundant to avoid being a single point of failure.	
SPEC-F8	<p>Data Gathering for Islanding & Optimisation. In SDN-microSENSE, the IIM component (which is part of the self-healing framework) is responsible for the gathering the data for implementing the islanding schemas on the grid. IIM is to receive the following data:</p> <ul style="list-style-type: none"> Information on the Security Incidents from Risk Level Assessment component, where all necessary information about location, involved asset and other parameters will be attached Grid model and field device information (from the Assets Inventory Database) <p>Operational data (from grid operator)</p>	AIDB
SPEC-F9	<p>Application of Islanding Schemas. Computing of optimal islanding schemes is performed by the IIM (Islanding and optimisation fraMework) within the self-healing framework. This IIM generates a set of isolation actions to ensure undisrupted operation and avoidance of failure in the grid. According to the computing results, IIM is to display recommended islanding schemes to a human operator (who can accept/reject the actual application of the suggested islanding schemas). This information is to be displayed directly in an IIM specific UI (which can be accessed from the SDN-microSENSE common UI) with all relevant data to the operator.</p>	IIM
SPEC-F10	<p>Network Reconfiguration based on the Electrical Data Analysis. The EDAE component (within the SDN-microSENSE self-healing framework) is to be responsible for sending the data about network reconfiguration directly to the SDN Controller. This action will utilize direct established interface between the EDAE component and the SDN Controller. The EDAE component will send information about the network reconfiguration parameters of the affected nodes in order to provide alternative network paths for the data flows.</p> <p>To do that, the EDAE component gathers and processes the following data:</p> <ul style="list-style-type: none"> Information on the Security Incidents from Risk Level Assessment component, where all necessary information about location, involved asset and other parameters will be attached. Network Topology, grid model and field device information (from Asset Inventory Database). <p>Operational data (from the grid infrastructure elements).</p>	EDAE, AIDB

Regarding the Operational specifications, they were extracted through the analysis of the non-functional requirements in Deliverable D2.2, collection has been made on a higher level of abstraction of the specification (due to maturity level and stage of the project when deliverable was made) and provided within Deliverable D2.3. They are provided in the tables in D2.3 and covering the following:

- TABLE 20. SPEC-OP1. RESILIENCE AND RELIABILITY
- TABLE 21. SPEC-OP2. DATA PROTECTION BY DESIGN.
- TABLE 22. SPEC-OP3. DATA SECURITY BY DESIGN.
- TABLE 23. SPEC-OP4. BACKUPS.
- TABLE 24. SPEC-OP5. USABILITY.

3.3 Technical constraints

There were defined three tables/groups with the technical constraints which actually are representing the high level technical constraints. The high level of abstraction was introduced during the preparation of the D2.3 because of the complexity of the system architecture and large number of components with very diverse technologies.

The deliverable presents the following three constraints that are to be used as the initial guidance:

- CONS-T1. SECURE APPLICATION PROGRAMMING INTERFACES
- CONS-T2. SECURE AUTHENTICATION
- CONS-T3. AUTHORIZING TRAFFIC (IP AND PORTS RESTRICTION).

Detailed description of the constraints could be found within the Deliverable D2.3. All Technical constraints are applicable to the all components of the SDN-microSENSE therefore the same are to be used and followed during development and implementation of the component within WP4.

3.4 Functional and non-functional requirements coverage

In this section, a detailed description of how each component will address the requirements defined in D2.2 and D2.3 will be described.

3.4.1 EDAE requirements

Functional Requirements:

The functional requirements that EDAE is responsible to address are presented in Table 4.

Table 4 Functional and Non-functional requirements for EDAE operation

Functional Requirements	Non-functional Requirements
FR-UR-03, FR-UR-04, FR-UR-05, FR-UR-06, FR-UR-07, FR-UR-08, FR-UR-09, FR-UR-10, FR-UR-11, FR-UR-12, FR-UR-13, FR-UR-14, FR-UR-15	NFR-DPT-27, NFR-SEC-03, NFR-RL-01, NFR-RL-04, NFR-RL-06

Each functional requirement of the aforementioned ones requires EDAE to mitigate a specific type of attack. The operation of EDAE allows to address all the functional requirements with the same mechanism, since EDAE obtains information for the security incidents to mitigate cyber-attacks. Specifically, the information of the security events alongside with the risk assessment of each event are provided to EDAE via S-RAF. Flowingly, EDAE consumes the information from S-RAF (network security status) and by taking into account the network constraints (bandwidth capacity, available

bandwidth), the network quality (delay, latency and jitter between the switches), the QoS needs of the applications and the sensitivity of the data that the applications handles, re-arranges the logical network connections. The decision about the re-arrangement of the network logical connections is performed using genetic algorithms and mixed integer linear programming (see section 6.2). The re-arrangement of the logical connections is performed in such way that the produced alternative communication paths are more secure for the applications that handle sensitive data and optimize the QoS needs of the applications. Given that an attacker has already infected a host, performing this re-arrangement makes it harder for an attacker to intrude to more sensitive hosts, since the attacker should pass from more secure links.

Non-functional requirements:

The non-functional requirements that EDAE is responsible to address are presented in Table 4 above.

NFR-DPT-27:

According to NFR-DPT-27 *“The system shall be able to utilize the right virtual machine backup strategy, since it ensures systems are delivered with the highest level of availability”*. EDAE operates in a Docker environment inside the virtual machine in order to ensure that its operation is resilient against changes that would occur in the virtual machine. As a second step, we are going to deploy EDAE Docker on a Kubernetes environment to further boost and ensure the resilience and the availability of EDAE.

NFR-SEC-03:

According to NFR-SEC-03 *“Connections among systems and equipment shall be secured by authentication and encryption”*. EDAE uses its interfaces to communicate with AIDB, EDAE Dashboard and SDN-Controller via RESTful APIs over the TLS protocol. The interfaces will use the Privacy Protection Framework which is placed inside the Management Plane of the SDN-microSENSE architecture in order to ensure secured connections among systems.

NFR-RL-01:

According to NFR-RL-01 *“The system shall be able to perform a required function under stated conditions for a specified time”*. The specified time should be defined for each action or operation of the system in order for the system to follow these time constraints of operation. EDAE operates continuously and monitors the network security and quality status in order to mitigate the cyber-attacks.

NFR-RL-04:

According to NFR-RL-04 *“The system shall implement mechanisms to ensure resilience against human errors and interactions within the system”*. There are two types of errors that could affect the EPES system and which are in the scope of EDAE's operation. The first is for an authorized user to change the table flows of the SDN switches which will results in suboptimal communication links. EDAE is resilient to this type of error since it frequently monitors the status of the network in order to restore it to its optimal configuration. Another type of error that could occur is a PDC to be disconnected, or blocked for whatever reason. The latest type of error can be mitigated by re-allocating the PMUs to leftover PDCs in order to maximize the observability of the EPES electrical grid.

NFR-RL-06:

According to NFR-RL-06 *“The system shall be designed for maintenance and automation. The system will need monitoring and regular updates in order to ensure proper operation over time”*. EDAE is designed in such a way that it consists of multiple logical components, each one independent of the development of the other. Each component performs a concrete function and is seen by the rest components as black box with certain input and output. This allows EDAE to be developed in a distributed way and to be easy for maintenance. Regarding the automation part, EDAE is fully automated and run on a Docker environment.

4 Architecture and detailed design

In this section we provide information about the design of the SDN-SELF and how it works as part of the overall SDN-microSENSE framework.

4.1 Architecture overview

This section presents how the components developed in this task fit in the general architecture of SDN-microSENSE, as presented in Figure 4 from deliverable D2.3 [43].

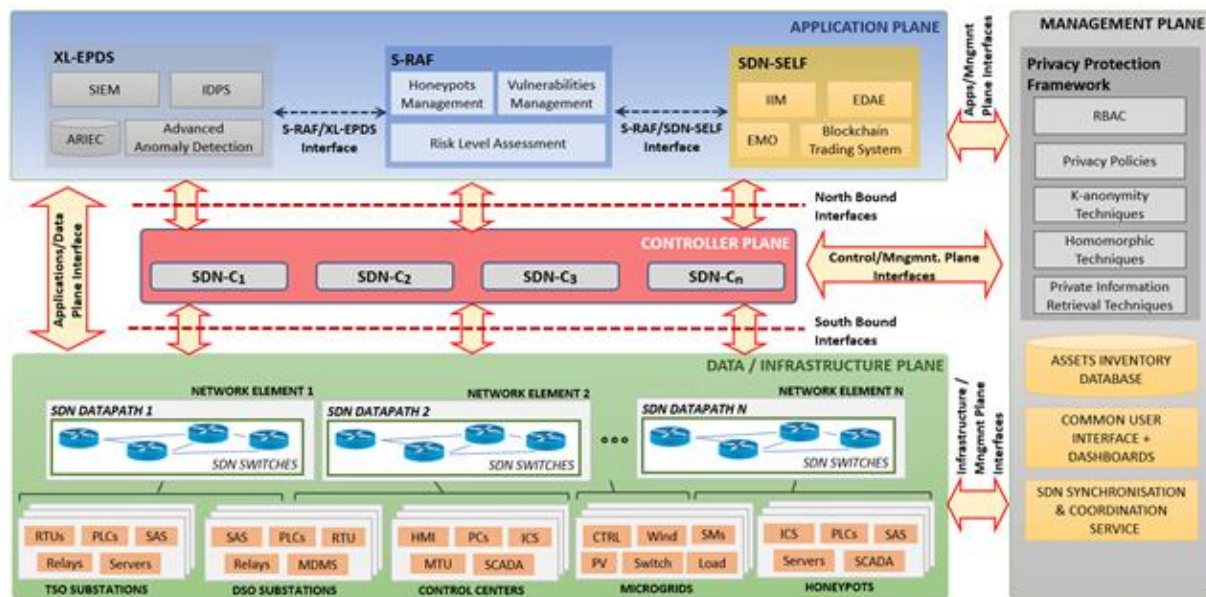


Figure 4: SDN-microSENSE Architecture Structural View

SDN-SELF is placed in the application plane and has been developed in WP4. It is interacting with the S-RAF that is responsible for assessing the level of risk in all the involved EPES devices and system.

Taking a closer look on the initial conceptual architecture, as presented in Figure 5, SDN-SELF includes the components: EDAE (developed in task 4.2), IIM (developed in task 4.3), EMO (developed in task 4.4) and the Blockchain Trading System (developed in task 4.5).

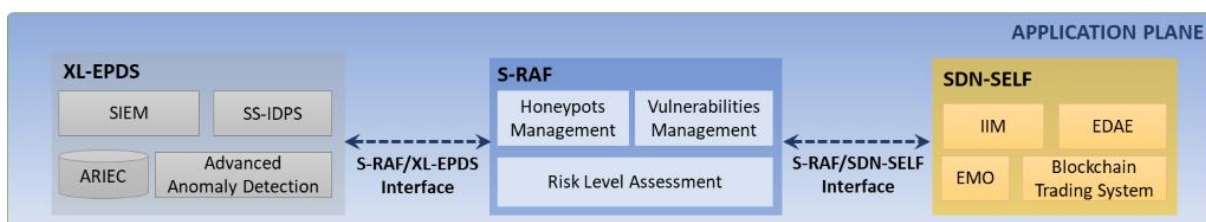


Figure 5: SDN-SELF in the application plane

In this deliverable it is detailed the interactions that affect the EDAE, the AIDB and the northbound interfaces of the SDN-C. The detailed interaction of the main components is presented in Figure 6.

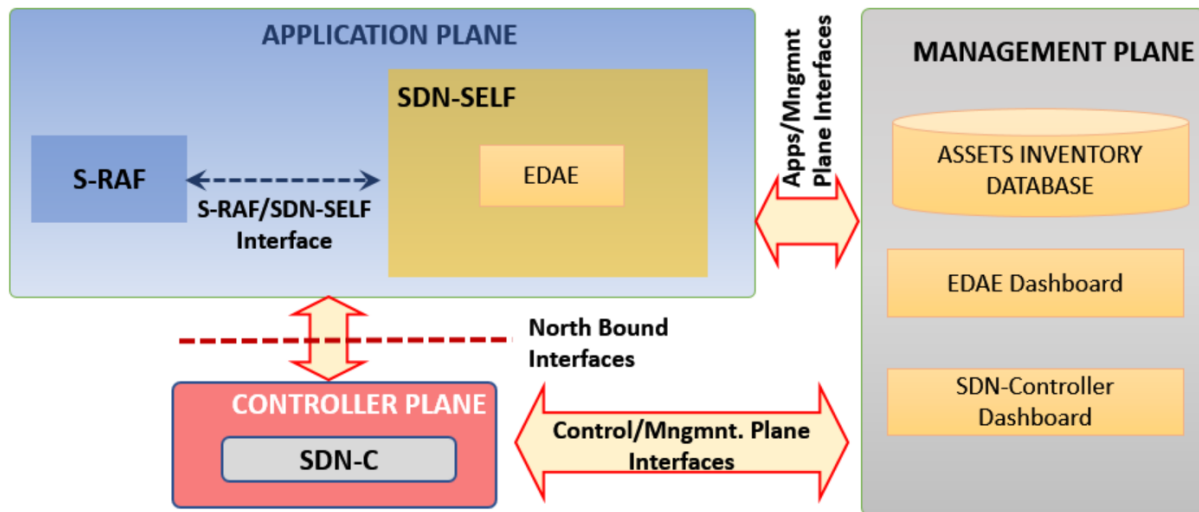


Figure 6: Detail of the architecture overview related to task 4.2

EDAE is an application located in the application plane and interfaces the SDN-C, located in the controller plane, through the northbound interfaces.

4.2 Components view

According the requirements on previous section two new components were needed and then developed in this task. These are the graphical user interface for EDAE and SDN-Controller (EDAE-Dashboard and SDN Dashboard). Summing up, for the proper operation of EDAE is needed the interaction of three other components and two user interfaces: S-RAF, SDN-C, SDN Dashboard, EDAE Dashboard and AIDB.

These components are described:

- In WP3, S-RAF.
- In Section 5, SDN-C and SDN Dashboard.
- In Section 6, EDAE and EDAE Dashboard.
- In Section 7, AIDB.

In Figure 7 it is displayed the interaction between components. The grey boxes are the interfaces defined in D2.3. Notice that the interface EDAE/EDAE-Dashboard was never defined before so this is the first time it appears in the component view. The blue boxes are the components described in this deliverable. In the orange boxes are described some of the functionalities that these interfaces can provide to the components. The arrows indicate the workflow of the information.

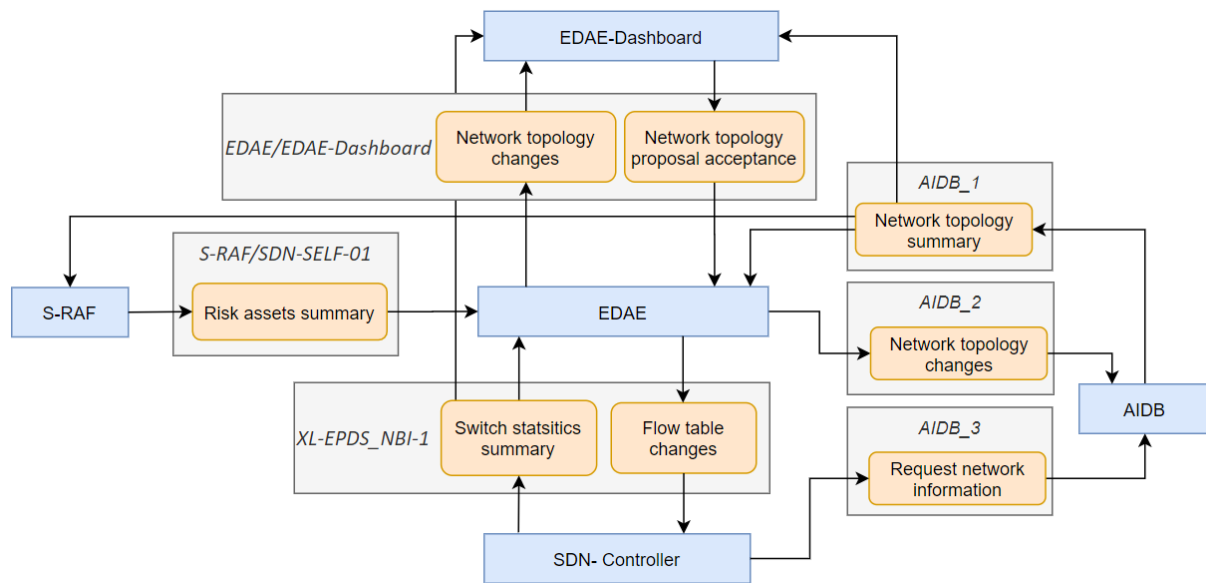


Figure 7: Interaction between components

EDAE will request the information from the AIDB to know the network topology and information of the assets (AIDB-1), it will need the information of the assets in risk through S-RAF (S-RAF/SDN-SELF-01) and all the relevant information from the SDN assets that are not present in the AIDB but its information can be retrieved from the SDN-C (XL-EPDS_NBI-1). With all those inputs, EDAE will propose a network topology change to EDAE-Dashboard in order to improve the network security, the observability of the EPES and improve the QoS between assets (EDAE/EDA-Dashboard). The topology change will be evaluated by an operator that will provide back a response to EDAE through the same interface. In case the topology change it is accepted, EDAE will propose some changes in the flow tables (XL-EPDS_NBI_1) to the SDN-C. EDAE will also update the AIDB with the new values of the asset's attributes that has changed (AIDB-2).

AIDB will request network information to the SDN-C often, since this information will be used for other components than EDAE (AIDB-3).

EDAE-Dashboard will retrieve information of the network topology form the AIDB too, and information of network switch statistics from the SDN-C (XL-EPDS_NBI-1).

4.2.1 Detailed information flow of the inputs and outputs

The exact inputs and outputs of the EDAE core and the data exchange with each interface are presented in Figure 8.

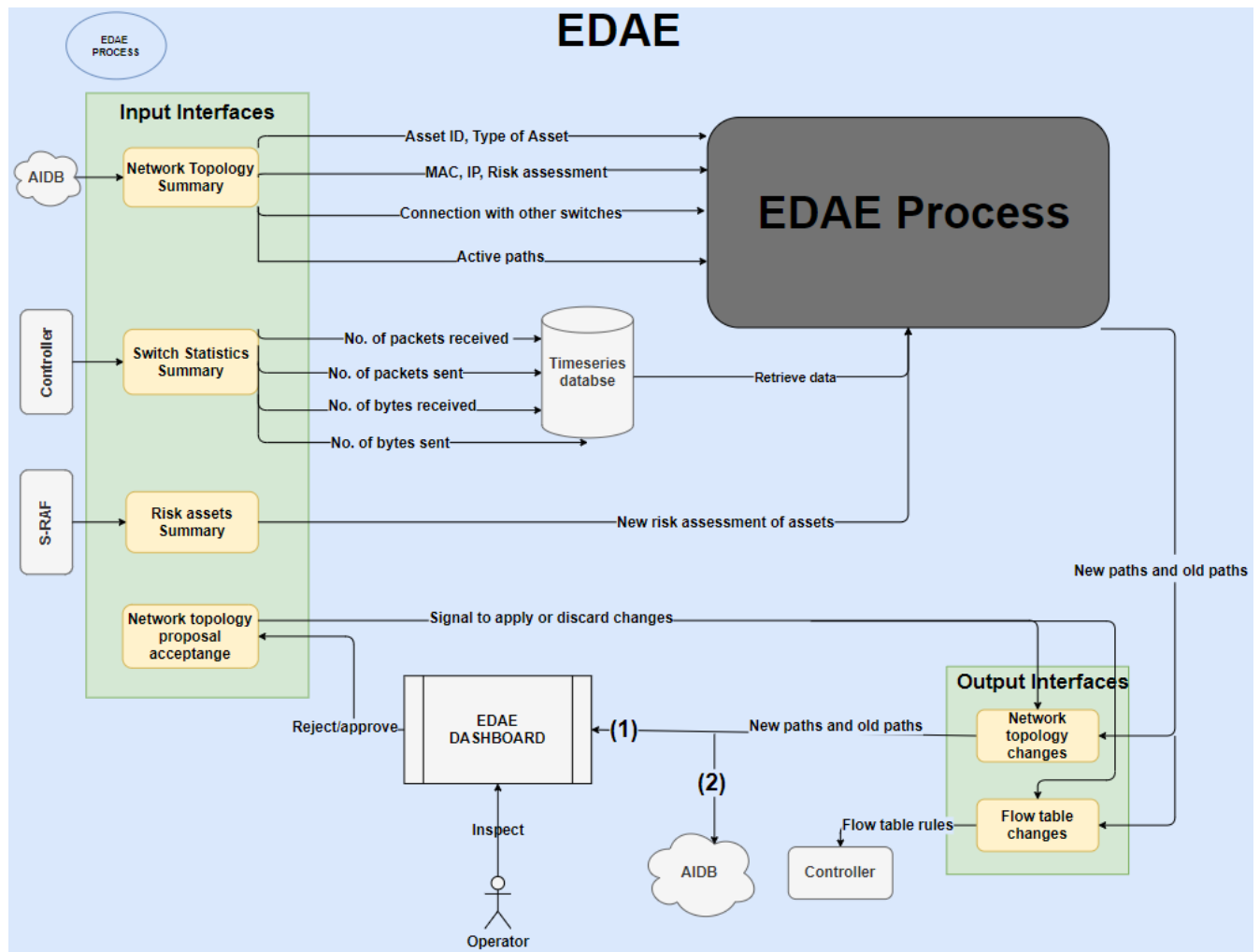


Figure 8 Detailed information flow of the Input and outputs

Inputs:

1) Network Topology Summary interface

The network Topology Summary interface requests from the AIDB for each asset the following attributes:

- Asset ID: (The friendly name of the host)
- Type of Asset: (PMU, PDC, SCADA, RTU, PLC etc.)
- MAC address: (The unique hardware's identifier)
- IP address: (Internet Protocol address assigned to the host)
- Risk assessment: (The security risk level of this host)
- Switch links: (The switches that the host is connected to)
- QoS and security constraints of the hosts (Defined in section 6.2.2.2.2)

2) Active Paths

This interface requests from the controller the current active communication paths between the hosts of the communication network. In SDN networking, multiple communication paths can be constructed between two hosts. With the use of rules, that are stored in flow tables

inside the switches, someone can define the path that a packet will follow from a source host to a destination host. The active communication path is the path that a packet will follow from a source host to a destination source.

3) Switch Statistics Summary

This interface frequently requests from the controller port statistics for each switch in order to store them in the timeseries database. The required statistics are the following:

- Number of packets received to a specific port
- Number of packets sent to a specific port
- Number of bytes received to a specific port
- Number of bytes sent to a specific port

4) Risk assets Summary

The Risk assets interface receives security events produced by S-RAF. This event informs EDAE which assets were affected and how much they were affected.

5) Network Topology proposal acceptance

Through this interface, the EDAE output interfaces are informed if the proposed communication paths of the EDAE core are accepted or not by the operator.

Outputs:

1) Network Topology Changes

This interface sends the old paths and the new paths as they are calculated from EDAE core to EDAE DASHBOARD in order to visualize the results and allow the operator to decide whether will apply the proposed solution or not. Additionally, the results are stored in AIDB.

2) Flow table changes

The flow table changes interface receives the old paths to be deleted and the new paths to be created. To this end, this interface uses the RESTful API provided by the controller in order to update the flow tables of the involved switches accordingly.

4.3 Interfaces Model

This section gathers information about the interfaces required for the implementation of the integrated solution of SDN-microSENSE by defining the communication between the components created in WP-3-4-5 and following the guidelines, requirements and specifications from WP2. The interfaces are in alignment with provided WP4 architecture from the previous section. Previously provided diagrams are representing the interaction between the components and those interactions represents the interfaces that should be further defined and clarified. In order to produce basic information on the interfaces the template for interfaces was developed in order to define those interactions between components. The following subsections describe these interfaces (organized per activity) by detailing the following information:

1. **Description:** Describes the purpose of the interface.

2. **Component providing the interface:** Describes the component that is offering the described interface.
3. **Consumer components:** Describes the components that are using the described interface.
4. **Used Technology:** REST, XML-RPC, GUI, Java API, etc.
5. **Input data:** describes how data that is required by the described interface (e.g.: Methods or Endpoints, values and parameters of the interface).
6. **Output data:** describes the data that is returned by the described interface (e.g.: the returned data of methods or REST call).
7. **API URL:** *URL* of the interfaces through which API is accessed.
8. **Constraints:** Any security or authentication related topics regarding this interface, specifically the need to use a secure transfer protocol. Also, any other constraints (e.g. specific prerequisites, data-types, encoding, transfer rates) which apply to the interface.
9. **State:** Synchronous/Asynchronous, Stream.
10. **Responsibilities:** Partner that is responsible for the implementation and usage of the interface.
11. **Documentation link:** Link toward the documentation of the tool or its interface (public address, or address to the project shared folder).

Previous numbered subsections of the interface can be represented through the table and thus producing the template that can look the same for all interfaces. On the other side, the interface model is not the same for all interfaces and this depends on the content of the tables, i.e. it depends on the descriptions of this subsections where each interface will represent its own unique model, format and used technology.

In this document and within WP4 following interfaces are covered. The interfaces were described in general manner in D2.3. The names used in this document have the intention to be more descriptive, however the interface identifier as it is named in D2.3 it is detailed between brackets [43].

- 1) AIDB – EDAE (*AIDB-1, AIDB-2*)
- 2) AIDB – EDAE-DASHBOARD (*AIDB-1*)
- 3) AIDB – S-RAF (*AIDB-1*)
- 4) AIDB – SDN-C (*AIDB-2, AIDB-3*)
- 5) EDAE – EDAE-DASHBOARD
- 6) EDAE – S-RAF (*S-RAF/SDN-SELF-01*)
- 7) EDAE – SDN-C (*XL-EPDS_NBI-1*)
- 8) EDAE-DASHBOARD - SDN-C (*XL-EPDS_NBI-1*)

4.3.1 AIDB – EDAE

This interface will request the following information from AIDB and it will provide to EDAE:

- Identifier of the asset in the network topology
- Asset type: Switch, Host, SDN-C.
- Additional information about the host purpose (PDC, PMU, RTU, SDN-Switch, SCADA...)
- IP: Internet Protocol address of the asset in communication to such asset.
- List of physical connections between assets
- Communication weights, indicator that represents the level of preference for the communication between two assets
- Bandwidth in Mbps of the physical network connection between two assets.

Additionally, EDAE requires to store into the AIDB the proposed changes in the network topology.

- IP changes: New Internet Protocol address of the asset in communication to such asset.
- Weight changes: New value of the calculated indicator that represents the level of preference for the communication between two assets.

Table 5: AIDB-1 interface for EDAE component

Interface name: AIDB-1	
Description	Network topology summary EDAE requires additional network information which cannot be provided by SDN-C and gets it from the AIDB. This information is taken into account by Self-healing algorithms to appraise the best topologic change purposes.
Component providing the interface	AIDB
Consumer components	EDAE
Used Technology	REST
State	Synchronous
Input data	AssetQuery()
Output data	- Switch(ExternalId) - Hosts (ExternalId, Description, sdnIP) - Network topology(Relationship, Weight, Bandwidth)
API URL	NA
Constraints	NA
Responsibilities	AYESA
Documentation link	AssetQuery()

Table 6: AIDB-2 interface for EDAE component

Interface name: AIDB-2	
Description	Network topology changes EDAE will indicate to AIDB to proposed changes in the network topology, those changes consist of IP redefinitions, and weight revision of directional communication between assets.
Component providing the interface	AIDB
Consumer components	EDAE
Used Technology	REST

State	Synchronous
Input data	AssetUpdate()
Output data	- Host(sdnIP) - Network topology(Weight)
API URL	NA
Constraints	NA
Responsibilities	AYESA
Documentation link	Asset Create/Update/Delete()

4.3.2 AIDB – EDAE Dashboard

The EDAE Dashboard requires additional information which is not provided by the SDN-C. This information is paramount for the Self-healing algorithms and supplies the base to which draw different topologic changes purposes given by the EDAE (from another interface).

Table 7: AIDB-1 interface for EDAE-Dashboard component

Interface name: AIDB-1	
Description	Advanced SDN Topology information This interface provides additional information managed by EDAE about the SDN topology, e.g. weight, which cannot be supplied by the SDN-C.
Component providing the interface	AIDB
Consumer components	EDAE Dashboard
Used Technology	REST
State	Synchronous
Input data	AssetQuery()
Output data	List of SDN Topologic relationships (Host/Switch - Host/Switch, Weight, Bandwidth)
API URL	
Constraints	
Responsibilities	AYESA
Documentation link	AssetQuery()

4.3.3 AIDB – S-RAF

S-RAF summons to AIDB to retrieve information about the SDN infrastructure. It seems strange involving the AIDB rather than the SDN-C, however on a second thought, the AIDB is needed due to not all required information can be provided by the SDN-C. The required information compasses following subjects:

- List of hosts. The AIBD centralizes the SDN asset information by the mean of an only endpoint, avoiding to know the list of SDN-Cs belonging to different subnetworks in the enterprise by the S-RAF side.
- Asset risk level. The risk level of each host is a worthy information to appraise the risk of an eventual attack to some part of the SDN infrastructure.

Table 8: AIDB-1 interface for S-RAF component

Interface name: AIDB-1	
Description	Get all the SDN asset information This interface provides information about all hosts (IP, hostname) belonging to the SDN infrastructure, as well its vulnerabilities.
Component providing the interface	AIDB
Consumer components	S-RAF
Used Technology	API REST
State	Synchronous
Input data	<i>AssetQuery(), AssetRiskQuery()</i>
Output data	<i>Complete list of Hosts(ExternalId, IP, MAC)</i> <i>Active Vulnerabilities(CVE, RiskLevel)</i>
API URL	NA
Constraints	The API User has to belong to a registered Use Case.
Responsibilities	AYESA
Documentation link	NA

Given the detected threats from the XL-EPDS framework and the vulnerabilities of the assets, S-RAF can review the risk level of one or more hosts. This interface requirement is covered by the already addressed in the Table 13: S-RAF/SDN-SELF-01 interface. Next table is identical to the already mentioned one. Notice the row “Consumer components” both EDAAE and AIDB receive the same message.

Table 9: S-RAF-AIDB interface: Risk asset summary

Interface name: S-RAF/SDN-SELF-01	
Description	Risk asset summary This interface allows sending the risk incidents to EDAAE/AIDB with designated criticality level. The interface is intended to provide the required information on the equipment/asset that is under attack and has a risk level. This is designed as the communication channel between S-RAF and SDN-SELF modules, including AIDB
Component providing the interface	S-RAF
Consumer components	EDAAE, AIDB
Used Technology	KAFKA
State	Asynchronous
Input data	SDN-SELF consumes the risk assessment results from the KAFKA queue.
Output data	Specific tables on the incidents containing the information about level of criticality and information about the involved asset. Example of the S-RAF output is given in section 13.1.
API URL	NA
Constraints	None
Responsibilities	UBITECH
Documentation link	<i>SharePoint Link or Public link with the documentation</i> or NA

4.3.4 AIDB – SDN-C

The AIDB stores the SDN infrastructure automatically without the User or any other component involvement. Next, the rest of component can update additional information not provided by SDN-C but available in AIDB. For this, the AIDB component contains a list of SDN-Cs and counts on a programmable task which queries to them all SDN-C.

Table 10: AIDB-3 interface for SDN-C component

Interface name: AIDB-3	
Description	AIDB Updating Interface AIDB utilises the Northbound Interface provided by the SDN-C to synchronize the asset inventory with the current status of the network topology.
Component providing the interface	SDN-C
Consumer components	AIDB daemon
Used Technology	HTTP (REST API)
State	Synchronous
Input data	<i>requestNetworkTopology()</i>
Output data	<i>outputNetworkTopology()</i>
Constraints	None
Responsibilities	UOWM

4.3.5 EDAE – EDAE-Dashboard

Through this interface, EDAE will provide to EDAE-Dashboard the following data:

- Connections: new and/or modified connections between network nodes proposed by EDAE.
- Routes: new and/or modified routes from one node to another proposed by EDAE.

This information will be represented by the EDAE- Dashboard in order the operator can accept or reject the proposals. Consequently, the EDAE-Dashboard must provide the following information to EDAE in return, using a second interface for that purpose:

- User acceptance: Acceptance or rejection of the proposed network topology by the operator or evaluator.

Table 11: EDAE/EDAE-Dashboard interface for EDAE-Dashboard component

Interface name: EDAE/EDAE-Dashboard	
Description	Network topology changes This interface allows EDAE to send the information related to the topological changes proposed by EDAE to the EDAE-Dashboard. These proposals will be represented in the EDAE-Dashboard to be finally accepted or not by the operator. This “evaluation” result is going to be received in this interface.
Component providing the interface	EDAE
Consumer components	EDAE - Dashboard
Used Technology	AMQP (RabbitMQ)
State	Asynchronous
Input data	User acceptance or rejection of the EDAE proposals in a True/False format.

Output data	New/modified connections and routes for each pair of nodes proposed by EDAE.
API URL	NA
Constraints	None
Responsibilities	IREC
Documentation link	NA

Table 12: EDAE/EDAE-Dashboard interface for EDAE component

Interface name: EDAE/EDAE-Dashboard	
Description	Network topology proposal acceptance This interface allows the EDAE – Dashboard to send the acceptance signal to the EDAE. This signal indicates if the topology changes proposed by EDAE have been accepted or rejected by the operator. The EDAE will actuate in accordance to the acceptance signal. To do that previously the interface received the information regarding these proposed changes for the topology.
Component providing the interface	EDAE - Dashboard
Consumer components	EDAE
Used Technology	AMQP (RabbitMQ)
State	Asynchronous
Input data	New/modified connections and routes for each pair of nodes proposed by EDAE.
Output data	User acceptance or rejection of the EDAE proposals in a True/False format.
API URL	NA
Constraints	None
Responsibilities	AYESA
Documentation link	NA

4.3.6 EDAE – S-RAF

This interface will request the following information from S-RAF and it will provide to EDAE:

- Source IP: Internet Protocol address of the asset from where the attack comes from
- Assets: List of the assets in risk
- Id: Identifier of the asset in risk
- Current risk: Current level of risk of the asset in risk

Table 13: S-RAF/SDN-SELF-01 interface for EDAE component

Interface name: S-RAF/SDN-SELF-01	
Description	Risk asset summary This interface allows sending the risk incidents to EDAE/AIDB with designated criticality level. The interface is intended to provide the required information on the equipment/asset that is under attack and has a risk level. This is designed as the communication channel between S-RAF and SDN-SELF modules, including AIDB
Component providing the interface	S-RAF
Consumer components	EDAE, AIDB
Used Technology	KAFKA

State	Asynchronous
Input data	SDN-SELF consumes the risk assessment results from the KAFKA queue.
Output data	Specific tables on the incidents containing the information about level of criticality and information about the involved asset. Example of the S-RAF output is given in section 13.1.
API URL	NA
Constraints	None
Responsibilities	UBITECH
Documentation link	<i>SharePoint Link or Public link with the documentation</i> or NA

4.3.7 EDAE – SDN-C

This interface will request the following information from SDN Controller and it will provide to EDAE the following information for each port of the SDN switch. Regarding the status of a port of a switch:

- rx_packets: Number of received packets
- tx_packets: Number of transmitted packets
- rx_bytes: Number of received bytes
- tx_bytes: Number of transmitted bytes
- rx_dropped: Number of packets dropped by RX
- tx_dropped: Number of packets dropped by TX
- rx_errors: Number of receive errors
- tx_errors: Number of transmit errors
- rx_frame_err: Number of frame alignment errors
- rx_over_err: Number of packets with RX overrun
- rx_crc_err: Number of CRC errors
- collisions: Number of collisions

On the other hand, EDAE will provide at least the following information to SDN-C:

- Dpid: Datapath identifier for the corresponding switch
- Table_id: Identifier of the table to put the flow entry in
- Priority: Priority level of flow entry
- Match: Fields to match
- Actions: Instructions set

Table 14: XL-EPDS_NBI-1 interface for EDAE component

Interface name: XL-EPDS_NBI-1	
Description	Interaction with SDN-C This interface allows EDAE to interact with the SDN-C by inserting, removing or modifying network flows. EDAE will gather the statistics of each port of each switch in time windows of 30 minutes in order to calculate packet loss percentage utilized bandwidth, jitter and other derived time series statistics.
Component providing the interface	SDN-C
Consumer components	EDAE
Used Technology	HTTP (REST API)

State	Synchronous
Input data	<code>requestNetworkStatistics()</code> <code>requestNetworkFlows()</code> <code>addNetworkFlow()</code> <code>deleteNetworkFlow()</code> <code>modifyNetworkFlow()</code>
Output data	<code>outputNetworkStatistics()</code> <code>outputNetworkFlows()</code>
API URL	NA
Constraints	None
Responsibilities	UOWM
Documentation link	https://documenter.getpostman.com/view/12079040/T1DjkfXx?version=latest

4.3.8 EDAE-Dashboard – SDN-C

Through this interface, the SDN Controller will provide to the EDAE-Dashboard the statistic information and the current status of each SDN switch in the network. Specifically, the SDN Controller will provide, for each SDN switch:

- Port description: current status and information of each port of the switch.
- Port statistics: statistics of each port of the switch.
- Table statistics: statistics of each table in the switch.
- Flow statistics: statistics of each flow entry currently installed in the switch.

A more detailed description of the information obtained with the Northbound Interface is presented in the corresponding section.

Table 15: XL-EPDS_NBI-1 interface for EDAE-Dashboard component

Interface name: XL-EPDS_NBI-1	
Description	Switch statistics summary EDAE – Dashboard uses the Northbound Interface provided by the SDN Controller to retrieve the current status and statistics of each SDN switch. This information is represented in the EDAE – Dashboard.
Component providing the interface	SDN-C
Consumer components	EDAE - Dashboard
Used Technology	HTTP (REST API)
State	Synchronous
Input data	<code>requestPortStats()</code> <code>requestPortDescription()</code> <code>requestSwitchFlows()</code> <code>requestSwitchTables()</code>
Output data	<i>A collection of json files with the outputs of the corresponding Northbound Interface requests.</i> <code>outputPortStats()</code> <code>outputPortDescription()</code> <code>outputSwitchFlows()</code> <code>outputSwitchTables()</code>
API URL	NA
Constraints	None



Responsibilities	UOWM
Documentation link	https://documenter.getpostman.com/view/12079040/T1DjkfXx?version=latest

5 SDN Controller design and implementation

This section provides the technical details concerning the development of the SDN Controller (SDN-C), focusing on the SDN-C interaction with the application and management planes. It should be noted that in D4.2, only the northbound interface of the SDN-C is provided, since the implementation of the SDN-C is fully described in D4.1. Moreover, this section provides the technical documentation of the SDN dashboard, a user interface that resides in the management plane and has been developed as a user-friendly interface with the northbound interface, allowing the system administrator to interact with the SDN-Cs and observe their status.

5.1 Interfaces Model - Northbound Interfaces

The Northbound Interface (NBI) is utilised by implementation-independent applications that aim to interact with the SDN-Cs by retrieving information and apply configurations to the OpenFlow-enabled intermediary devices (SDN switches). The API endpoints offered by SDN-C are grouped into two categories:

- **Ofctl_rest** commands, based on the API specification provided by the Ryu project [44], aims to retrieve information and update configuration of SDN switches, utilising the OpenFlow protocol. All endpoints provided by this REST API are translated to a corresponding synchronous OpenFlow command by the SDN-C and then, are instructed to the specified SDN switch. In comparison to the original implementation, ofctl_rest has been improved in terms of compatibility. In more detail, the response format of the command that retrieves the network flows has been adequately re-formatted to use the correct keywords of match fields defined in version 1.3 of OpenFlow. For example, instead of the `dl_dst` keyword that the default ofctl_rest API utilises to indicate the destination Ethernet address, the keyword “eth_dst” is utilised by the ofctl_rest provided by SDN-microSENSE, in compliance with OpenFlow 1.3. And, more importantly, TCP and UDP related match keywords are now distinguishable.
- **Rest_topology** commands, based on the corresponding application provided by the Ryu project [45], aim to retrieve the network topology. In comparison with the original implementation provided by the Ryu framework, the customised version of rest_topology offered by SDN-microSENSE supports inputs and outputs only in decimal form (datapath IDs and port numbers) to achieve compliance between rest_topology and ofctl_rest. Moreover, rest_topology also integrates the responses of the delay monitoring expansion of the topology module (technical details concerning how the delay monitoring is implemented, are provided in D4.1).

The NBI provided by each SDN-C is secured via TLS by integrating an nginx server in each SDN-C, which acts as a TLS termination proxy. Moreover, to ensure that each client is properly authorised to access the NBI, Basic HTTP Authentication is configured on the nginx proxy of each SDN-C, requiring by the client to provide the appropriate credentials in the header of each HTTP request. The details of this implementation, in the context of the SDN-C architecture and implementation, is provided in D4.1.

5.1.1 Rest_topology API

This information is considered as confidential and it is found in annex 13.5

5.1.2 Ofctl_rest API

This information is considered as confidential and it is found in annex 13.5

5.1.3 Components Model

5.1.3.1 SDN dashboard architecture

An overview of the SDN dashboard architecture is presented in Figure 9. The SDN dashboard follows the Model-View-Controller (MVC) architecture [46], according to which, the system functionality is segmented into discrete MVC applications that are mainly characterised by:

- **Models:** Data structures that abstract components of the data model. The data model is stored in a relational database that is part of the SDN dashboard.
- **Views:** Human-friendly interfaces that interact with the user.
- **Controllers:** Generate the views by retrieving the data contained in the models and applying any necessary data transformations.

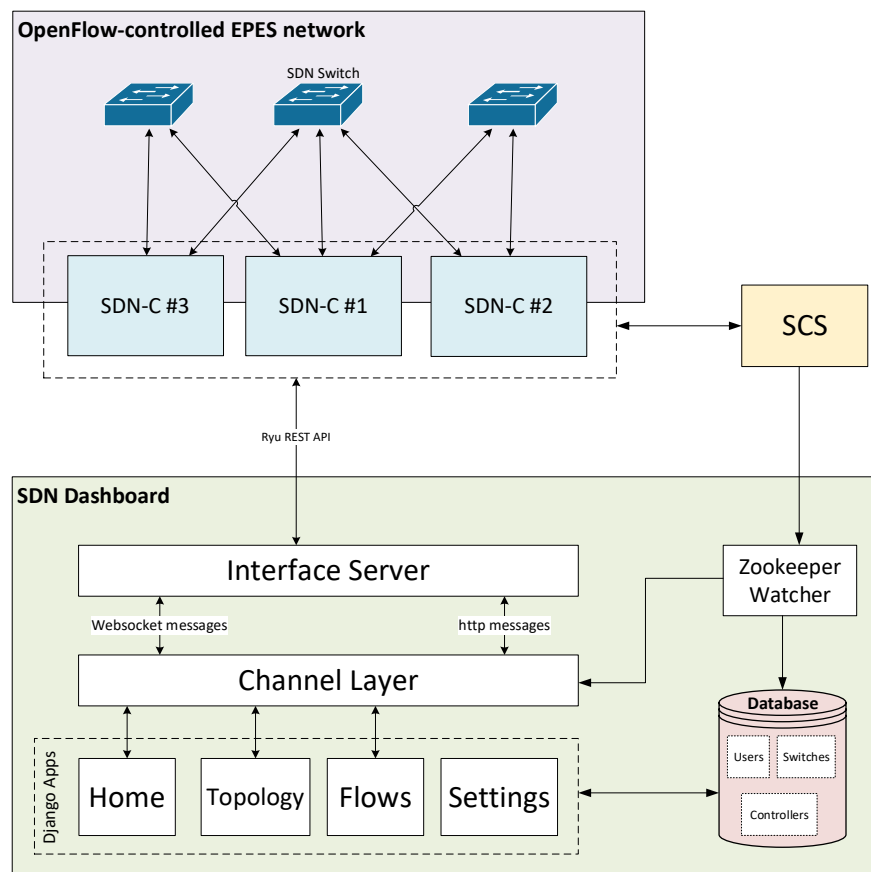


Figure 9: Architecture of the SDN dashboard

In a nutshell, the SDN dashboard is comprised of the following MVC applications (their details will be analysed in the next subsections):

- **Home:** Provides an overview of the switches and the SDN-Cs.
- **Flows:** Provides the means to add, modify, delete, and view network flows.
- **Settings:** Provides the views to edit system preferences along with the corresponding models to retrieve and save settings.

- **Topology:** Renders the topology by retrieving the topology from the SDN-C.
- **Users:** Provides all the necessary views to view and manage users as well as for users to reset their lost password.

As mentioned earlier, the SDN dashboard communicates transparently with the underlying SDN-Cs by determining the master SDN-C for each SDN switch. To accomplish this, the **Zookeeper Watcher** component of the SDN dashboard runs in parallel with the Django framework and maintains an asynchronous communication channel with SCS (Synchronisation and Coordination Service). SCS is an Apache Zookeeper database utilised by the SDN-Cs to elect a single master SDN-C for each SDN switch (the details of this component are provided in D4.1). The Zookeeper Watcher is connected asynchronously with SCS via the Python-based Kazoo library [47], and, during the initialisation, dumps the relevant tables and replicates the contents of SCS to the internal database of the SDN dashboard. After the initialisation phase, the Zookeeper Watcher listens for any change (caused by a new SDN-C connection or by an SDN-C failure) and replicate those changes directly to the internal database and the SDN dashboard applications.

Finally, all internal applications of the SDN dashboard, including Zookeeper Watcher, interface with the **Channel Layer**. This is an architectural component which acts as a shared message queue and allows real-time message exchange between the various components of the SDN dashboard. The Channel Layer proves to be useful by transmitting real-time changes to the end-user frontend via WebSocket. For example, when the master SDN-C changes or a new SDN-C is connected to an SDN switch, while the user has opened the homepage, this information is transmitted via the WebSocket protocol to the active webpage and the corresponding page section is changed via JavaScript.

5.1.3.2 Database schema

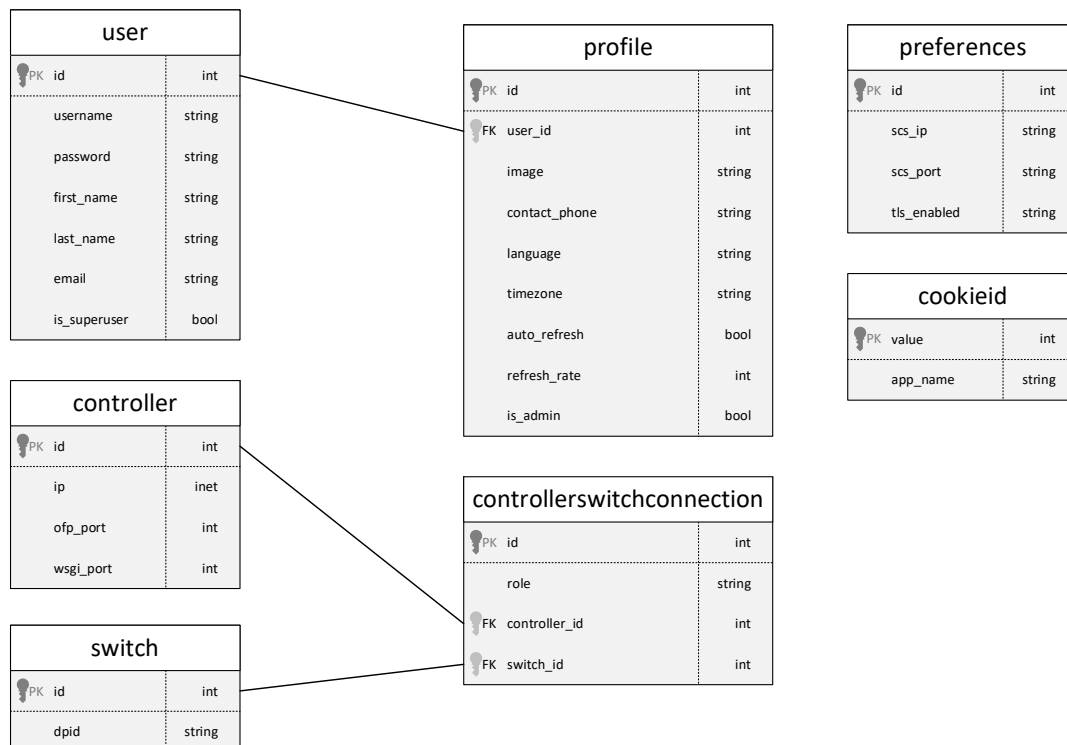


Figure 10: The ER diagram of the SDN dashboard

The SDN dashboard uses an internal database to store operational states, the users and their privileges as well as system customisation and preferences. In summary, the internal database consists of the following tables:

- **User:** This is the default table used by the system to store the users that are able to login to the system.
- **Profile:** This table has one-to-one relationship with the user table (by using the field `user_id` as foreign key) and stores additional information about the user. Additional information includes a contact/emergency phone number, language and time zone preferences, the path of the profile picture uploaded by the user, and the options whether the user prefers that the SDN dashboard UIs should auto refresh as well as the refresh rate.
- **Preferences:** This table represents system-wide configurations, including the TCP socket of the SCS component and whether the communication with SCS should be secured via TLS.
- **Cookieid:** Each SDN app in the context of SDN-microSENSE reserves a unique cookie ID to facilitate the management of the flow entries. This table stores the reserved cookie IDs as well as the name of the corresponding SDN app.

Regarding the state of the underlying SDN infrastructure, the following tables are used and maintained by the Zookeeper Watcher:

- **Controller:** This table stores all SDN-Cs detected in SCS. For each controller, the IP is stored as well as the OpenFlow port and the TCP port used for the NBI requests.
- **Switch:** This table stores all SDN switches detected in SCS. For each switch, the datapath ID is stored in its decimal form.
- **Controllerswitchconnection:** Since a controller may connect to multiple switches and a switch may connect to multiple controllers, this extra table has been created to realise the many-to-many relationships. For each such relationship detected in SCS, except of the two foreign keys that correspond to the switch and the controller, the field `role` is inserted that represents the role of the controller (1 for EQUAL, 2 for MASTER or 3 for SLAVE). Django reassures that only a single controller/switch combination exists in this table.

5.1.3.3 *User roles and privileges*

The SDN dashboard uses three user roles to ensure that only authorised users manipulate the programmable network and the registered users. The roles and their privileges are outlined below:

- **Simple user:** This is the user that has no extra privileges. The simple user can view the homepage and the topology, although view-only access is granted to the network flows, meaning that they cannot access the Flow Control page and modify, add, or delete network flows. Moreover, access to the user management system is not allowed.
- **Security administrator:** This user has additional access to the Flow Control page, meaning that they can modify, add, and delete network flows. However, the security administrator has no access to the user management system. The role of security administrator is indicated by the field `is_admin` of a user profile. Note that only the superuser can change this value for any user.
- **Superuser:** This user has access to all views of the SDN dashboard and is able to manage the registered users as well as to create new ones. This role is indicated by the field `is_superuser` in a user instance.

5.1.4 User Interfaces

5.1.4.1 Users

The user application provides all the necessary functionalities to view and manage users as well as for users to reset their lost password.

First, when an unauthenticated user tries to access a view that requires authentication, then this user is redirected to the login page (`/login` view), illustrated in Figure 11:

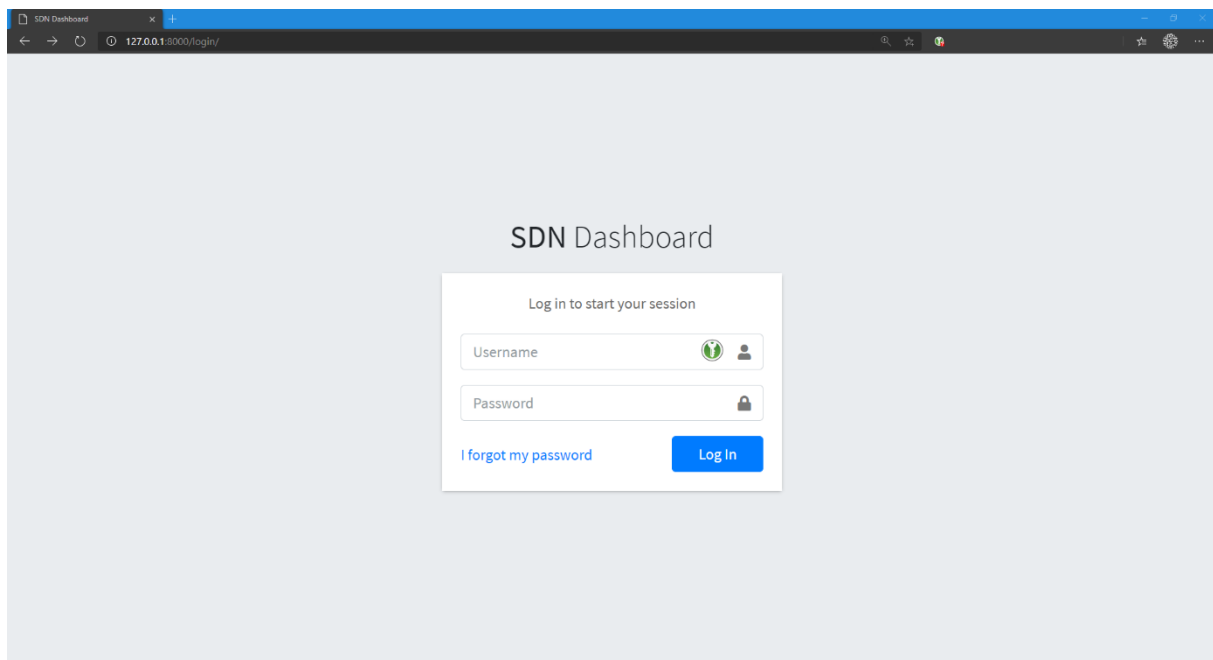


Figure 11: Login page of the SDN dashboard

After successful login, the user is redirected to the homepage, described in the next subsection.

If the user has forgotten their password, then they can click the option “I forgot my password” and the user will be redirected to `/password-reset` in order to provide their email and receive instructions for resetting their password. The corresponding pages are illustrated in Figure 12 and Figure 13.

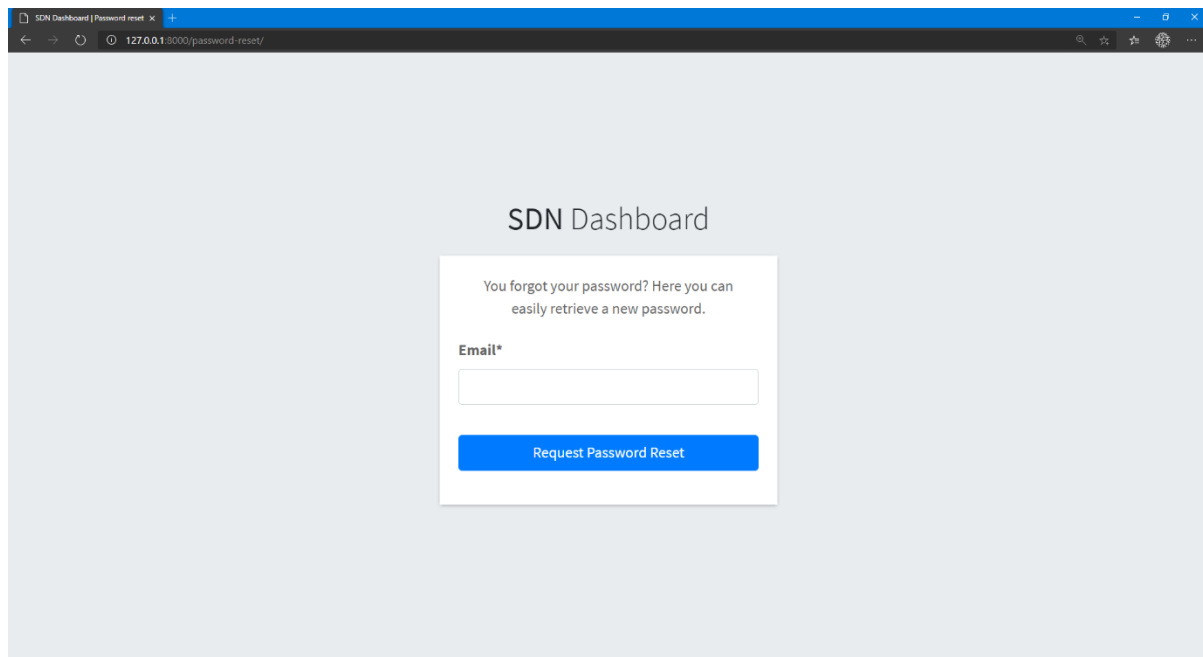


Figure 12: Reset password page of the SDN dashboard

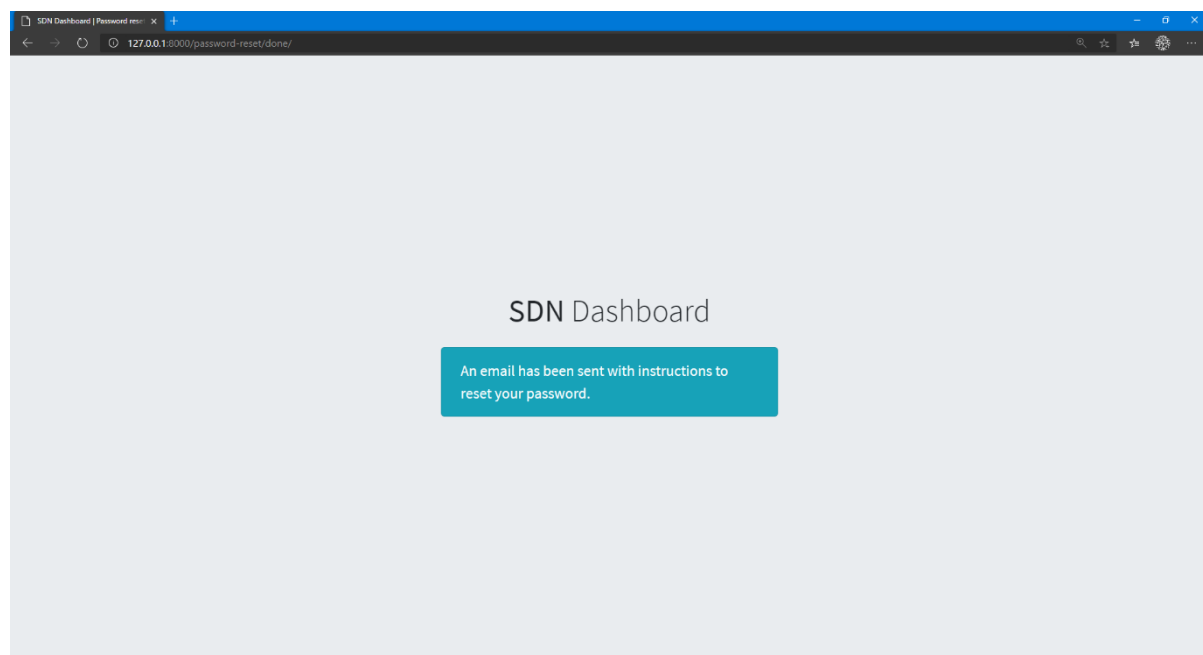


Figure 13: Reset password notification of the SDN dashboard

If the logged-in user has the superuser privilege, then access is granted to the user management system (/users). The user management page is illustrated in Figure 14, where all users are displayed along with their privileges and their most basic attributes, including their username, full name, contact phone and email. For each user, the options Edit and Delete are provided. When Delete is clicked and the user confirms their option, then the corresponding user is deleted. Naturally, the system prevents a user from deleting themselves.

When the Edit option is clicked, then the user is redirected to the user editing view located in `/users/edit/[username]`, where username corresponds to the selected user. The user editing menu is illustrated in Figure 15. In this menu, the user can edit the username, the first and last name, the profile image, the contact phone, the language and the time zone. Only the superuser can alter the `is_admin` field of the user profile to grant the corresponding user additional privileges.

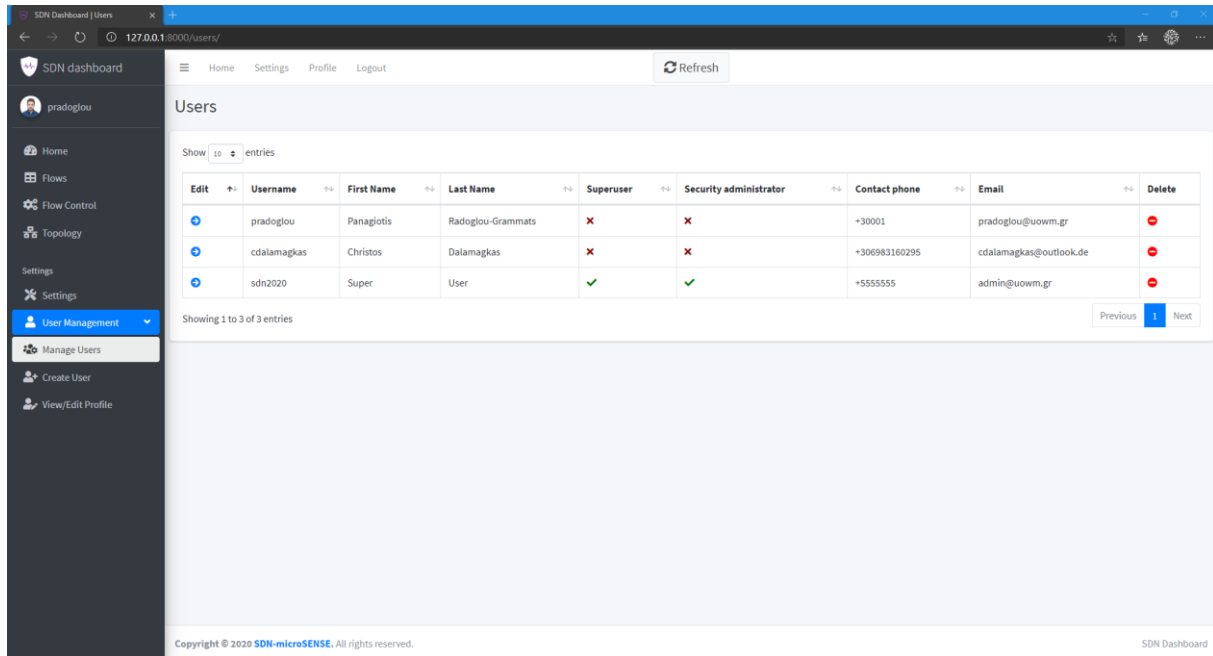


Figure 14: The user management system of the SDN dashboard

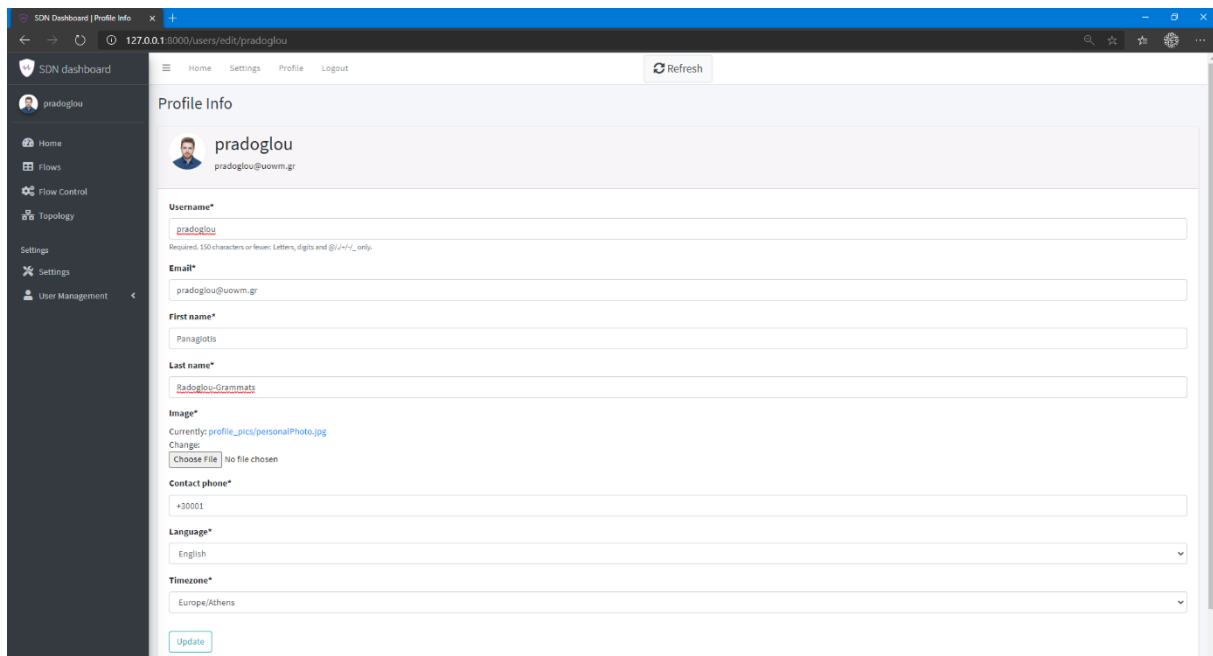


Figure 15: Profile editing menu of the SDN dashboard

To create a new user, the `/users/create` view is provided, illustrated in Figure 16. In this form, the superuser can insert basic information about the new user and upon the user is created, additional information can be inserted by editing the corresponding profile in the user management system.

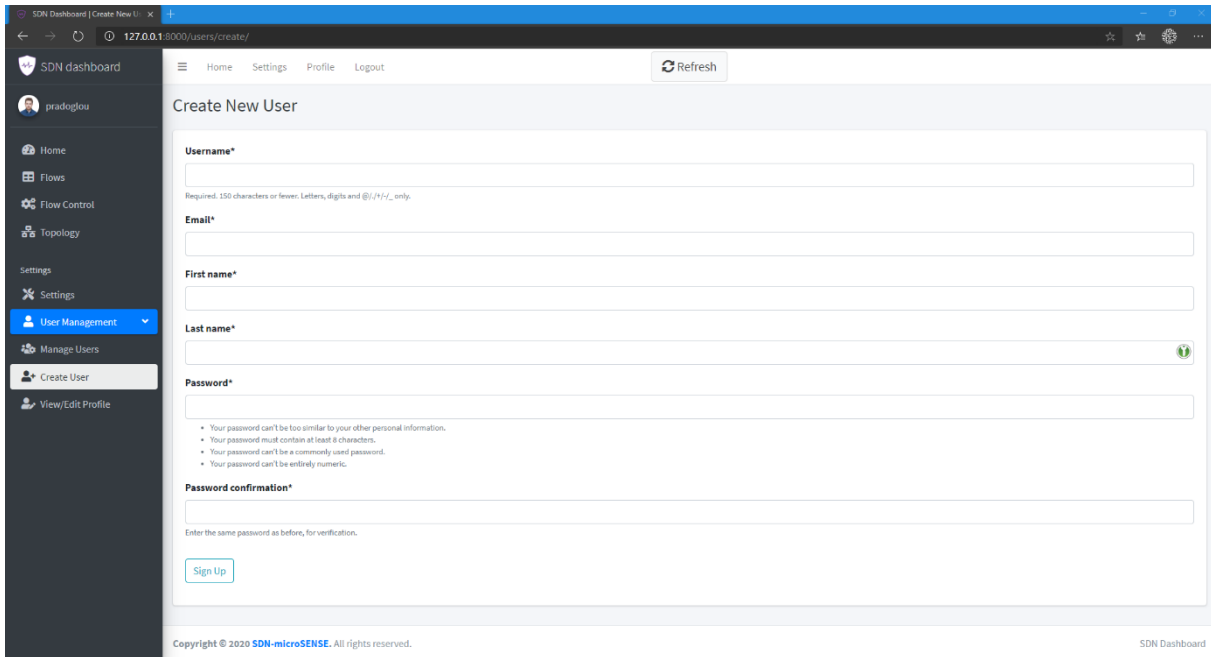


Figure 16: New user menu of the SDN dashboard

In summary, all views of the user application are provided bellow:

Table 16: User Management Views

Method	URI	Description	Parameters
GET	<code>/admin</code>	Administration portal provided by Django to manage the data models.	-
GET	<code>/login</code>	The view utilised by users to login.	-
GET	<code>/logout</code>	The view utilised by users to logout.	-
GET	<code>/password-reset</code>	The view utilised by users to reset their password.	-
GET	<code>/password-reset/done</code>	The view displayed after a successful password reset.	-
GET	<code>/users</code>	Renders the page that displays all users.	-
GET	<code>/users/create</code>	Renders the form utilised by the superuser to create a new user.	-
GET	<code>/users/edit/[username]</code>	Allows to edit a user's attributes.	The username at the end of the URL.
GET	<code>/users/get</code>	Returns all users in JSON format.	-
GET	<code>/users/delete/[username]</code>	Deletes a user specified by their username.	The username at the end of the URL.

5.1.4.2 Homepage

The homepage is an application consisting of a single view that renders the landing page of the SDN dashboard. This page, illustrated in fig. XX, provides an overview about the available SDN switches and, for each SDN switch, the connected SDN controllers along with their details and roles as well as information and statistics about the ports controlled by OpenFlow and the configured tables.

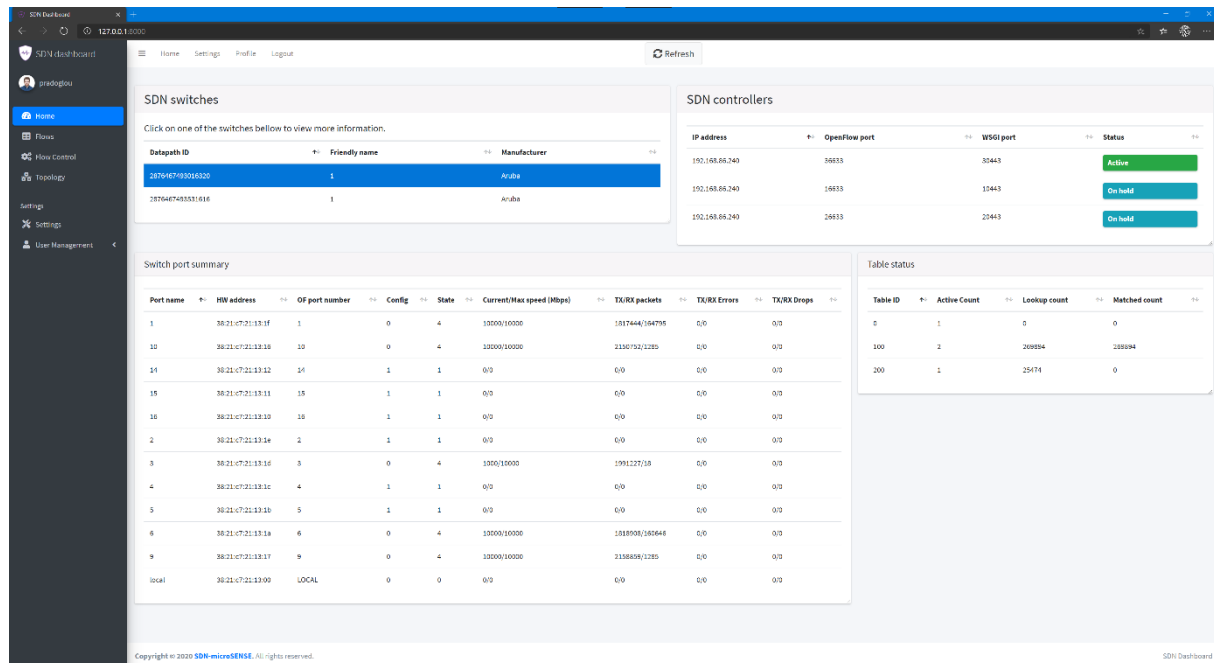


Figure 17: Homepage of the SDN dashboard

5.1.4.3 Flows

The Flows application encompasses all the necessary views to view and manage network flows. Two pages are provided by this application:

- **Flows** to view the network flows from all SDN switches and for all non-empty OpenFlow tables.
- **Flow Control** to enable addition, modification, and deletion on the flow tables.

The Flows page (/flows) is directly accessible from the main sidebar menu and is rendered dynamically by retrieving a) the SDN switches from the internal database, b) the flow tables of each switch and c) the flows of each SDN switch. A separate HTML table is created for each switch/table combination and each table displays basic attributes of each flow, including priority, match fields, the assigned cookie ID and SDN app, the duration of the flow in seconds, the idle/hard timeouts, the packet/byte counters, and the actions (instructions) applied upon matching. For each flow, there are two buttons, Edit and Delete. The Edit button passes the network flow attributes to the Flow Control page (/flow/edit) via a GET request whereas Delete issues a delete strict command to the corresponding SDN switch via the SDN dashboard backend (/flows/action/delete-strict) in order to delete the specified flow. User confirmation is ensured to avoid possibly costing human errors.

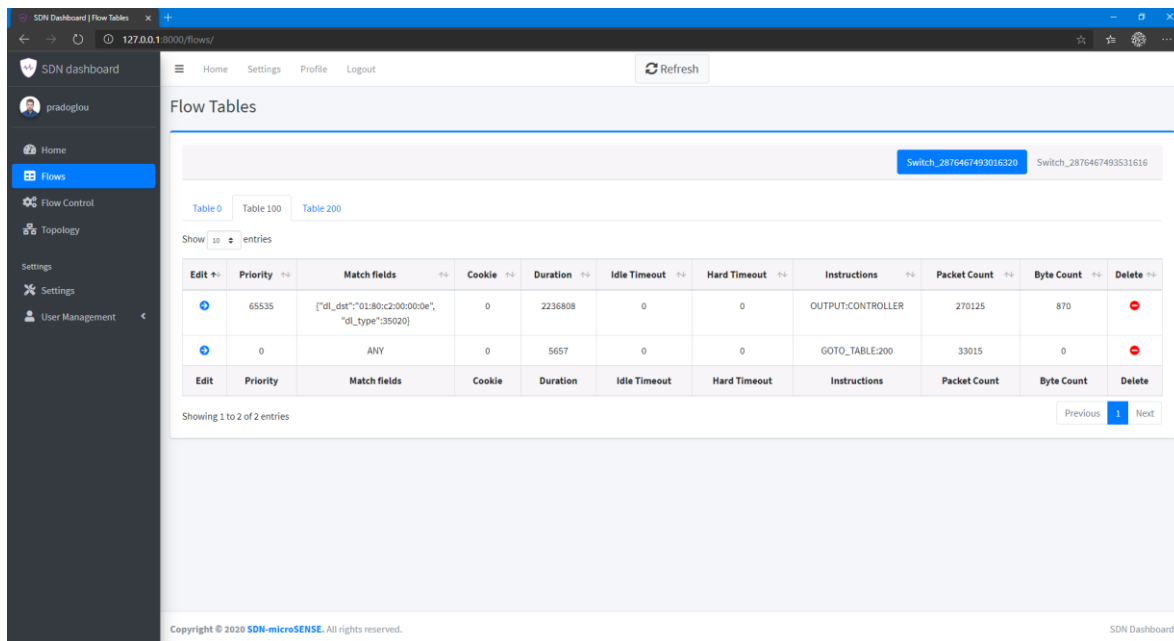


Figure 18: Flows menu of the SDN dashboard

Last, the Flow Control page is illustrated in Figure 19 and is also directly accessible via the main sidebar. This menu allows the user to apply all the supported flow operations, including Add, Modify/Modify Strict, and Delete/Delete Strict. The user can specify all the network flow options that are utilised in the context of SDN-microSENSE, including undefined number of match fields, flow priority, timeouts, cookie IDs and instructions (GOTO_TABLE and OUTPUT). When a user chooses to submit, the appropriate backend API endpoint (depending on the selected flow action) receives the command content in JSON format, determines the master SDN-C by consulting the internal database and issues the appropriate NBI command. Note that only the Security Administrator and the Superuser have access to this page. To aid the user navigate through the various options, a number has been placed in the upper-left corner of each card.

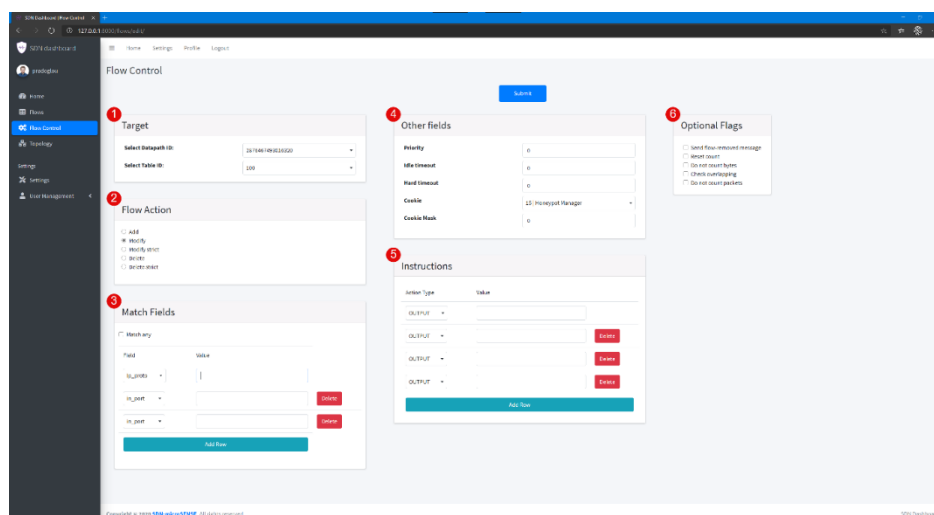


Figure 19: The Flow Control menu of the SDN dashboard

All the views implemented by this application are provided in the table below:

Table 17: Flows Views - SDN Dashboard

Method	URI	Description	Parameters
GET	/flows	Renders the Flows page, including the layout, the DataTables, the pills and the tabs dynamically.	-
GET	/flows/edit	Renders the Flow Control page.	-
GET	/flows/get-flows	This view returns the flows of a specific datapath ID by asking the appropriate master SDN-C.	Dpid
GET	/flows/get-switches	Returns all switches in JSON format, tailored to be consumed by the select2 dropdown widget.	-
GET	/flows/get-tables	Returns all table IDs of a specific datapath ID in JSON format, tailored to be consumed by the select2 dropdown widget.	Dpid
POST	/flows/action/add	Instructs an “Add flow” request to the master SDN-C of the dpid specified, by parsing the data specified in the request body.	dpid, priority, match, cookie, idle_timeout, hard_timeout, actions
POST	/flows/action/modify	Instructs a “Modify flow” request to the master SDN-C of the dpid specified, by parsing the data specified in the request body.	
POST	/flows/action/delete	Instructs a “Delete flow” request to the master SDN-C of the dpid specified, by parsing the data specified in the request body.	
POST	/flows/action/modify-strict	Instructs a “Modify strict” request to the master SDN-C of the dpid specified, by parsing the data specified in the request body.	
POST	/flows/action/delete-strict	Instructs a “Delete strict” request to the master SDN-C of the dpid specified, by parsing the data specified in the request body.	

5.1.4.4 Topology

The topology is the application that renders the page which displays the network topology (/topology). The page is directly accessible via the main sidebar. The topology page contains a JavaScript application that communicates with the appropriate backend view (/topology/get-topology), to retrieve the network topology in JSON format from any SDN-C holding the master role. The JSON object is then delivered unedited to the JavaScript application [XX], which draws the SDN switches, the hosts and the links between switches. The topology application is illustrated in Figure 20.

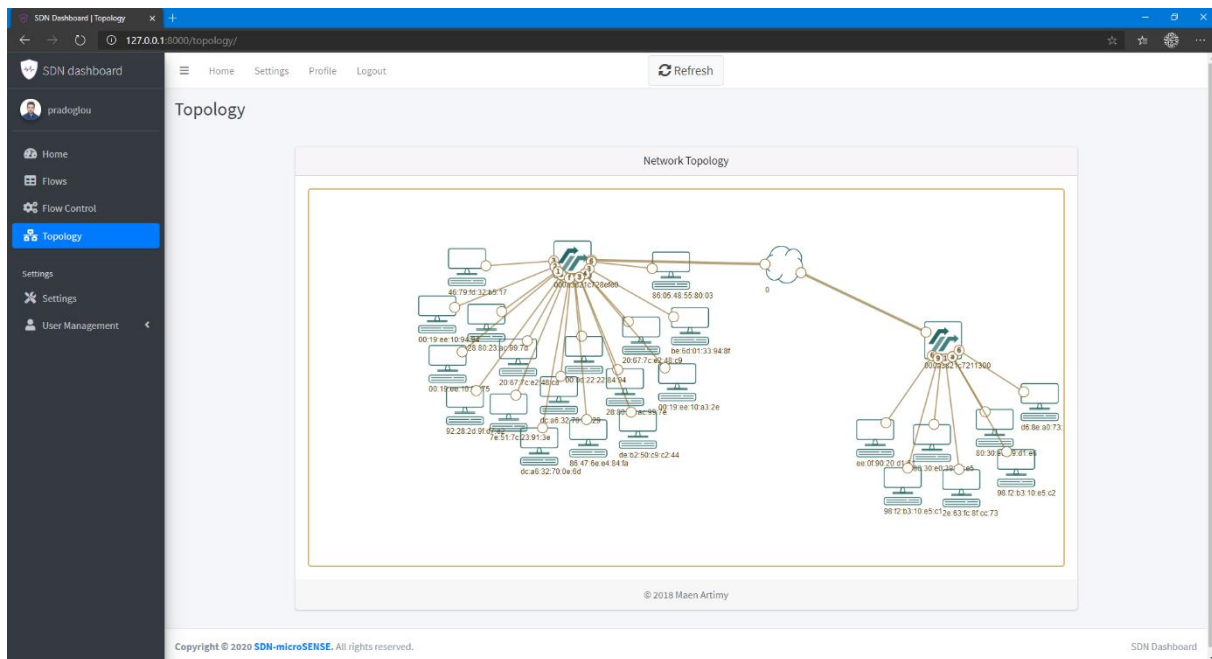


Figure 20: The topology menu of the SDN dashboard

All views provided by the topology application are outlined below:

Table 18: Topology Views - SDN Dashboard

Method	URI	Description	Parameters
GET	/topology	This view renders the topology page and the D3.js application that illustrates the topology.	-
GET	/topology/get-topology	This view returns the topology as a single JSON object.	-

5.1.4.5 Settings

The Settings application undertakes the management of user and system preferences. The main view that renders the settings menu (/settings) is depicted in Figure 21. This menu provides three main groups of configurations, divided in separate menu cards:

- **Zookeeper parameters:** The user can provide all the necessary connection parameters with SCS (Zookeeper Server), including the IP address and the TCP port. If TLS is enabled in the port specified, then the user can check the Zookeeper over TLS option and provide the KeyStore file & password, the certificate file of SCS and the CA certificate used to sign the certificate of SCS. This menu card is only available for the security administrator and the superuser.
- **Cookies customisation:** This menu card provides the means to add and delete mappings between cookie IDs and SDN apps utilised by SDN-microSENSE. This card is only available for the security administrator and the superuser.
- **System preferences:** This menu card provides user-tailored configuration options, bound to the profile of each user, including the option to auto-refresh all tables across the SDN dashboard as well as the refresh interval. This menu card is available to all users.

At this point, it should be mentioned that the settings menu is displayed differently, depending on the role of the logged-in user. Since only the privileged users (superusers and security administrators) are able to configure the communication with SCS and determine the cookies mapping, these options are not shown to users not having the appropriate privileges.

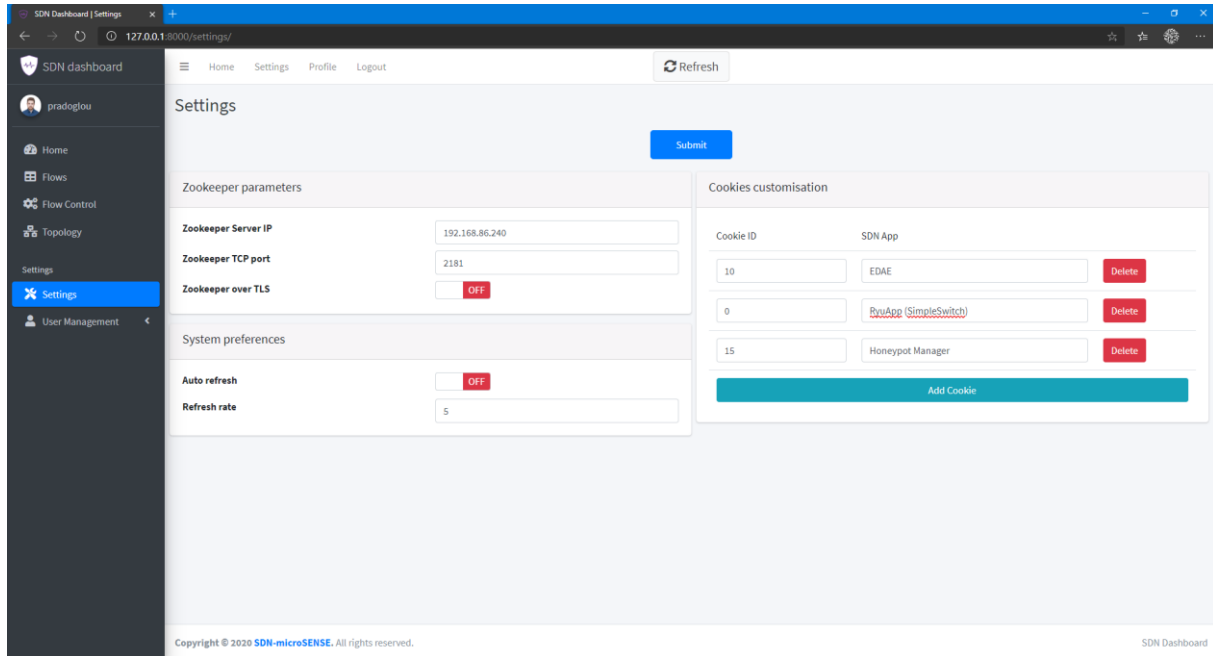


Figure 21: The Settings menu of the SDN dashboard

All views provided by the Settings application are summarised below:

Table 19: Settings Views - SDN Dashboard

Method	URI	Description	Parameters
GET & POST	/settings	The GET version of this view renders the settings menu, while the POST version saves the configurations stored in the forms.	POST only: scs_ip, scs_port, tls_enabled, auto_refresh, refresh_rate, keystore_file, keystore_password, scs_cert, ca_file.
GET	/settings/cookies	This view returns the existing cookies in JSON format.	-

5.1.5 SDN dashboard prototype deployment

This subsection describes the implementation details and offers the necessary information for the deployment of the SDN dashboard in operational environments. An illustration of the deployed SDN dashboard architecture is illustrated in Figure 22.

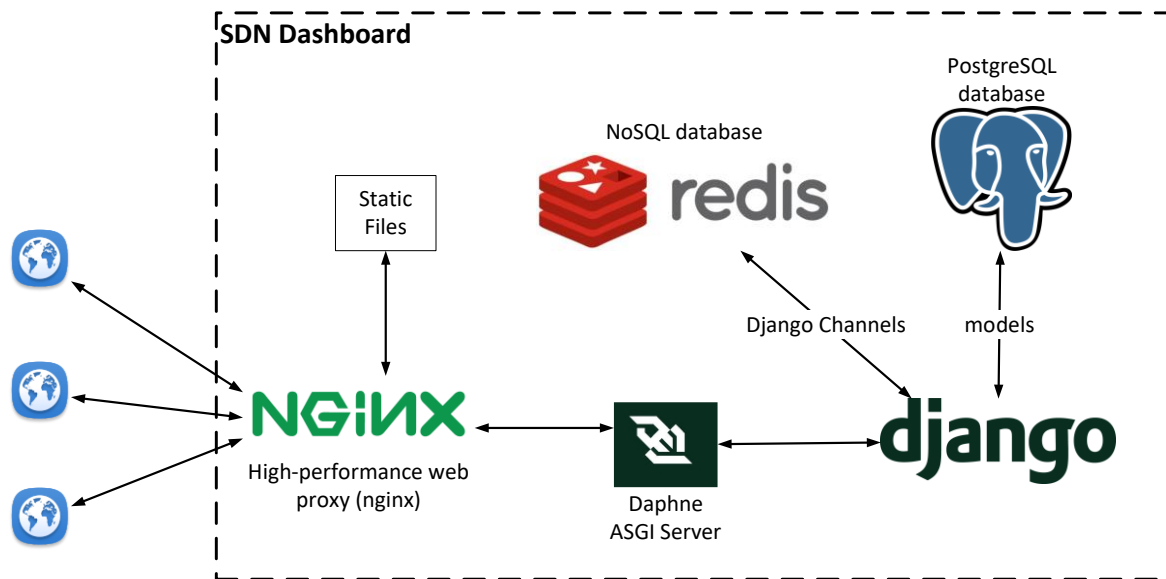


Figure 22: Deployment architecture of the SDN dashboard

The deployment architecture follows a three-tier hierarchy, where a high-performant web server receives all web requests and serves static files and applies end-to-end TLS encryption, while it proxies dynamic requests to an Asynchronous Server Gateway Interface (ASGI), that in turn delivers the requests in a standardised way to the SDN dashboard applications residing in the innermost tier. NoSQL database is utilised to offer dynamic interaction between the backend and the frontend, while persistent storage is ensured with SQL database.

Regarding the implementation details of the SDN dashboard, the Python-based Django framework was chosen to implement the backend of the SDN dashboard, one of the most popular programming frameworks that follows the MVC paradigm. To store the models, the registered users, user preferences and any other static data described in the data model, a PostgreSQL database is utilised by Django. PostgreSQL is an open-source relational database management system (RDBMS) that offers persistent, SQL-compliant, and production-grade storage. In the front tier of the SDN dashboard, nginx is employed to process the web requests and: a) serve static files, including images and HTML/CSS files, b) apply end-to-end TLS encryption, c) apply load balancing and caching where required, and d) proxy dynamic requests to Django. For Django to interpret web requests in a standardised way, the Daphne ASGI framework is used. In comparison to WSGI and Gunicorn, ASGI is the new emerging standard and spiritual successor to WSGI, that supports asynchronous web interfaces and the WebSocket protocol. Finally, to facilitate the transmission of asynchronous messages between the various modules of the SDN dashboard, Django Channels are used, a framework that abstracts the producers and consumers of an asynchronous system and uses the Redis database to implement this abstraction in a transparent way.

Regarding the front-end development, several JavaScript-based libraries and frameworks were used. jQuery is a library that simplifies the creation of powerful widgets in the frontend, including select2 dropdowns, DataTables and switch buttons, while it undertakes asynchronous communication with the backend via AJAX requests. The WebSocket protocol was also used to asynchronously change the frontend by immediately reflecting changes occurred in the backend. Finally, the topology application is based on the D3.js and the FlowManager project [48].

5.1.5.1 Prerequisites and Installation

The SDN dashboard can be provided as a virtual appliance (.ova file) that can be used by multiple hypervisors, including VMware Workstation, VMware ESXi, Oracle Virtualbox, Citrix Hypervisor, Proxmox VE etc. The following minimum computing resources should be provided to the SDN dashboard VM:

- x2 virtual CPU cores
- 1 GB RAM
- 20 GB HDD
- Ubuntu 20.04

Using the Oracle VirtualBox hypervisor as an example, the following steps can be executed to deploy the SDN dashboard. The installation process should be adapted accordingly depending on the actual hypervisor.

Step 1: From the tab named “File” of the Oracle VirtualBox, click the option called “Import Appliance...” as illustrated in Figure 23:

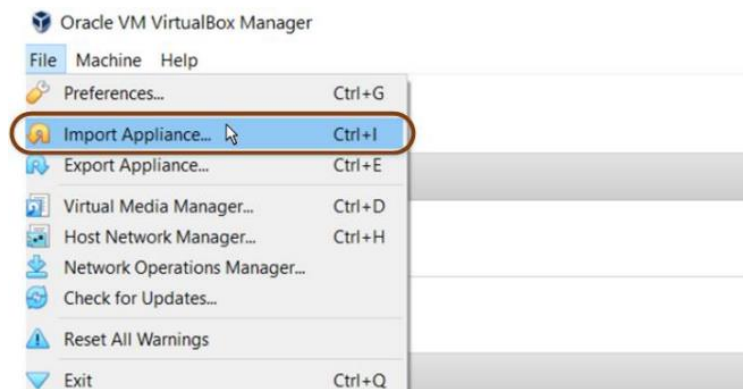


Figure 23: Import Appliance via Oracle VirtualBox

Step 2: From the new window, insert the location of the provided SDN dashboard OVA file, as depicted in Figure 24. Next, click the option “Next”.

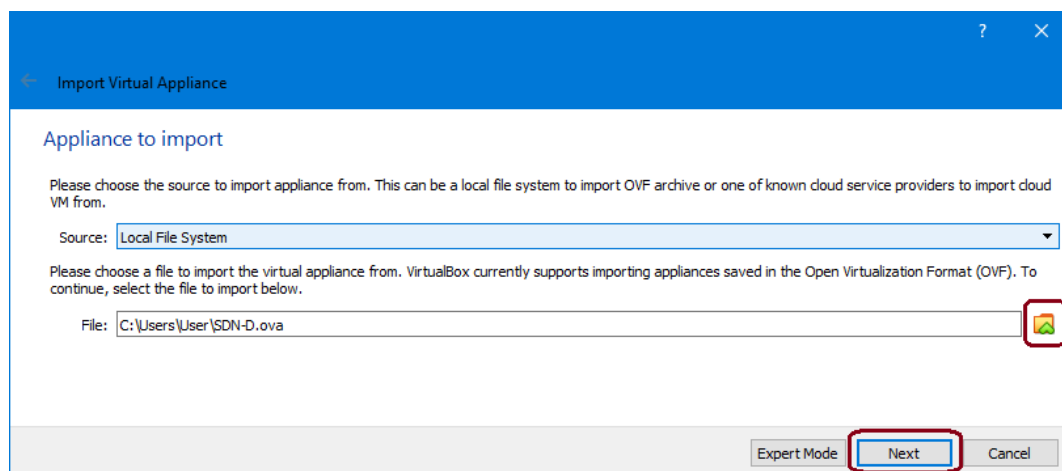


Figure 24: Locating the SDN dashboard OVA file

Step 3: From the new window, click the option “Import”, as depicted in Figure 25, using the predefined options.

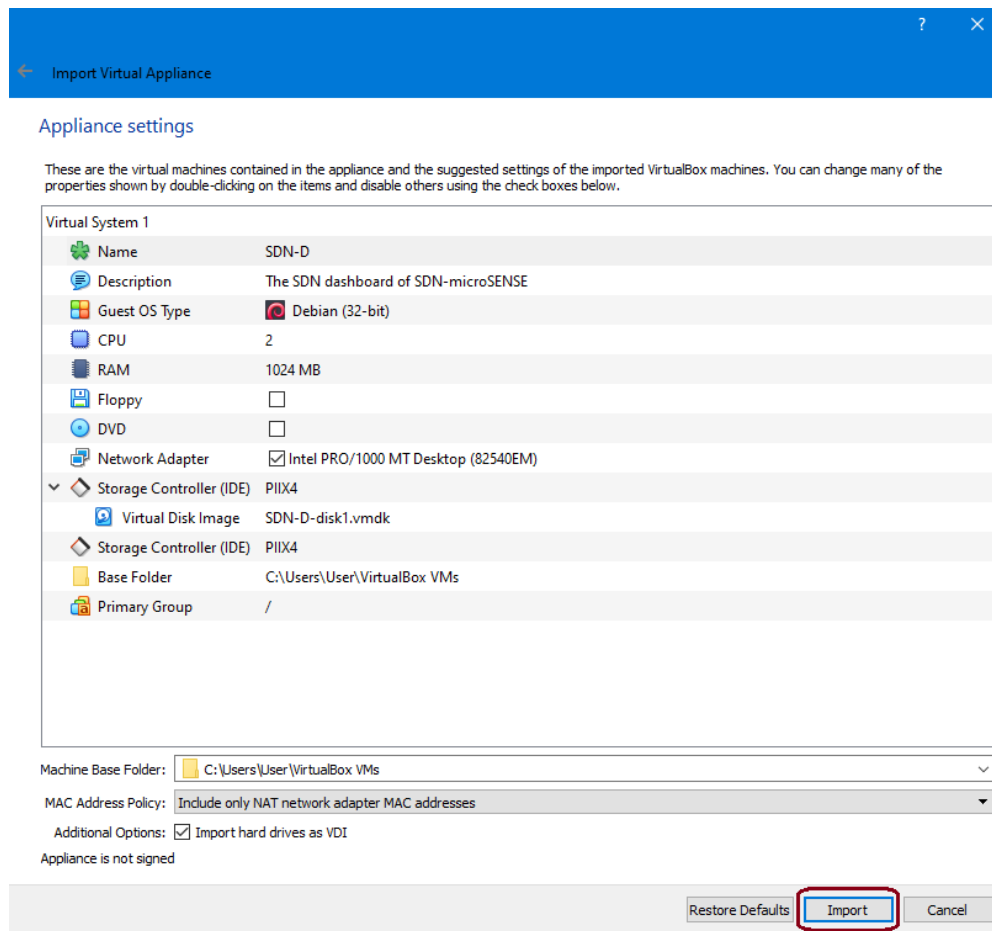


Figure 25: Import options for the SDN dashboard

Step 4: Wait for VirtualBox to finalise the importing procedure of the SDN dashboard virtual image, as illustrated in Figure 26.



Figure 26: Wait VirtualBox to finish importing the SDN dashboard OVA

Step 5: Start the SDN dashboard virtual machine by choosing the corresponding VM and clicking the Start button, as depicted in Figure 27.

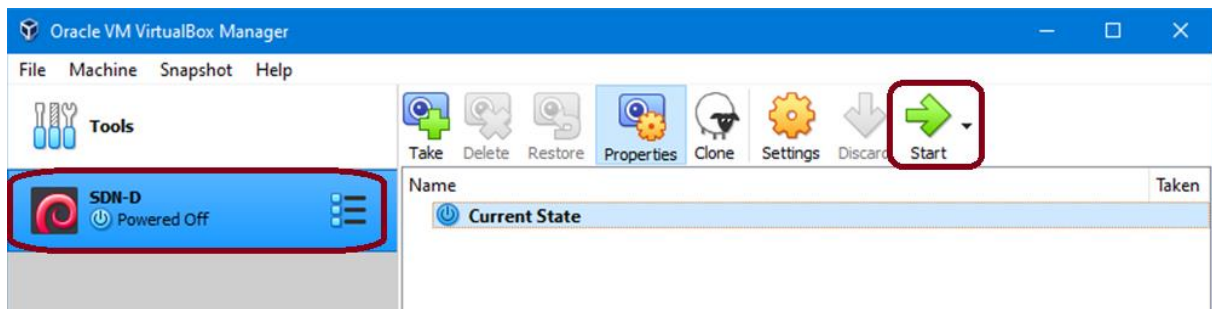


Figure 27: Start the SDN dashboard image

Step 6: Use the following credentials for login, as illustrated in Figure 27:

Username: user

Password: user

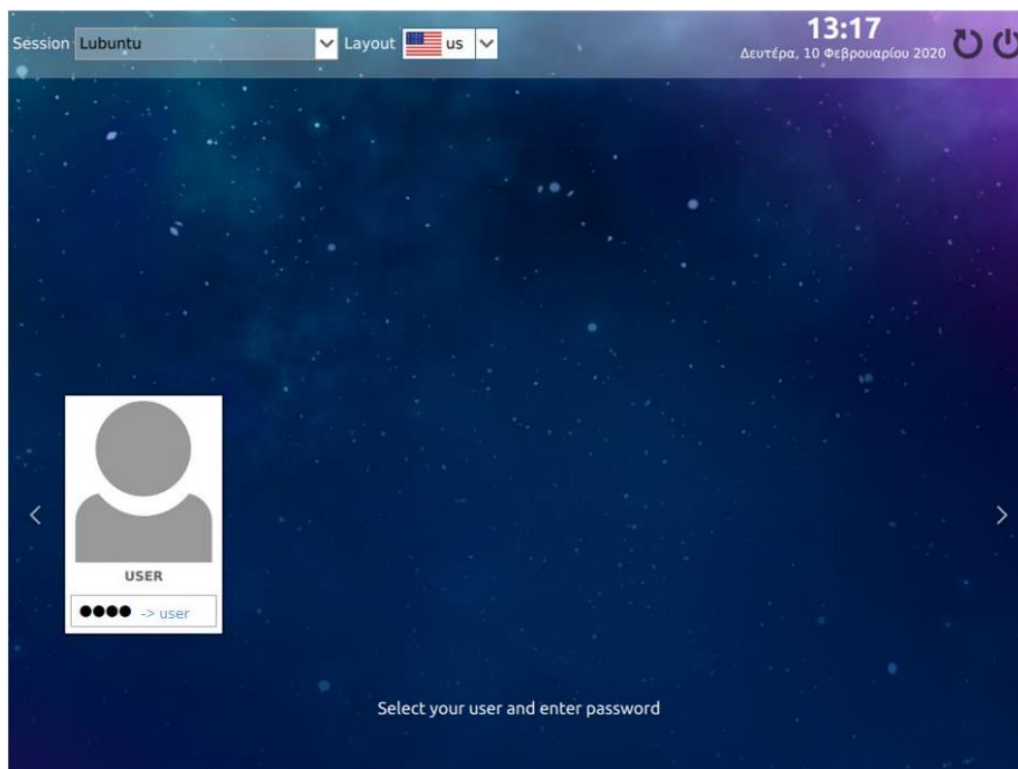


Figure 28: SDN dashboard VM credentials

5.1.5.2 Source code repository

For the development of the SDN dashboard, modern development techniques were followed to ensure maximum code quality with continuous testing. To this aim, the SDN dashboard has been developed (and will continue its development during the project lifetime and beyond), by using Gitlab, hosted by UOWM, an open-source DevOps lifecycle tool that follows the Git Version Control System (VSC). The repository is hosted under the following link:

https://snf-15142.ok-kno.grnetcloud.net/sdn-microsense/sdn_dashboard

Access to this repository is provided only to UOWM and PPC developers as well as to authorised partners of the SDN-microSENSE consortium, upon request.

6 Electric data Analysis engine (EDAE) design and implementation

6.1 SDN-based System Model

According to the grant agreement, the task 4.2 will deploy the network management processes that will increase the observability of the EPES. When a part or unit of the EPES ecosystem is under attack, either by accident or due to a cyber-attack, the corresponding logical connections will be rearranged, establishing new communication paths.

The Electric Data Analysis Engine (EDAE) is the component that will provide the solution to maximize the observability of the EPES. No other application was considering the interaction between the SDN-C, the SRAF and the AIDB in order to propose an alternative routing path between monitoring/control units and measuring units in the power system. EDAE has been designed as a special need beyond what it is written in the Grant Agreement.

EDAE main functionalities are:

- To provide an alternative network path from a measurement unit to a control/monitoring unit in case it exists a redundancy in the control/monitoring system where the information could be forwarded.
- To isolate a network element that is assessed to be in risk. Either if the risk affects a control/monitoring unit, a measurement unit or if it is identified as an unknown attacker.
- To provide an alternative network path from a measurement unit to a control/monitoring unit in case a network element is attacked or under risk and it exists a network loop between such elements.

EDAE is not only maximizing the observability but improving the quality of service and the security of the network elements:

- Considering the network latency between nodes so the path with less latency will be selected when it exists different options.
- Considering the risk assessment of the assets present in the network topology in order to avoid potential risks.
- Considering the port statistics of the network switches in order to prevent overloading or other

6.1.1 Network

There are two types of network elements to consider:

- **Network switches:** It is required that the network switches can be managed by the OpenFlow protocol. In Figure 29, an Aruba 2930F is a network switch that supports OpenFlow protocol, hence, it can be used for SDN applications. They can be controlled by the SDN Controller. In D4.1 a detailed description of SDN-Controller and SDN-Switches configuration is explained and it is very relevant for the development of this task and the whole integration of tools and components.



Figure 29: Aruba 2930F as a network switch

- **Hosts:** It is considered all the network elements or power grid elements connected to a network switch. They belong to the infrastructure plane.

6.1.2 Power Grid

There are two types of power grid elements to consider:

- **Monitoring/control Units:** Are those elements that gather information from the measurement units such as Phasor Data Concentrators (PDC) or central elements that monitor the grid behaviour and control some network elements, such a SCADA.
- **Measurement Units:** Are those elements that are taking measurements such as Phasor Measurement Units (PMU) or Remote Terminal Units (RTU). In Figure 30 an example of a Schneider Electric RTU that is going to be used in the different pilots.



Figure 30: Schneider Electric RTU

6.2 EDAE core: Architecture

In the previous section, the main EDAE functionalities were described. This section is devoted to describe the operation of the EDAE core. Figure 31 depicts the EDAE core Architecture.

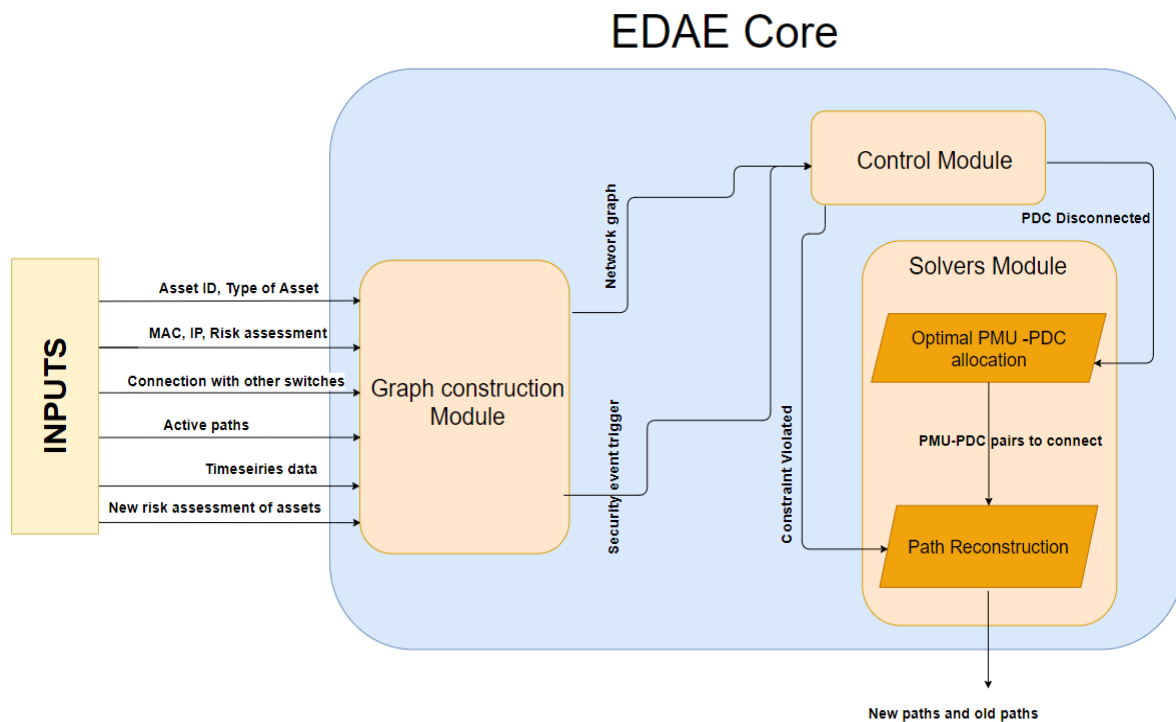


Figure 31: EDAE Core Architecture

6.2.1 Graph Construction Module

The graph construction module requests from the AIDB information related to network topology in order to create the network graph in a representation useful for the process. Concretely, each node (host or switch) of the graph is augmented with risk assessment metadata (packet loss percentage, latency, jitter and the available bandwidth of the links) from AIDB. Finally, the graph is augmented with metadata that contain a list with the active paths between the hosts.

6.2.2 Control Module

The Control module inspects each active path for QoS and Security constraints violation or if a PDC is disconnected (packet loss percentage above a certain threshold). The inspection is periodical (every 1 minute) or asynchronous if an S-RAF event occurs. If a security constrain is violated the control module initializes the genetic algorithm for path reconstruction. If a PDC is disconnected, the control module initializes the Mixed Integer Linear Programming (MILP) algorithm for the optimal PMU-PDC allocation.

6.2.3 Solvers Module

The solvers Module consists of the multi-objective genetic algorithm and the MILP algorithm. Concretely, if an objective for a host is violated, the genetic algorithm search for an alternative path that will not violate the constraints of the host, by taking into account also the constraints of the rest of the hosts. If a solution is found, it is provided to the output interfaces. A detailed explanation about the genetic algorithm is given in section 6.3.2.2.

If a PDC disconnects from the network due to a cyber-attack or a malfunction, then a matchmaking algorithm for optimal PDC-PMU re-allocation using Mixed Integer Linear programming is invoked. The objective of this algorithm is to restore the redundant observability of the EPES infrastructure by re-assigning the PMUs to the rest of the PDCs, taking into account the bandwidth constraints of the

PDCs/PMUs and the average latency between each PMU and PDC connection. More information about the MILP algorithm is given in section 6.3.3.2.3. If the optimal solution is achieved, the new PDCs and PMUs allocation is passed to genetic algorithm to construct the communication paths of the whole network.

6.3 EDAE Algorithm formulation

6.3.1 Related methods from the literature

In this paragraph, various techniques regarding rerouting mitigation strategies in an SDN-based communication network are presented. Specifically, the techniques to be analysed cover the aspects of Quality of Service (QoS) optimization, security optimization on a cyber-attacked network and observability optimization.

Table 20: Objective coverage per work displayed

	QoS	Security	Observability	Energy
[49]	✓	✓	X	✓
[50]	✓	✓	X	X
[51]	✓	X	✓	X
[52]	X	✓	X	X
[53]	✓	✓	X	X
[54]	X	X	✓	X

The QoS describes how a given service meets the user and functional requirements. Network depended services are heavily relied on the qualitative performance of the network service. SDN architecture allows the reconfiguration of the network in order to allow achieving certain objectives such as security, latency and observability. Thereby, an application can be developed such that it takes into account the QoS of each network depended-service and try to reconfigure the network in a way that will meet the needs of each service.

In the work presented in [49] the aspects of energy consumption of the communication infrastructure, the network QoS and security criteria are taken into account. Regarding QoS, the team quantifies it as the sum of the delay of each intermediate links between two nodes and the objective is to minimize the aforementioned path. The Pareto front set of the optimal solutions are extracted using a genetic algorithm and the deviation of the current structure of the network with respect to the optimal solutions is calculated and recommendations to restructure the network (flow tables) are provided to the network operator.

The work presented in [50] is unique because it incorporates Deep Reinforcement Learning (DRL) using Deep Deterministic Policy Gradients (DDPG) [55] in its solution. They propose an online routing self-trained decision-making algorithm that takes decisions based on the current state of the network while optimizing QoS and security. Specifically, they add an additional layer in the SDN architecture called Agent-layer, which is similar to the application layer but it differs in the way that it actually moves the intelligent and intense by means of processing power decision-making part of the controller to a separate application. The formulation of their problem is separated into three entities. First, they determine the state vector, which is a vector that contains the frequency of packet-in message, the

occupancy of the flow table and the channel occupancy rate between node and the controller for every switch. Next, they determine the action vector that is used to decide on which switch the next hop will be assigned. They define it as a tensor that holds the weights that a node J will be assigned as the next hop from the node i . Finally, and most important, is the Reward function that consists of two sub rewarding functions that are related with security and QoS rewards respectively. The QoS reward function takes into account the propagation delay and the link packet loss between the nodes. Finally, they continue with the formulation of the loss function that incorporates the aforementioned entities with training details.

A QoS aware mechanism suited for SDN-Based Wide Area Measurement System (WAMS) infrastructure is presented in [51]. The critical components of a WAMS infrastructure are the Phase Measurement Units (PMUs), the Phasor Data Concentrators (PDCs) and the relevant applications that make use of them. There are applications that their functionality is depended on the timely arrival of the necessary data. Not meeting the demanded QoS of some applications might lead to disastrous decision-making. The authors, of [52] propose the classification of the traffic flows into different class of services depending on the target application with an application of QoS aware mechanism alongside with a content-aware queuing algorithm in order to provide low latency for critical WAMS applications. Specifically, they initially classify the traffic into two classes, namely normal and QoS0. QoS0 traffic is the traffic between PMU and PDC, while each other traffic that passes from PDC is tagged as Normal. Consequently, the class QoS0 is further classified between PDC and control centre as QoS1 and QoS2. The Normal traffic passes without any changes. QoS1 class contains traffic that should be sent to the control centre immediately since they carry critical information (e.g. voltage above a certain threshold). QoS2 class has a lower priority than QoS1 but is less delay-sensitive. Afterwards, they develop a routing algorithm that finds the shortest path between a PMU and PDC. The routing algorithm is formed as an Integer Linear Programming (ILP) problem by taking into account the leftover resources of each switch, the leftover resources of each link, the required resources of the PMU and the cost of each link without specifying what resources are or how the cost is calculated. After a solution has been found, the algorithm searches for a backup solution but the links that are included in the first solution are discouraged to be included in the backup solution by adding a high cost for those links. Furthermore, they include a queuing method, which treats each aforementioned traffic class differently in order to prioritize newer acquired data and to selectively drop the less important data due to the limited capacity of the queue. Finally, in the traditional WAMS the PDC has a timer with fixed value in which time interval waits for the PMUs to send their data. If the latency between all PMUs and PDCs is lower than this threshold then a delay is injected. Hence, they dynamically calculate the timer's value in specific time intervals as the maximum latency observed between a PMU and PDC.

The work proposed in [52] is regards reconfiguration of the network topology bun in the application layer. For each host they identify a set of characteristics that form their diversity ID. Such characteristics can be the type of the operating system, version of the operating system, known vulnerabilities of the system, type of antivirus, software versions, type of administrative personnel etc. Their goal is to construct the paths in such way that in order an attacker to reach a target host h_t in the network through an already attacked host h_a to have to pass through a lot of diverse hosts. They propose a Moving Target Defence (MTD) technique, solving a Shuffle Assignment Problem (SAP). Their technique has two modes, an active and a reactive mode. In the active mode, the technique calculates a number of different configurations, which alternates over time in order to make harder for an

attacker that has not been detected yet to reach a specific target. The reactive mode is enabled when an attacker has been identified, and chooses the configuration that is optimal for the specific attacked host. Consequently, in the reactive mode the algorithm iterates over a predefined number of iterations in order to find the shortest attack path and increase its length by adding a neighbour host in the path.

RouteGuardian is the technique named by the author of [53]. In this paper, their goal is to construct fast paths that take into account the capacity and security requirements of the application. In order to achieve that they collect a list with the security devices. After the acquisition of the list, they compute the K shortest paths between h_1 and the first security device in the list using k-shortest algorithm, which is an extension of Dijkstra. From the k shortest paths, they choose the one that has the maximum capacity (bandwidth). Finally, they iterate over the same procedure to connect the first security device with next one in the list etc. until the last device in the list is connected with the destination host.

Finally, the work of [54] is a self-healing mechanism specified for wide-area monitoring protection and control (WAMPAC) applications in modern electrical grids. This work has to do with PMU and PDC devices. The notion of observability in the context of their work is simplified on how many buses are measured through PMUs, if all the buses are measured by PMUs then the grid is observable. The aim of authors is to restore the observability, in case a PDC is disconnected, by reconnecting the PMUs as fast as possible in such way that all the bases are observable. Their technique consists of two stages. The first one is aided to recover observability by connecting that many PMUs such that all the bases are observable and the PDCs are capable to host such many connections of PMUs. The PMU-PDC allocation problem is formed as an Integer Linear Programming (ILP) problem. Consequently, after the PMU-PDC pairs have been identified, the second stage of algorithm is to find the paths that will serve the connection of the PMU-PDC pairs. Again, they formulate a second ILP problem that aims to find paths that will require the less possible installation rules such that the total traffic load on each link of the path never exceeds the bandwidth.

6.3.2 Problem formulation

First it is described the algorithm that finds the optimal path between two hosts by means of QoS and security and finally the algorithm that assigns the leftover PMUs to the PDCs. Table 19 presents the notation of variables and symbols and a description of them.

6.3.2.1 Variables and Symbols

Table 21: Notations and description of variables and symbols used in EDAE algorithm formulation

Notation	Description
SW	Set of communication network forwarding devices (switches)
BS	Set of buses in the power transmission network
PMU	Set of PMUs in the network
PDC	Set of PDCs in the network
L	Set of active communication paths between all hosts with QoS and/or Security constraints
$LT_c(h_i)$	Latency constraint for host i
$AB_c(h_i)$	Available bandwidth constraint for host i
$PL_c(h_i)$	Packet loss constraint for host i
$JTR_c(h_i)$	Jitter constraint for host i
$SR(h_i)$	Security constraint for host i
L_{bs_i, bs_j}	Set of transmission lines between buses
$PDC_c \in PDC$	Set of compromised/faulty PDCs



$PMU_d \in PMU$	Set of disconnected PMUs
p_{h_i, h_j}	A full communication path between two hosts
bs_i	Bus i
h_i	End-host i
s_i	Switch i
a_i	A network asset could be switch or host.
$T_k \in \mathbb{N}$	Discrete measurement of time in seconds e.g. (kth sec)
$g \in [0,1]$	Degree of weighting decrease for exponential weighted average
$< a_i, a_j >$	Link between two assets of the network.
$t_{x[i \rightarrow j]}(T_k)$	Number of packets transmitted from switch i to switch j until time T_k
$r_{x[i \rightarrow j]}(T_k)$	Number of packets received switch j from switch i until time T_k
$t_{xb[i \rightarrow j]}(T_k)$	Number of bytes transmitted from switch i to switch j until time T_k
$r_{xb[i \rightarrow j]}(T_k)$	Number of bytes received switch j from switch i until time T_k
$b_{<a_i, a_j>}(T_k)$	Bandwidth between two network assets a specific time T_k
$pl_{<a_i, a_j>}(T_k)$	Packet loss percentage between two network assets within a specific time window
$th_{<a_i, a_j>}(T_k)$	Throughput between two network assets within a specific time window
$ab_{<a_i, a_j>}(T_k)$	Available bandwidth between two network assets a specific time T_k
$jtr_{<a_i, a_j>}(T_k)$	Jitter between two network assets a specific time T_k
$lt_{<a_i, a_j>}(T_k)$	Latency between two network assets a specific time T_k
$sr_{<a_i, a_j>}$	Security risk level between two switches
sr_{s_i}	Security risk level of switch i
sr_{h_i}	Security risk level of host i
$PL : P_{h_i, h_j} \mapsto \mathbb{R}^+$	Function that maps a full communication link to its packet loss
$B : P_{h_i, h_j} \mapsto \mathbb{R}^+$	Function that maps a full communication link to its bandwidth
$AB : P_{h_i, h_j} \mapsto \mathbb{R}^+$	Function that maps a full communication link to its available bandwidth
$JTR : P_{h_i, h_j} \mapsto \mathbb{R}^+$	Function that maps a full communication link to its jitter
$LT : P_{h_i, h_j} \mapsto \mathbb{R}^+$	Function that maps a full communication link to its latency
$SR : P_{h_i, h_j} \mapsto \mathbb{N}$	Function that maps a full communication link to its security risk level
A	Set of available PMUs to connect with PDCs in MILP problem
B	Set of available PDCs to connect with PMUs in MILP problem
$x(a,b)$	Decision Variable of the MILP model

6.3.2.2 Search for optimal path

The goal of this algorithm is to find the optimal path between two hosts by means of security and QoS given a number of constraints. The section is structured as follows, first is presented the mathematical formulation of the variables and functions, consequently the objective function and the constraints are formulated and finally is presented the solving method.

6.3.2.2.1 Variables and functions

In this section, the definition and the origin of the variables and functions that construct the input to EDAAE takes place.

Below are presented the variable s that are collected through various APIs that interface EDAAE with other components:

$$b_{<a_i, a_j>}(T_k) = requested_from_AIDB \text{ (Switch Statistics summary)}$$

$$lt_{\langle a_i, a_j \rangle}(T_k) = \text{requested_from_PostgreSQL_server_timeseries_database}$$

$$t_{x[i \rightarrow j]}(T_k) = \text{requested_from_PostgreSQL_server_timeseries_database}$$

$$r_{x[i \rightarrow j]}(T_k) = \text{requested_from_PostgreSQL_server_timeseries_database}$$

$$t_{x_b[i \rightarrow j]}(T_k) = \text{requested_from_PostgreSQL_server_timeseries_database}$$

$$r_{x_b[i \rightarrow j]}(T_k) = \text{requested_from_PostgreSQL_server_timeseries_database}$$

$$sr_{\langle a_i, a_j \rangle} = \text{requested_from_AIDB}$$

$$sr_{s_i} = \text{requested_from_AIDB}$$

$$sr_{h_i} = \text{requested_from_AIDB}$$

Below are presented the mathematical equations that are used to calculate other variables from variables collected from the interfaces. In the context of the application T_K will be the time that the latest measurement of the aforementioned variables occurred.

Equation (1) calculates the packet loss percentage from switch i to switch j based on the packets transmitted from switch i and the packets received from switch j over a period.

$$pl_{\langle a_i, a_j \rangle}(T_k) = \frac{[t_{x[i \rightarrow j]}(T_k) - r_{x[i \rightarrow j]}(T_{k-N})] - [r_{x[i \rightarrow j]}(T_k) - r_{x[i \rightarrow j]}(T_{k-N})]}{t_{x[i \rightarrow j]}(T_k) - t_{x[i \rightarrow j]}(T_{k-N})} * 100 \quad (1)$$

Equation (2) calculates the throughput (utilized bandwidth) as the exponential weighted average of the throughput over a period. The reason that the latest measurement is not taken as the actual value of the throughput is to avoid spikes that might not represent the actual state of the link by means of throughput between the network assets. Older values have less weight in the calculation of the mean value.

$$th_{\langle a_i, a_j \rangle}(T_k) \quad (2)$$

$$= g * \sum_{n=N}^{n=1} (1 - g)^{n-1} \frac{[t_{x_b[i \rightarrow j]}(T_{k-n+1}) - t_{x_b[i \rightarrow j]}(T_{k-n})] + [r_{x_b[i \rightarrow j]}(T_{k-n+1}) - r_{x_b[i \rightarrow j]}(T_{k-n})]}{T_{k-n+1} - T_{k-n}}$$

Equation (3) describes the available bandwidth as the deference of the maximum bandwidth (capacity of the link) and the throughput (utilized bandwidth).

$$ab_{\langle a_i, a_j \rangle}(T_k) = b_{\langle a_i, a_j \rangle}(T_k) - th_{\langle a_i, a_j \rangle}(T_{k-N}, T_k) \quad (3)$$

Equation (4) quantifies jitter as the average of the first discrete derivative of the latency. Jitter expresses whether the latency of the link will frequently deviate from the mean value or not.

$$jt_{<a_i,a_j>}(T_k) = \sum_{n=N}^{n=1} \frac{|lt_{<a_i,a_j>}(T_{k-n+1}) - lt_{<a_i,a_j>}(T_{k-n})|}{N-1} \quad (4)$$

Below are presented the functions that map sets to values.

Function (5) assigns the bandwidth of a full communication path as the minimum bandwidth between the underlying links.

$$B(p_{h_i, h_j}) = \min_{\forall <a_i,a_j> \in p_{h_i, h_j}} b_{<a_i,a_j>} \quad (5)$$

Function (6) assigns the available bandwidth of a full communication path as the minimum available bandwidth between the underlying links.

$$AB(p_{h_i, h_j}) = \min_{\forall <a_i,a_j> \in p_{h_i, h_j}} ab_{<a_i,a_j>} \quad (6)$$

Function (7) assigns the jitter of a full communication path as the maximum jitter between the underlying links.

$$JTR(p_{h_i, h_j}) = \max_{\forall <a_i,a_j> \in p_{h_i, h_j}} jtr_{<a_i,a_j>} \quad (7)$$

Function (8) assigns the packet loss percentage of a full communication path as the difference of the probability of a packet to pass through n links from the probability of the sure event. Since the probability of a packet to pass from a link to another is an independent event then the probability (packet loss percentage) can be calculated as follows.

$$PL(p_{h_i, h_j}) = 1 - \prod_{\forall <a_i,a_j> \in p_{h_i, h_j}} (1 - pl_{<a_i,a_j>}) \quad (8)$$

Function (9) assigns the latency of a full communication path as the sum of the latency of each underlying link.

$$LT(p_{h_i, h_j}) = \sum_{\forall <a_i,a_j> \in p_{h_i, h_j}} lt_{<a_i,a_j>}(T_k) \quad (9)$$

Function (10) assigns the security risk of a full communication path as the weighted mean of the security risk of each. Specifically, the risk values are six [1, 6] the higher the risk the higher the value, hence each risk value has the same weight with itself. The weighted mean favours higher values and thus penalizes paths that contain links with high-risk values.

$$SR(p_{h_i, h_j}) = \frac{\sum_{\forall \langle a_i, a_j \rangle \in p_{h_i, h_j}} SR_{\langle a_i, a_j \rangle}(T_k)^2}{\sum_{\forall \langle a_i, a_j \rangle \in p_{h_i, h_j}} SR_{\langle a_i, a_j \rangle}(T_k)} \quad (10)$$

6.3.2.2.2 Objective functions and Constraints

In this section, the objective functions and the constraints will be presented. The objective functions are related to QoS and security aspects. All the objective functions are transformed to dimensionless by dividing the objective of the function by the constrain of the host. Additionally, a threshold value is introduced that works as a regularizer parameter that controls the degree of the resource over allocation. The lower the threshold the higher the over-allocation is permitted.

The goal of objective function (11) is to minimize the latency between the two hosts.

$$\min: J_{LT}(p) = \max \left(threshold, \frac{LT(p_{h_i, h_j})}{LT_c(h_i)} \right) \quad (11)$$

The objective function (12) has a negative sign, since the objective is to find the path that maximizes the available bandwidth. For this objective no threshold is used since the available bandwidth will be usually 10ths to 100s of times more than the required. Hence, whenever the remaining bandwidth is less than 10% than its capacity, death penalty is applied to the chromosome by assigning the highest possible value that is able to be produced by the specific machine.

$$\min: J_{AB}(p) = - \frac{AB(p_{h_i, h_j})}{\max((h_i))} \quad (12)$$

Objective function (13) should minimize the packet loss percentage.

$$\min: J_{PL}(p) = \max \left(threshold, \frac{PL(p_{h_i, h_j})}{PL_c(h_i)} \right) \quad (13)$$

Objective function (14) should minimize the jitter.

$$\min: J_{JTR}(p) = \max \left(threshold, \frac{JTR(p_{h_i, h_j})}{JTR_c(h_i)} \right) \quad (14)$$

Finally, objective function (15) is related to security and should minimize the security risk of the path. If the security requirement is not fulfilled then death penalty is applied to restrict using this path.

$$\min: J_{security}(p) = \frac{SR(p_{h_i, h_j})}{SR_c(h_i)} \quad (15)$$

$$s. t \left\{ \begin{array}{l} LT(p_{h_i, h_j}) \leq \min(LT_c(h_i), LT_c(h_j)) \\ AB(p_{h_i, h_j}) \geq \max(AB_c(h_i), AB_c(h_j)) \\ PL(p_{h_i, h_j}) \leq \min(PL_c(h_i), PL_c(h_j)) \\ JTR(p_{h_i, h_j}) \leq \min(JTR_c(h_i), JTR_c(h_j)) \\ SR(p_{h_i, h_j}) \leq \min(SR_c(h_i), SR_c(h_j)) \end{array} \right. \forall p_{h_i, h_j} \in (L + p - p')^2$$

6.3.2.2.3 Solver

In order to solve the aforementioned problem, genetic algorithms will be used. Specifically, the problem has been formed as a multi-objective constrained problem and the convergence to a near optimal solution should be fast. Hence, PaDe [56] multi-objective genetic algorithm is proposed as the solution since, it allows the parallel solving of multi-objective problems. For each chromosome in the population, PaDe decomposes a multi-objective problem into single-objective ones and using the asynchronous generalized island model [57] to distribute the solution process to multiple processors. At the end of the evolution, the population is set as the best individual in each single-objective island.

PaDe is not suitable to solve constrained problems; therefore, the problem should be transformed into a non-constrained one. A solution to this is to add an additional objective, which will quantify the violation of the constraints and will be used as a penalty to the overall objective function. The additional, objective function is the norm of the total constraint violation.

A constrain violation of type is calculated as (e.g. for latency and available bandwidth) :

$$C_vl_i = \max\left(0, LT(p_{h_i, h_j}) - \min(LT_c(h_i), LT_c(h_j))\right) \quad (16)$$

$$C_vl_i = \max\left(0, \max\left(AB_c(h_i), AB_c(h_j) - AB(p_{h_i, h_j})\right)\right) \quad (17)$$

Therefore, the additional objective function is formed as:

$$\min: J_{violation}(p) = \sqrt{\sum_{k \forall p_{h_i, h_j} \in (L+p-p')} C_vl_k^2} \quad (18)$$

Since, the problem is multi-objective, six dimensional specifically; there will be probably multiple solutions, each one being optimal at different objectives or combinations of objectives. After the final population of solutions has been calculated, the Pareto Front of solutions with respect to the function below is calculated:

$$\arg \min_p: \alpha J_{LT}(p) + \beta J_B(p) + \gamma J_{PL}(p) + \delta J_{JTR}(p) + \varepsilon J_{security}(p) + \zeta J_{violation} \quad (19)$$

Where $\alpha + \beta + \gamma + \delta + \varepsilon + \zeta = 1$. The values of $\alpha, \beta, \gamma, \delta, \varepsilon, \zeta$ define the user's preference.

² $(L + p - p')$ means that the old path (p') is removed from set L and the new path p is added

The solutions with the minimal value are chosen in order to construct the Pareto Front of solution with respect to the given policy. Finally, an additional filtering step takes place in order to select the value that utilizes the minimum number of the resource allocation.

6.3.2.2.4 Representation of the chromosomes

The representation of the chromosomes is a critical factor for the scalability of the model. The straightforward approach to represent the chromosome is the absolute one. A chromosome representation is called absolute such that length of the chromosome is equal to the number of the switches and the range of values that each genome of the chromosome can't take is also equal to the number of the switches. This approach allows the existence of each possible combination of a path including paths that are not sensible (a switch can be present multiple times) or paths that are not present considering the links of the network topology. This method overpopulates the genetic algorithm with solutions that are not feasible, and when a feasible solution is found the genetic is biased to produce solutions very similar to this one. The authors of [58] suggest a repairer method based on Dijkstra algorithm in order to repair the chromosome that led to unfeasible solution. The repair method based on Dijkstra has a complexity in worst case scenario equal to $O(|V|^3(M + \log|V|))$ where V is the total number of nodes and M is the total number of QoS objectives. Even though this technique increases the computational complexity, the chromosomes are always feasible paths and hence, decreases the required number of generations for convergence. It should be also noted that the repair method suggested in [58] repairs the chromosomes based on solutions of Dijkstra applied on each QoS requirement and thus the repaired chromosomes are not repaired in random but in an optimal way dictated by Dijkstra. In order to produce feasible chromosomes, a dynamic and relativistic approach with repair was introduced for EDAs. This method has the advantage that the additional computational complexity is $O(V * \log|V|)$, but the disadvantage is that the repair is random. An illustration of the naïve representation and EDAs representation can be seen in Figure 32.

The dynamic and relativistic representation with repair of the chromosome requires the introduction of a new variable, which will be referred as connectivity. Connectivity (C_i) measures the total number of the connected switches to switch i . Therefore, the range of each genome of the chromosome is equal to the amount of connectivity of the previous genome (i.e switch). This approach shrinks the search space of the hypervolume of the solution allowing the genetic examine more feasible solutions. The relativistic part of this representation is that in order to map the representation of the chromosome to the true representation of a path containing the actual values of the switches, requires to decode the value of a specific genome relative to its previous one. The dynamic part is that each genome does not always have the same range of possible values but it always changes with respect to C_i . Finally, whenever a mutation or cross-over takes place, the new chromosomes might produce solutions that are not feasible. Hence, a random repair mechanism takes place to repair the genomes that have values higher than its connectivity value.

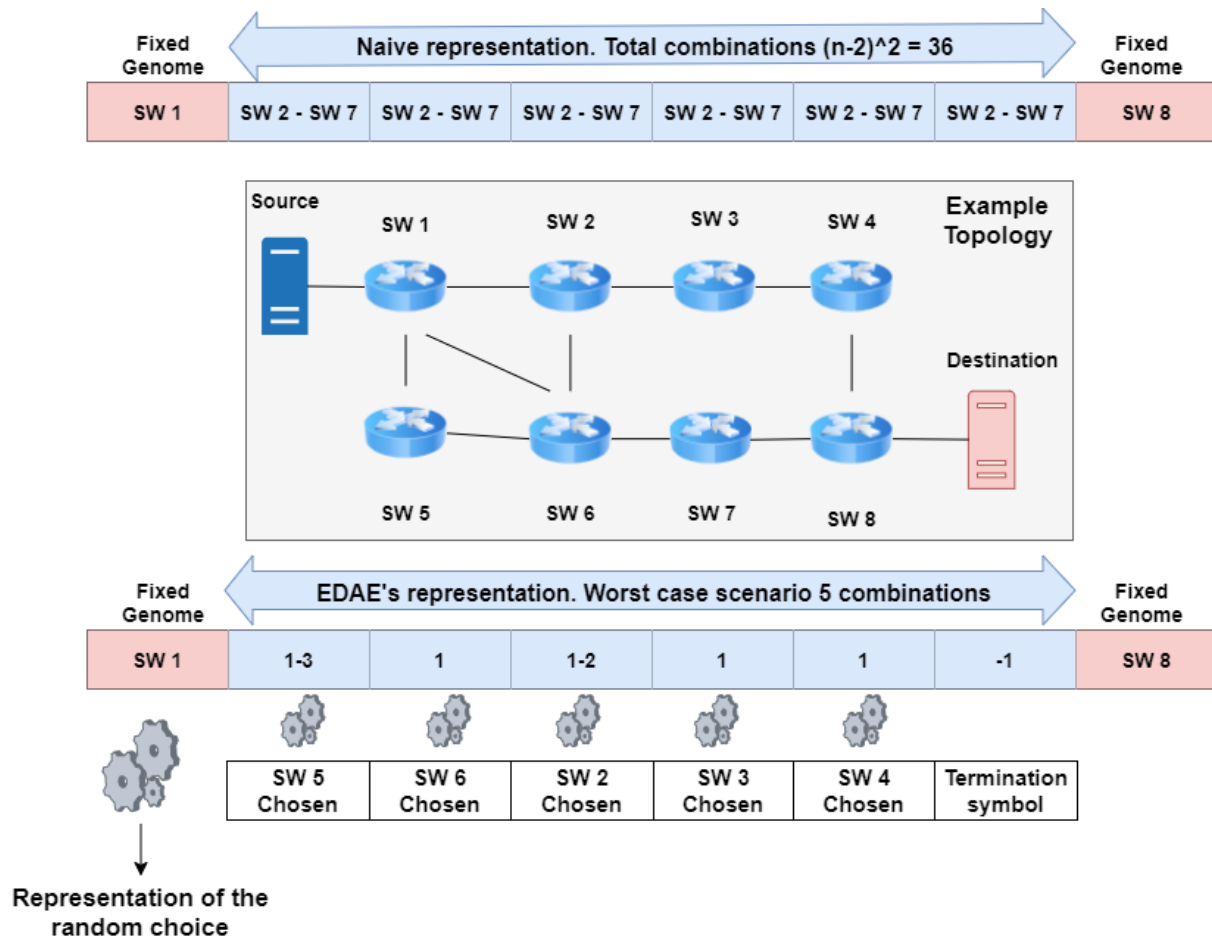


Figure 32: Illustration of the Naive (top) and EDAA's (Bottom) chromosome representation

6.3.2.3 PMU-PDC allocation for EPES observability

This section presents two MILP algorithms responsible for maximizing the observability of the EPES after a cyberattack or a malfunction on a PDC. The bandwidth constrained MILP and the PMU constrained MILP. The difference between each algorithm relies on the data transfer rate of each PMU. Normally, each PMU is configured to send data over a specific data transfer rate³ [64]. If a PMU modifies each data transfer rate in minutes, then the PMU constrained MILP algorithm will be selected, since it allocates one PMU to at most one PDC. In this way, we avoid any potential conflicts regarding the connection of a PMU to multiple PDCs, which will demand the rearrangement of the communication paths in a short period of time. In general, the bandwidth constrain MILP is modified in a Binary ILP in order to add the constrain of one PMU to send data to at most one PDC.

³ Almas, Muhammad Shoaib, and Luigi Vanfretti. "A method exploiting direct communication between phasor measurement units for power system wide-area protection and control algorithms." *MethodsX* 4 (2017): 346-359.

The MILP algorithms are applicable to every device that transfer current or voltage or active/reactive power, not only with PDCs and PMUs, since the observability of an EPES is related with the availability of electricity measurements from each bus of the EPES^{4 5}.

6.3.2.3.1 Bandwidth constrained MILP

The proposed MILP algorithm focuses on the matchmaking between PMUs and PDCs with constrains regarding the communication bandwidth of PMUs and PDCs. The constrains are chosen in accordance with the IEC/IEEE 60255-118-1 standard "Synchrophasor for power systems – Measurements" [59], with the IEEE C37.118.2-2011 [60], which both describe the functional requirements for a PMU and in accordance with the IEEE C37.247-2019 standard for Phasor Data Concentrators for Power Systems [61].

Let $A = \{PMU_1, PMU_2, \dots, PMU_N\}$, a set of N available PMUs to be connected with all the available PDCs. Let $B = \{PDC_1, PDC_2, \dots, PDC_M\}$, a set of M available PDCs to be connected with all the available PMUs, where $M \leq N$.

Furthermore, the Decision Variables of the model are defined as the data that can be transferred from all the available routes from PMUs to PDCs, therefore the Decision Variables can be expressed as:

$$x(a, b) \geq 0 \dots \forall a \in A, b \in B$$

$$x(a, b) \in Z \dots \forall a \in A, b \in B$$

The objective function of the problem is expressed as the minimization of the average communication network latency between the available PDCs and PMUs. In this case, the objective function is described as:

$$\min \sum_{a \in A, b \in B} \text{latency}(a, b) x(a, b)$$

The constrains of the MILP problem are the available bandwidth of each PMU and the available bandwidth of each PDC. Concretely, A PMU cannot connect to i+1 PDCs if the data rate exceeds its bandwidth capacity d_A .

$$\sum_{a \in A} x(a, b) \leq d_A \dots \forall b \in B$$

A PDC cannot accept data from j+1 PMUs if the data rate of the PMUs exceeds the bandwidth capacity, d_B of a PDC.

$$\sum_{b \in B} x(a, b) \leq d_B \dots \forall a \in A$$

⁴ Baldwin, Thomas L., et al. "Power system observability with minimal phasor measurement placement." IEEE Transactions on Power systems 8.2 (1993): 707-715.

⁵ Krumpholz, G. R., K. A. Clements, and P. W. Davis. "Power system observability: a practical algorithm using network topology." IEEE Transactions on Power Apparatus and Systems 4 (1980): 1534-1542.

6.3.2.3.2 PMU constrained MILP

To ensure the reliable operation of the EDAAE in case a PMU data transfer rate is shifting in minutes, the above MILP formulation is replaced by a binary ILP with the constrain of each PMU to send data to only one PDC.

Concretely, let $A = \{PMU_1, PMU_2, \dots, PMU_N\}$, a set of N available PMUs. Let $B = \{PDC_1, PDC_2, \dots, PDC_M\}$, a set of M available PDCs.

The decision variables are the sets of PMU-PDC pairs $\{a, b\} \forall a \in A, \forall b \in B$ with binary values:

$$\{a, b\} = 1, \quad \text{if PMU } a \text{ sends electrical data to PDC } b$$

Or

$$\{a, b\} = 0, \quad \text{if PMU } a \text{ and PDC } b \text{ are not connected}$$

The objective function of the problem is to select the PMU-PDC pairs which minimize the average communication network latency. In this case, the objective function is described as:

$$\min \sum_{a \in A, b \in B} \text{latency}(a, b) \{a, b\}$$

A PDC cannot accept data from $j+1$ PMUs if the connected PMUs exceed the amount k_B of PMUs that a PDC can host.

$$\sum_{a \in A} \{a, b\} \leq k_B \dots \forall b \in B$$

Finally, each PMU should send data to at most one PDC:

$$\sum_{b \in B} \{a, b\} \leq 1 \dots \forall a \in A$$

6.3.3 Problem Definition and Use Cases

The goal of EDAAE is to enhance the resiliency of Electrical Power Energy Systems (EPES) by adding a self-healing technique that leverages the network capabilities offered by SDN technology. Specifically, EDAAE should mitigate the impact of an attack, optimize the observability of the EPES infrastructure and optimize the QoS of the applications. EDAAE needs to enable its self-healing mechanism whenever one of the three following scenarios takes place.

Scenario 1) QoS of monitoring equipment is not sufficient

The necessity of monitoring the electrical network is of high criticality, since it allows other applications or specialized personnel on decision-making related to activities that affect the health of the equipment, the prevention of dangerous disastrous events and the economy of the whole EPES life cycle. Phasor Measurement Units (PMUs) and Phasor Data Concentrators (PDCs) are the main components responsible to monitor the underline electrical network of an EPES. Other components responsible for monitoring are the RTUs, PLCs and IDEs. All the aforementioned components interface the physical equipment of the electrical network and make it observable to the rest ecosystem

(applications/personnel). A strong requirement is that the measurements from the electrical grid should be broadcasted in the targeted applications in a specific time window. Therefore, QoS of monitoring instrument directly affects the observability in EPES.

EDAE as a self-healing mechanism should ensure that the QoS of the monitoring equipment is met. Whenever some of the QoS criteria is not met then EDAE should rapidly find an alternative communication path (if existent) such that the QoS are met and the security level of this path is sufficient with respect to the sensitivity of the equipment's data with respect to the QoS constraints of the rest monitoring devices.

Scenario 2) PDC is disconnected from the network

This scenario is related with scenario 1 since its objective is to maximize the observability of the EPES. In this scenario, a PDC is disconnected due to a malfunction or a cyber-attack from the network and the PMUs that were connected to the PDC should forward their measurements to another PDC if possible. Multiple PMUs can be connected to a PDC. PMUs are attached to a bus in the electrical grid and are able to take measurements from the bus and neighbouring buses. When a PDC is disconnected from the network, EDAE should assign the most PMUs that is possible (target PDCs have sufficient leftover bandwidth) that maximize the observability (observable buses) by finding communication paths that meet the QoS and security constraints of the PMU and does not affect the QoS constraints from the rest of the monitoring devices. In this scenario, the core algorithm of EDAE proposes the paths that do not violate QoS constraints to the MILP algorithm, and the MILP algorithm assigns the PMUs to the PDCs by minimizing the average latency of the paths between each PMU and PDC. Afterwards, the matchmaking is send back to the core algorithm and consequently, the core algorithm applies the new communication paths. It should be noted that the scenario is functional even without the usage of PDCs and PMUs. Any device that could transmit and receive electrical data (current,voltage,active/reactive power) can be handled. More details are presented in section 6.3.2.1.

Scenario 3) The security level of a network asset changes

S-RAF reports to EDAE whenever the risk assessment of a host is altered. Whenever the risk assessment of a host is increased, the risk assessment of the switches that forward information from this host is also changed. EDAE, checks if there are existing applications that require that their data should follow a path up to a certain level of risk and if this update of the risks violates those constraints. If so, then EDAE should find an alternative path for those applications such that the risk is less or equal to the specified risk level, the QoS are met (if present) and the QoS of the rest applications is intact. This scenario uses the same algorithm as scenario 1.

6.4 EDAE Dashboard

EDAE-Dashboard is a friendly user interface that has a triple functionality. First, the current state of the SDN network is monitored. Second, this tool provides a representation of the EDAE's proposals, i.e. the proposed topological changes derived from the EDAE algorithm. Finally, the user of the EDAE-Dashboard (the operator) is able to accept or not the proposals provided by the EDAE.

These functionalities are available through a simple and intuitive web interface that can be operated by the user. Figure 33 shows a graphic schema of the different views that conform the EDAE-Dashboard

user interface. As it was previously described both the current state of the network topology and the EDAE proposed changes are represented in two different views. On the left view, the current SDN network is shown, and additional information can be depicted depending on the user actions. Thus, in the traffic metrics pop-up, the user can obtain network performance information from the nodes (switches and hosts) just positioning the cursor over the desired node. Moreover, with a click over the node statistical information of the component is also shown in the switch statistics dialog box.

On the right view of the interface, the EDAE proposal is represented. In this case, the operator can accept or reject the proposed changes totally or partially, which means that the operator could accept only part of the EDAE proposal. To that end, a dialog box (the *proposed paths dialog box* in the figure) with the proposed changes appears when the user clicks on the host icon. These proposals are a list of the proposed connections/path between the selected host and the others in the network. The user can accept or reject each change using an available button inside the dialog box.

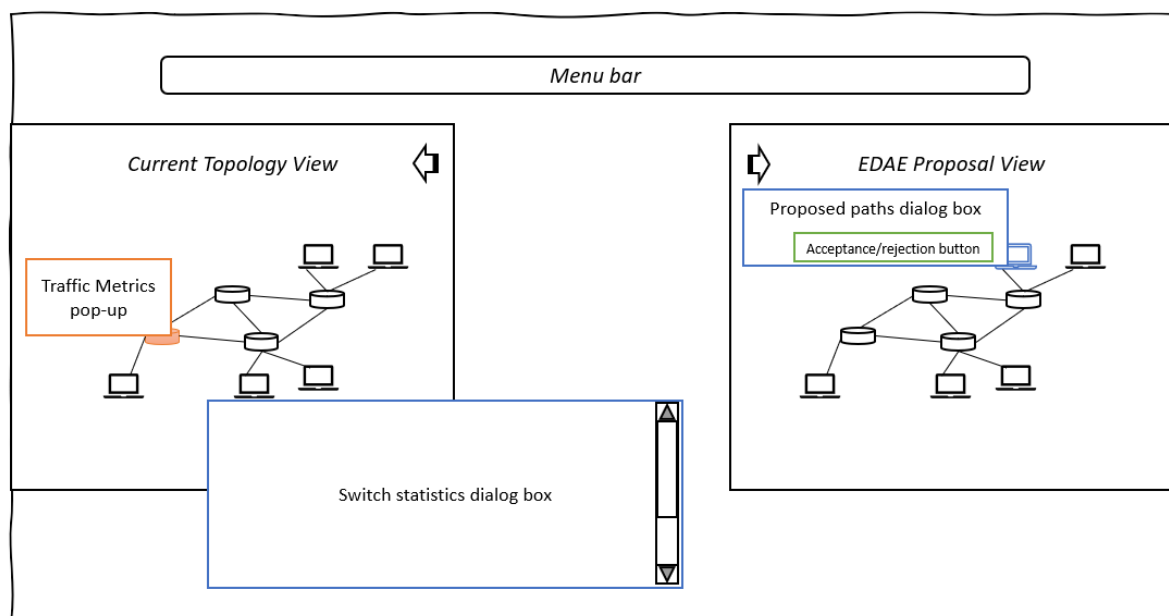


Figure 33: EDAE-Dashboard user interface schema.

The development of the EDAE-Dashboard has been divided in two parts or tools, the back-end and the front-end, which is a common practice in web design. The back-end tool is in charge of extracting the necessary data (statistics, topology and proposed changes) from other parts of the self-healing system (i.e. the SDN-C, AIDB and EDAE) and process it in order to obtain valuable information, the network performance metric in this case. Additionally, all the retrieved statistics and the calculated metrics are stored in databases. On the other hand, the front-end tool is responsible of representing all the information. To that end, the databases are consulted.

After this first review of the EDAE-Dashboard functionality and web interface, a deeper explanation of its architecture and used technology is described in the following subsections.

6.4.1 Architecture and functionality

As commented before, the EDAA-Dashboard functionality can be divided in three parts: the monitoring of the current network, the representation of the EDAA proposal and the acceptance of the EDAA proposal. Each functionality and its architecture are described separately in the following.

a) Current network topology

With the aim to represent the status of the SDN network, the current topology of the network and a set of stats and metrics are showed. On the one hand, the topological information is retrieved from the AIDB, using the API services available to that end (see the Interfaces section below). On the other hand, the statistic information of the SDN network is obtained from the SDN Controller through the available NBI. A python application, from now on *python adapter*, has been developed to request the previous information periodically.

The python adapter is also in charge of calculating some network performance metrics, which provides valuable information of the current status of the network traffic, based on previous obtained statistic information. Specifically, the necessary stats are the ones related to the flow entries and ports information for each SDN switch in the network. A list of the calculated performance metrics is presented below:

- *Current port throughput [Mbps]*: is the bits per second consumed by each port of a switch at the calculation moment.
- *Current switch throughput [Mbps]*: is the bits per second consumed by a switch at the calculation moment.
- *Current end-to-end throughput [Mbps]*: is the bits per second consumed by a path between two end points (hosts) of the network at the calculation moment.

The python adapter has another functionality that consists in storing the retrieved information of the network along with the calculated performance metrics. This storage is divided into two categories. Firstly, a backup of the statistic and network performance metrics is stored in a MongoDB. To ease the management of the historical information, this data is saved only for 30 days. The idea is to create an accurate historical record that can be consulted on demand by the users. Secondly, the updated data, i.e. the last obtained data from the SDN-C and the last values of the performance metrics, is stored in a Redis database. Redis database is selected for this end since it is used as a database cache, which means faster speed queries.

The schema in Figure 34 shows the architecture of the data acquisition procedure from the SDN-C and the AIDB, and the information storage in both internal databases. This storage is done by the python adapter, which is continuously running in order to capture the statistics (provided by the SDN-C) and to calculate the network performance metrics. To simplify the schema, the interfaces have not been represented, but they can be consulted in the interface section below.

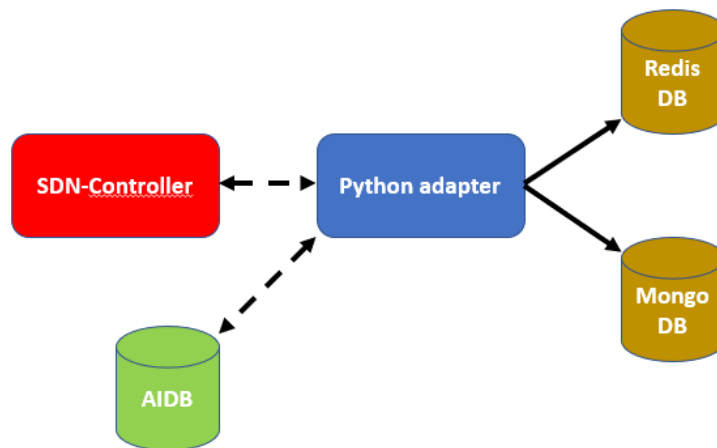


Figure 34: Architecture of the data acquisition and storage

The front-end tool is connected to both internal databases (MongoDB and Redis) in order to consult and represent the topology, statistics and network performance metrics. In Figure 32, it is represented a simplification of the workflow that permits the representation of the actual network state by the front-end. It must be clarified that the Redis database is consulted to represent the latest network status, while the MongoDB will be consulted to retrieve historical information if it is required by the operator. To that end, an API has been developed in order to consume the data from both databases, but that API is omitted in Figure 35 to simplify the representation.

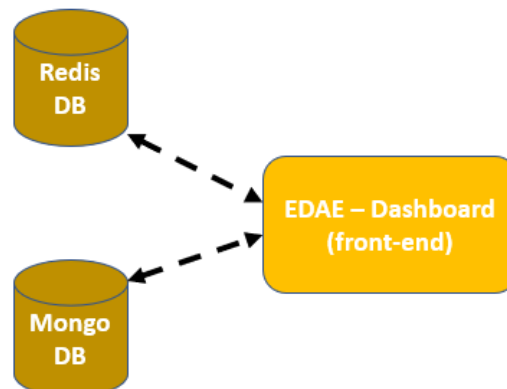


Figure 35: Architecture of the front-end tool

b) EDAE proposal representation

Apart from the current status of the network, the EDAE-Dashboard is configured to represent the EDAE proposals, which can be seen as the principal functionality of this dashboard. To that end, the communication between EDAE tool and EDAE-Dashboard is mandatory. This communication is set through RabbitMQ (AMQP protocol), which works as a message-queueing software. Additionally, the approval signal of the EDAE proposals is send back to the EDAE with another communication in the inverse direction. This one is also established through RabbitMQ.

Figure 36 shows the workflow regarding the EDAE proposal representation, from the event-generation by the EDAE-tool proposal until the storage of the proposal into the Redis and Mongo databases. The way of working is: EDAE tool provides an event, which is queued through RabbitMQ, this one is

captured by a python application, from now *event listener*, and finally the proposal message is stored in both databases.

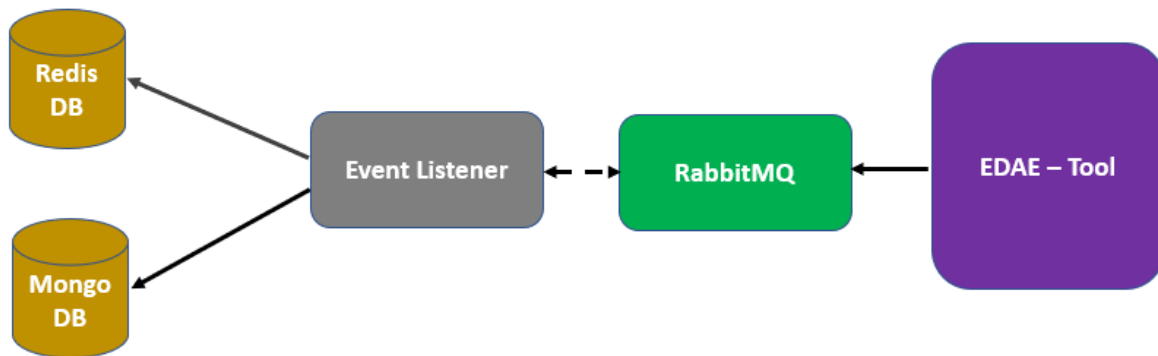


Figure 36: Architecture of the EDAE proposal representation

c) Acceptance of the EDAE proposal

As part of the user interface, a functionality to accept or reject the proposal by the EDAE tool is developed. This one corresponds to the *Acceptance/Rejection button* (see Figure 30). In case of being accepted, the EDAE will act accordingly sending the required control actions to the SDN Controller. After the application of the accepted proposals, the network status (the current network view) will be updated by the python adapter. It must be notice that this functionality will be only available is the EDAE proposal requires approval (see the description in Section 6.3.2).

In this case the workflow is as follows. The acceptance is sent to EDAE tool throughout RabbitMQ. The event listener also captures the changes in the RedisDB, these changes can be the acceptance or not of the EDAE proposal by the user of the EDAE dashboard. The event listener queues the message (accept or reject the EDAE proposal) in the queue. Then, the EDAE-tool captures the message and this one provides the change, or not, to SDN-C.

In Figure 37 it is shown the workflow from the moment the Event Listener detects the changes in a specific field of the RedisDB, which was added for that end, until the EDAE tool captures that change and the control actions are sent to the SDN-C.

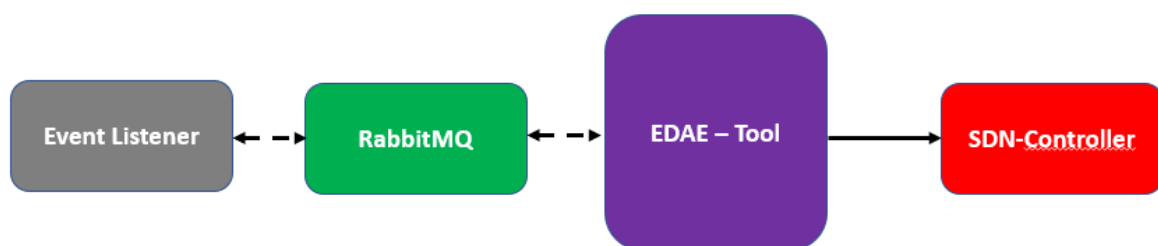


Figure 37: Architecture of the EDAE proposal acceptance

6.4.2 Interfaces

In order to provide the functionalities described in the previous section, the EDAE – Dashboard interfaces with other tools of the self-healing system to retrieve and send the necessary information (see Figure 7). These interfaces are described in the following. However, some of them has been

already defined in their specific sections in this document, so a simple description with the reference to the corresponding section will be exposed.

6.4.2.1 EDAE-Dashboard – EDAE: Network topology changes

The representation of the EDAE proposals is done on the basis of the information retrieved from this interface. The AMQP protocol (RabbitMQ) is employed and the EDAE send the topology proposal messages that is inserted in the network topology changes queue. This message contains the information described in Table 22: Network topology changes message data.

Table 22: Network topology changes message data

Field	Type	Description
PROPOSAL_ID	Int	Identifier of the proposal.
DATA:	Dict	Information that describes the topological changes proposed by EDAE.
APPROVAL_REQUIRED	True/False	If true, the proposal requires the acceptance of the EDAE-Dashboard operator and the EDAE will wait for an acceptance/rejection signal. If false, no approval is required by EDAE. In this case, the acceptance/rejection functionality is not available in the EDAE-Dashboard.
PMU	List of dicts	List of changes in terms of connections proposed of PMUs with PDCs. This list contains specific information of each proposed connection: <ul style="list-style-type: none"> ID: identifier of the PMU PDC: list with the IDs of the PDCs that the PMU is sending data to. traffic_demand: minimum traffic demand of the PMU.
PATHS	List of dicts	Set of proposed new paths between pairs of edge nodes (PMUs and PDCs). These paths are given as a list of IDs of the nodes that compounds the path.

6.4.2.2 EDAE-Dashboard – EDAE: Network topology proposal acceptance

Once the EDAE proposal is represented, the operator can accept or reject them if the approval is required by EDAE. In that case, a proposal acceptance message is sent by the EDAE-Dashboard using the proposal acceptance queue. This message consists on the information described in Table 23.

Table 23: Network topology proposal acceptance message data

Field	Type	Description
PROPOSAL_ID	Int	Identifier of the proposal.
APPROVAL	True/False	If true, the operator has accepted the changes proposed by EDAE and the EDAE will act in accordance sending the required actions to the SDN Controller. If false, the operator has rejected the changes proposed by EDAE.

6.4.2.3 EDAE-Dashboard – AIDB

The API services of the AIDB, which will be described in Section 7, are employed to get current topological information of the SDN network as well as additional information managed by EDAE that cannot be retrieved from the SDN Controller, such as the link weights. Specifically, the *AssetQuery()* method is used to obtain the required information (see Section 7.3 for a detailed explanation of the method).

6.4.2.4 EDAE-Dashboard – SDN Controller

The representation of additional statistical information for each switch is another functionality provided by the EDAE-Dashboard. This information must be retrieved from the SDN Controller through its corresponding Northbound Interface, which was described in Section 5.1. This interface provides two groups of API services, *ofctl_rest* and *rest_topology*, from which the following endpoints are used in this interface:

- Get port description.
- Get port statistics.
- Get table stats.
- Get flow stats (all).

A description of the output information that can be retrieved with the above endpoints is presented in the corresponding sections.

6.5 Component Model

From implementation point of view, some tasks to be done has been determined in order to satisfy the requirements.

These tasks are independent processes implemented in **python** with its unitary tests. Some of these tasks are dedicated to get information from external APIs and transform data in a friendlier format for internal purposes. Some other are dedicated to save the tracking information in our own database. Moreover, the most important are those that apply the required functionality, the main purpose for the whole workflow.

To put all together has been decided to use an open source software called **Apache Airflow** to act as orchestrator. Using this software, a workflow with those individual tasks can be built and then schedule the whole workflow, manage globally and individually all the logs, errors, email notifications ... that are generated during the execution.

Airflow uses a database to manage itself. From the set of possibilities Airflow proposes, **PostgreSQL** has been chosen and also was decided to have a separated server so it could be used from everywhere in the workflow.

Although Airflow have a great tracking system, some more information is stored in the PostgreSQL database. This information stored relates to the tasks and it is valuable for tracking, to reproduce certain situations in order to debug them and also it's needed from the dashboard to show the aggregate information useful for the "decision making" process.

This is the workflow representation with its tasks and the way how they interact with each other (image from Airflow):

1. **Main workflow:** from inputs to notify proposal or implement changes if it needs to be approved or not respectively.
2. **Approval Result workflow:** from EDAE-Dashboard response to implement changes or do nothing if the proposal was approved or not respectively.



Figure 38: Airflow workflow

In terms of deployment, **Docker** is used for the development process so it lets to simulate the different servers that are supposed to exist in the real implementation.

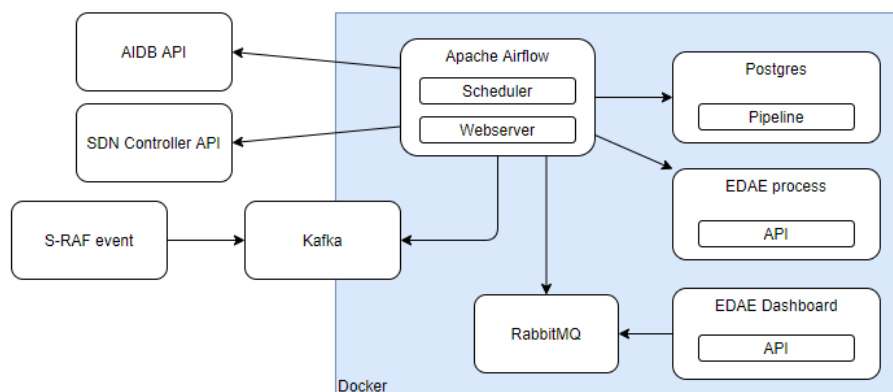


Figure 39: Docker containers and APIs

Several Docker containers are going to be used for this purpose:

- **Apache Airflow server:**
Created from **Airflow** Docker image and having shared local folders for logs and dags.
- **PostgreSQL server:**
Created from **PostgreSQL 11** Docker image with Pipeline extension (for time series) installed.
It's going to be used by:
 - o Airflow to manage itself.

- o Workflow on itself or tasks inside the workflow to save inputs, outputs, execution times, errors, data transportation from one task to another (for large data exchanging).
 - o EDAE process as a time series database to get some statistics from historical information stored previously.
- *EDAE process:*
Created from a **python 3.7** image.
It will contain the main EDAE process and a rest API to expose it.
 - *Kafka:*
This will be the communication mechanism between S-RAF and EDAE workflow. So, EDAE workflow will receive information from S-RAF about incidences as a starting point of our process who is in charge to give a response.
 - *RabbitMQ:*
This will be the communication mechanism between EDAE workflow and EDAE-Dashboard. The proposal to be approved will go in one direction and the result of this evaluation should go in the opposite direction in order to manage the actions to implement within the proposal.
 - *APIs:*
While the original (or a draft of it) can't be called, a dummy API is going to be used. A Docker container will contain our own rest API returning a json example for each API.

Each of these containers is supposed to be a different machine/server in production environment, so we decided to set an IP for each one in order to do the call to its resources as much similar to the real world as possible. Dummy APIs are still useful in production environment in order to run acceptance tests. Figure 34 shows the topology of the ecosystem created in the virtual machine for that matter:

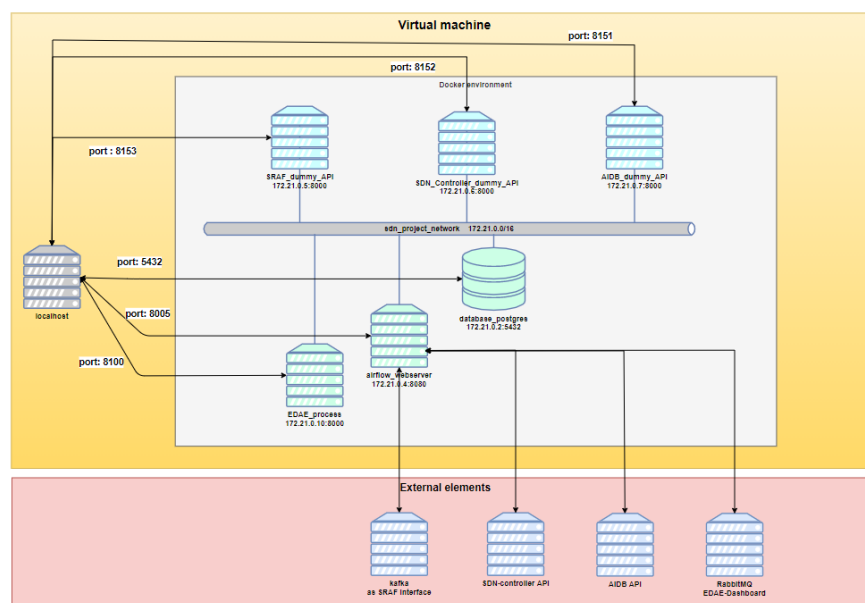


Figure 40: Different machine/server in EDAE product environment

6.6 Interfaces Model

For the proper operation of EDAE is needed at least four interfaces for the different components: S-RAF, SDN-C, AIDB and EDAE-Dashboard. In the following lines, it is described the relation between them according Figure 7:

1) S-RAF and EDAE

This interface has to provide to EDAE the information about the risk level of the assets of the network topology. S-RAF will output a set of .json files (Section 13.1) that will be read by the interface S-RAF/EDAE and it will provide another .json file to EDAE with the information needed for the proper operation.

From the set of .json files detailed in the annex, the relevant information for EDAE is highlighted in Figure 41:



Figure 41: Relevant information for EDAE operation from S-RAF

2) AIDB and EDAE

This interface has to provide to EDAE the summary of the network topology configuration. The AIDB will output a .json file that will be read by the interface AIDB/EDAE and it will provide some .json files to EDAE with the information needed for the proper operation.

An example of the following configuration is prepared for EDAE (Figure 42):

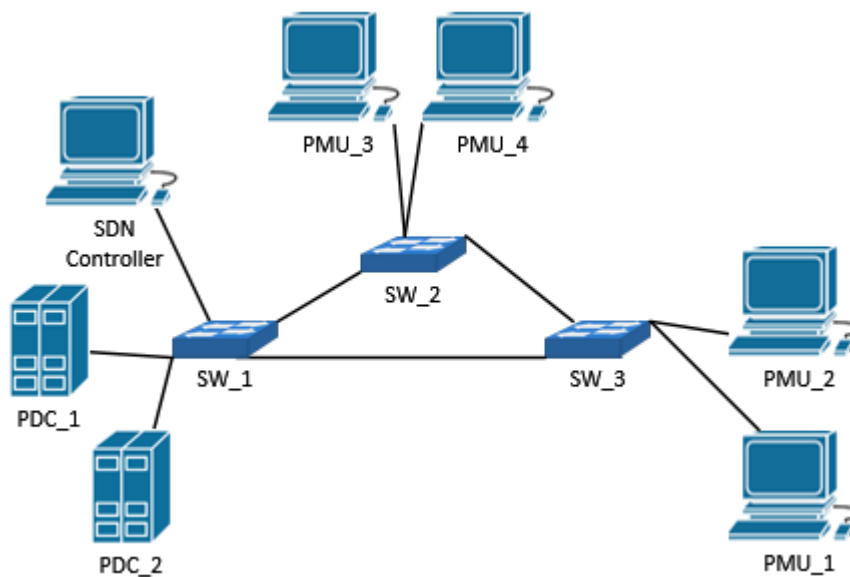


Figure 42: Network topology selected to test EDAE

From the .json detailed in (13.1) the following information can be obtained:

PDC_input.json

```
[
{
  "ID": "PDC_1", // ID of the requested PDC
  "ipv4": "10.0.0.1", // Ip address of Internet Protocol Version 4
  "MAX_PMUS": 20, // Maximum number of PMUs that can be connected to the PDC
  "PMUS_con": 2, // Number of connected PMUs
  "PMUS": ["PMU_1", "PMU_2"] // A list with the IDs of the PMUs that are connected to the PDC
},
{
  "ID": "PDC_2", // ID of the requested PDC
  "ipv4": "10.0.0.2", // Ip address of Internet Protocol Version 4
  "MAX_PMUS": 30, // Maximum number of PMUs that can be connected to the PDC
  "PMUS_con": 2, // Number of connected PMUs
  "PMUS": ["PMU_3", "PMU_4"] // A list with the IDs of the PMUs that are connected to the PDC
}
]
```

PMU_input.json

```
[
{
  "ID": "PMU_1", // ID of the requested PMU
  "PDC": ["PDC_1"] // A list with IDs of the PDCs that the PMU is sending data to
  "traffic_demand": "10 Mbps" // Minimum traffic demand of the PMU
},
{
  "ID": "PMU_2", // ID of the requested PMU

```

```

"PDC": ["PDC_1"], // A list with IDs of the PDCs that the PMU is sending data to
"traffic_demand": "20 Mbps" // Minimum traffic demand of the PMU
},
{
  "ID": "PMU_3", // ID of the requested PMU
  "PDC": ["PDC_2"] // A list with IDs of the PDCs that the PMU is sending data to
  "traffic_demand": "10 Mbps" // Minimum traffic demand of the PMU
},
{
  "ID": "PMU_4", // ID of the requested PMU
  "PDC": ["PDC_2"], // A list with IDs of the PDCs that the PMU is sending data to
  "traffic_demand": "20 Mbps" // Minimum traffic demand of the PMU
}
]

```

3) SDN- Controller and EDAE

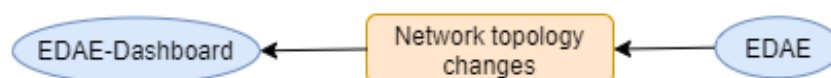
This have been already implemented by us using the restful API presented in the link below and transforming the input accordingly.

https://ryu.readthedocs.io/en/latest/app/ofctl_rest.html

4) EDAE and EDAE Dashboard

In this case, two independent interfaces are implemented between EDAE and EDAE – Dashboard applying in both cases the AMQP protocol (using RabbitMQ), which permits to send messages from one tool to the other using two separate queues. These messages are .json files containing the information, as it was described in Section 6.3.2.

The first interface permits to send the topology changes proposed by EDAE to the EDAE-Dashboard. Thus, the topology proposal message send from EDAE is a .json file with the information required by the dashboard to represent the EDAE proposals:



```

{
  "PROPOSAL_ID":53427606,
  "DATA": {
    "APPROVAL_REQUIRED":true,
    "PDC": [
      {
        "ID": "PDC_1",

```

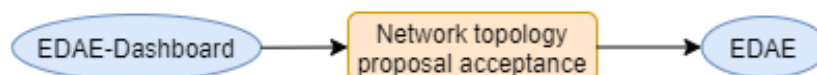


```

    "PMUS_con": 2,
    "PMUS": ["PMU_1", "PMU_4"]
  },
  {
    "ID": "PDC_3",
    "PMUS_con": 3,
    "PMUS": ["PMU_1", "PMU_2", "PMU_4"]
  }
],
"PMU": [
  {
    "ID": "PMU_1",
    "PDC": ["PDC_1", "PDC_13"],
    "traffic_demand": "10 Mbps"
  },
  {
    "ID": "PMU_2",
    "PDC": ["PDC_12"],
    "traffic_demand": "20 Mbps"
  }
]
}
[
  {
    "path": ["PMU_1", "SW_3", "SW_2", "SW_1", "PDC_2"]
  },
  {
    "path": ["PMU_4", "SW_2", "SW_1", "PDC_1"]
  }
]
]

```

The second interface allows the EDAE-Dashboard to send the approval signal (the acceptance or rejection of the EDAE proposals from the operator) to the EDAE, when this approval is required. Thus, the proposal acceptance message send by the dashboard is a .json file like this one:



```

{
  "PROPOSAL_ID":53427606,
  "APPROVAL": "True"
}

```

7 EDAE Core Engine Evaluation

This section presents a series of experiments performed to evaluate the performance of EDAE core engine by means of computational complexity and the quality of solutions. The section consists of five subsections with the following context. Subsections 7.1 will describe the evaluation framework and methodology that was followed in order to evaluate the rerouting functionality of EDAE. Subsection 7.2 presents the results followed by a discussion upon them. Respectively, subsections 7.3 and 7.4 present the evaluation framework and the results with respect to the MILP re-allocation functionality. Finally, in subsection 7.5 conclusions and open points will be discussed.

7.1 Evaluation framework of re-routing functionality

The performance of an algorithm depends on the specific scenario in which it is executed. Therefore, the evaluation framework is based on two network topologies, One Ring Bottleneck (ORB) and Two Ring Bottleneck (TRB), that are common in and representative of data center, metro, grid, and enterprise networks [62]. Additionally, the aforementioned topologies are also suitable for the evaluation framework since they are characterized by a high number of connected hosts, which means that multiple QoS requirements shall be satisfied with limited hardware resources. Details will be presented in subsection 7.1.1.

Besides of the structure of the network topology, another critical factor is the size of the topology. Therefore, experiments are conducted for various sizes of the two network topologies in order to assess the scalability capability of the re-routing functionality. Details will be presented in subsection 7.1.2.

Another important aspect that should be taken into account during the design of the evaluation framework is the mixture of the type of hosts and the applications that are aided for. In this evaluation framework the hosts reflect devices that are common in an EPES infrastructure, such as PMUs, SCADA, PLC, PDC, RTUs, and IEDs. Each host usually implements more than one functionality, hence the QoS requirements vary for each host with respect to the application that implements. Therefore, for each host, classes of QoS requirements, with respect to the underlying application that serve, were identified. Details will be presented in subsection 7.1.3.

The modelling of the status of the network (i.e., delay, bandwidth, and packet loss) is a parameter that should be taken into account during the design of the evaluation framework. Details will be presented in subsection 7.1.4.

Finally, a comparison between EDAE's re-routing mathematical formulation of the problem with the one proposed in [63] takes place. Details regarding the modeling of [63] are presented in 7.1.5.

7.1.1 Structure of the network topologies

ORB topology is discussed in section 7.1.1.1 while TRB topology in section 7.1.1.2.

7.1.1.1 One Ring Bottleneck Topology (ORB)

As it observed from Figure 42, the ORB topology consists of a base ring of $m + 1$ switches. One and only one device is connected to the base ring and specifically to the $m + 1$ th switch. Each switch of the base of the ring is connected to a column that connects n hosts. In order to allow the communication between two hosts that are present in different columns the information should pass from the base ring switches, hence the communication is bottlenecked by the base ring.

The parameters that define the size of the topology are the width (m) and the height (h). The total number of switches in ORB topology is $m + 1 + n * m$ and the total number of connected devices is $n * m + 1$.

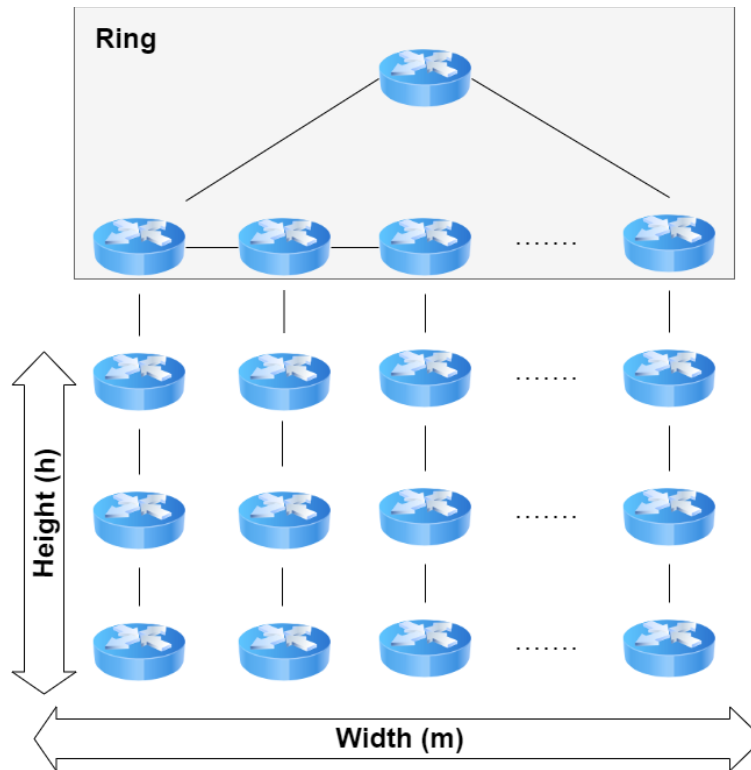


Figure 43: One Ring Bottleneck Topology

7.1.1.2 Two Ring Bottleneck Topology (TRB)

The TRB topology extends ORB topology with an additional ring consisting of $m + 1$ switches at the bottom of the ORB topology. This topology adds additional hardware resources and increases and the number of available re-routing solutions.

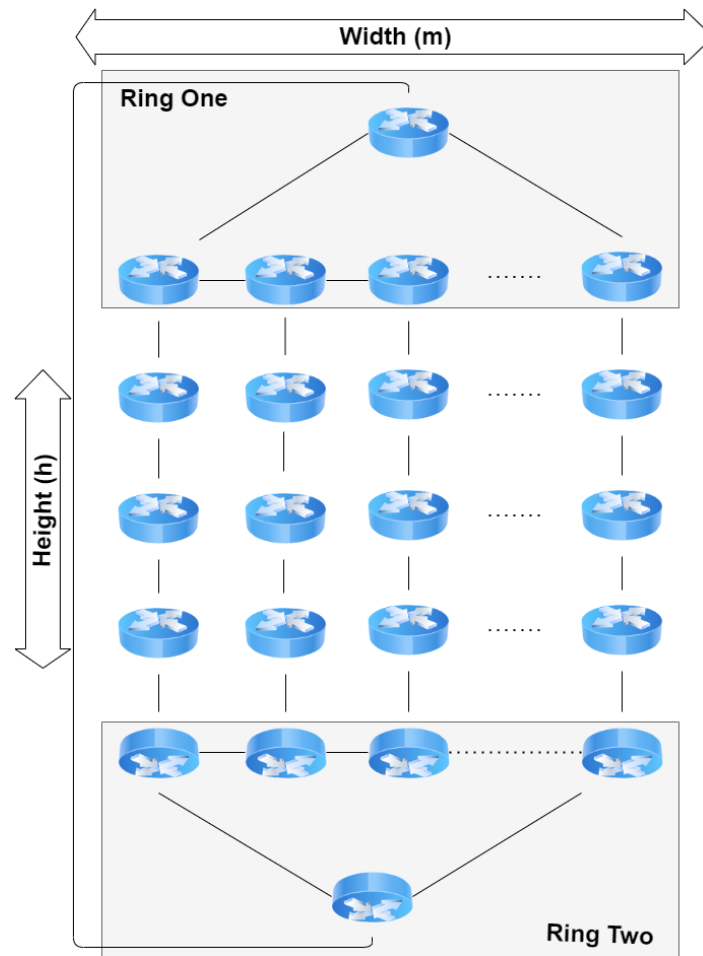


Figure 44: Two Ring Bottleneck Topology

7.1.2 Scale of the network topologies

The scale of the network topology is controlled by two parameters, the width (m) and height (h). For the evaluation infrastructure all the available combinations (49) between m and h were utilized, with both m and h varying from 4 to 10.

7.1.3 QoS requirements

The objective of this section is to present the rational and the assumptions that were made in order to extract the QoS requirements of the hosts. The QoS requirements for various applications are presented in section 7.1.3.1, while section 7.1.3.2 presents the modelling of the QoS requirements with respect to the evaluation framework.

7.1.3.1 Definition of QoS requirements

In order to make the evaluation framework specified for EPES applications, the types of the various hosts and their respective QoS requirements were selected based on sources that were found in the literature regarding the EPES QoS requirements. Specifically, table 22 presents a summary of the information gathered from the following sources [64] [65] [66] [67] [68]. Requirements regarding jitter in EPES related applications were not found in the literature. From [69] the maximum acceptable value for jitter regarding VoIP applications is set to 30ms while the delay to 150ms. Given that the quality of

VoIP applications is highly affected by jitter, the jitter requirements for all the applications bellow were specified as the 10 percent of the required latency.

Table 24 QoS requirements in applications related to an EPES

Asset Type	Application	Requirements				
		Bandwidth ⁶ (Mbps)	Delay (ms)	Jitter (ms)	Packet Loss (%)	Security (Categorical)
PMU	PMU to PDC data transfer	0.064, 0.088, 0.128, 0.176, 0.181, 0.362	20	2	99,99	High
	State Estimation	>>	1000	100	>>	>>
	Transient Stability	>>	100	10	>>	>>
	Small Signal Stability	>>	1000	100	>>	>>
	Voltage Stability	>>	1000	100	>>	>>
	Postmortem analysis	>>	N/A	N/A	>>	>>
AMI (Smart Meters)	Various functions	0.01 – 0.5	2000-15000	200-1500	99.99	High
Multiple	Demand Response	0.014 – 0.1	500	50	99.99	High
Multiple	Wide Area Situational Awareness	0.6 – 1.5	20-200	2-20	99.99	High
Multiple	Distribution Energy Resources and Storage	0.0096 – 0.056	20-15000	2-1500	99.99	High
Multiple	Electric Transportation	0.1	2000-15000	200-1500	99.99	Medium
Multiple	Distribution Grid Management	0.0096 – 0.1	100-2000	10-200	99.99	High
Distributed Agents	Multi Agent applications ⁷	0.01 – 0.1	50	5	99.99	High

7.1.3.2 Modelling of QoS requirements

Taking into consideration Table 22 that was presented in section 7.1.3.1, the most demanding applications were filtered and were used for the evaluation framework. As a second step, the assets of the EPES that will perform the aforementioned applications were selected. The filtered applications and the assets that will perform the applications are the following:

Table 25 Filtered applications and the required assets

Application	Source asset	Destination asset
-------------	--------------	-------------------

⁶ The required bandwidth of the PMU depends on the frames/sec ratio and the number of phasors. The values were taken from [68] and where multiplied by a factor of 2 in order to take into account the bandwidth overhead that would be introduced by encrypting the data.

⁷ The requirements regarding applications that are based on Multi Agent architectures were extracted from tools that will be used in the project, such as MAS tool.

PMU to PDC data transfer	PMU	PDC
Transient Stability	PMU	Application Servers
Wide Area Situational Awareness	PDC	SCADA
Distribution Energy Resources and Storage	Smart meters, RTUs, IEDs	SCADA, Agents
Distribution Grid Management	Smart meters, RTUs, IEDs	SCADA, Agents
Multi Agent applications	Agents	Agents, SCADA

After defining the applications and the assets that will be included in the evaluation framework, the next step is quantify the ratio of each specific class of the assets with respect to the total number the assets. It should be noted here that the SCADA for all the experiments is only 1 and is places at the tip of the ring. The following table presents the percentage of presence for each type of asset.

Table 26 Ratio of each type of asset with respect to the total number of assets

SCADA	PMU	PDC	IEDS	RTUs	Smart Meters	Application Servers	Agents
1	15%	2%	30%	30%	10%	3%	10%

Assumptions

The following assumptions, which are derived from table 23, apply in the evaluation framework:

- 1) All agents can communicate with the rest of the agents
- 2) The agents communicate only with agents and the SCADA
- 3) Smart meters, RTUs and IEDs communicate with SCADA and agents
- 4) PMUs communicate with application servers and PDCs
- 5) PDCs communicate with SCADA

Most of the QoS requirements, as shown in table 22, are not crisp values but lie in a closed range of values. Therefore, the QoS requirements were transformed to crisp values by uniformly sampling a number from the given range. It should be noted that the upper bound of their range for latency and jitter requirements was thresholded to 120 ms and 12ms respectively if it was higher than the aforementioned threshold. The reason that this threshold applied is to include in the evaluation framework only the most demanding applications.

7.1.4 Modelling the status of the network

This section presents the modelling of the status of each switch in the network. Based on [70] the typical delay value for SDN switch is 7.5μs and support duplex mode. The delay in our evaluation framework was set from 0.5ms up to 4ms. For the evaluation framework, delay, jitter, packet loss percentage and the security level of each switch was split into multiple categories and probabilities assigned on each category in order to sample a category. Delay and packet loss are represented as lists of ranges. After the category has been sampled the value is uniformly sampled from the selected range. Table 25 presents the categories and the distribution of the probabilities for each metric that describes the status of each switch.

Metric	Categories	Distribution of probabilities
Delay (ms)	[(0.5, 0.1), (1, 1.5), (1.5,2), (2,2.5), (2.5,3), (3., 3.5), (3.5, 4)]	Uniform
Jitter (int) ⁸	[2, 3, 4, 5]	[10%, 20%, 30%, 40%]
Packet loss (%)	[(0, 0.01), (0.01, 0.02), (0.02, 0.05), (0.05, 0.1)]	[30%, 20%, 30%, 20%]
Security (int)	[1, 2, 3]	[70%, 20%, 10%]

7.1.5 Comparison of EDAE's modelling with something

The work of [63] is the most recent attempt to solve the re-routing problem in order to produce paths that are characterized by low delay and packet loss values and in the same time they try to avoid traffic congestion to occur. In order to do so, the authors of [63], make use of genetic algorithms and specifically NSGA II [71]. The major differences between EDAE's mathematical formulation and the one presented in [63] are the following.

- 1) EDAE includes in its problem two additional objectives, jitter and security.
- 2) EDAE formulates the problem as a Multi Objective problem with constraints, while the aforementioned work is not constrained.
- 3) EDAE thresholds the optimal solution per objective, whenever the objective is over-achieved by a factor of 20% as shown in 6.3.2.2.2.
- 4) The final difference is that EDAEs uses a dynamic and relativistic representation with repair of the chromosomes while in the work of [63] the representation is absolute. The differences between the relativistic representation and the absolute are described in 6.3.2.2.2.

7.2 Results over the evaluation framework

This section presents the results for both EDAEs modelling and the one used in [63]. The aspects that are covered through the experiments are the computational time of each approach, the success ratio of the objectives and the capability to manage the available resources. The structure of the section is the following. Subsection 7.2.1 describes the evaluation details. Subsections 7.2.2 presents the results that cover the aspect of the computational time, while subsection 7.2.3 covers the results regarding the success ratio of the objectives and the resource management.

7.2.1 Evaluation details

For each topology, ORB and TRB, the values of width (m) and height (h) range from 4 to 10 which results in 49 combinations per topology structure. For each pair of (m, h) the experiment repeated 50 times. For each number of iteration and (m, h) pair a unique seed was created in order to ensure that both algorithms are tested on topologies with the exact same network status. The results regarding the execution time were concentrated and presented aggregated using mean, standard deviation, min and max values. The results regarding the QoS objectives are shown as box plots. Three box plots are produces for each topology configuration, the FAIL, DEV and Over.Alloc. For both the modelling approaches the same genetic algorithm with relativistic chromosome representation was utilized. Fail box plot shows the percentage of the applications for which the specific requirement is not met. The DEV box plot shows the percentage of the deviation from the specific requirement. Finally, Over.Alloc box plot shows the percentage that each specific requirement is covered. The values of over alloc that

⁸ The jitter is computed by dividing the delay metric with the integer that was sampled.

are closer to threshold of the objectives, which was set as 20%, are better since the paths do not over allocate resources to applications that don't demand it.

7.2.2 Computational time

This section presents statistics regarding the computational time required to construct a path for each given topology for both modelling approaches.

As it can be seen the performance for both the modelling approaches is almost identical. The modelling approach of [63] has slightly lower execution time, which is expected since the number of objectives is 3 without constraints, while EDAE's objectives are 5 with constraints and also because the filtering of the Pareto front solutions is not performed for the modelling approach of [63]. It can be seen that the average execution time and the standard deviation, increases linear with a slope near to 0.5, since for a topology with two times more nodes the increase in average and standard deviation of execution time is slightly less that two times more. As far as concerns the maximum values it can be seen that also the increase of executional time is linear with respect to the number of nodes but with a slope close to 3. Some spikes, and deviations from the linear increasing pattern are observed and are subject to the stochastic nature of the genetics algorithms. Finally, the aforementioned comments apply for both topologies.

ORB Topology:

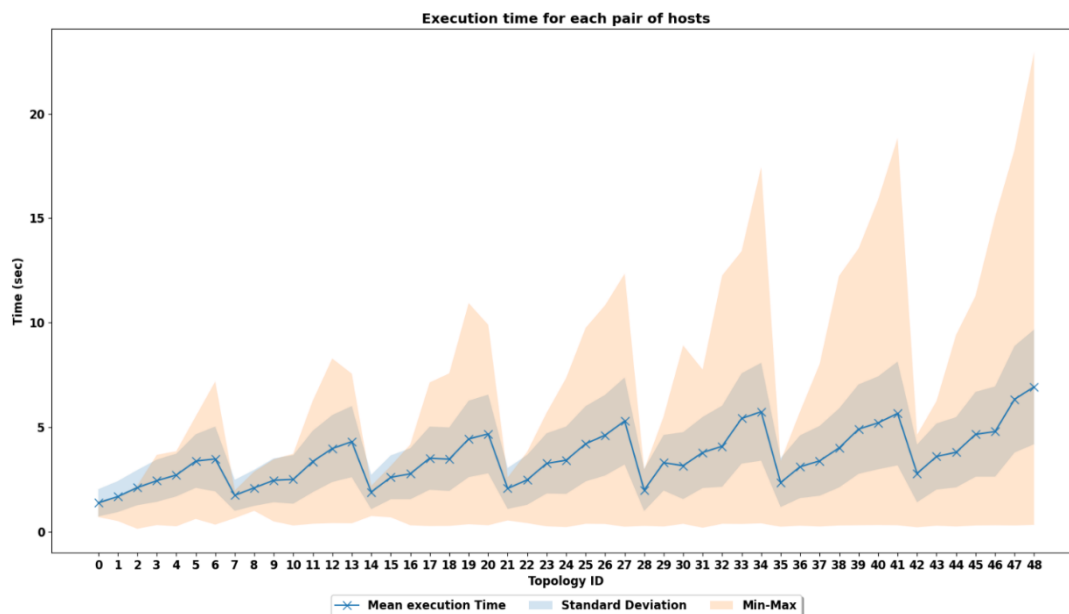


Figure 45: Execution time to find a path for a pair of hosts for a specific topology for EDAE (ORB)

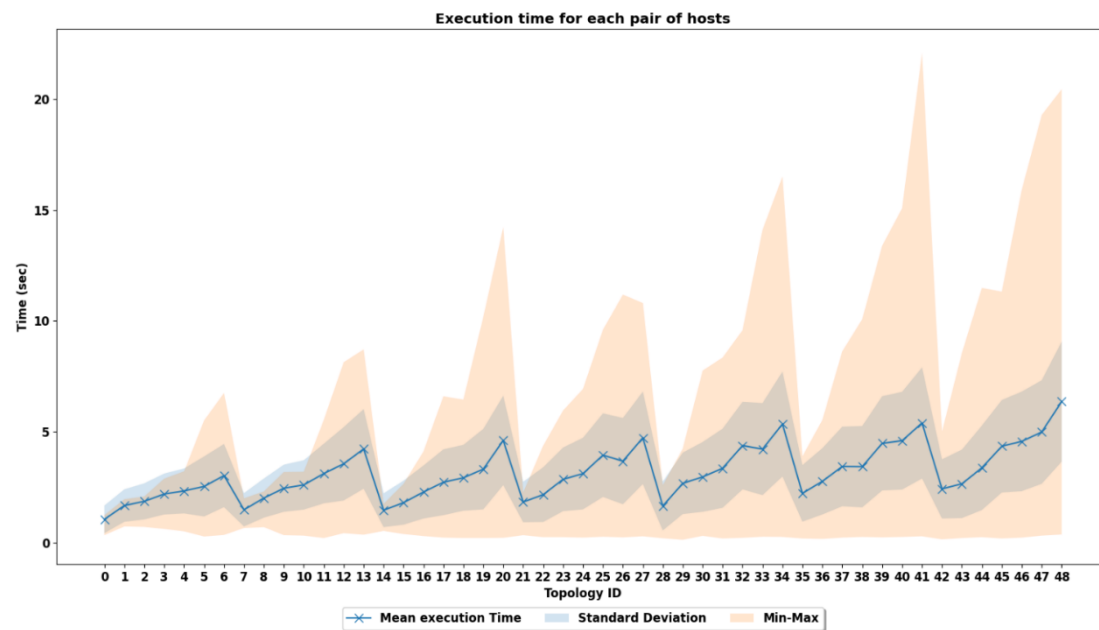


Figure 46: Execution time to find a path for a pair of hosts for a specific topology for [63] (ORB)

TRB Topology

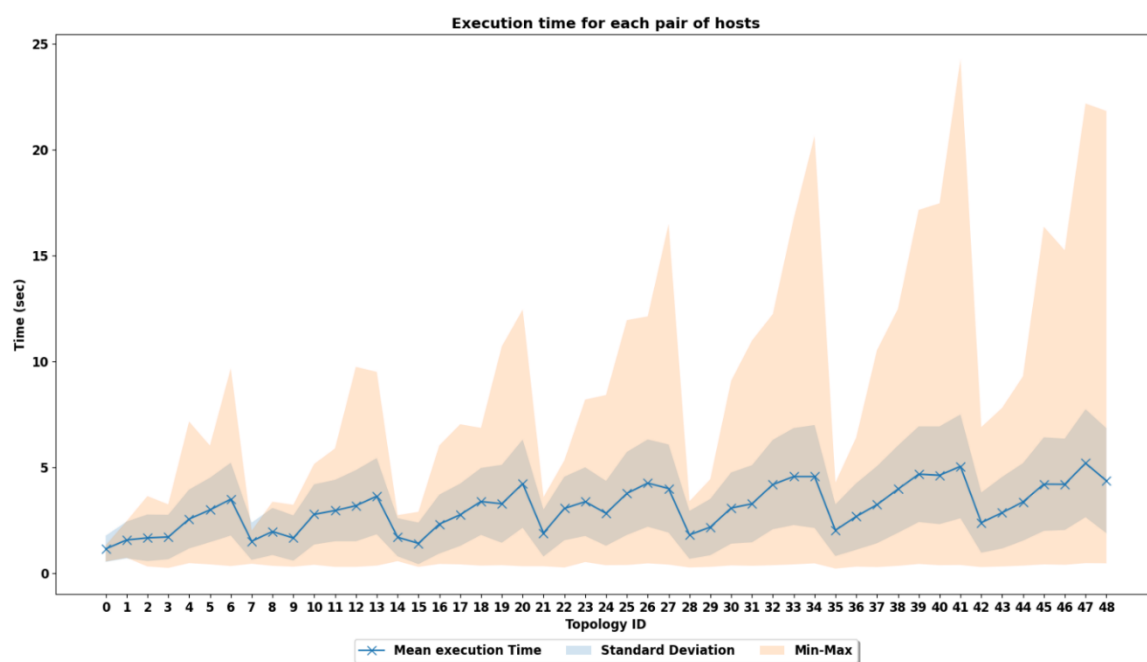


Figure 47: Execution time to find a path for a pair of hosts for a specific topology for EDAE (TRB)

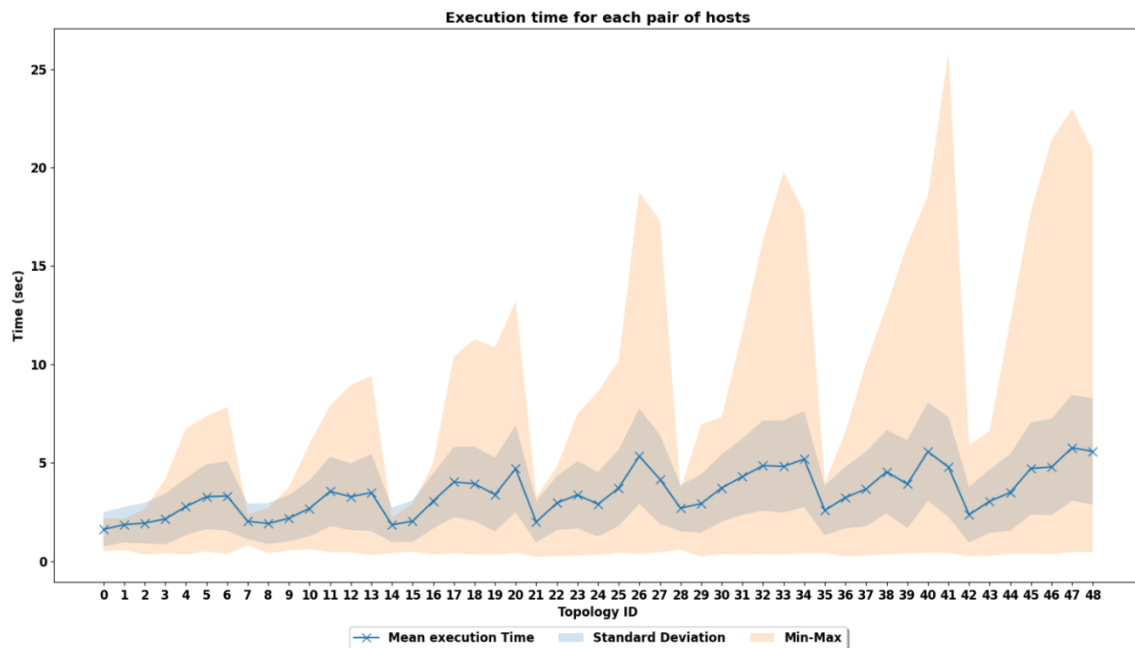


Figure 48: Execution time to find a path for a pair of hosts for a specific topology for [63] (TRB)

7.2.3 Objective success ratio and resource management

This section presents boxplots for each objective for each topology. The results are depicted in sections 7.2.3.1, 7.2.3.2, 7.2.3.3, 7.2.3.4 and 7.2.3.5 for delay, jitter, bandwidth, packet loss and security objectives respectively.

The percentage of the applications that EDAE's approach failed to find a path that meets the delay requirements is between 2 to 4 times, depending on the (m,h) configuration, less than [63]. The deviation from the requirement expressed as percentages are 2 to 10 times less than [63]. Finally, in almost all cases EDAE was able to find paths that utilize less resources than [63].

The results regarding Jitter, as shown in 7.2.3.2, are identical for both the algorithms and none of the m produces paths that violate the jitter constraints. Even though EDAE has smaller values in most cases regarding the over allocation of jitter, those values are negligible. With respect to the given evaluation framework, Jitter seems to be easily covered, since [63] does not take into account jitter objective and stills does not violate this constraint.

It can be clearly shown that EDAE outperforms [63] in all cases for both topologies by means of bandwidth and packet loss percentage. Specifically, for TRB topology EDAE is able to produce paths with the required demand on bandwidth on the total number of the cases.

Finally, as it was expected, EDAE outperforms [63] in security requirements, since [63] does not take into account those objectives. Nevertheless, the diagrams are included to showcase EDAE's performance and to validate that indeed that security is optimized.

Conclusions

Inducing constraints in the mathematical modelling regarding the optimization of QoS requirements results in superior results. Additionally, modelling the re-routing problem as a Multi Objective with constraints allows someone to increase the number of objectives and still have better results than a

Multi Objective problem with less objectives and not constraints. Finally, the thresholding that was introduced in the definition of the objective functions and the filtering of the Pareto front solutions are capable to produce solutions that exploits the network resources more optimal while the QoS requirements are still preserved.

7.2.3.1 Delay objective

ORB topology

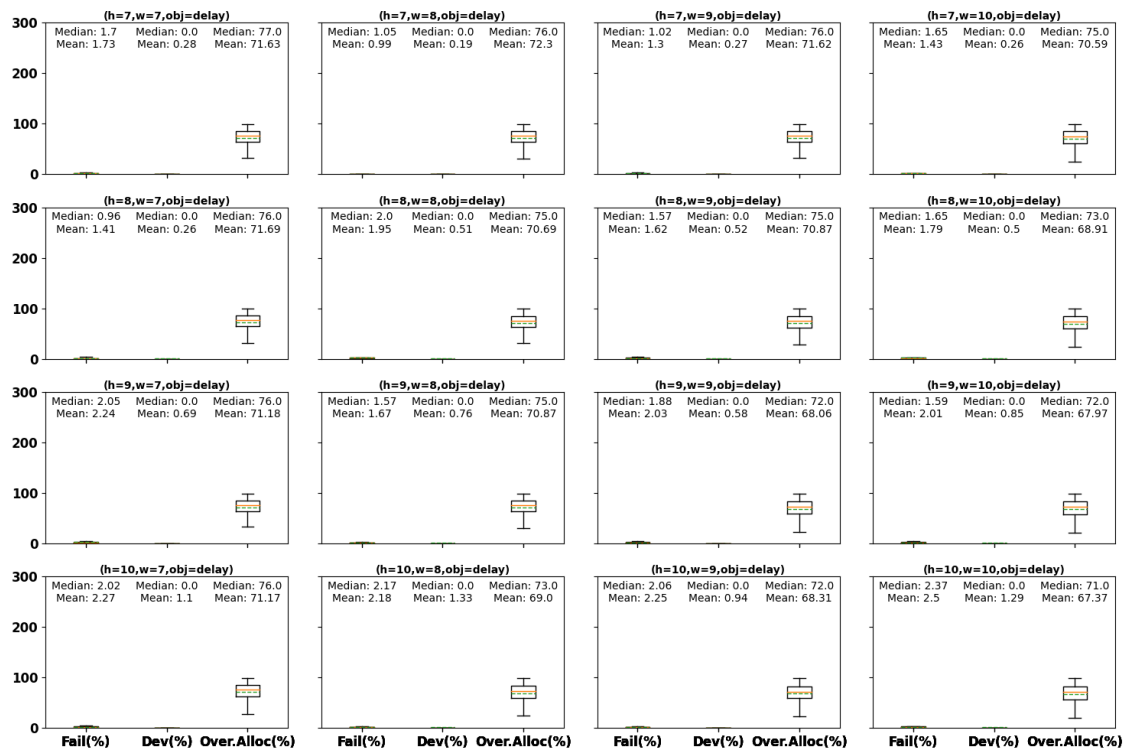


Figure 49: Statistics for delay objective for each topology [EDAE - (ORB)]

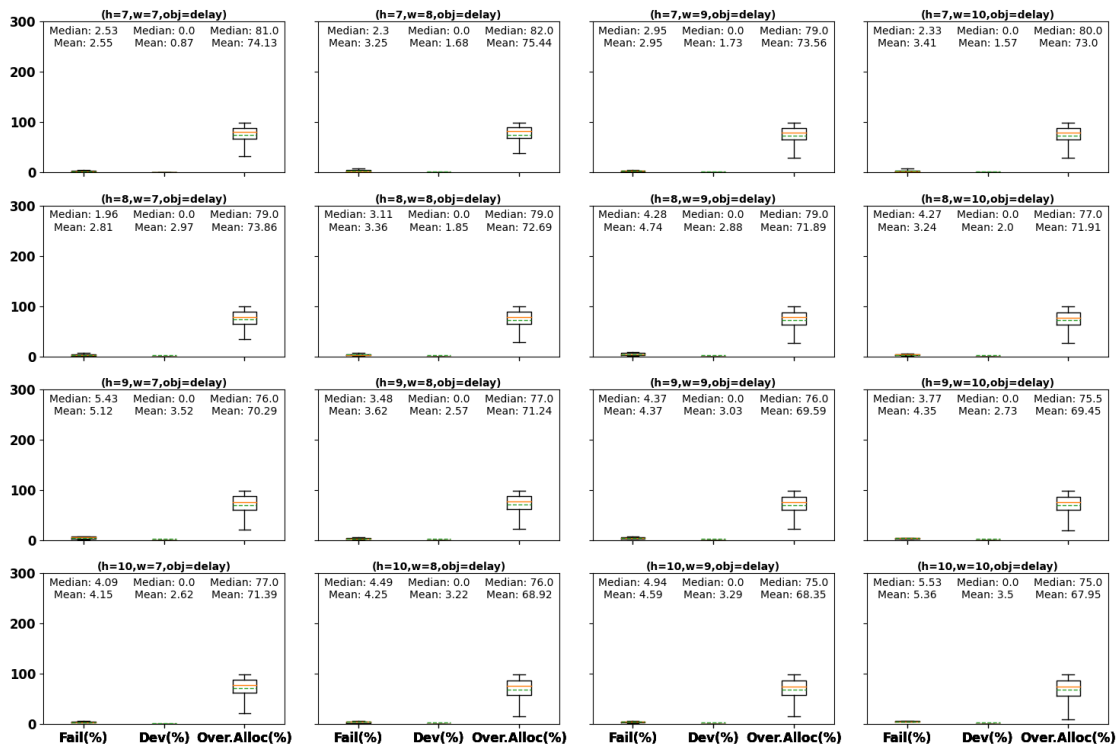


Figure 50: Statistics for delay objective for each topology [63] - (ORB)

TRB Topology

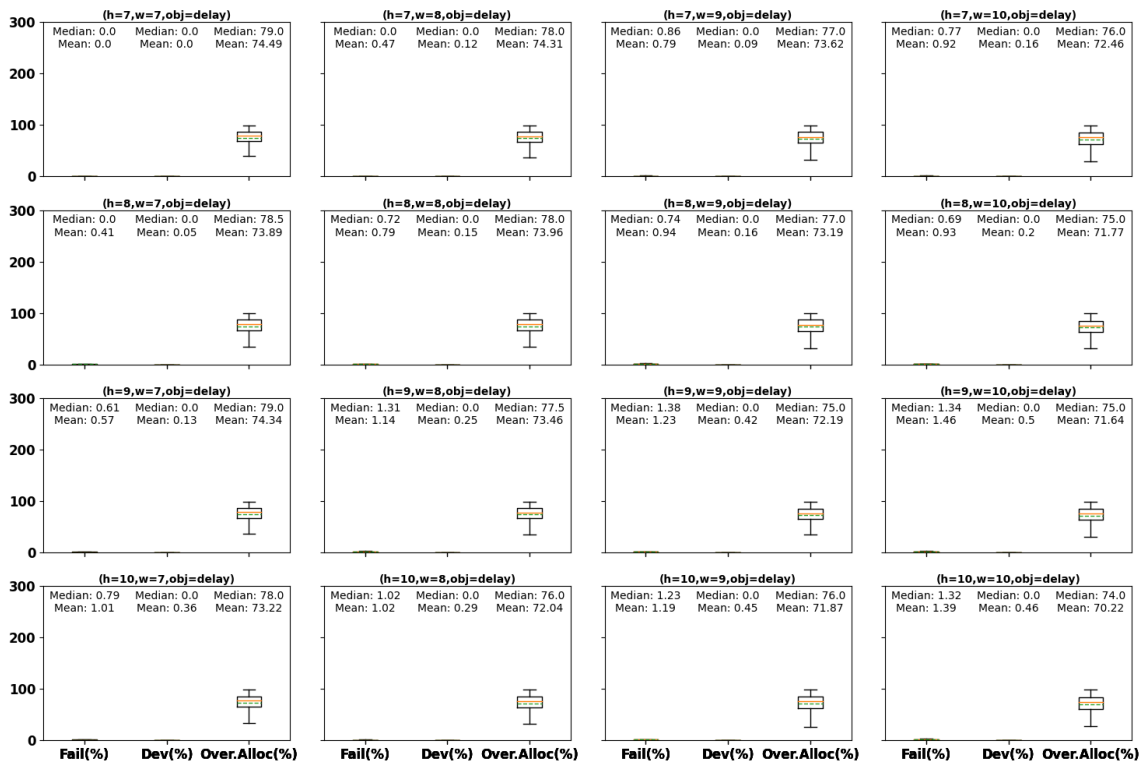


Figure 51: Statistics for delay objective for each topology [EADAE - (TRB)]

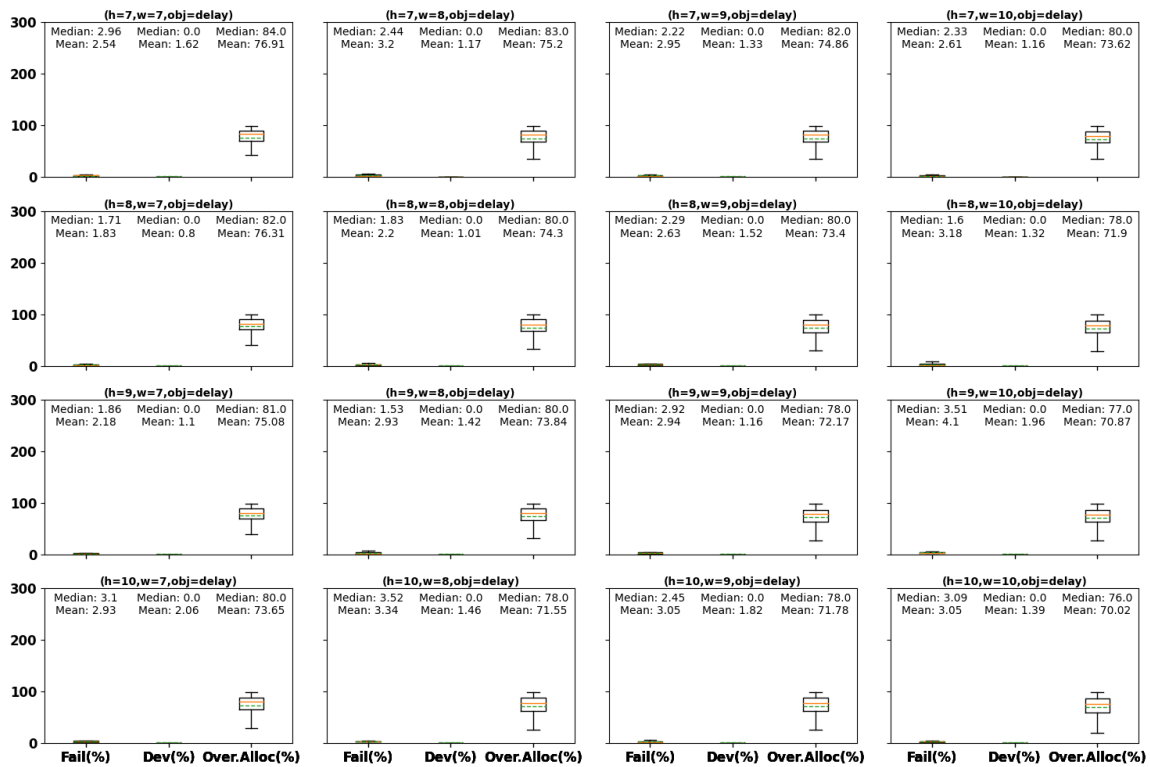


Figure 52: Statistics for delay objective for each topology [63] - (TRB)]

7.2.3.2 Jitter objective

ORB Topology

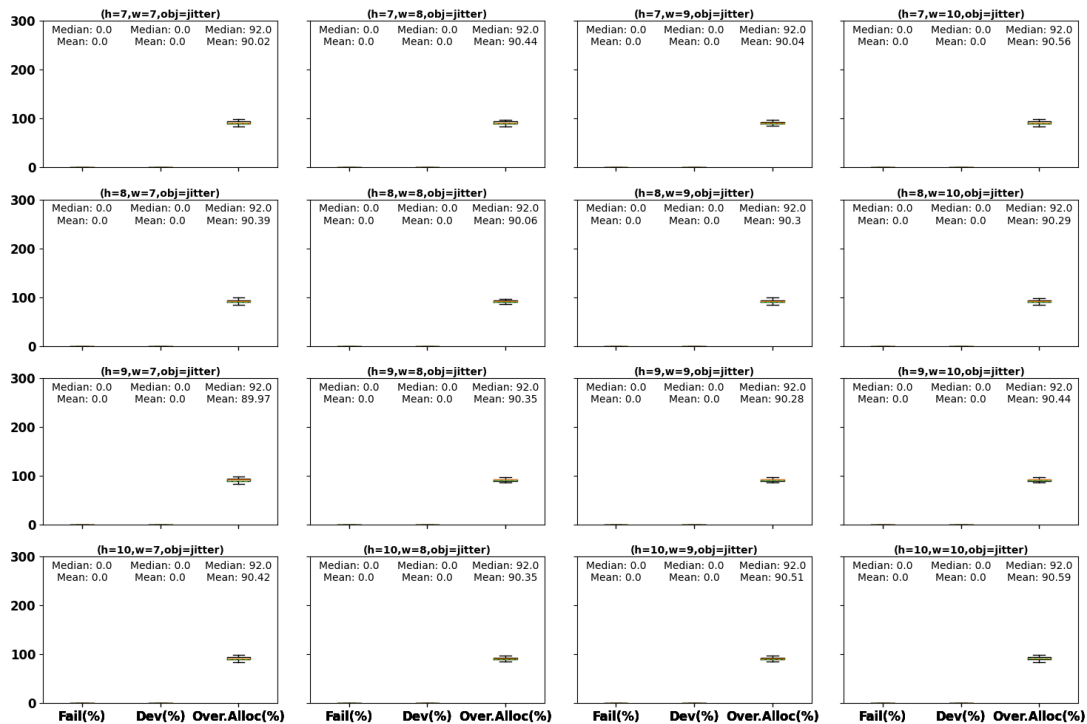


Figure 53: Statistics for jitter objective for each topology [EDAE - (ORB)]

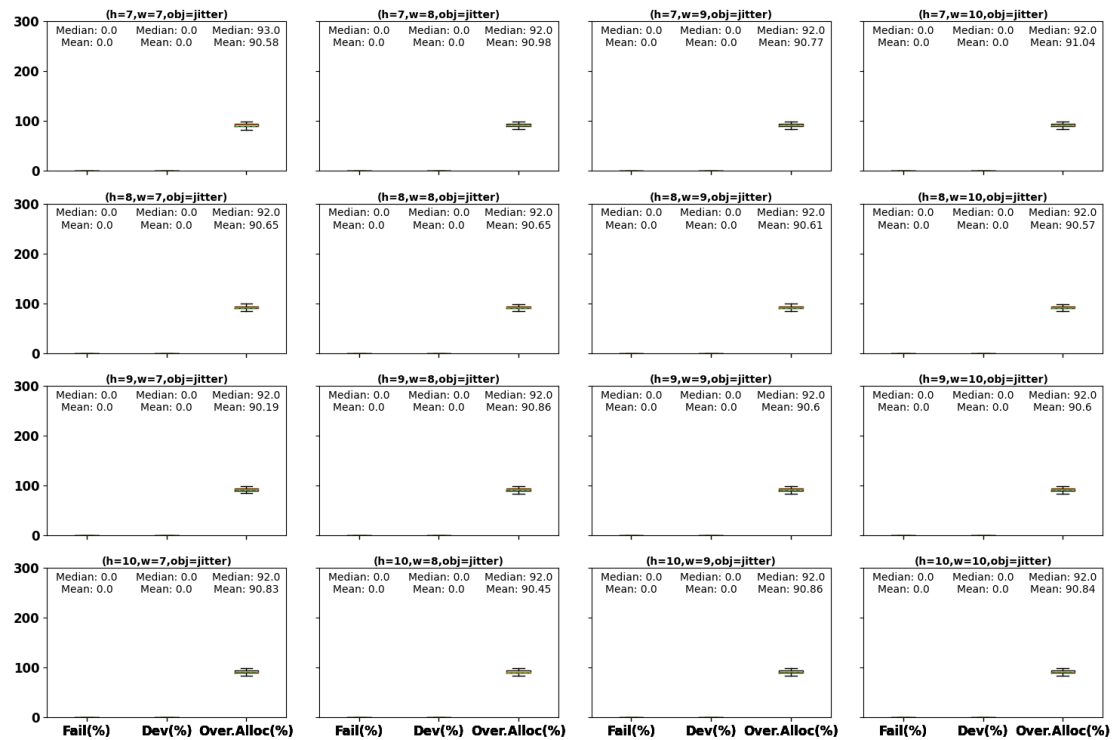


Figure 54: Statistics for jitter objective for each topology [63] - (ORB)]

TRB Topology



Figure 55: Statistics for jitter objective for each topology [EDAE - (TRB)]

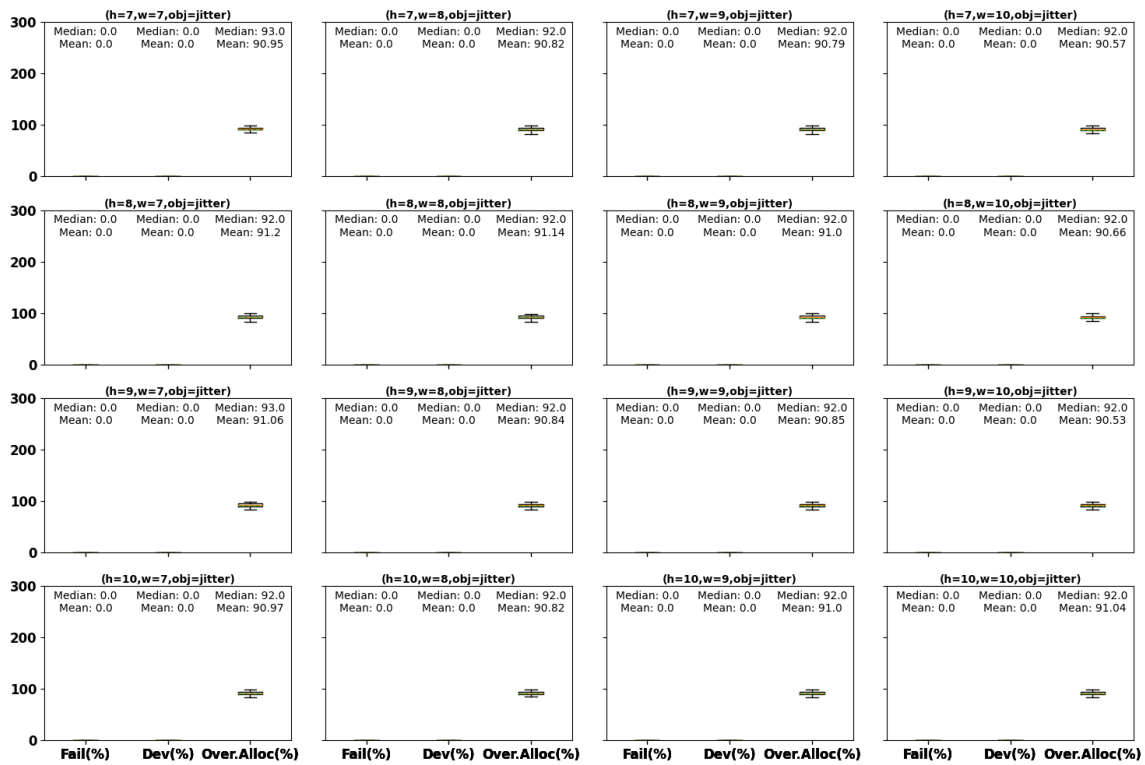


Figure 56: Statistics for jitter objective for each topology [63] - (TRB))

7.2.3.3 Bandwidth objective

ORB Topology

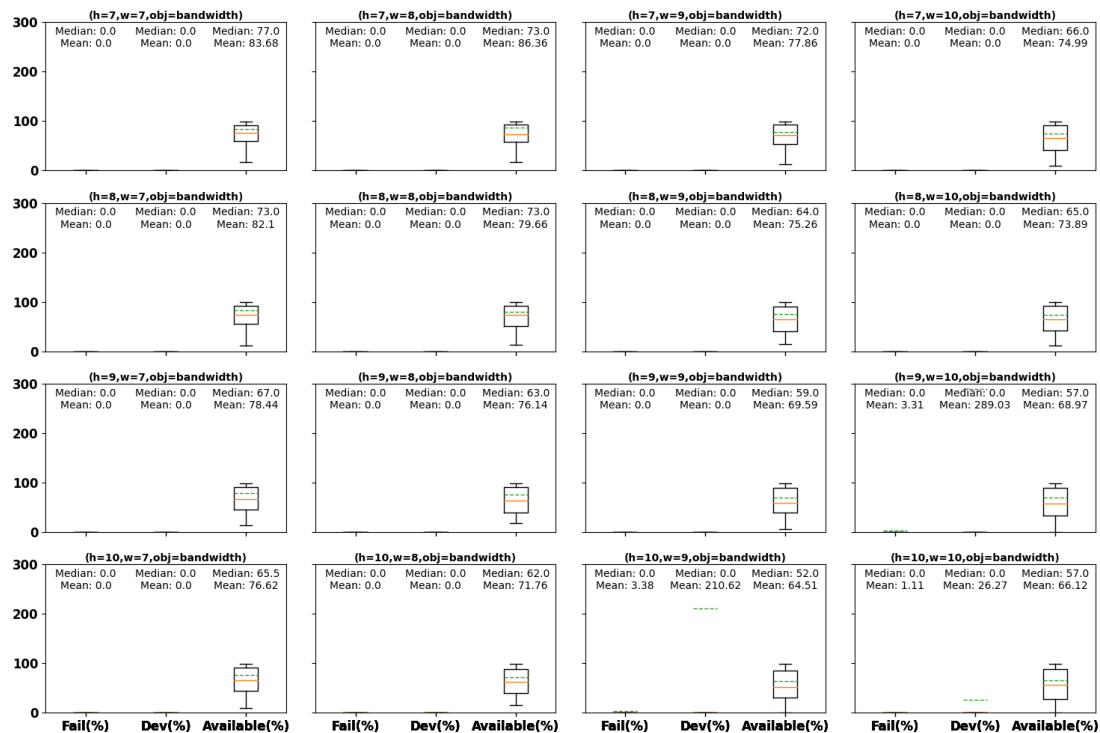


Figure 57: Statistics for bandwidth objective for each topology [EDAE - (ORB))

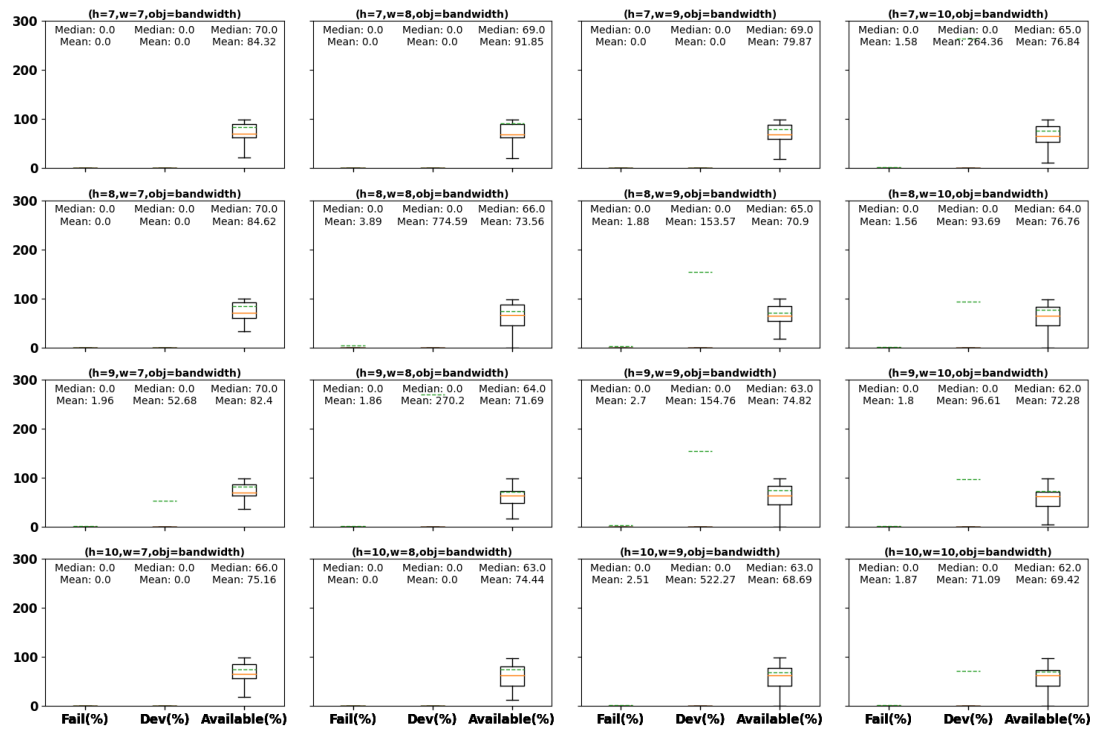


Figure 58: Statistics for bandwidth objective for each topology [63] - (ORB))

TRB Topology

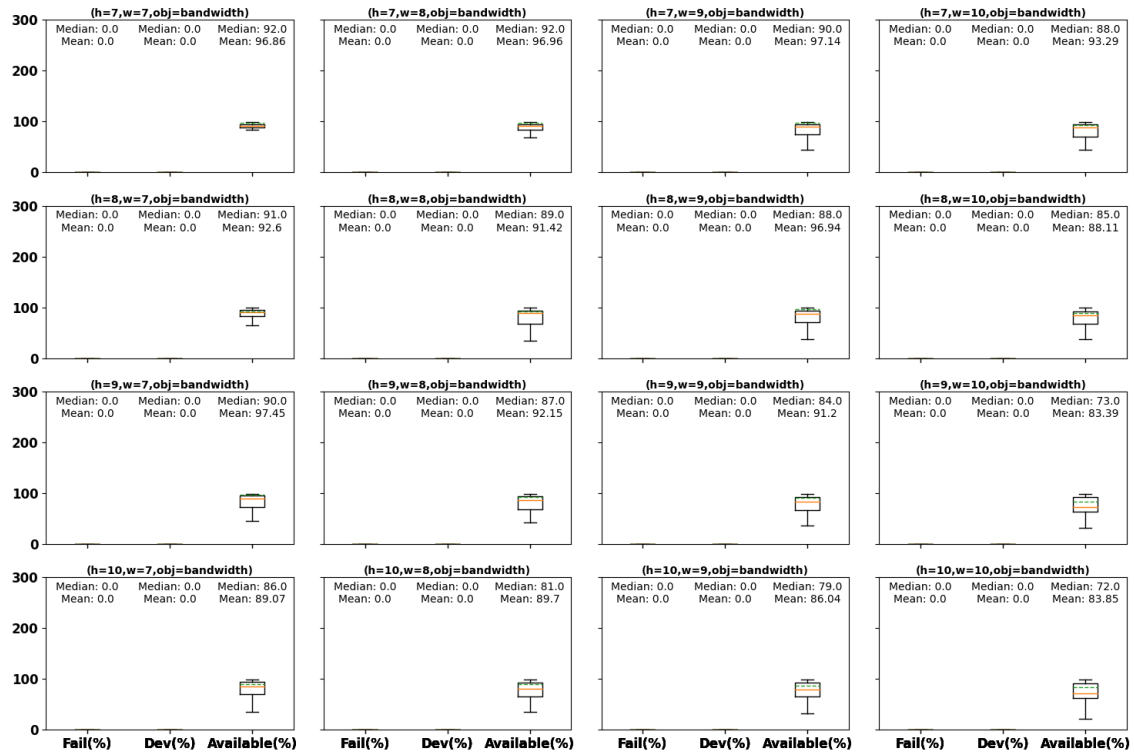


Figure 59: Statistics for bandwidth objective for each topology [EDAE - (TRB))

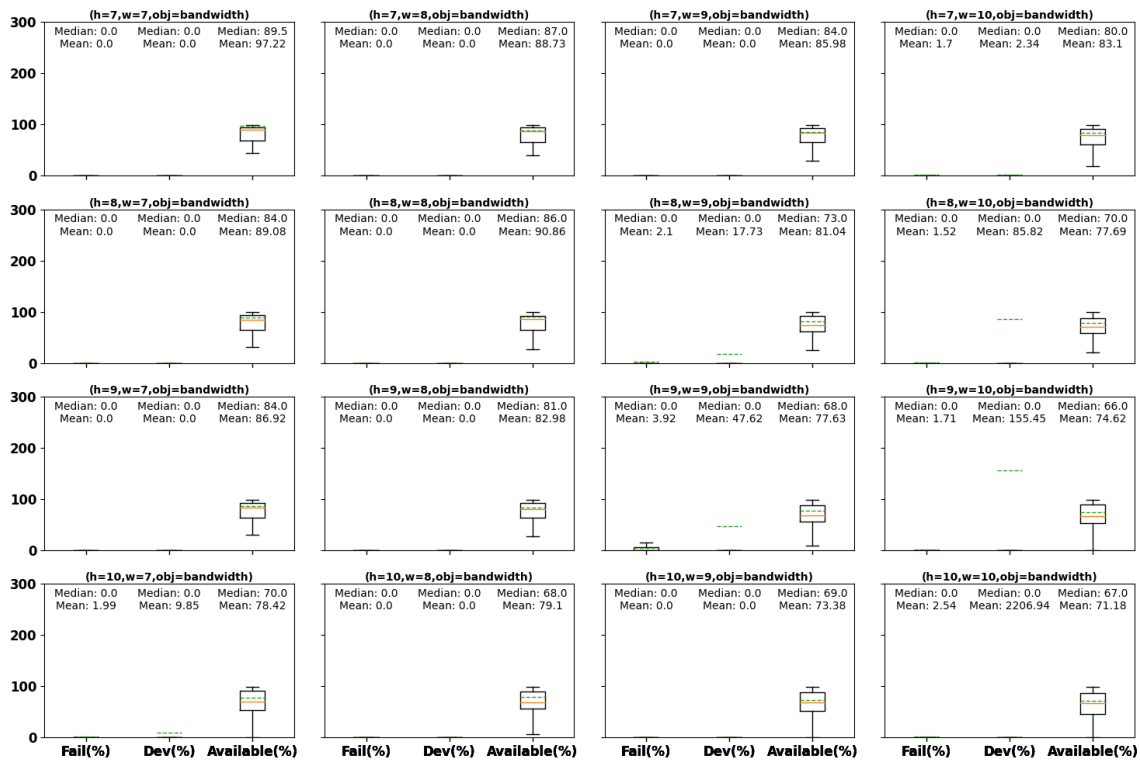


Figure 60: Statistics for bandwidth objective for each topology [[63] - (TRB)]

7.2.3.4 Packet loss objective

ORB Topology

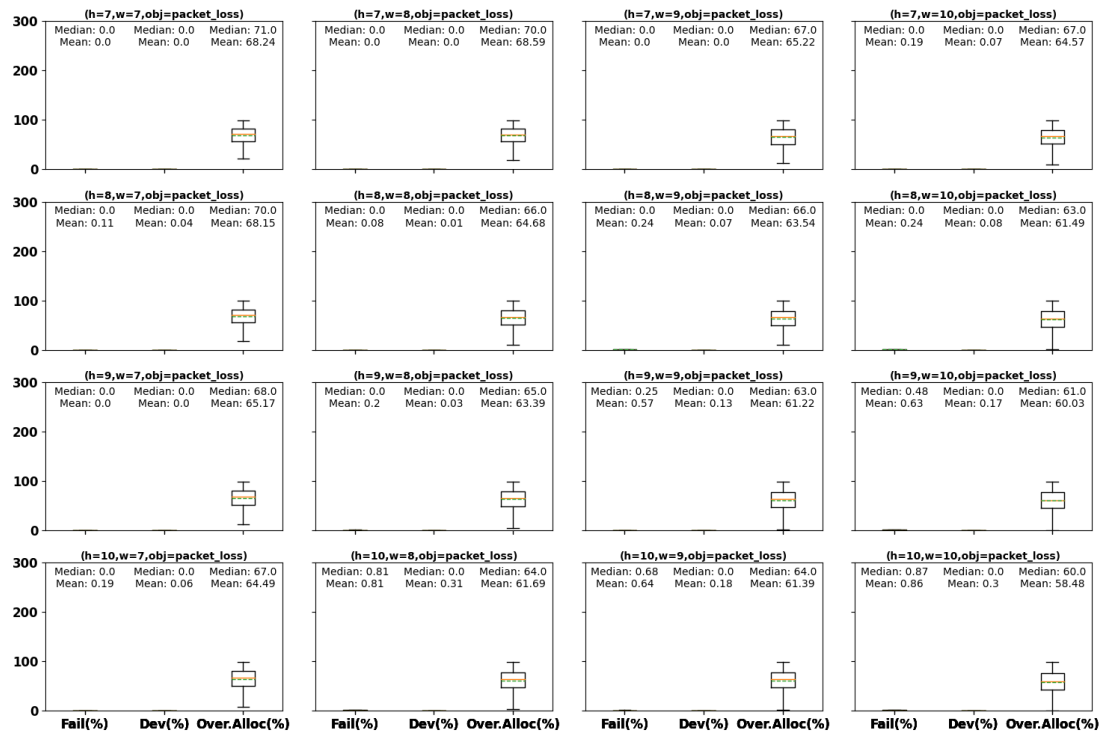


Figure 61: Statistics for packet loss objective for each topology [EDAE - (ORB)]

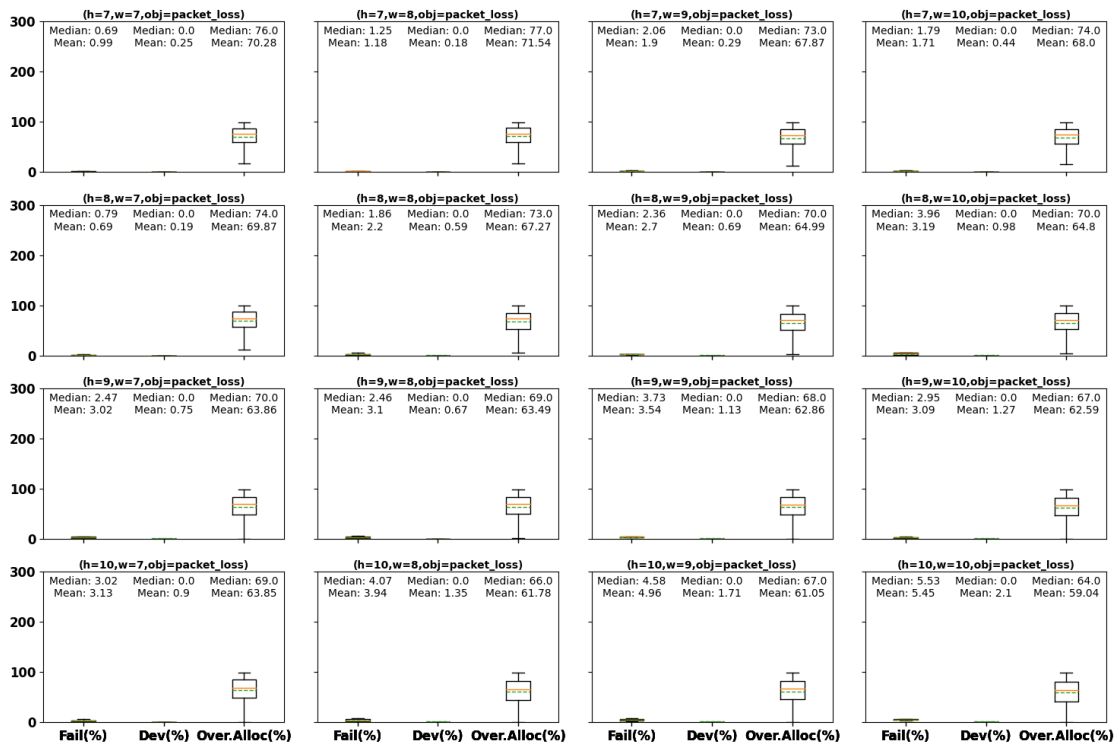


Figure 62: Statistics for packet loss objective for each topology [63] - (ORB)]

TRB Topology

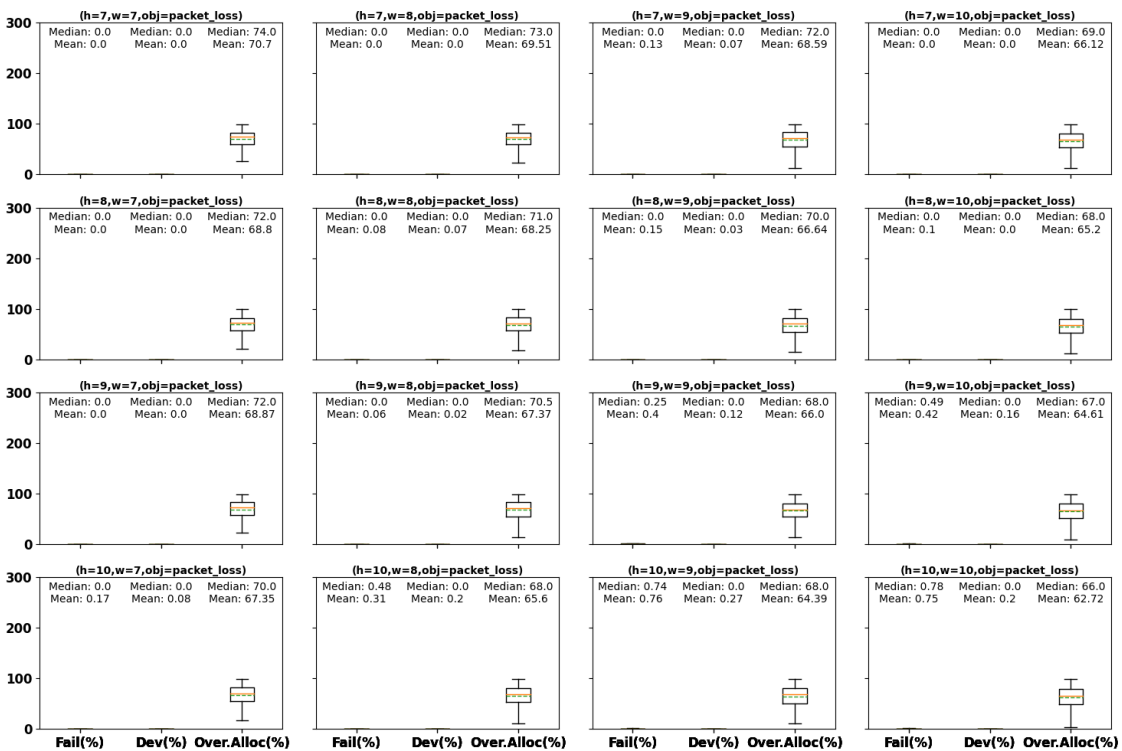


Figure 63: Statistics for packet loss objective for each topology [EDAE - (TRB)]

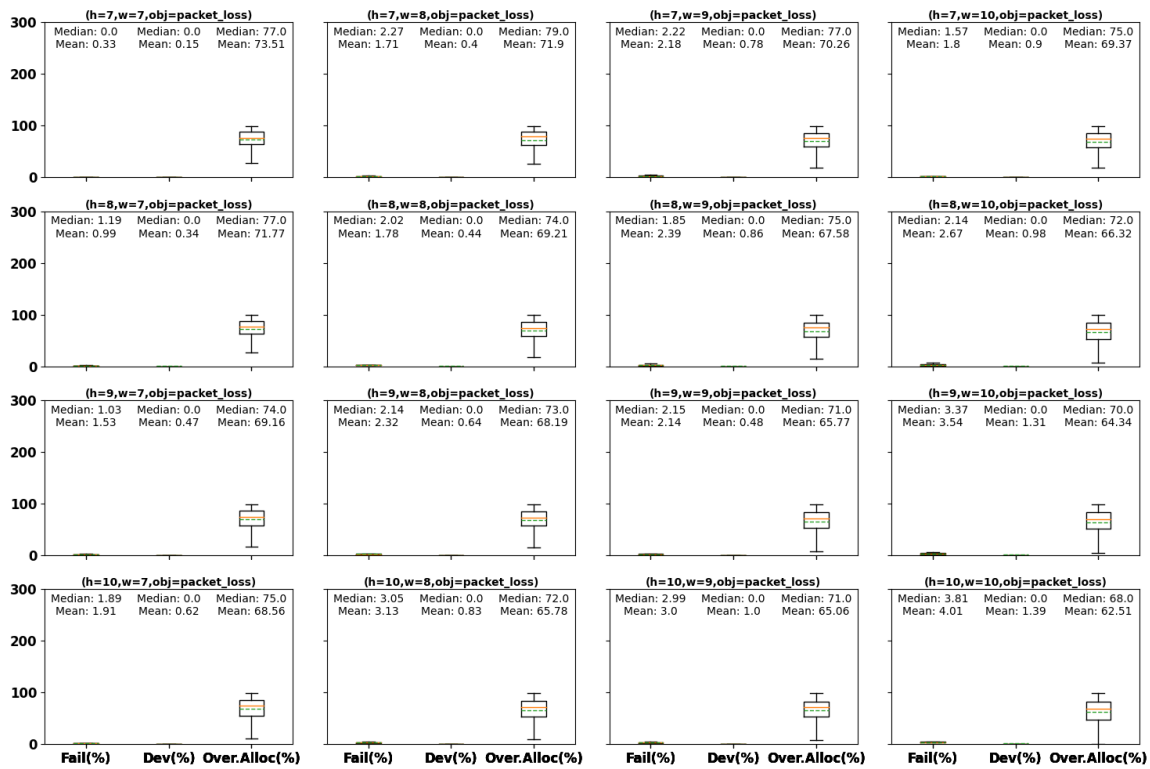


Figure 64: Statistics for packet loss objective for each topology [63] - (TRB))

7.2.3.5 Security objective

ORB Topology

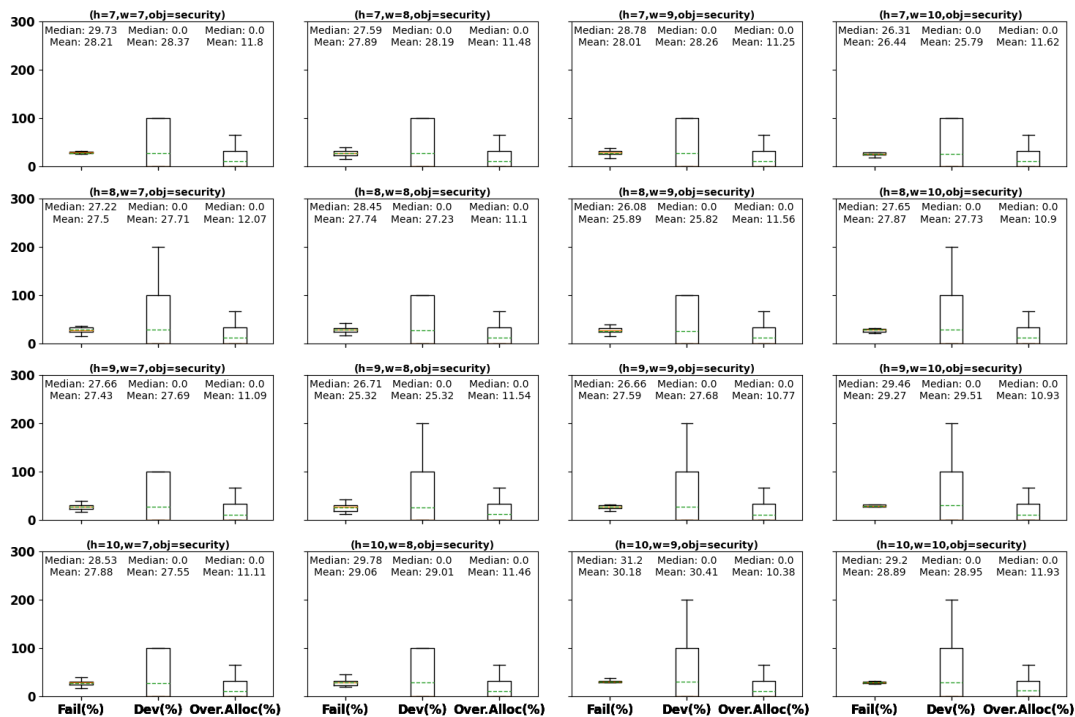


Figure 65: Statistics for security objective for each topology [EDAE - (ORB))

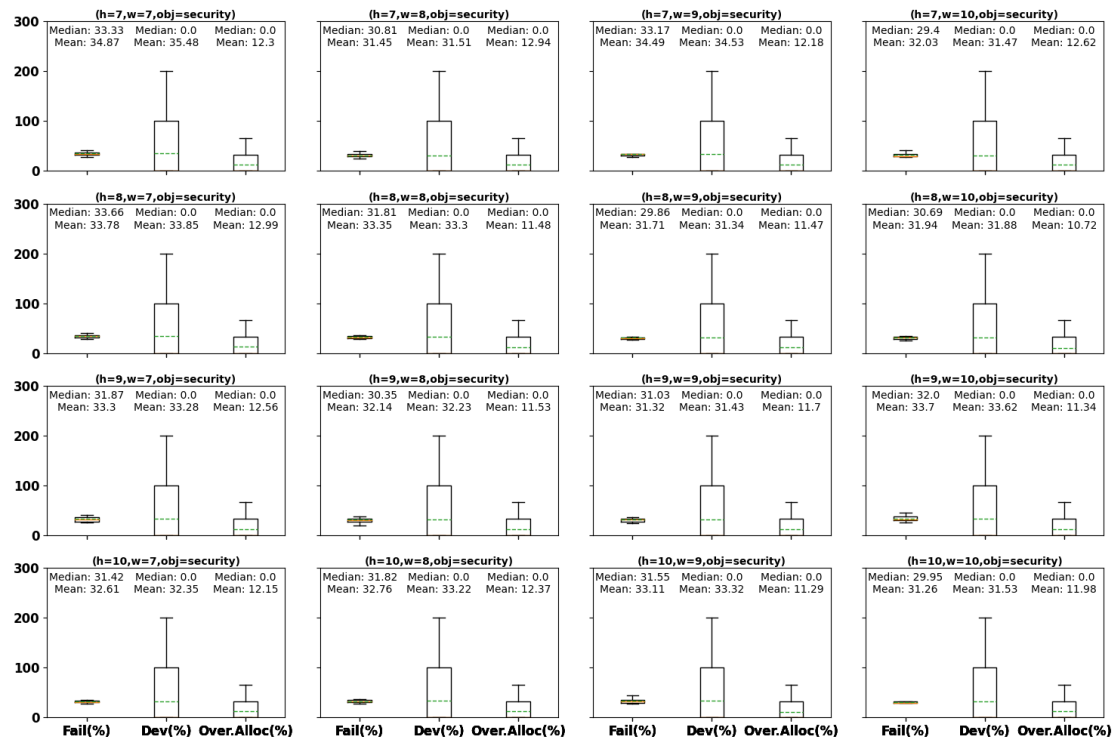


Figure 66: Statistics for security objective for each topology [63] - (ORB)]

TRB Topology

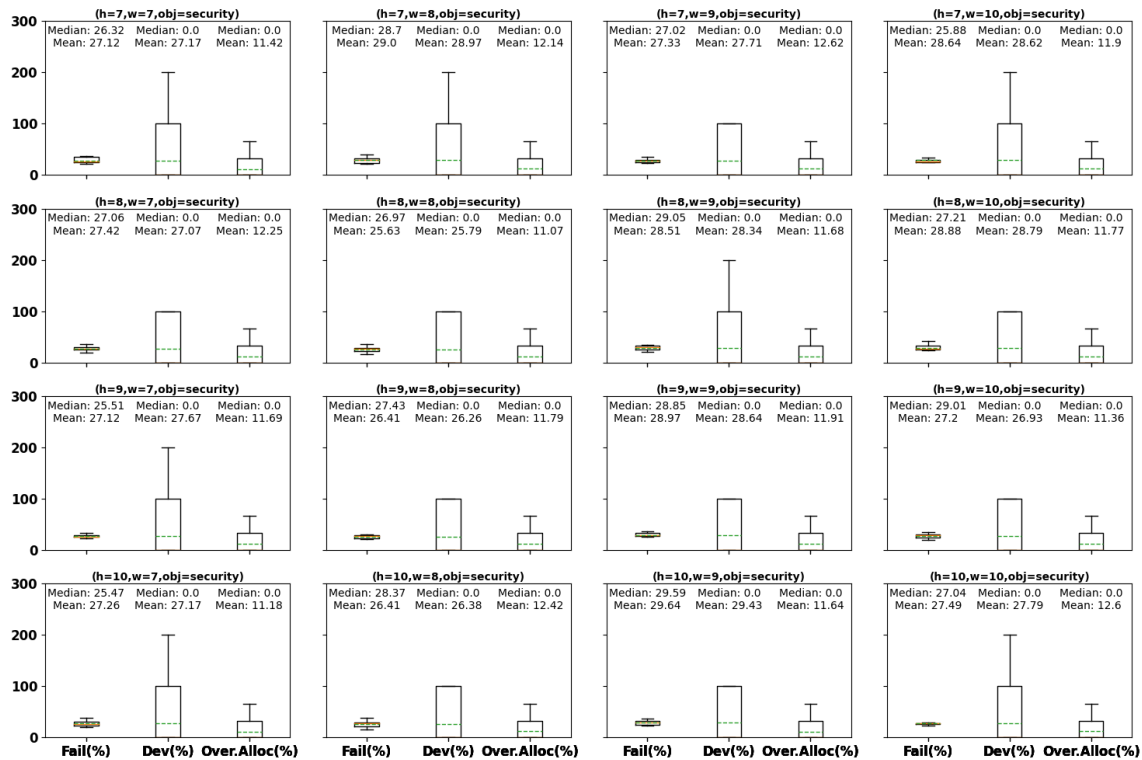


Figure 67: Statistics for security objective for each topology [EDAE - (ORB)]

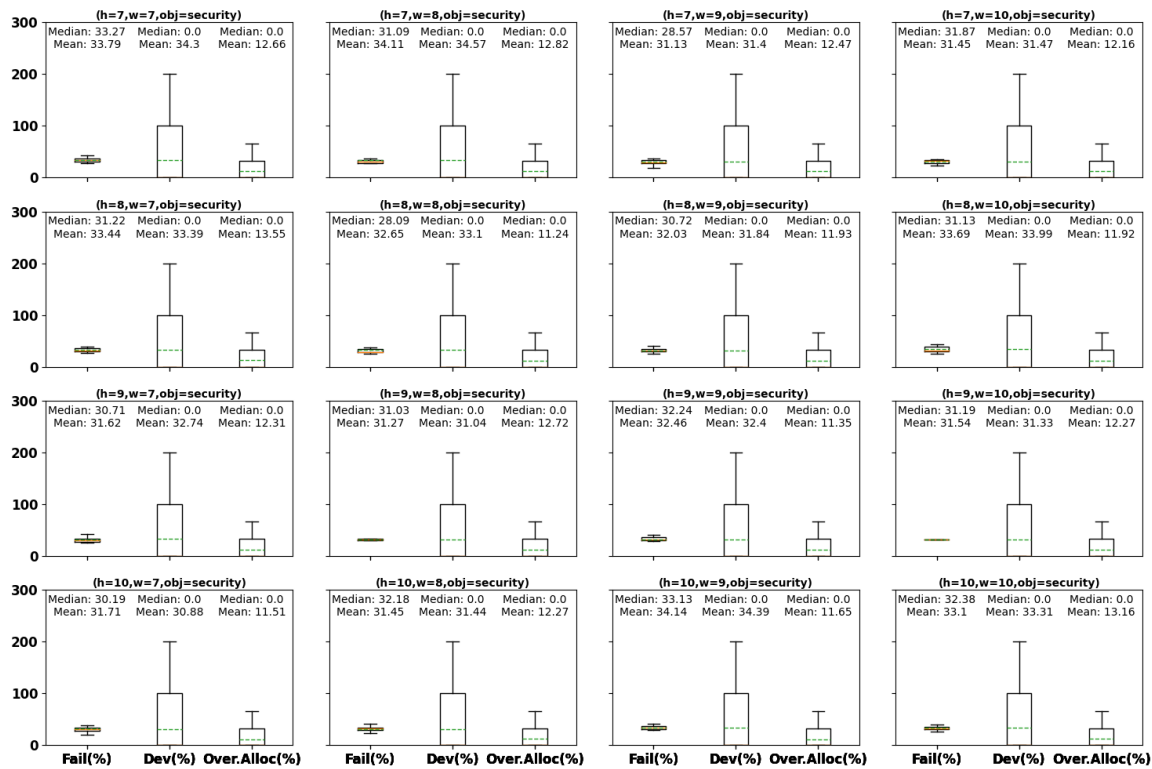


Figure 68: Statistics for security objective for each topology [[63] - (TRB)]

7.3 Evaluation framework for the maximization of the EPES observability

This section presents the comparative tests between our two proposed MILP algorithms and the MILP algorithm proposed in [59]. Qu et al. in [59] proposed a MILP formulation which is based on neighbouring PMUs. In details, their solution assigns one PMU from each neighbourhood to a PDC, when a PDC is compromised. The objective of their formulation is to select the minimum pairs of PDC and PMU.

Two differences exist between our two MILP algorithms and the proposed one in [59]. First, both in our two MILPs our objective function is to select the PMU and PDC pairs that minimize the latency of the network. Second, we reassign all the PMUs to the leftover PDCs if a PDC is compromised. Structure of the MILP evaluation topology

For the evaluation of the algorithms, we used the modelled IEEE 30 Bus system as in [59]. The topology consists of 30 PMUs and 10 PDCs. Each PDC is connected to a router.

Assumptions

The following assumptions apply in the evaluation framework:

1. Each PDC can connect up to 20 PMUs
2. Each PMU offers a traffic load of 10000 Mbps
3. The latency between each PMU-PDC path is set from 0.5ms up to 4ms as it is presented in Table 25 in section 7.1.4.

The evaluation of the three optimization models is done in terms of the MILP's computation time and the average latency of the network after the matchmaking. For the experiments, we disconnect 3 to 8 PDCs of the total 10. For each number of disconnected PDCs we run 50 experiments since the latency of the network varies. For each experiment, we randomly chose the compromised PDCs. Table 25 depicts the results:

Table 27: Evaluation results of MILP formulations

Number of disconnected PDCs		3	5	6	7	8
Qu et al. MILP [59]	Avg. Computational Time (s)	0.051	0.053	0.032	0.030	0.029
	Avg. Latency (ms)	21.80	22.00	22.16	21.90	21.10
Bandwidth constrained MILP	Avg. Computational Time (s)	0.032	0.029	0.027	0.022	0.024
	Avg. Latency (ms)	8.82	10.46	11.71	12.998	16.10
PMU constrained MILP	Avg. Computational Time (s)	0.034	0.036	0.026	0.024	0.022
	Avg. Latency (ms)	8.84	10.31	11.65	13.28	16.04

8 Assets inventory database design and implementation

Before the adoption of any Self-healing solution, EPES already count on its proper databases to manages the own functionality related to some aspect of the asset management and operation. The asset inventory database that it addressed in this paragraph, is a cross component which not only supplies information to the Application/Management Plane and the Control / Management Plane layers but also interacts with EPES in order to obtain information about those same Grid assets. This last interaction is depicted (see “Update Grid Info (via API)”) in the following scenario diagram already included in the D2.3 document [43]. In the SDN-microSENSE Self-healing project, this interaction will be manually done for each use case, at the beginning of the use case life time. In future solution adoptions, this will be the way to synchronize the real Grid assets from the EPES with the AIDB and it in turn shared with the SDN-microSENSE architecture.

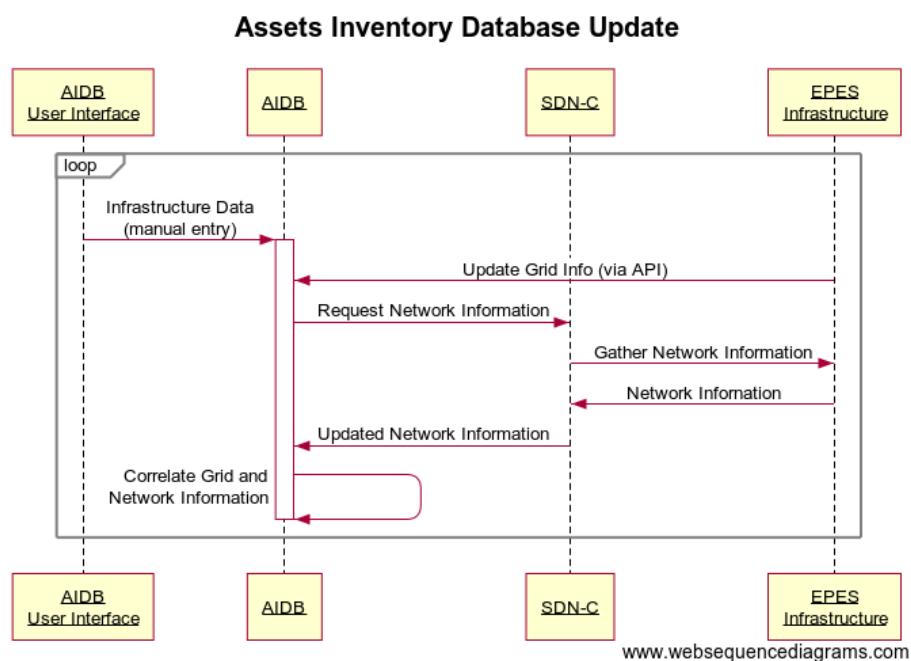


Figure 69: AIDB - Asset Inventory update

For the SDN asset inventorying, it is designed a mechanism which allows us to inventory all SDN assets automatically, without human intervention by the mean of the AIDB daemon. This in turn will queries to the SDN Controller through the Northbound interface all SDN Switches and Host connected to the SDN network. These information flows are depicted above using the name “Request Network Information” and “Updated Network Information”.

These components access to the AIDB is via API, notwithstanding a web application interface is also available to allow the Self-Healing User to query assets details. In the following diagram, internal AIDB component have been depicted:

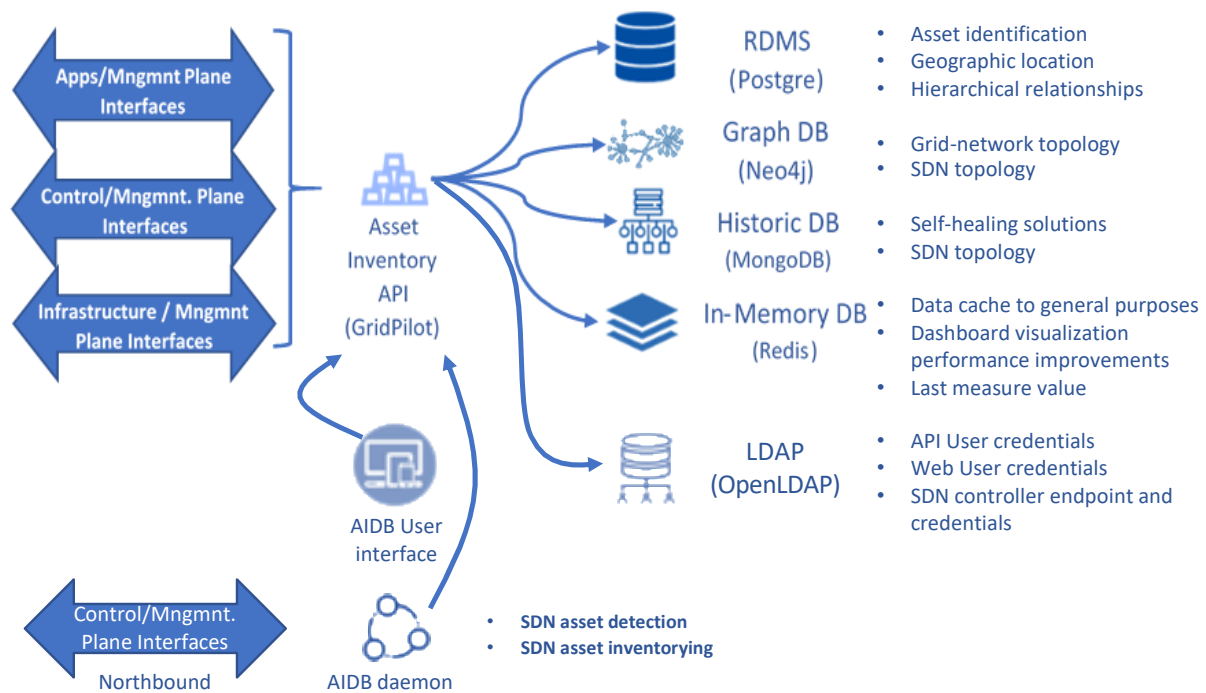


Figure 70: AIDB internal components

8.1 Interface model

AIDB keeps interfaces with SDN-C, EDAE, EDAE Dashboard, and S-RAF. The next figure contains all interfaces where the AIDB is involved, as well as the identification of both input and output information.

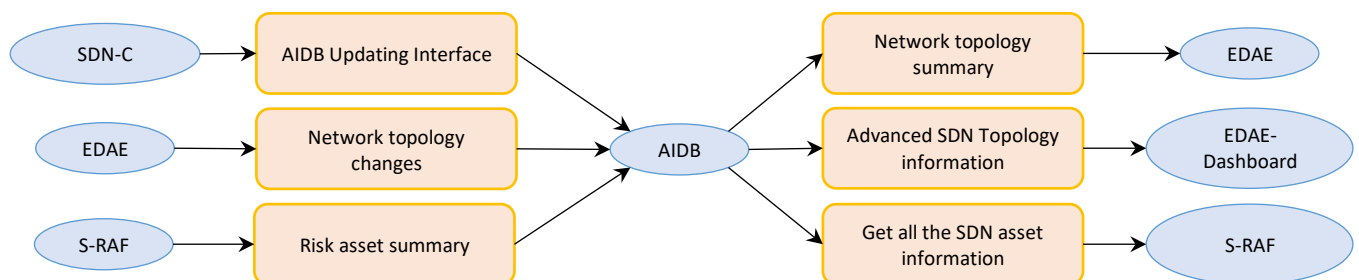


Figure 71: Interfaces with AIDB

These interfaces are been addressed previously. Architecture and detailed design, in the following tables:

- Table 5: AIDB-1
- Table 6: AIDB-2
- Table 8: AIDB-1 interface for S-RAF
- Table 10: AIDB-3 interface
- Table 7: AIDB-1

8.2 AIDB Information

According to the ISO 27001 definition, an asset is something that has value to the organization. An asset extends beyond physical goods or hardware, and includes software, information, people, and reputation. More concretely, the AIDB information encompasses both ICT network and grid assets. In the microSENSE Self-healing project the ICT network is based on SDN architecture. Figure 72 depicts an insightful summary of the information managed by the AIDB.

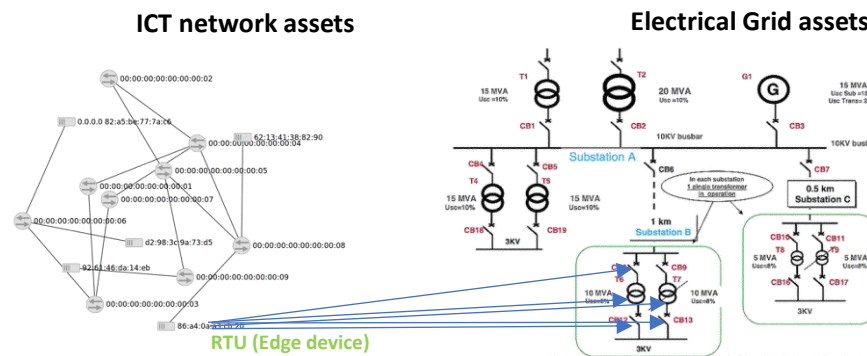


Figure 72: AIDB - Information areas

- **SDN assets.** These entities stored are SDN Switches, Hosts and SDN Controllers.
- **SDN topology.** The AIDB has to store relationships among switch-switch and switch-host assets. Additionally, the relationship between host-SDN Controller is also stored aims to register what SDN Controllers are connected to each subnetwork.
- **Grid assets.** These entities are defined according to the CIM standard.
- **Grid model.** The Islanding and Energy balance mechanisms, that are included in the Self-healing features, are needed additional information which are not actually grid assets. In fact, this information models electric abstractions, things such as Impedances, External grids...
- **Grid topology.** The electric topology is a part of the grid model information.
- **SDN – Grid topology.** These relationships are among host-grid assets. In these relationships the host side models the edge computing (RTU, PLC, Smart meters, gateways) which are involved in telemetry processes, whereas the grid side is about the grid assets from which the edge computer gets measures, or to which sends control orders. This information is paramount to figure out the grid asset impacted by an ICT attack.

8.2.1 SDN asset modelling

In the microSENSE Self-healing architecture, the SDN Control layer supplies a first-hand information about the current status of SDN assets and the relationships among it. Having this into account it brings up following questions: Why is it really needed an SDN asset storage into a database? What difference does it make? ...As result of the microSENSE Self-healing project, we have convened the SDN asset database to cover following features the most partners have considered as paramount:

- **SDN asset history.** The AIDB allows us to know SDN change logs. This information cannot be provided by the Northbound interface owing to the subjacent SDN protocol, Openflow, does not manages historic information. In fact, the SDN protocol can provide information about what new host or switch has been detected, notwithstanding this eventual information is not internally saved.
- **Hosts with several Ethernet connections.** In the SDN protocols, there are not difference between a host and an Ethernet card. In fact, if we have a host with two Ethernet cards the SDN layer will not manage information about that host and will consider only two hosts that actually are Ethernet cards. For companies where this information is relevant, the AIDB can help to model this situation nesting a host within another host. The host-leaf will be the Ethernet card and the parent will actually be the host.
- **Additional information.** The AIDB is the selected place to storage additional information which is not provided by neither the subjacent SDN protocol, OpenFlow, nor the Northbound interface. Information related to maintenance and operator users of the SDN infrastructure cloud be stored in the AIDB. For instance, physic location, operator comments, last revision date, last incidents, pending work orders...
- **Multi SDN subnetwork Management.** The SDN information is enclosed in an SDN scope. This scope is actually made up of switches and hosts. All of these are managed by an only SDN Controller. It gets paramount the multi subnetwork management due to a medium-high ICT infrastructure counts on several subnetworks. The AIDB allows to share SDN subnetworks information providing a centralized management.
- **SDN Controller location.** The SDN Controller is indeed a service which is running in a host connected to a switch of the SDN network. The host where the SDN controller is running is a relevant information to figure out the SDN subnetworks. The AIDB stores all SDN controller locations and even the Northbound endpoint and credentials. Notice the Northbound credentials will be stored in a centralized LDAP used by all AIDB components.
- **Vulnerability information.** The vulnerability information which is managed by the *eVul* tool, is actually stored in the AIDB. The vulnerability API queries are addressed in this document.

On the right, an example of SDN network has been depicted (Figure 73). It can see, five switches, three hosts and the location of an SDN Controller. Blue lines depict topologic relationships, those relationships are bidirectional and indicate a direct communication (wired or not) between two SDN Assets. Yellow lines indicate a “belong to” relationship. Basing on the picture, we can say that SDN Controller manages a subnetwork made up of five switches and 3 hosts. Remember only an SDN Controller is needed to manage all the SDN subnetworks.

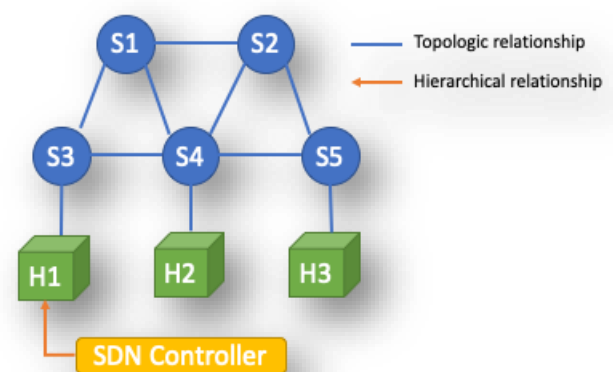


Figure 73: AIDB - SDN asset inventory example

Database description considerations

Next paragraphs contain the SDN asset data model. For a better understanding, it is relevant having into account following considerations:

- **Asset generalization.** The AIDB implementation is based on an open model definition where each asset regardless its type inherits a set of generic properties which are addressed in the generic asset paragraph. This way is often used by asset inventory databases.
- **Mandatory/optional fields.** Mandatory/optional fields do not have to be considered as mandatory in other AIDB implementations.
- **Fix values.** Some fields contain a fix value. These fields are included due to the open asset definition which allows software developers to implement features in unified way.
- **SDN asset identifiers.** For future microSENSE Self-healing implementations, the SDN asset IDs ought to be the MAC address returned by the own Northbound interface, this in turn is returned as Datapath Id by SDN Controller, and this in turn by SDN switches. This is the best way to guarantee switches and host identifications, with oneness and immutable value. Nonetheless, aims to easy the identification by human during the implementation of microSENSE use cases, we have decided employing the IP address rather than the MAC address as SDN asset identifier. Hence, in use case implementations, IP address must be fix values.
- **Relational and graph databases.** Switches, hosts and SDN Controller information is stored into a relational database, PostgreSQL, whereas SDN topology is stored in a graph database. The microSENSE Self-healing project does not intend to establish the best storage way. This design should be considered as a possible implementation option.
- **ExternalId/InternalId.** The ExternalId is an identifier which is environment independent whereas the InternalId depends on the environment. In the descriptions, the InternalId has been hidden. This mechanism allows the software promotion along different development environments (development, integrated tests, preproduction and production...)
- **Credential information.** All credential managed by the AIDB will be stored in a centralized LDAP database. Hence, all databases: PostgreSQL, Mongo, Neo4J, Redis will not contain information such as user id, password.

8.2.1.1 Switch

This entity models switches devices belonging to the ICT infrastructure. Though the microSENSE project is based on the SDN assets, the switch definition is also valid for no-SDN switches.

Table 28: AIDB – Switch table definition

Field	Req.	FieldId	Type	Description
ExternalId	Yes	InvExternalId	String	See <i>SDN asset identifiers</i> consideration.
Asset type	Yes	InvAssetType	String	Fix value 'SDN_SWITCH'
Net level	Yes	InvNetLevel	String	Fix value 'SWITCH_LAYER'
Scope	Yes	InvScope	String	It indicates the use case, i.e. 'UC1'...'UC6' (*)

Name	No	InvName	String	Human readable description of the switch datapath.
Description	No	InvDescription	String	Any human comment
MAC	Yes	sdnMAC	String	Mac address it matches the Datapath ID up.
IP	Yes	sdnIP	String	IP address
Manufacturer	No	sdnManufacturer	String	This value is automatically obtained through the Northbound interface
Hardware description	No	sdnHwDesc	String	This value is automatically obtained through the Northbound interface
Software description	No	sdnSwDesc	String	This value is automatically obtained through the Northbound interface
serial_num	No	serial_num	String	Serial number

(*) Notice, though there is a field to indicate the use case more than one subnetwork can be defined within a scope.

8.2.1.2 Host

In the AIDB, the host entity can refer to either a physical host or an Ethernet card. As already mentioned, an SDN switch is not able to distinguish between an Ethernet card and the physic host. If this detail is relevant for the SDN operators, the AIDB might model this situation with two hosts: a host for the physic host and other hosts for each Ethernet card belonging to.

Regarding physical hosts, there is not a field devoted to the type of host (PC, Server, PLC, smart meters...). This kind of information is not mandatory to a minimal solution for the microSENSE Self-healing. Nevertheless, the AIDB User can register it in the Name or Description fields somehow. On the contrary, knowing if a host actually is an Edge-Computer like a RTU, PLC, Smart meter..., is a valuable information to the SDN-microSENSE Self-healing project. In fact, these edge computers are responsible for the data acquisition in field. The AIDB does not store any especial flag to indicate this, however it counts on mechanisms to find out what host plays the edge computer role in the SDN network. This mechanism is addressed in the paragraph SDN and Grid asset relationships. Table 29 gathers the Host table definition.

Table 29: AIDB – Host table definition

Field	Req.	FieldId	Type	Description
ExternalId	Yes	InvExternalId	String	See <i>SDN asset identifiers</i> consideration.
Asset type	Yes	InvAssetType	String	Fix value 'SDN_HOST'
Net level	Yes	InvNetLevel	String	Fix value 'HOST_LAYER'
Scope	Yes	InvScope	String	It indicates the use case, i.e. 'UC1'...'UC6'

Name	No	InvName	String	Human readable description of the host.
Description	No	InvDescription	String	Any human comment. Here, the user can include any comment, even can contain any text that could indicate the host usage things such as 'PLC', 'RTU', 'PMU', 'SCADA', 'SMARTMETER'.
MAC	Yes	sdnMAC	String	Mac address it matches the Datapath ID up.
IP	Yes	sdnIP	String	IP address
ParentHost	No	InvFather	String	This field is used in two cases, with different meanings: <ul style="list-style-type: none"> • Edge computer. If the Host is a Smart meter, PLC, or RTU, this field might contain the ExternalId corresponding to the SCADA or other high level component to which the information is sent, or control operations are received. • Ethernet card. When the host is actually an Ethernet card within a physic host, this field might contain the ExternalId corresponding to the physic host.

8.2.1.3 SDN Controller

The SDN Controller is a software service which is bound to a host. That is the SDN network hub. This definition is used by the AIDB daemon to find out SDN assets. The AIDB daemon queries all SDN controllers registered in the database and makes a polling to get SDN information.

Table 30: AIDB – SDN Controller table definition

Field	Mandatory	FieldId	Description
ExternalId	Yes	InvExternalId	SDN Controller identifier
Asset type	Yes	InvAssetType	Fix value 'SDN_CONTROLLER'
Description	No	InvDescription	Any human comment
sdnDriver	Yes	sdnDriver	This can be 'Northbound' 'Ryu' 'floodlight'. 'Northbound' by default.
LDAP Endpoint	Yes	sdnEndpoint	This reference to a LDAP entry which define the credentials to access to Thirds party (url, user, pass)

8.2.2 SDN Topology

The SDN Topology managed by the AIDB is focussed on the connectivity among switch-switch and switch-host. AIDB employs a database based on graphs, Neo4J, to store these relationships. This database is arranged in nodes and arcs. Nodes can be both Switches and Hosts, whereas relationships model the connectivity between a Switch/Host with other Switch. Relationships between Host-Host are banned.

8.2.2.1 SDN Topology Asset node

The Node definition at least has to contain the SDN asset ExternalId to refer to the complete SDN asset information stored in the relational database. Nevertheless, the node definition gathers some information already stored in the relational database. This redundancy has been appraised by data architects and it supplies some advantages. The approach is exploiting this information by software components devoted to the graphic visualization of the SDN architecture and involving the relational database only when needed, to consult asset details.

Table 31: AIDB – SDN topology asset node table definition

Field	Req.	FieldId	Type	Description
ExternalId	Yes	InvExternalId	String	See <i>SDN asset identifiers</i> consideration.
Asset type	Yes	InvAssetType	String	Fix value 'SDN_HOST'
Scope	Yes	InvScope	String	It indicates the use case, i.e. 'UC1'...'UC6'
Name	No	InvName	String	Human readable description of the switch datapath.
MAC	Yes	sdnMAC	String	Mac address it matches the Datapath ID up.
IP	Yes	sdnIP	String	IP address

8.2.2.2 SDN Topology Relationship

Internally, the relationships have been modelled as directional way, i.e. there are one register for the relationship NodeA-NodeB, and other for the NodeB-NodeA. This way allows us to add more detailed information to each direction.

Table 32: AIDB – SDN Controller table definition

Field	Req.	FieldId	Type	Description
ExternalIdA	Yes	InvExternalIdA	String	ExternalId of the asset in the side A of the relationship.
ExternalIdB	Yes	InvExternalIdB	String	ExternalId of the asset in the side B of the relationship.
Enable	No	InvEnable	Boolean	It is a flag that indicates if the relationship is valid.
Weight	No	sdnWeight	Number	Weight of the relationship. This value will be used to the path ranking.

MaxLatency	No	sdnMaxLatency	Number	Max communication latency (milliseconds) from the asset identified by ExternalA towards the asset identified by ExternalB.
Bandwidth	No	sdnBandWidth	Number	Communication bandwidth (in bps) from the asset identified by ExternalA towards the asset identified by ExternalB.

8.2.3 Grid asset modelling

The first approach about Grid asset modelling was focussed on the CIM protocol, IEC 61968 as the best way to model Grid assets. A later deep dive into the informational requirements of the Self-healing algorithm brought up that the CIM model was not enough. Basically, this gap is owing to the Electrical models which are used by the OPF, Islanding and Energy Balance algorithms require information that are not in CIM. In fact, these algorithms employ following entities which do not entirely exist in the CIM model:

- **External grid.** The CIM's entities "Equivalent Network", "Equivalent load" and "Equivalent Source" do not contain the data required by the Grid model such as:
 - Voltage - voltage at the slack node in per unit.
 - Voltage degree - voltage angle at the slack node in degrees
- **Impedance.** The impedance is a property which is present in several grid assets. However, it is not actually an asset. In this sense, the entity Impedance managed by grid model is useful to electric abstract a section of the grid. This could be modelled using a fictitious "AC line segment" in CIM but this compels the AIDB we have a mix of real and fictitious assets in CIM.
- **Ward.** A ward equivalent is a combination of an impedance load and a PQ load. In the same way, this entity is an electric abstraction of part of the grid.
 - Active power of the PQ load
 - Reactive power of the PQ load
 - Active power of the impedance load in kW at 1.pu voltage
 - Reactive power of the impedance load in kVAr at 1.pu voltage
- **Extended ward.** A ward equivalent is a combination of an impedance load, a PQ load and as voltage source with an internal impedance. Newly, this entity is an abstraction of part of the electrical grid.
- **Storage systems.** CIM standard is updating to add this entity in the model and others related to the distributed generation through DERs.

Essentially, the problem is due to the Self-healing algorithms requires an electrical model of the Grid for electrical analysis whereas the CIM protocol offers a model more suitable for the asset management. The electrical model is deep-in addressed in the EDAE tool description. Here, software design is focussed on searching the closet standard format to the desired grid model and the software base chosen to store it. As result of this research, in the state of art we have found the electric model *pandaPower*. It is not only a suitable model for our purposes but also provides a compatible software library that supplies several algorithms such Status Estimation, OPF and Short-circuit, as well as geographic location and representation of the Grid. Next, the pandaPower entities have been enumerated:

- Bus
- Line
- Switch
- External grid
- Transformer
- Three winding transformer
- Generator
- Shunt
- Impedance
- Ward
- Extended ward
- DC Line
- Measurement
- Storage

The *pandaPower* model format can be gathered in an only JSON file. The file structure is very similar to the panda dataframe definition where basic entities (already mentioned) are defined like datasets. The entities count also on catalogue definition with model specifications which in turn have got preconfigured values for an entity type. This configuration is referenced by a name which might be used in the electric entity.

8.2.4 Grid topology

The grid topology is partially covered by the CIM due to the electric abstractions are not considered. Fortunately, the **pandaPower** includes too the electric topology model suitable for Self-Healing algorithms. The pandapower provides a function to translate pandapower networks into network graphs. Once the electric network is translated into an abstract network graph following features and more are available:

- Find the shortest path between two nodes.
- Calculate the shortest distance between a source bus and all buses connected to it.
- Find all buses that are connected to a certain bus.
- Cluster all buses graph that are connected to each other.
- Find if there are cycles in a network.
- Find buses that are not connected to an external grid.

AIDB does not store electric topology information in the Graph-oriented database. In contrast, entire information about Grid

8.2.5 SDN and Grid asset relationships

The AIDB manages relationships among an SDN host and a set of Grid assets. In the SDN side, the relationship involves an edge device (as well as edge computer device) such as an RTU, PLC or smart meter, even an inverter, whereas in the Grid side, any grid asset can be related. This relationship models two reality aspects:

- **Monitoring.** The edge device can get physic measures or status about a grid asset.



- **Control.** The edge device can receive orders which itself in turn can make comply against a grid asset. For instance, open/close orders against a grid switch, transformer tap position.

The AIDB distinguishes between these two realities, understanding that if a relation exists from Grid Asset towards an SDN Host then the Grid monitoring operation will be able. By contrast, if the relationship is from an SDN Host towards a Grid Asset then a Grid control operation will be able.

Internally, the relationships have been modelled as directional way, i.e. there are one register for the relationship NodeA-NodeB, and other for the NodeB-NodeA. This design aspect is motivated for unification purposes, allowing to employ the same graph searching algorithm in the SDN-SDN, and the SDN-GRID topology queries.

Table 33: AIDB – SDN – Grid relationship table definition

Field	Req.	FieldId	Type	Description
ExternalIdA	Yes	InvExternalIdA	String	ExternalId of the SDN/Grid asset in the side A of the relationship.
ExternalIdB	Yes	InvExternalIdB	String	ExternalId of the Grid/SDN asset in the side B of the relationship.
Enable	No	InvEnable	Boolean	It is a flag that indicates if the relationship is valid.

8.2.6 Vulnerability information

The AIDB contains information about the asset vulnerabilities. Although, vulnerabilities are detected and managed by the eVul tool which though belongs to the WP3, the risk levels related to an asset is stored into the AIDB. An asset can have a few open vulnerabilities each one of them in a different state. The Figure 74 depicts the relationship a Host and a Vulnerability.

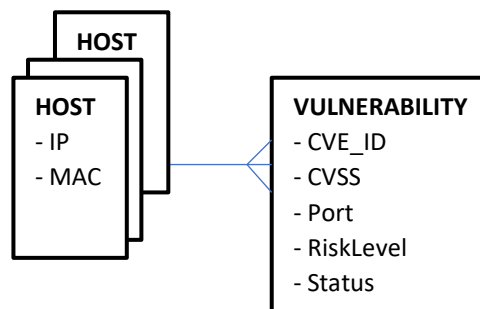


Figure 74: Vulnerability info

All of SDN Self-Healing vulnerabilities follows the Common Vulnerabilities and Exposures codification, CVE-ID which are gathered in the Open Source Vulnerability Database (OSVDB). Each vulnerability is bound to a metric known as Common Vulnerability Scoring System, CVSS. Table 34 contains CVSS ranges and his severity or risk level.

Table 34: Vulnerability risk level ranges

CVSS	Severity
------	----------

0	Informative
(0, 4)	Low
[4,7)	Medium
[7,9)	High
[9,10]	Critical

Nonetheless, the vulnerability manager, eVul, might intentionally change the risk level having into account concrete hosts. Below the vulnerability table definition is included:

Table 35: AIDB – vulnerability information table definition

Field	Req.	FieldId	Type	Description
ExternalIdA	Yes	InvExternalId	String	ExternalId of the SDN Host in which a vulnerability has been detected.
IP	No	IP	String	Host IP
MAC	No	MAC	String	Ethernet MAC
CVE	No	CVE_ID	String	Common Vulnerabilities and Exposures identifier
CVSS	Yes	CVSS	Number	Common Vulnerability Scoring System
Risk Level	Yes	RiskLevel	String	It indicates the severity of the vulnerability according to the Table 34: Vulnerability risk level ranges.
Port	No	Port	Number	Port number to which the vulnerability has been detected
Status	No	Status	String	It indicates the state according the vulnerability life cycle.

Vulnerability manager component creates a new instance workflow to follow up every detected vulnerability. Below, the vulnerability life cycle is depicted by a state diagram. The Vulnerability API will return only active vulnerabilities.

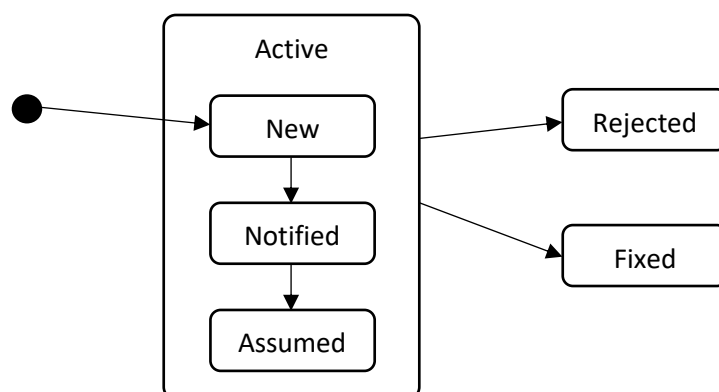


Figure 75: AIDB/Vulnerability manager - Vulnerability life cycle states

8.3 Asset inventory database Implementation

8.3.1 Architecture

The AIDB is based on open source components and additional microservices devised and implemented in Java for the SDN Self-healing project purposes (Table 36). All those components can be deployed in both local and cloud environments by the mean of Docker compose containers. Regarding the *cloudification*, it deems the AIDB component as cloud agnostic software, free of dependences on specific cloud platforms.

Table 36: List of components used for the AIDB

Base software component	Version
Java	OpenJDK 14
PostgreSQL	v9.4.8
Mongo DB	v4.0.16
Neo4J	v3.5.17
REDIS	v4.0.5
LDAP	LDAP

8.3.2 LDAP

The main security consideration in AIDB is avoiding the storage of credential information in the own AIDB database. For this, a devoted database will be used to store credential information for following purposes:

- AIDB Web application User credentials. This will be used to authentication of Web Users.
- AIDB API services User credentials. This will be used to authentication of API Users.
- Thirds party credentials. This will be used to gather the endpoint, the user, and password of external APIs which are employed by the AIDB. This need brought up to SDN Controller API.

A side effect of the LDAP usage is the User responsible for the security maintenance will be able to admin the credentials regardless the AIDB Web application and any AIDB logic.

8.4 Asset inventory API

AIBD allows other tools to access the functionality using a few API services. The functionality offers a lot of usage possibilities which have been drawn up below:

- **Asset query operations.** This functionality allows to know SDN assets stored in the database including its relationships SDN-SDN assets and SDN-Grid assets.
 - Lookup every SDN assets. This is the best way to get the entire ICT infrastructure
 - Lookup every SDN assets belonging to a certain type SDN Switch, Hosts or SDN controller.
 - Lookup every Grid assets. To get grid asset information, the grid model query is recommended. This functionality is the way to get the relationships among Grid assets and SDN Hosts.

- **Asset update operations.**
 - Create an SDN Switch, an SDN Host or an SDN Controller, including relationships with other assets
 - Update properties of a specified SDN Asset
 - Update properties of a specified Grid Asset. This functionality is not recommended due to grid asset will be managed through the grid model operations.
- **Topology query operations.**
 - Lookup all immediate connected assets. Given an asset, this operation gets only that assets to which are directly connected.
 - Lookup all connected assets, starting from a given asset. In ICT terms, this functionality returns the transitive closure of a given asset and answers the question whether and asset is achievable from other. For instance, this operation allows to know from what grid assets a RTU is able to get measures.
 - Lookup all connected network assets. The topology manages two networks, SDN and Grid. This operation is equal to the previous one, except only assets belonging to the same network to the given asset will be returned. For instance, if the given asset is a host then all achievable switches and host from it will be returned, and Grid assets connected to some RTU or PLC will not be included.
- **Grid model query operation.** This service provides a complete grid model definition in json format according to the **pandapower** library specifications. Grid asset properties, electric abstraction properties, the electric topology, and geographic location are included in the same json message.
- **Grid model update operation.** This service allows to store a complete grid model definition in json format according to the **pandapower** library specifications. Grid asset properties, electric abstraction properties, the electric topology, and geographic location are included in the same json message.
- **Open Vulnerability query.** The AIDB stores asset vulnerabilities which have been detected by the vulnerability tool, eVul. This information is offered by the AIDB for general purposes, especially by the components belonging to the S-RAF.

8.4.1 Code lists

Table 37: AIDB - Asset type code list

Asset type	Asset Class	Asset Level
GRD_AC_LINE_SEGMENT	Installation	GRID_MODEL
GRD_BUSBAR_SECTION	Installation	GRID_MODEL
GRD_DC_LINE_SEGMENT	Element	GRID_MODEL
GRD_EXTERNAL_GRID	Installation	GRID_MODEL
GRD_GENERATION_UNIT	Element	GRID_MODEL
GRD_GENERATION_UNIT	Element	GRID_MODEL
GRD_IMPEDANCE	Element	GRID_MODEL
GRD_LOAD	Installation	GRID_MODEL
GRD_SHUNT_COMPENSATOR	Element	GRID_MODEL
GRD_SWITCH	Element	GRID_MODEL
GRD_TRANSFORMER	Element	GRID_MODEL
GRD_TRANSFORMER	Element	GRID_MODEL
GRD_WARD	Element	GRID_MODEL
GRD_XWARD	Element	GRID_MODEL
INV_BATTERY	Element	GRID_MODEL
INV_SMARTMETER	Element	GRID_MODEL
SDN_CONTROLLER	Element	SDN_CONTROLLER_LAYER
SDN_HOST	Element	SDN_HOST_LAYER
SDN_SWITCH	Element	SDN_SWITCH_LAYER

Table 38: AIDB - Asset level code list

Asset Level
GRID_MODEL
SDN_CONTROLLER_LAYER
SDN_HOST_LAYER
SDN_SWITCH_LAYER

Table 39: AIDB - Asset status code list

Asset State	Description
CONN_E	Connected
DISCONN_E	Disconnected
UNINST_E	Uninstalled
WAREH_E	Warehouse

Table 40: AIDB - Asset Attribute code list

Asset type	Attribute codes
Any	InvExternalId
	InvAssetType
	InvClass
	InvName
	InvDescription
	InvState
	InvScope
	InvFather
SDN_SWITCH	sdnManufacturer
	sdnHwDesc
	sdnSwDesc
SDN_HOST	sdnIP
	sdnMAC
SDN_CONTROLLER	sdnEndpoint
	sdnDriver

Notice, Grid asset attributes have been hidden in the code list due to entire grid asset definition will be contained in the Grid model json format.

8.4.2 Data partitions

The AIDB is partitioned in 6 use cases, and counts on mechanisms that avoid the access to data belonging to different use cases. For this, these partitions require the ExternalId independence, owing to different asset belonging to different use cases might have got the same ExternalId. In fact, this is likely to happen to emulated SDN environments by Mininet. In these cases, Mininet spawns the same IP, MAC, and DIPD for hosts and switches for each new emulation. Thanks to the AIDB partitions those identifiers will not get hindered among them and assets which have been created within a use case cannot be seen from other use cases.

The API User setup establishes a use case, except the administrator User who has a complete access. Follow API return or manage information about all assets belonging to the use case to which the API User belongs to.

8.4.3 AssetQuery()

This API returns asset information which match up several search criteria. These criteria compel the AND logic, forcing all criteria have to be true to be included in the answer.

Table 41: AIDB – Asset query method

Field	Type	Req.	Description
Request			
invExternalId	String	No	If filled, the query will return information about that asset whose invExternalId matches with the given one. Otherwise, every asset will be returned.
invName	String	No	If filled, the query will return information about that asset whose name matches with the given one.
installation	YES/NO	Yes	Once Yes, and invExternalId is not informed, all installation assets will be included in the answer.
element	YES/NO	Yes	Once Yes, and invExternalId is not informed, all element assets will be included in the answer.
topology	YES/NO	No	If YES, the immediate relationships of each asset will be included in the answer.
assetType	String	No	When filled, the search is constrained to assets whose type contains the assetType string. See Table 37
Level	String	No	When filled, the search is constrained to assets whose level contains the Level string.
Response			
invExternalId	String	Yes	Already mentioned
invClass	String	Yes	INSTALLATION or ELEMENT. See Table 37
invAssetType	String	Yes	See Table AIDB - Asset type code list.
invNetLevel	String	Yes	Table AIDB - Asset level code list.
invFather	String	Yes	This field is only for SDN_Controller type. In this case, this field contains the externalId of the Host where the SDN controller is running.
attributes:	Array		
Attribute	String		See Table 37
Value	String		Attribute string value
relationships:	Array		
relationshipId	String	Yes	This is the internal id generated by the graph database.

externalIdA	String	Yes	Identification of the asset which plays the source role in the relationship.
externalIdB	String	Yes	Identification of the asset which plays the target role in the relationship.
sdnWeight	Number	No	This field appears in relationships between both Switch-Switch and Switch-Host.
sdnMaxLatency	Number	No	Max communication latency (milliseconds) from the asset identified by ExternalA towards the asset identified by ExternalB.
sdnBandWidth	Number	No	Communication bandwidth (in bps) from the asset identified by ExternalA towards the asset identified by ExternalB.

Example:

GET: https://apidemo.gridpilot.tech:8085/assetInventory/search?invExternalId=&installations=NO&downstream=NO&topology=YES&assetType=SDN_HOST&invName=&elements=YES&level=SDN

8.4.4 Asset Create/Update/Delete()

This operation has got an interface quite similar to the response of the previous API. To deal with different operations following considerations have to be taken into account:

- **Creation.** The creation operation is be made in two way depending on if the ExternalId is informed or not.
 - ExternalId is informed but not exist into the AIDB. In this case, the AIDB API will create an asset using the given ExternalId.
 - ExternalId is not informed. In this case, the AIDB will automatically generate a new ExternalId. This option will linger on disable.
- **Update.** All provided attribute values will be updated for the asset whose ExternalId matches up the given ExternalId.
- **Delete.** No physical deleting is possible in the AIDB. This operation is actually an update operation where the attribute invState is set to the most suitable value. See the Table 39: AIDB - Asset status code list.

Regarding the topology, following instruction have to be considered:

- **Creation.** To create a new relationship, the field relationshipId must be null and both externalIdA and externalIdB must be informed pointing at valid assets.
- **Update.** In this case, the relationshipId must be informed and the rest of field values will be updated into the graph database. It is possible a momentary disabling assignation a weight 0.
- **Delete.** In this case, the relationshipId must be informed and any of externalIdA or externalIdB must be null. This will provoke the relationship is removed from the graph database and certainly related assets will not be affected.

Table 42: AIDB - Create/Update/Delete asset operation methods

Field	Type	Req.	Description
Request			
invExternalId	String	Yes	Already mentioned
invAssetType	String	Yes	See Table 37: AIDB - Asset type code list
attributes:	Array		

Attribute	String	Yes	See Table 40: AIDB - Asset Attribute code list
Value	String	No	Attribute string value
relationships:	Array		
relationshipId	String	Yes	This is the internal id generated by the graph database.
externalIdA	String	Yes	Identification of the asset which plays the source role in the relationship.
externalIdB	String	Yes	Identification of the asset which plays the target role in the relationship.
Enable	YES/NO	No	Yes by default
sdnWeight	Number	No	This field appears in relationships between both Switch-Switch and Switch-Host.
sdnMaxLatency	Number	No	Max communication latency (milliseconds) from the asset identified by ExternalA towards the asset identified by ExternalB.
sdnBandWidth	Number	No	Communication bandwidth (in bps) from the asset identified by ExternalA towards the asset identified by ExternalB.

PUT: <http://apidemo.gridpilot.tech:8085/assetInventory/update>

8.4.5 TopologicQuery()

Table 43: AIDB – Topologic Query method

Field	Type	Req.	Description
Request			
invExternalId	String	No	If filled, the topology search will start from that asset. Otherwise, complete topology will be returned.
invExternalIdTarget	String	No	If informed, the query will try to find out a path between the asset identified by invExternalId and invExternalIdTarget. If that is not exist the answer will be empty. If not informed, a broadcast search will be returned, starting by the asset's invExternalId.
installation	YES/NO	Yes	Once Yes, relationships with installations will be included in the answer.
element	YES/NO	Yes	Once Yes, relationships with element will be included in the answer.
downstream	YES/NO	No	This field applies whether invExternalId is informed. If YES, the query will make a deep-in search. Otherwise, only immediate relationships will be returned.
border	YES/NO	No	If YES, only relationships between SDN and Grid assets will be included.
Response			
The answer is the same to the AIDB – Asset query API response			

GET: <https://apidemo.gridpilot.tech:8085/assetInventory/search?invExternalId=0000000000000004&invExternalIdTarget=&installations=NO&elements=YES&downstream=YES&topology=YES>

8.4.6 GridModelQuery()

This service will return a json message according to the **pandapower** specifications corresponding to the use case to which the API User belongs.

Invocation example:

GET: <https://apidemo.gridpilot.tech:8085/assetInventory/search?>

8.4.7 GridModelUpdate()

This service allows to store into the AIDB the json file which contains the grid model. It will only update the grid model of the use case to which the API User belongs.

Invocation example

PUT: <http://apidemo.gridpilot.tech:8085/assetInventory/update>

8.4.8 AssetRiskQuery()

This service allows to know the list of active vulnerabilities of hosts belonging to the infrastructure. Notice, only active vulnerabilities with the max Risk Level by host will be returned. Hence, if a host has several vulnerabilities only that one with max Risk Level will be returned.

Table 44: AIDB – Asset risk query method

Field	Type	Req.	Description
Request			
No input information is need for the query.			
Response			
invExternalId	String	No	It is the ExternalId of the Asset.
invName	String	No	It is the Name of the Asset.
invDescription	String	No	Asset comments
sdnIP	String	Yes	IP address of the Host
sdnMAC	String	No	MAC address of the Host
cve_id	String	Yes	Common Vulnerabilities and Exposures identifier
cvss_score	Number	Yes	Common Vulnerability Scoring System
sdnPort	Number	No	Port number to which the vulnerability has been detected

Invocation example:

GET: https://apidemo.gridpilot.tech:8085/assetInventory/asset_risk_level

```
[
{
  "invExternalId": "10.0.0.1",
  "sdnIP": "10.0.0.1",
  "sdnMAC": "00:00:00:00:00:01",
  "vulnerability": [
```

```
    "cve_id": "CVE-2013-7518"
    "cvss_score": 5.45
    "sdnPort": null
  },
  {
    "cve_id": "CVE-2014-1234"
    "cvss_score": 9.45
```

```
    "sdnPort": null
  }
]
},
{
  "invExternalId": "10.0.0.2",
  "sdnIP": "10.0.0.2",
  "sdnMAC": "00:00:00:00:00:02",
  "vulnerability": [
    {
      "cve_id": "CVE-2013-7518"
```

```
    "cvss_score": 5.45
    "sdnPort": null
  },
  {
    "cve_id": "CVE-2014-1234"
    "cvss_score": 9.45
    "sdnPort": null
  }
]
}
```

9 Unit Testing

In this section it is detailed unit tests designed for each component. It is noted that EDAE component is divided in two sections. The first one regarding the EDAE core which aims to test the MILP algorithm once all the inputs are gathered in the proper format. On the other side the EDAE workflow aims to test the process of gathering all the inputs from different components to EDAE core and manage the outputs for the other components, in this section the orchestrator of the component it is tested before the integration phase arrives.

9.1 Northbound Interface unit tests

9.1.1 Technical environment

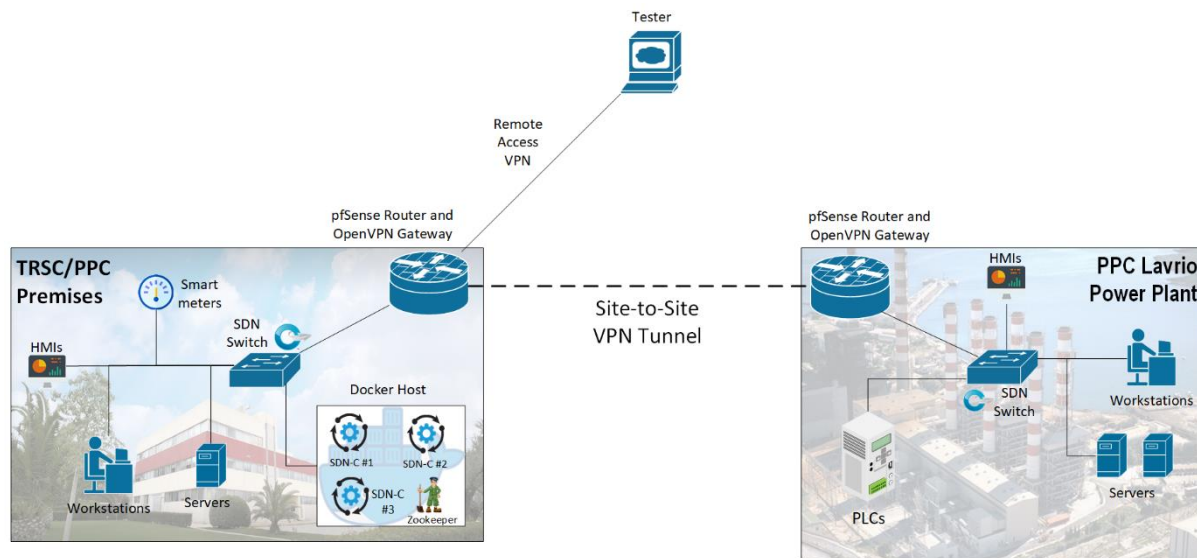


Figure 76: Testing environment of the NBI

The testing environment utilised to test the Northbound Interface replicates a realistic SDN-enabled pre-pilot environment and is depicted in Figure 76. The environment is comprised of two sites, namely the laboratories of TRSC/PPC (Athens, Greece) as well as the pilot environment of Use Case 3, located in the PPC power plant of Lavrio, Greece. Each site utilises one SDN switch (Aruba 2930F JL261A), where smart meters, sensors (TRSC) as well as PLCs and HMIs (Lavrio) are connected to. The two sites are interconnected via two virtualised *pfSense* gateways that form a site-to-site *OpenVPN* tunnel and remote access to the software tester is provided by the central *pfSense* gateway located in TRSC.

The SDN-Cs that integrate the NBI are hosted as Docker containers in a virtualised Docker host in TRSC, as depicted in the figure. The two SDN switches are connected to all SDN-Cs and each SDN-C communicates with SCS (Zookeeper) to determine its role. The details of the connection with SCS and the election process are covered by D4.1.

The testing procedure is initialised by deploying the SDN-Cs and initialising the OpenFlow tables of each switch. The SDN-Cs run a Ryu application that imitates the switch logic of a conventional switch, thus filling accordingly the flow tables and enable instant communication in both locations, thus enabling the generation of realistic traffic as well as realistic responses for the under-testing NBI.

9.1.2 Unit tests

Table 45: NBI_01 unit test description

Test Case ID	NBI_01	Component	SDN-C
Description	Test the retrieval of network statistics via the REST API		
Spec ID	SPEC-F6, XL-EPDS_NBI-2	Priority	High
Prepared by	UOWM, PPC	Tested by	UOWM, PPC
Pre-condition(s)	<ul style="list-style-type: none">At least one SDN-C has been deployedSCS has been deployed and is reachable by SDN-C(s)At least one SDN Switch is connected to at least one SDN-CThe role of SDN-C is MASTER or EQUAL		
Test steps			
1	The tester opens a web browser		
2	The tester provides the URL: https://192.168.86.240:10443/stats/port/35628848924874		
3	The tester provides any credentials, if requested		
4	The response is appeared in the browser in JSON format, depicting statistics from each port of the switch specified.		
Input data	https://192.168.86.240:10443/stats/portdesc/35628848924874		
Result	{"2876467493531616": {"rx_packets": 14580008, "tx_packets": 32909058, "rx_bytes": 4467889637, "tx_bytes": 2376423503, "rx_dropped": 0, "tx_dropped": 0, "rx_errors": 0, "tx_errors": 0, "rx_frame_err": 0, "rx_over_err": 0, "rx_crc_err": 0, "collisions": 0, "duration_sec": 1822004867, "duration_nsec": 2851118606, "port_no": 2}, {"rx_packets": 5205440, "tx_packets": 9731181, "rx_bytes": 342023577, "tx_bytes": 819547595, "rx_dropped": 0, "tx_dropped": 0, "rx_errors": 0, "tx_errors": 0, "rx_frame_err": 0, "rx_over_err": 0, "rx_crc_err": 0, "collisions": 0, "duration_sec": 1822004867, "duration_nsec": 3701710606, "port_no": 5}, {"rx_packets": 9803, "tx_packets": 3850080, "rx_bytes": 765830, "tx_bytes": 429961448, "rx_dropped": 0, "tx_dropped": 0, "rx_errors": 0, "tx_errors": 0, "rx_frame_err": 0, "rx_over_err": 0, "rx_crc_err": 0, "collisions": 0, "duration_sec": 1822004867, "duration_nsec": 1130012686, "port_no": 15}, {"rx_packets": 18773, "tx_packets": 4567898, "rx_bytes": 2126672, "tx_bytes": 474766906, "rx_dropped": 0, "tx_dropped": 0, "rx_errors": 0, "tx_errors": 0, "rx_frame_err": 0, "rx_over_err": 0, "rx_crc_err": 0, "collisions": 0, "duration_sec": 1822004867, "duration_nsec": 1124770062, "port_no": 7}, {"rx_packets": 58290579, "tx_packets": 32981632, "rx_bytes": 3907631191, "tx_bytes": 9358752714, "rx_dropped": 0, "tx_dropped": 0, "rx_errors": 0, "tx_errors": 0, "rx_frame_err": 0, "rx_over_err": 0, "rx_crc_err": 0, "collisions": 0, "duration_sec": 1822004867, "duration_nsec": 1136304398, "port_no": 1}, {"rx_packets": 14581391, "tx_packets": 32913091, "rx_bytes": 4468340219, "tx_bytes": 2376694962, "rx_dropped": 0, "tx_dropped": 0, "rx_errors": 0, "tx_errors": 0, "rx_frame_err": 0, "rx_over_err": 0, "rx_crc_err": 0, "collisions": 0, "duration_sec": 1822004867, "duration_nsec": 1986896398, "port_no": 4}, {"rx_packets": 997042, "tx_packets": 2771912, "rx_bytes": 88495675, "tx_bytes": 334643169, "rx_dropped": 0, "tx_dropped": 0, "rx_errors": 0, "tx_errors": 0, "rx_frame_err": 0, "rx_over_err": 0, "rx_crc_err": 0, "collisions": 0, "duration_sec": 1822004867, "duration_nsec": 2837488398, "port_no": 0}}		

	<pre> 3}, {"rx_packets": 9739, "tx_packets": 3850048, "rx_bytes": 755491, "tx_bytes": 429953898, "rx_dropped": 0, "tx_dropped": 0, "rx_errors": 0, "tx_errors": 0, "rx_frame_err": 0, "rx_over_err": 0, "rx_crc_err": 0, "collisions": 0, "duration_sec": 1822004867, "duration_nsec": 1993188110, "port_no": 16}, {"rx_packets": 6937526, "tx_packets": 7852196, "rx_bytes": 482244410, "tx_bytes": 668709662, "rx_dropped": 0, "tx_dropped": 0, "rx_errors": 0, "tx_errors": 0, "rx_frame_err": 0, "rx_over_err": 0, "rx_crc_err": 0, "collisions": 0, "duration_sec": 1822004867, "duration_nsec": 2843780110, "port_no": 6}, {"rx_packets": 0, "tx_packets": 0, "rx_bytes": 0, "tx_bytes": 0, "rx_dropped": 0, "tx_dropped": 0, "rx_errors": 0, "tx_errors": 0, "rx_frame_err": 0, "rx_over_err": 0, "rx_crc_err": 0, "collisions": 0, "duration_sec": 1822004867, "duration_nsec": 1983751438, "port_no": 14}, {"rx_packets": 123835, "tx_packets": 3714472, "rx_bytes": 29627390, "tx_bytes": 397983823, "rx_dropped": 0, "tx_dropped": 0, "rx_errors": 9, "tx_errors": 0, "rx_frame_err": 0, "rx_over_err": 0, "rx_crc_err": 9, "collisions": 0, "duration_sec": 1822004867, "duration_nsec": 284665102, "port_no": 10}, {"rx_packets": 123874, "tx_packets": 4435003, "rx_bytes": 29636808, "tx_bytes": 442432671, "rx_dropped": 0, "tx_dropped": 0, "rx_errors": 0, "tx_errors": 0, "rx_frame_err": 0, "rx_over_err": 0, "rx_crc_err": 0, "collisions": 0, "duration_sec": 1822004867, "duration_nsec": 3702760974, "port_no": 9}, {"rx_packets": 0, "tx_packets": 0, "rx_bytes": 0, "tx_bytes": 0, "rx_dropped": 0, "tx_dropped": 0, "rx_errors": 0, "tx_errors": 0, "rx_frame_err": 0, "rx_over_err": 0, "rx_crc_err": 0, "collisions": 0, "duration_sec": 1822004867, "duration_nsec": 1137354254, "port_no": "LOCAL"}}} </pre>
Test Case Result	Achieved

Table 46: NBI_02 unit test description

Test Case ID	NBI_02	Component	SDN-C
Description	Test the retrieval of the network topology via the REST API		
Spec ID	SPEC-F6, XL-EPDS_NBI-1, AIDB-3	Priority	High
Prepared by	UOWM, PPC	Tested by	UOWM, PPC
Pre-condition(s)	<ul style="list-style-type: none">At least one SDN-C has been deployedSCS has been deployed and is reachable by SDN-C(s)At least one SDN Switch is connected to at least one SDN-CThe role of SDN-C is MASTER or EQUAL		
Test steps			
1	The tester opens a web browser		
2	The tester provides the URL: https://192.168.86.240:30443/v1.0/topology/switches		
3	The tester provides any credentials, if requested		
4	The response is appeared in the browser in JSON format, which includes all switches		
5	The tester provides the URL: https://192.168.86.240:30443/v1.0/topology/hosts		
6	The tester provides any credentials, if requested		
7	The response is appeared in the browser in JSON format, including all hosts		

8	The tester provides the URL: https://192.168.86.240:30443/v1.0/topology/links
9	The response is appeared in the browser in JSON format, which includes all links between switches
Input data	https://192.168.86.240:30443/v1.0/topology/switches https://192.168.86.240:30443/v1.0/topology/hosts https://192.168.86.240:30443/v1.0/topology/links
Result	<pre>[{"dpid": "2876467493531616", "ports": [{"dpid": "2876467493531616", "port_no": "2", "hw_addr": "38:21:c7:28:ef:fe", "name": "2"}, {"dpid": "2876467493531616", "port_no": "5", "hw_addr": "38:21:c7:28:ef:fb", "name": "5"}, {"dpid": "2876467493531616", "port_no": "15", "hw_addr": "38:21:c7:28:ef:f1", "name": "15"}, {"dpid": "2876467493531616", "port_no": "7", "hw_addr": "38:21:c7:28:ef:f9", "name": "7"}, {"dpid": "2876467493531616", "port_no": "1", "hw_addr": "38:21:c7:28:ef:ff", "name": "1"}, {"dpid": "2876467493531616", "port_no": "4", "hw_addr": "38:21:c7:28:ef:fc", "name": "4"}, {"dpid": "2876467493531616", "port_no": "3", "hw_addr": "38:21:c7:28:ef:fd", "name": "3"}, {"dpid": "2876467493531616", "port_no": "16", "hw_addr": "38:21:c7:28:ef:f0", "name": "16"}, {"dpid": "2876467493531616", "port_no": "6", "hw_addr": "38:21:c7:28:ef:fa", "name": "6"}, {"dpid": "2876467493531616", "port_no": "14", "hw_addr": "38:21:c7:28:ef:f2", "name": "14"}, {"dpid": "2876467493531616", "port_no": "10", "hw_addr": "38:21:c7:28:ef:f6", "name": "10"}, {"dpid": "2876467493531616", "port_no": "9", "hw_addr": "38:21:c7:28:ef:f7", "name": "9"}]}, {"dpid": "2876467493016320", "ports": [{"dpid": "2876467493016320", "port_no": "2", "hw_addr": "38:21:c7:21:13:1e", "name": "2"}, {"dpid": "2876467493016320", "port_no": "5", "hw_addr": "38:21:c7:21:13:1b", "name": "5"}, {"dpid": "2876467493016320", "port_no": "16", "hw_addr": "38:21:c7:21:13:10", "name": "16"}, {"dpid": "2876467493016320", "port_no": "9", "hw_addr": "38:21:c7:21:13:17", "name": "9"}, {"dpid": "2876467493016320", "port_no": "1", "hw_addr": "38:21:c7:21:13:1f", "name": "1"}, {"dpid": "2876467493016320", "port_no": "4", "hw_addr": "38:21:c7:21:13:1c", "name": "4"}, {"dpid": "2876467493016320", "port_no": "3", "hw_addr": "38:21:c7:21:13:1d", "name": "3"}, {"dpid": "2876467493016320", "port_no": "6", "hw_addr": "38:21:c7:21:13:1a", "name": "6"}, {"dpid": "2876467493016320", "port_no": "15", "hw_addr": "38:21:c7:21:13:11", "name": "15"}, {"dpid": "2876467493016320", "port_no": "14", "hw_addr": "38:21:c7:21:13:12", "name": "14"}, {"dpid": "2876467493016320", "port_no": "10", "hw_addr": "38:21:c7:21:13:16", "name": "10"}]}], [{"mac": "00:0d:22:22:84:94", "ipv4": ["192.168.20.251"], "ipv6": [], "port": {"dpid": "2876467493531616", "port_no": "5", "hw_addr": "38:21:c7:28:ef:fb", "name": "5"}}, {"mac": "20:67:7c:e2:48:ca", "ipv4": [], "ipv6": [], "port": {"dpid": "2876467493531616", "port_no": "6", "hw_addr": "38:21:c7:28:ef:fa", "name": "6"}}, {"mac": "20:67:7c:e2:48:c9", "ipv4": [], "ipv6": [], "port": {"dpid": "2876467493531616", "port_no": "3", "hw_addr": "38:21:c7:28:ef:fd", "name": "3"}}, {"mac": "28:80:23:ac:99:7e", "ipv4": [], "ipv6": [], "port": {"dpid": "2876467493531616", "port_no": "10", "hw_addr": "38:21:c7:28:ef:f6", "name": "10"}}, {"mac": "28:80:23:ac:99:7d", "ipv4": [], "ipv6": [], "port": {"dpid": "2876467493531616", "port_no": "9", "hw_addr": "38:21:c7:28:ef:f7", "name": "9"}}, {"mac": "00:19:ee:10:a3:2e", "ipv4": ["192.168.10.15"], "ipv6": [], "port": {"dpid": "2876467493531616", "port_no": "4", "hw_addr": "38:21:c7:28:ef:fc", "name": "4"}}, {"mac": "00:19:ee:10:94:94", "ipv4": ["192.168.10.10"], "ipv6": [], "port": {"dpid": "2876467493531616", "port_no": "2", "hw_addr": "38:21:c7:28:ef:fe", "name": "2"}},</pre>

	{"mac": "dc:a6:32:70:14:29", "ipv4": [], "ipv6": [], "port": {"dpid": "2876467493531616", "port_no": "16", "hw_addr": "38:21:c7:28:ef:f0", "name": "16"}}
Test Case Result	Achieved

Table 47: NBI_03 unit test description

Test Case ID	NBI_03	Component	SDN-C
Description	Test that TLS has been properly configured on the NBI		
Spec ID	S-RAF_NBI-1, CONS-T1, CONS-L1	Priority	High
Prepared by	UOWM, PPC	Tested by	UOWM, PPC
Pre-condition(s)	<ul style="list-style-type: none">At least one SDN-C has been deployedSCS has been deployed and is reachable by SDN-C(s)At least one SDN Switch is connected to at least one SDN-CThe role of SDN-C is MASTER or EQUAL		
Test steps			
1	The tester opens a web browser		
2	The tester starts a packet capture using Wireshark		
3	The tester provides a REST API command, e.g. https://192.168.86.240:30443/v1.0/topology/hosts		
4	The tester verifies that the session with SDN-C is encrypted by checking the relevant indication of the web browser		
5	The tester stops the Wireshark capture and notices that the session is encrypted using TLS 1.2		
Input data	None		

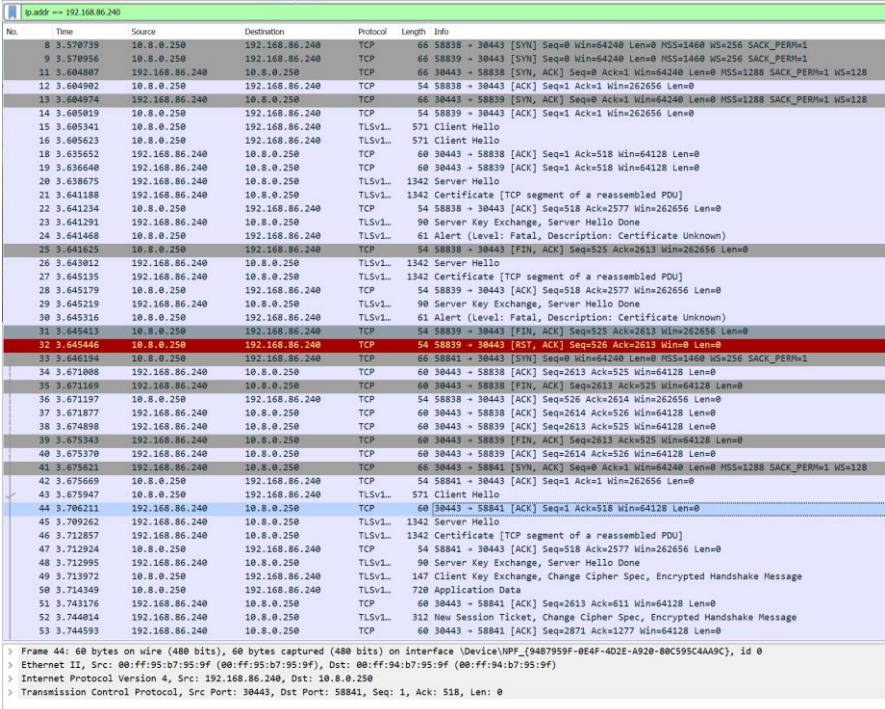
Result	<p>By investigating the TLS negotiation depicted below, is concluded that the TLS session is secured since TLS 1.2 is used as well as the TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 cipher. Although, the certificate used by the SDN-C is signed by a self-signed Certificate Authority (CA) deployed in TRSC/PPC premises. This results to the error “Certificate Unknown” generated by the browser, which is latter ignored. However, this error is not considered significant, since the user is able to overcome this error by adding the CA into to Trusted Root Certificate Authorities, hence trusting the CA.</p>  <p style="text-align: center;">Figure 77: TLS traffic of NBI - Unit tests</p>
Test Case Result	Achieved, to be tested in Pilot

Table 48: NBI_04 test unit description

Test Case ID	NBI_04	Component	SDN-C
Description	Test that authorisation is required for accessing the northbound interface		
Spec ID	S-RAF_NBI-1, CONS-T2	Priority	High
Prepared by	UOWM, PPC	Tested by	UOWM, PPC
Pre-condition(s)	<ul style="list-style-type: none">At least one SDN-C has been deployedSCS has been deployed and is reachable by SDN-C(s)At least one SDN Switch is connected to at least one SDN-CThe role of SDN-C is MASTER or EQUAL		
Test steps			
1	The tester opens a web browser		

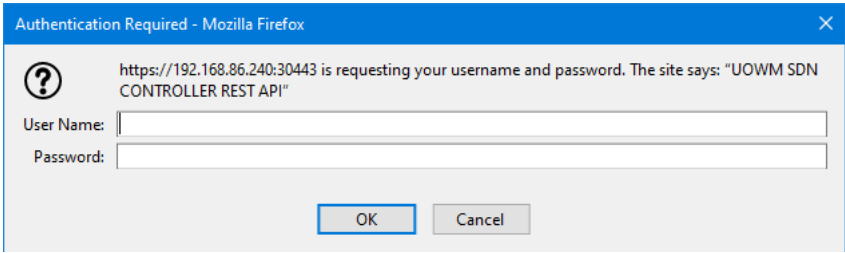

2	The tester provides a REST API command, e.g. https://192.168.86.240:30443/v1.0/topology/hosts
3	The web browser prompts for Authentication 
4	The tester provides false credentials
Input data	None
Result	The HTTP Error 401 – Unauthorized is displayed 
Test Case Result	Achieved, to be tested in Pilot

Table 49: NBI_05 test unit description

Test Case ID	NBI_05	Component	SDN-C
Description	A network flow is successfully added via the REST API		
Spec ID	SPEC-F6, S-RAF_NBI-1, XL-EPDS_NBI-1	Priority	High
Prepared by	UOWM, PPC	Tested by	UOWM, PPC
Pre-condition(s)	<ul style="list-style-type: none">At least one SDN-C has been deployedSCS has been deployed and is reachable by SDN-C(s)At least one SDN Switch is connected to at least one SDN-CThe role of SDN-C is MASTER or EQUAL		
Test steps			
1	The tester opens a terminal and issues the following command: curl --location --request POST 'https://192.168.86.240:30443/stats/flowentry/add' -u 'user:secret' --data-raw '{"dpid":2876467493531616,"priority":15,"cookie":0,"idle_timeout":0,"hard_timeout":0,"actions":[{"type":"WRITE_ACTIONS","actions":[{"type":"OUTPUT","port":2}]}],"match":{"in_port":1,"ipv4_src":"192.168.10.1","ipv4_dst":"192.168.10.2","ip_proto": 6,"tcp_dst": 8800,"eth_type": 2048},"table_id": 2}' --insecure		
2	The tester accesses the URL: https://192.168.86.240:10443/stats/flow/2876467493531616 to retrieve the flow table, noticing that the specified flow entry has been added		
Input data	A POST request with the appropriate body request, as defined in step 1		

Result	<p>The flow table, including the new flow:</p> <pre>{ "2876467493531616": [{"priority": 1, "cookie": 0, "idle_timeout": 0, "hard_timeout": 0, "byte_count": 0, "duration_sec": 18154, "duration_nsec": 631000000, "packet_count": 0, "length": 80, "flags": 0, "actions": ["GOTO_TABLE:1"], "match": {"eth_dst": "ff:ff:ff:ff:ff:ff", "eth_type": 34984}, "table_id": 0}, {"priority": 1, "cookie": 0, "idle_timeout": 0, "hard_timeout": 0, "byte_count": 0, "duration_sec": 18154, "duration_nsec": 631000000, "packet_count": 0, "length": 80, "flags": 0, "actions": ["GOTO_TABLE:1"], "match": {"eth_dst": "ff:ff:ff:ff:ff:ff", "eth_type": 33024}, "table_id": 0}, {"priority": 1, "cookie": 0, "idle_timeout": 0, "hard_timeout": 0, "byte_count": 739200, "duration_sec": 18154, "duration_nsec": 631000000, "packet_count": 12320, "length": 80, "flags": 0, "actions": ["GOTO_TABLE:1"], "match": {"eth_dst": "ff:ff:ff:ff:ff:ff", "eth_type": 2054}, "table_id": 0}, {"priority": 65535, "cookie": 0, "idle_timeout": 0, "hard_timeout": 0, "byte_count": 7558172, "duration_sec": 18154, "duration_nsec": 631000000, "packet_count": 54231, "length": 96, "flags": 0, "actions": [{"WRITE_ACTIONS": ["OUTPUT:CONTROLLER"]}], "match": {"eth_dst": "01:80:c2:00:00:0e", "eth_type": 35020}, "table_id": 0}, {"priority": 1, "cookie": 0, "idle_timeout": 0, "hard_timeout": 0, "byte_count": 244213, "duration_sec": 18154, "duration_nsec": 631000000, "packet_count": 1120, "length": 80, "flags": 0, "actions": ["GOTO_TABLE:1"], "match": {"eth_dst": "ff:ff:ff:ff:ff:ff", "eth_type": 2048}, "table_id": 0}, {"priority": 0, "cookie": 0, "idle_timeout": 0, "hard_timeout": 0, "byte_count": 4949779237, "duration_sec": 18154, "duration_nsec": 631000000, "packet_count": 32553918, "length": 64, "flags": 0, "actions": ["GOTO_TABLE:2"], "match": {}, "table_id": 0}, {"priority": 0, "cookie": 0, "idle_timeout": 0, "hard_timeout": 0, "byte_count": 983413, "duration_sec": 18154, "duration_nsec": 631000000, "packet_count": 13440, "length": 64, "flags": 0, "actions": ["GOTO_TABLE:2"], "match": {}, "table_id": 1}, {"priority": 15, "cookie": 0, "idle_timeout": 0, "hard_timeout": 0, "byte_count": 0, "duration_sec": 206, "duration_nsec": 766000000, "packet_count": 0, "length": 120, "flags": 0, "actions": [{"WRITE_ACTIONS": ["OUTPUT:2"]}], "match": {"in_port": 1, "eth_type": 2048, "ipv4_src": "192.168.10.1", "ipv4_dst": "192.168.10.2", "ip_proto": 6, "tcp_dst": 8800}, "table_id": 2}, {"priority": 0, "cookie": 0, "idle_timeout": 0, "hard_timeout": 0, "byte_count": 4950762650, "duration_sec": 18154, "duration_nsec": 631000000, "packet_count": 32567358, "length": 64, "flags": 0, "actions": ["GOTO_TABLE:3"], "match": {}, "table_id": 2}, {"priority": 1, "cookie": 0, "idle_timeout": 0, "hard_timeout": 0, "byte_count": 0, "duration_sec": 18154, "duration_nsec": 631000000, "packet_count": 0, "length": 104, "flags": 0, "actions": ["OUTPUT:3"], "match": {"in_port": 6, "eth_dst": "46:79:fd:32:b5:17", "eth_src": "7e:51:7c:23:91:3e"}, "table_id": 3}]</pre>
Test Case Result	Achieved

Table 50: NBI-06 test unit description

Test Case ID	NBI_06	Component	SDN-C
Description	A network flow is successfully deleted via the REST API		
Spec ID	SPEC-F6, S-RAF_NBI-1, XL-EPDS_NBI-1	Priority	High
Prepared by	UOWM, PPC	Tested by	UOWM, PPC
Pre-condition(s)	<ul style="list-style-type: none"> At least one SDN-C has been deployed SCS has been deployed and is reachable by SDN-C(s) 		

		<ul style="list-style-type: none"> At least one SDN Switch is connected to at least one SDN-C The role of SDN-C is MASTER or EQUAL
Test steps		
1	The tester opens a terminal and issues the following command <pre>curl --location --request POST --user user:secret 'https://192.168.86.240:10443/stats/flowentry/delete_strict' --data-raw '{"dpid":2876467493531616,"cookie":0,"cookie_mask":0,"table_id":2,"match":{"in_port":1}}'</pre>	
4	The tester opens a browser and provides the URL: https://192.168.86.240:10443/stats/flow/2876467493531616 to retrieve the flow table, noticing that the specified flow does not exist anymore	
Input data		A POST request with the appropriate body request, as defined in step 1
Result		The flow table does not include the deleted flow: <pre>{ "2876467493531616": [{"priority": 1, "cookie": 0, "idle_timeout": 0, "hard_timeout": 0, "byte_count": 0, "duration_sec": 18154, "duration_nsec": 631000000, "packet_count": 0, "length": 80, "flags": 0, "actions": ["GOTO_TABLE:1"], "match": {"eth_dst": "ff:ff:ff:ff:ff:ff", "eth_type": 34984}, "table_id": 0}, {"priority": 1, "cookie": 0, "idle_timeout": 0, "hard_timeout": 0, "byte_count": 0, "duration_sec": 18154, "duration_nsec": 631000000, "packet_count": 0, "length": 80, "flags": 0, "actions": ["GOTO_TABLE:1"], "match": {"eth_dst": "ff:ff:ff:ff:ff:ff", "eth_type": 33024}, "table_id": 0}, {"priority": 1, "cookie": 0, "idle_timeout": 0, "hard_timeout": 0, "byte_count": 739200, "duration_sec": 18154, "duration_nsec": 631000000, "packet_count": 12320, "length": 80, "flags": 0, "actions": ["GOTO_TABLE:1"], "match": {"eth_dst": "ff:ff:ff:ff:ff:ff", "eth_type": 2054}, "table_id": 0}, {"priority": 65535, "cookie": 0, "idle_timeout": 0, "hard_timeout": 0, "byte_count": 7558172, "duration_sec": 18154, "duration_nsec": 631000000, "packet_count": 54231, "length": 96, "flags": 0, "actions": [{"WRITE_ACTIONS": ["OUTPUT:CONTROLLER"]}], "match": {"eth_dst": "01:80:c2:00:00:0e", "eth_type": 35020}, "table_id": 0}, {"priority": 1, "cookie": 0, "idle_timeout": 0, "hard_timeout": 0, "byte_count": 244213, "duration_sec": 18154, "duration_nsec": 631000000, "packet_count": 1120, "length": 80, "flags": 0, "actions": ["GOTO_TABLE:1"], "match": {"eth_dst": "ff:ff:ff:ff:ff:ff", "eth_type": 2048}, "table_id": 0}, {"priority": 0, "cookie": 0, "idle_timeout": 0, "hard_timeout": 0, "byte_count": 4949779237, "duration_sec": 18154, "duration_nsec": 631000000, "packet_count": 32553918, "length": 64, "flags": 0, "actions": ["GOTO_TABLE:2"], "match": {}, "table_id": 0}, {"priority": 0, "cookie": 0, "idle_timeout": 0, "hard_timeout": 0, "byte_count": 983413, "duration_sec": 18154, "duration_nsec": 631000000, "packet_count": 13440, "length": 64, "flags": 0, "actions": ["GOTO_TABLE:2"], "match": {}, "table_id": 1}, {"priority": 0, "cookie": 0, "idle_timeout": 0, "hard_timeout": 0, "byte_count": 4950762650, "duration_sec": 18154, "duration_nsec": 631000000, "packet_count": 32567358, "length": 64, "flags": 0, "actions": ["GOTO_TABLE:3"], "match": {}, "table_id": 2}, {"priority": 1, "cookie": 0, "idle_timeout": 0, "hard_timeout": 0, "byte_count": 0, "duration_sec": 18154, "duration_nsec": 631000000, "packet_count": 0, "length": 104, "flags": 0, "actions": ["OUTPUT:3"], "match": {"in_port": 6, "eth_dst": "46:79:fd:32:b5:17", "eth_src": "7e:51:7c:23:91:3e"}, "table_id": 3}]</pre>
Test Case Result		Achieved

Table 51: NBI_07 test unit description

Test Case ID	NBI_07	Component	SDN-C
Description	Retrieval of network flows in compatible format.		
Spec ID	SPEC-F6	Priority	High
Prepared by	UOWM, PPC	Tested by	UOWM, PPC
Pre-condition(s)	<ul style="list-style-type: none">At least one SDN-C has been deployedSCS has been deployed and is reachable by SDN-C(s)At least one SDN Switch is connected to at least one SDN-CThe role of SDN-C is MASTER or EQUAL		
Test steps			
1	The tester opens a web browser		
2	The tester provides the URL: https://192.168.86.240:10443/stats/flow/2876467493531616		
3	The tester provides any credentials, if requested		
4	The response is appeared in the browser in JSON format, including all flow entries of the switch specified. Additionally, the dl_st and dl_src have been replaced by eth_dst and eth_src, thus achieving compliance with OpenFlow 1.3.		
Input data	https://192.168.86.240:10443/stats/flow/2876467493531616		
Result	{ "2876467493531616": [{"priority": 1, "cookie": 0, "idle_timeout": 0, "hard_timeout": 0, "byte_count": 0, "duration_sec": 18154, "duration_nsec": 631000000, "packet_count": 0, "length": 80, "flags": 0, "actions": ["GOTO_TABLE:1"], "match": {"eth_dst": "ff:ff:ff:ff:ff:ff", "eth_type": 34984}, "table_id": 0}, {"priority": 1, "cookie": 0, "idle_timeout": 0, "hard_timeout": 0, "byte_count": 0, "duration_sec": 18154, "duration_nsec": 631000000, "packet_count": 0, "length": 80, "flags": 0, "actions": ["GOTO_TABLE:1"], "match": {"eth_dst": "ff:ff:ff:ff:ff:ff", "eth_type": 33024}, "table_id": 0}, {"priority": 1, "cookie": 0, "idle_timeout": 0, "hard_timeout": 0, "byte_count": 739200, "duration_sec": 18154, "duration_nsec": 631000000, "packet_count": 12320, "length": 80, "flags": 0, "actions": ["GOTO_TABLE:1"], "match": {"eth_dst": "ff:ff:ff:ff:ff:ff", "eth_type": 2054}, "table_id": 0}, {"priority": 65535, "cookie": 0, "idle_timeout": 0, "hard_timeout": 0, "byte_count": 7558172, "duration_sec": 18154, "duration_nsec": 631000000, "packet_count": 54231, "length": 96, "flags": 0, "actions": [{"WRITE_ACTIONS": ["OUTPUT:CONTROLLER"]}], "match": {"eth_dst": "01:80:c2:00:00:0e", "eth_type": 35020}, "table_id": 0}, {"priority": 1, "cookie": 0, "idle_timeout": 0, "hard_timeout": 0, "byte_count": 244213, "duration_sec": 18154, "duration_nsec": 631000000, "packet_count": 1120, "length": 80, "flags": 0, "actions": ["GOTO_TABLE:1"], "match": {"eth_dst": "ff:ff:ff:ff:ff:ff", "eth_type": 2048}, "table_id": 0}, {"priority": 0, "cookie": 0, "idle_timeout": 0, "hard_timeout": 0, "byte_count": 4949779237, "duration_sec": 18154, "duration_nsec": 631000000, "packet_count": 32553918, "length": 64, "flags": 0, "actions": ["GOTO_TABLE:2"], "match": {}, "table_id": 0}, {"priority": 0, "cookie": 0, "idle_timeout": 0, "hard_timeout": 0, "byte_count": 983413, "duration_sec": 18154, "duration_nsec": 631000000, "packet_count": 13440, "length": 64, "flags": 0, "actions": ["GOTO_TABLE:2"], "match": {}, "table_id": 1}, {"priority": 0, "cookie": 0, "idle_timeout": 0, "hard_timeout": 0, "byte count": 0, "duration_sec": 0, "duration_nsec": 0, "packet_count": 0, "length": 0, "flags": 0, "actions": [], "match": {}, "table_id": 0}] }		

	4950762650, "duration_sec": 18154, "duration_nsec": 631000000, "packet_count": 32567358, "length": 64, "flags": 0, "actions": ["GOTO_TABLE:3"], "match": {}, "table_id": 2}, {"priority": 1, "cookie": 0, "idle_timeout": 0, "hard_timeout": 0, "byte_count": 0, "duration_sec": 18154, "duration_nsec": 631000000, "packet_count": 0, "length": 104, "flags": 0, "actions": ["OUTPUT:3"], "match": {"in_port": 6, "eth_dst": "46:79:fd:32:b5:17", "eth_src": "7e:51:7c:23:91:3e"}, "table_id": 3}]
Test Case Result	Achieved

Table 52: NBI_08 test unit description

Test Case ID	NBI_08	Component	SDN-C
Description	Retrieval of switch information		
Spec ID	SPEC-F6, XL-EPDS_NBI-1, AIDB-3	Priority	High
Prepared by	UOWM, PPC	Tested by	UOWM, PPC
Pre-condition(s)	<ul style="list-style-type: none">At least one SDN-C has been deployedSCS has been deployed and is reachable by SDN-C(s)At least one SDN Switch is connected to at least one SDN-CThe role of SDN-C is MASTER or EQUAL		
Test steps			
1	The tester opens a web browser		
2	The tester provides the URL: https://192.168.86.240:10443/stats/desc/2876467493016320		
3	The tester provides any credentials, if requested		
4	The response is appeared in the browser in JSON format, depicting general information of the switch specified		
Input data	https://192.168.86.240:10443/stats/desc/2876467493016320		
Result	Note that the results bellow have been filtered to remove sensitive information (device serial number): {"2876467493016320": {"mfr_desc": "Aruba", "hw_desc": "2930F-24G-PoE+-4SFP Switch", "sw_desc": "WC.16.05.0007", "serial_num": "*****", "dp_desc": "1"}}		
Test Case Result	Achieved		

9.2 SDN Dashboard

9.2.1 Technical environment

The testing environment utilised to conduct the unit testing of the SDN dashboard is depicted in Figure 78. The underlying infrastructure is the same as described in the Northbound Interface unit testing section. The additional component of this unit testing section, the SDN dashboard, is hosted in the tester's premises by running the Django server in development mode using the PyCharm IDE.

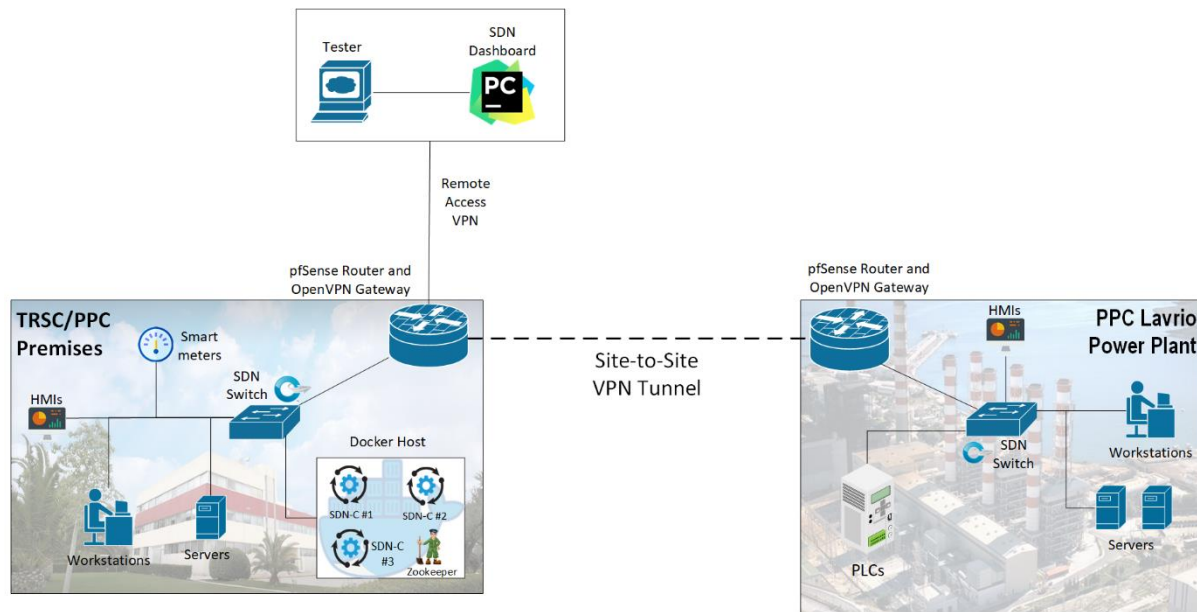


Figure 78: Testing environment of the SDN dashboard

9.2.2 Unit tests

Table 53: SDNGUI_01 unit test description

Test Case ID	SDNGUI_01	Component	SDN dashboard
Description	All SDN switches are accessible from the SDN Dashboard by using the corresponding master controller		
Spec ID	SPEC-F6, SPEC-OP1	Priority	High
Prepared by	UOWM, PPC	Tested by	UOWM, PPC
Pre-condition(s)	At least one SDN-C is deployed and connected to at least one SDN switch		
Test steps			
1	The tester starts Zookeeper Watcher to initialise the database containing the SDN switches, the SDN-Cs, and the relationships amongst them.		
2	The tester uses the PyCharm IDE to launch the Django server hosting the SDN dashboard		
3	The tester visits the SDN dashboard at http://127.0.0.1		
4	The tester provides the necessary login credentials to access the system		
5	The tester is redirected to the SDN dashboard homepage		
6	The tester notices in the homepage that port and table statistics are illustrated for each switch		
7	The tester visits the Flow page and views the flow entries of each switch		
Input data			

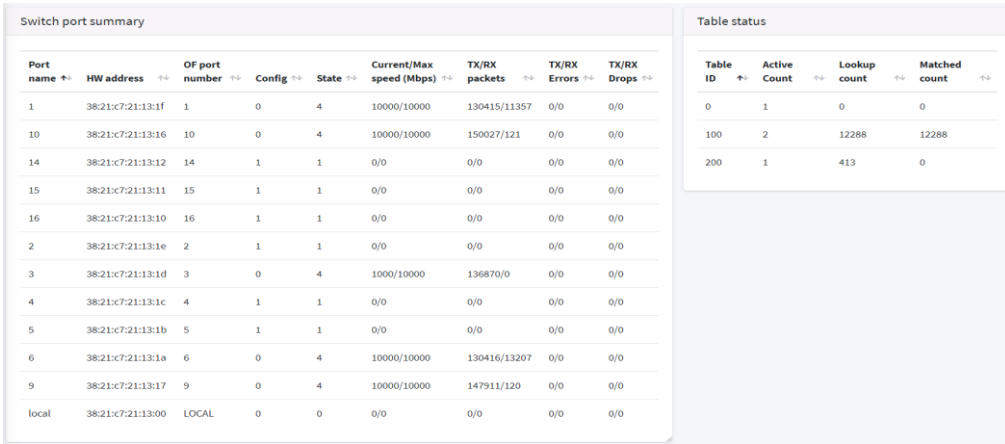
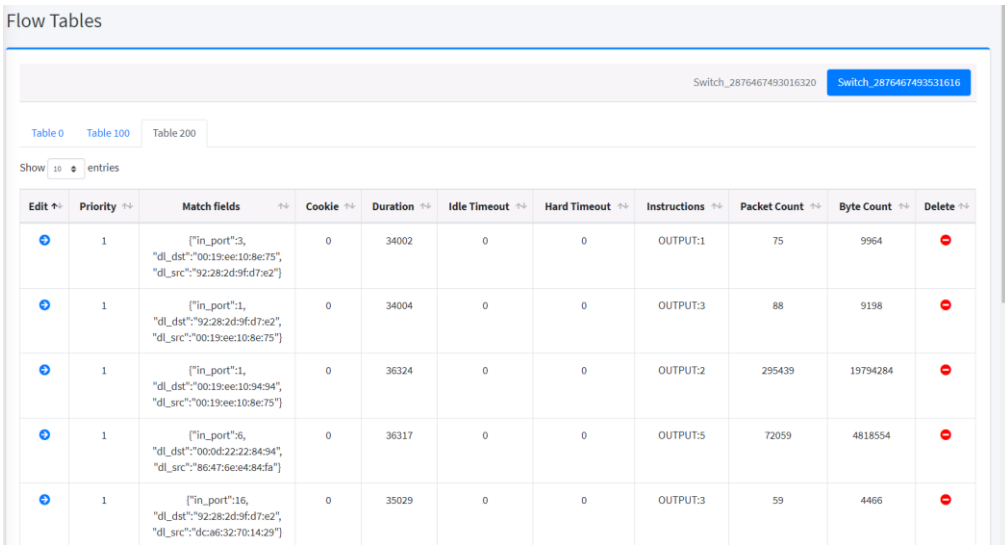
Result	<p>The statistics for each switch are depicted in the homepage</p>  <p>The flows are illustrated in the Flows page</p> 
	<p>Test Case Result</p> <p>Achieved, to be tested in Pilot</p>

Table 54: SDNGUI_02 unit test description

Test Case ID	SDNGUI_02	Component	SDN dashboard
Description	The SDN dashboard home page changes asynchronously if the role of an SDN-C changes or a new SDN-C is connected to an SDN switch		
Spec ID	SPEC-F6	Priority	Medium
Prepared by	UOWM, PPC	Tested by	UOWM, PPC
Pre-condition(s)	At least one SDN-C is deployed and connected to at least one SDN switch		
Test steps			

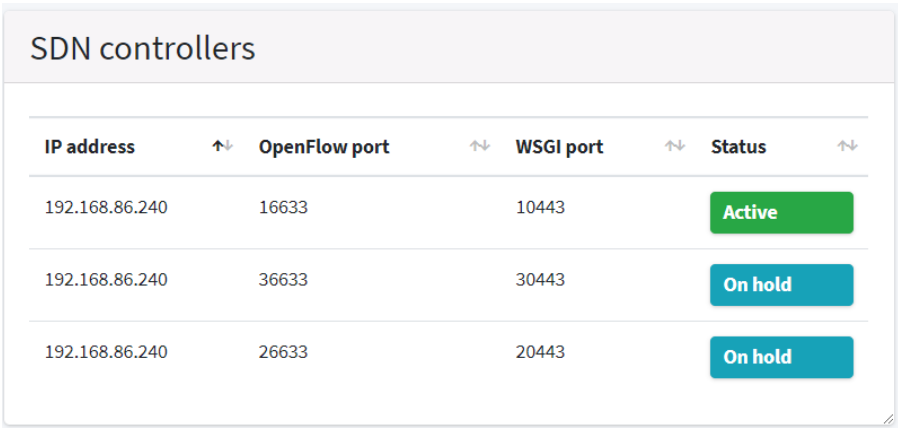
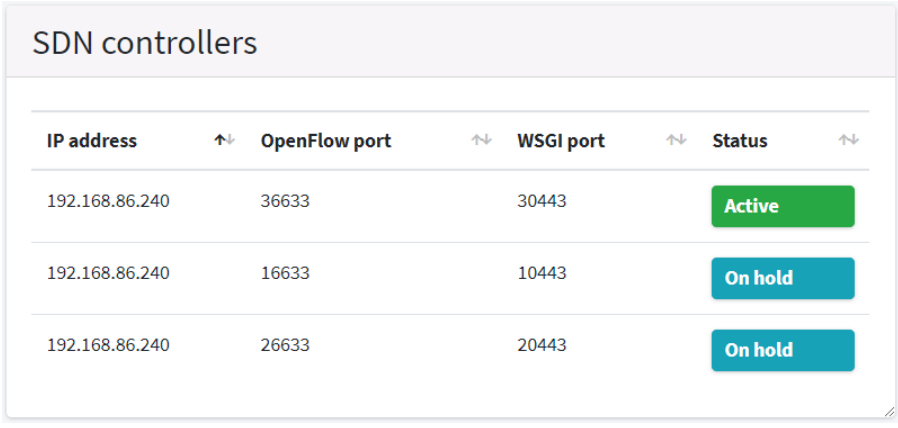
1	The tester starts Zookeeper Watcher to initialise the database containing the SDN switches, the SDN-Cs, and the relationships amongst them.
2	The tester uses the PyCharm IDE to launch the Django server hosting the SDN dashboard
3	The tester visits the SDN dashboard at http://127.0.0.1
4	The tester identifies the master SDN-C through the SDN dashboard
5	Since the SDN-C is deployed as Docker container, the tester connects to the corresponding Docker host and shuts down the identified SDN-C
6	The tester notices that the master SDN-C is changed in the dashboard without needing to refresh the home page
Input data	
Result	<p>The SDN Controllers status before the manual disruption</p>  <p>The SDN Controllers after the manual disruption</p> 
Test Case Result	Achieved, to be tested in Pilot

Table 55: SDNGUI_03 unit test description

Test Case ID	SDNGUI_03	Component	SDN dashboard
---------------------	-----------	------------------	---------------

Description	The SDN dashboard permits only authorised access to the system.		
Spec ID		Priority	Medium
Prepared by	UOWM, PPC	Tested by	UOWM, PPC
Pre-condition(s)	At least one SDN-C is deployed and connected to at least one SDN switch		
Test steps			
1	The tester starts Zookeeper Watcher to initialise the database containing the SDN switches, the SDN-Cs, and the relationships amongst them.		
2	The tester uses the PyCharm IDE to launch the Django server hosting the SDN dashboard		
3	The tester visits a random page in the SDN dashboard: http://127.0.0.1/flows		
4	The remote web server redirects the tester to the login page		
5	The steps 1-3 are repeated for each URL endpoint specified by the SDN dashboard		
Input data			
Result	For all URL endpoints, the SDN dashboard redirects the unauthenticated user to the login page.		
Test Case Result	Achieved		

Table 56: SDNGUI_04 unit test description

Test Case ID	SDNGUI_04	Component	SDN dashboard
Description	The SDN dashboard limits access to users depending on their role		
Req ID		Priority	Medium
Prepared by	UOWM, PPC	Tested by	UOWM, PPC
Pre-condition(s)	At least one SDN-C is deployed and connected to at least one SDN switch		
Test steps			
1	The tester starts Zookeeper Watcher to initialise the database containing the SDN switches, the SDN-Cs, and the relationships amongst them.		
2	The tester uses the PyCharm IDE to launch the Django server hosting the SDN dashboard		
3	The tester logs in the SDN dashboard using the credentials of a simple user		
4	The tester tries to access the page, e.g., http://127.0.0.1:8000/flows/edit and a 403 error page is displayed.		
5	The tester logs out and logs in by using the credentials of a user holding the Security Administrator role		
6	The tester successfully gains access to the page http://127.0.0.1:8000/flows/edit		

7	The tester tries to access the User Management System: http://127.0.0.1:8000/users , but a 403-error page is displayed
8	The tester logs out and logs in by using the credentials of a user holding the superuser role
9	The tester successfully gains access to the page http://127.0.0.1:8000/users
Input data	Attempt to access a page that requires elevated privileges.
Result	For the restricted pages, the SDN dashboard renders a HTTP 403 error (Forbidden) page, indicating that the logged-in user does not have the appropriate privileges for accessing the page.
Test Case Result	Achieved

Table 57: SDNGUI_05 unit test description

Test Case ID	SDNGUI_05	Component	SDN dashboard																																												
Description	The SDN dashboard allows the insertion and deletion of network flows																																														
Spec ID	SPEC-F6	Priority	Medium																																												
Prepared by	UOWM, PPC	Tested by	UOWM, PPC																																												
Pre-condition(s)	At least one SDN-C is deployed and connected to at least one SDN switch																																														
Test steps																																															
1	The tester starts Zookeeper Watcher to initialise the database containing the SDN switches, the SDN-Cs, and the relationships amongst them.																																														
2	The tester uses the PyCharm IDE to launch the Django server hosting the SDN dashboard																																														
3	The tester logs in the SDN dashboard using the credentials of a Security Administrator																																														
4	The tester records the existing network flows in Table 100 of the switch 2876467493531616																																														
<div>Flow Tables</div> <div><div>Switch_2876467493016320</div><div>Switch_2876467493531616</div></div> <div><div>Table 0</div><div>Table 100</div><div>Table 200</div></div> <div>Show 10 entries</div> <table><tr><th>Edit</th><th>Priority</th><th>Match fields</th><th>Cookie</th><th>Duration</th><th>Idle Timeout</th><th>Hard Timeout</th><th>Instructions</th><th>Packet Count</th><th>Byte Count</th><th>Delete</th></tr><tr><td></td><td>65535</td><td>{"dl_dst":"01:80:c2:00:00:0e", "dl_type":35020}</td><td>0</td><td>41467</td><td>0</td><td>0</td><td>OUTPUT:CONTROLLER</td><td>21132</td><td>1751</td><td></td></tr><tr><td></td><td>0</td><td>ANY</td><td>0</td><td>41469</td><td>0</td><td>0</td><td>GOTO_TABLE:200</td><td>1319640</td><td>0</td><td></td></tr><tr><td>Edit</td><td>Priority</td><td>Match fields</td><td>Cookie</td><td>Duration</td><td>Idle Timeout</td><td>Hard Timeout</td><td>Instructions</td><td>Packet Count</td><td>Byte Count</td><td>Delete</td></tr></table> <div>Showing 1 to 2 of 2 entries</div> <div><div>Previous</div><div>1</div><div>Next</div></div>				Edit	Priority	Match fields	Cookie	Duration	Idle Timeout	Hard Timeout	Instructions	Packet Count	Byte Count	Delete		65535	{"dl_dst":"01:80:c2:00:00:0e", "dl_type":35020}	0	41467	0	0	OUTPUT:CONTROLLER	21132	1751			0	ANY	0	41469	0	0	GOTO_TABLE:200	1319640	0		Edit	Priority	Match fields	Cookie	Duration	Idle Timeout	Hard Timeout	Instructions	Packet Count	Byte Count	Delete
Edit	Priority	Match fields	Cookie	Duration	Idle Timeout	Hard Timeout	Instructions	Packet Count	Byte Count	Delete																																					
	65535	{"dl_dst":"01:80:c2:00:00:0e", "dl_type":35020}	0	41467	0	0	OUTPUT:CONTROLLER	21132	1751																																						
	0	ANY	0	41469	0	0	GOTO_TABLE:200	1319640	0																																						
Edit	Priority	Match fields	Cookie	Duration	Idle Timeout	Hard Timeout	Instructions	Packet Count	Byte Count	Delete																																					
5	The tester visits the Flow Control page and fills the form accordingly and clicks Submit																																														

Flow Control

Submit

1

Target

Select Datapath ID:

2876467493531616

Select Table ID:

100

2

Flow Action

☒ Add
☐ Modify
☐ Modify strict
☐ Delete
☐ Delete strict

3

Match Fields

☐ Match any

Field

Value

eth_src

00:00:00:00:00:01

Add Row

4

Other fields

Priority

5

Idle timeout

0

Hard timeout

0

Cookie

15 | Honeypot Manager

Cookie Mask

65535

5

Instructions

Action Type

Value

OUTPUT

1

Add Row

6

By visiting the Flows page, the new flow has been added successfully

Flow Tables

Switch_2876467493016320

Switch_2876467493531616

Table 0

Table 100

Table 200

Show 10 entries

Edit	Priority	Match fields	Cookie	Duration	Idle Timeout	Hard Timeout	Instructions	Packet Count	Byte Count	Delete
	5	["dl_src":"00:00:00:00:00:01"]	2	5	0	0	OUTPUT:1	0	0	
	65535	["dl_dst":"01:80:c2:00:00:0e", "dl_type":"35020"]	0	44071	0	0	OUTPUT:CONTROLLER	22435	1577	
	0	ANY	0	44072	0	0	GOTO_TABLE:200	139448	0	
Edit	Priority	Match fields	Cookie	Duration	Idle Timeout	Hard Timeout	Instructions	Packet Count	Byte Count	Delete

Showing 1 to 3 of 3 entries

Previous

1

Next

7

The tester clicks on the Delete icon placed near the new flow entry

8

After user confirmation, the corresponding flow entry has been removed

Flow Tables

Switch_2876467493016320

Switch_2876467493531616

Table 0

Table 100

Table 200

Show 10 entries

Edit	Priority	Match fields	Cookie	Duration	Idle Timeout	Hard Timeout	Instructions	Packet Count	Byte Count	Delete
	65535	["dl_dst":"01:80:c2:00:00:0e", "dl_type":"35020"]	0	41467	0	0	OUTPUT:CONTROLLER	21132	1751	
	0	ANY	0	41469	0	0	GOTO_TABLE:200	1319640	0	
Edit	Priority	Match fields	Cookie	Duration	Idle Timeout	Hard Timeout	Instructions	Packet Count	Byte Count	Delete

Showing 1 to 2 of 2 entries

Previous

1

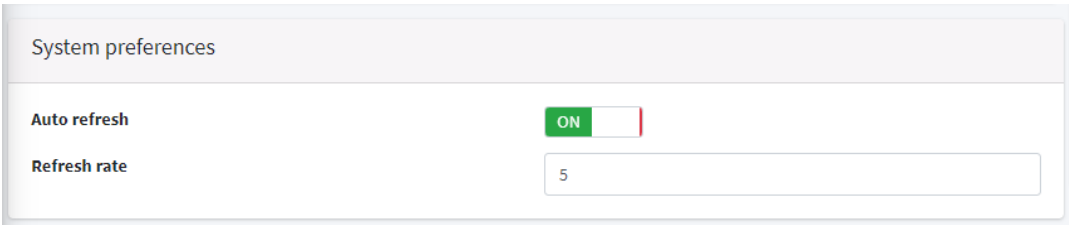
Next

Input data

A new flow via the Flow Control page.

Result	The tester was able to add and remove a network flow.
Test Case Result	Achieved

Table 58: SDNGUI_06 unit test description

Test Case ID	SDNGUI_06	Component	SDN dashboard
Description	The statistics depicted in the homepage of the SDN dashboard are automatically refreshed on a custom interval, configured by the logged-in user.		
Spec ID	SPEC-F6	Priority	Medium
Prepared by	UOWM, PPC	Tested by	UOWM, PPC
Pre-condition(s)	At least one SDN-C is deployed and connected to at least one SDN switch		
Test steps			
1	The tester starts Zookeeper Watcher to initialise the database containing the SDN switches, the SDN-Cs, and the relationships amongst them.		
2	The tester uses the PyCharm IDE to launch the Django server hosting the SDN dashboard		
3	The tester logs in the SDN dashboard using the credentials of a simple user		
4	The tester visits the Settings menu and configures a custom refresh rate <div></div>		
5	The tester visits the homepage and notices that the port and table statistics are automatically refreshed at the defined interval		
6	The tester visits the Flows page		
7	The tester launches a second web browser window and visits the Flow Control page		
8	The tester adds a new network flow via the Flow Control page		
9	The tester notices that the new flow is rendered in the Flows page without manually refreshing the page		
Input data	New values for “Auto refresh” and “Refresh rate” preferences		
Result	The port and table statistics as well as the flow tables are refreshed automatically, according to the configured refresh interval		
Test Case Result	Achieved		

9.3 EDAE core

9.3.1 Technical environment

The EDAE core engine has been developed under python 3.7, while the simulation environment that was used to assist the development is Mininet. Mininet is built inside an Ubuntu 18.04 LTS virtual machine of the host machine that executes EDAE, which operates on Ubuntu 20.04 LTS. Additionally, Ryu manager is used to execute the controller's software, ryu_ofctl API and flow manager. Table 59 details the required packages to run EDAE:

Table 59: List of software and version used for EDAE unit testing

Package Name	Version
networkx	v2.5
pandas	v1.1.2
pygmo	v2.16.0
absl-py	v0.10.0
requests	v2.24.0
pulp	v2.3.1

9.3.2 Unit Tests

Table 60 to **Error! Reference source not found.** describe 5 unit tests for EDAE core.

Table 60: EDAE_GENETIC_01 unit test description

Test Case ID	EDAE_GENETIC_01	Component	EDAE
Description	In this unit test, the S-RAF component informs EDAE for a security risk. EDAE consumes the event, identifies a risk violation to a communication path and invokes genetic algorithm to construct alternative path. This unit test is linked with scenario 3 described in 2.6.2.3		
Req ID	FR-UR-03, FR-UR-04, FR-UR-05, FR-UR-06, FR-UR-07, FR-UR-08, FR-UR-09, FR-UR-10, FR-UR-11, FR-UR-12, FR-UR-13, FR-UR-14, FR-UR-15, FR-UC3-01, FR-GR-14	Priority	High
Prepared by	CERTH	Tested by	CERTH
Pre-condition(s)	Predefined communication paths has been constructed through the EDAE – Controller interface. The statistics of the switches have been manually selected in order to know in advance the correct solution. The incident affects the communication path of [00:00:00:00:00:01 -> 00:00:00:00:00:03] such that a sensitivity constraint of host 00:00:00:00:00:01 is violated.		
Test steps			
1	Created a custom topology in Mininet with 4 hosts, 9 switches and one SDN-C with predefined security risk levels.		

2	Created the network graph through using the restful API of the controller
3	Augmented the network graph with host metadata and switch link metadata by reading the constraints of the hosts from and the port statistics of the switches by json files.
4	Generated S-RAF event through postman API.
5	Re-calculated the security risk of the communication paths.
6	Sensitivity constraint violated for host 00:00:00:00:00:01 and genetic algorithm invoked in order to find an alternative path.
7	Genetic algorithm correctly produced the correct path that meets the sensitivity constraint of host 00:00:00:00:00:01 and does not violate bandwidth constraint of host 00:00:00:00:00:04

Input data

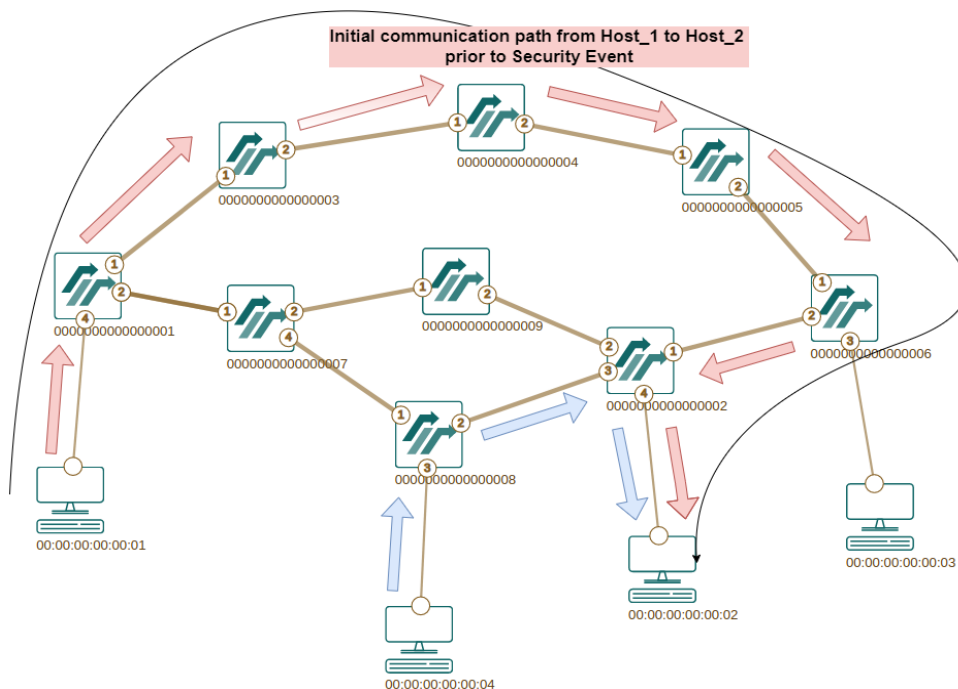


Figure 79: Initial communication paths

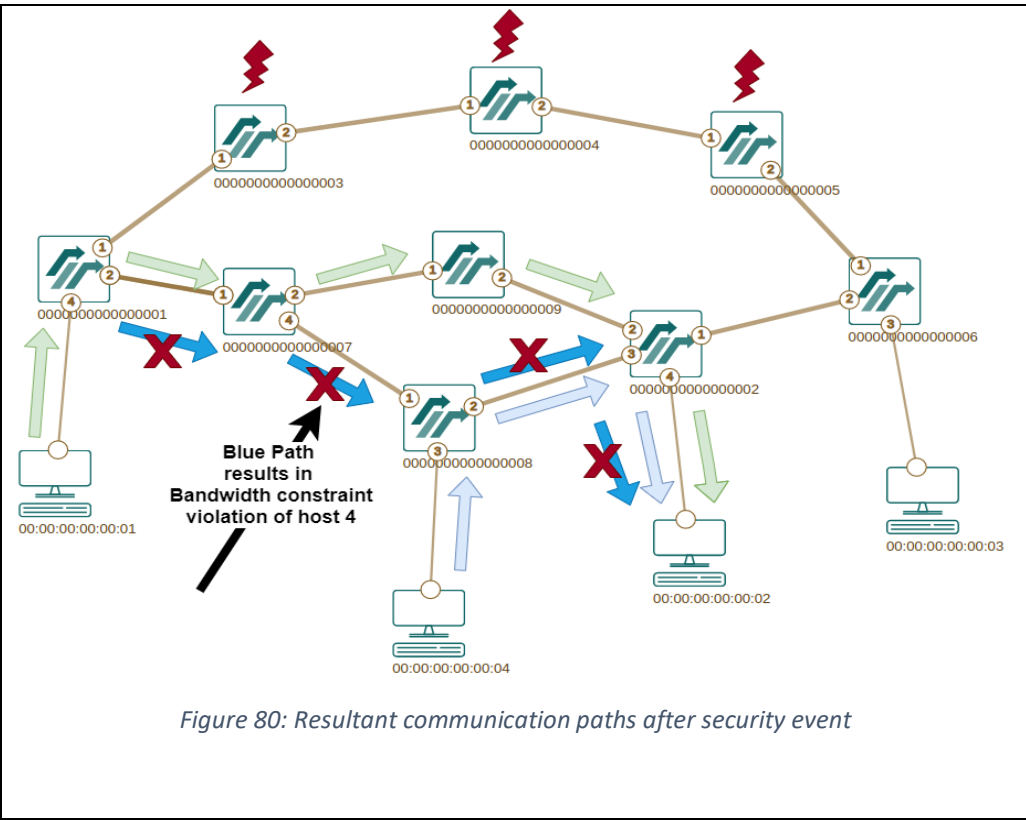
	 <p>Figure 80: Resultant communication paths after security event</p>
Result	EDAE chooses the path, which meets all the constraints and minimizes the security risk.
Test Case Result	Partially Achieved, to be tested with more data

Table 61: EDAAE_GENETIC_02 unit test description

Test Case ID	EDAE_GENETIC_02	Component	EDAE
Description	In this unit test, EDAE monitors the QoS constraints and latency constraint violation is identified. EDAE invokes genetic algorithm to compute an optimal path. This unit test is related with Scenario 1 described in 6.3.2.3		
Req ID	FR-UC3-01, FR-GR-14	Priority	Medium
Prepared by	CERTH	Tested by	CERTH
Pre-condition(s)	Predefined communication paths has been constructed through the EDAE – Controller interface. The statistics of the switches have been manually selected in order to know in advance the correct solution. The incident affects the communication path of [00:00:00:00:00:01 -> 00:00:00:00:00:03] such that bandwidth constraint of host 00:00:00:00:00:01 is violated by 30 mbps.		
Test steps			
1	Created a custom topology in Mininet with 4 hosts, 9 switches and one SDN-C with predefined security risk levels.		
2	Created the network graph through using the restful API of the controller		

3	Augmented the network graph with host metadata and switch link metadata by reading the constraints of the hosts from and the port statistics of the switches by json files.
4	Changed the latency of the switches of the json file while EDAE was monitoring for constraint violations.
5	When EDAE identified that latency constraint of host, 00:00:00:00:00:01 is violated for more than 2 minutes it invoked genetic algorithm in order to find a new path.
6	Genetic algorithm correctly produced the correct path that meets the latency constraints of host 00:00:00:00:00:01 and does not violate bandwidth constraints of host 00:00:00:00:00:04

Input data

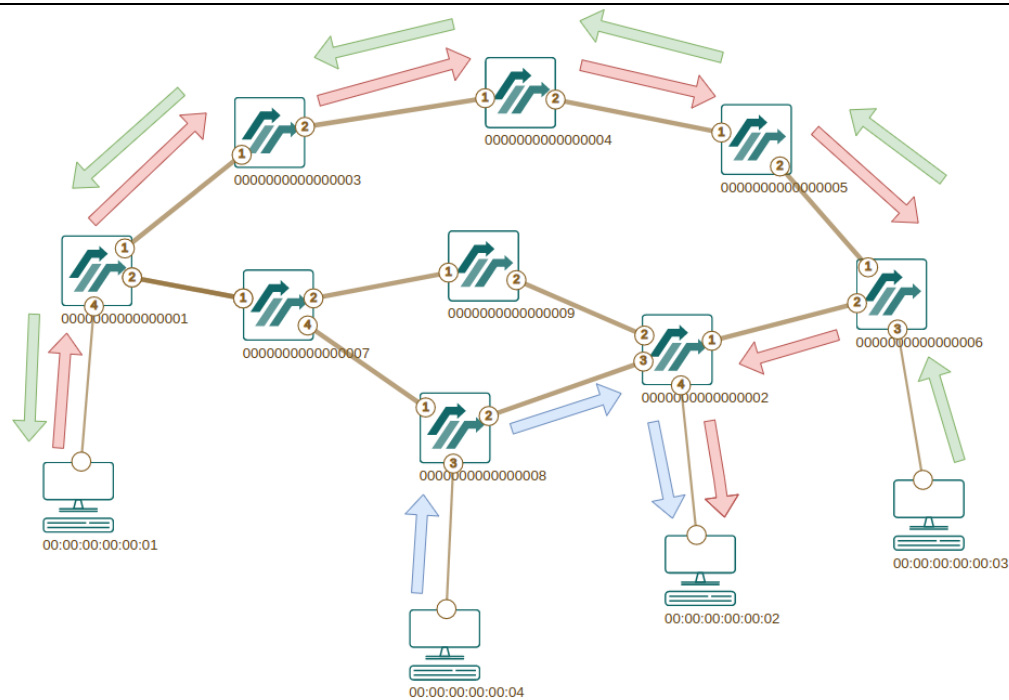


Figure 81: Initial communication paths

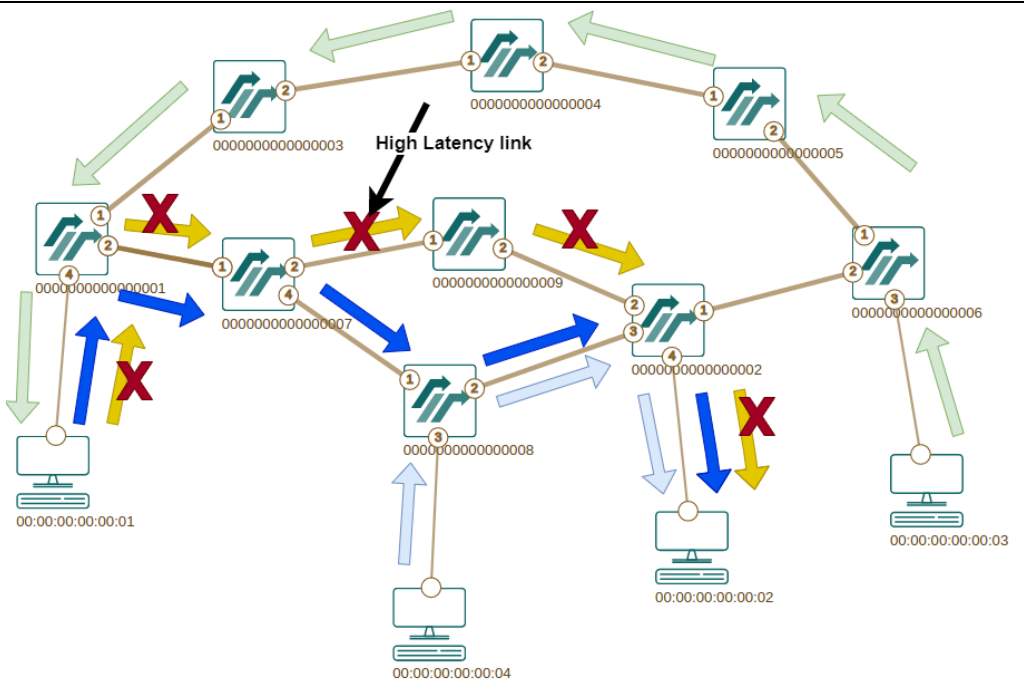
	 <p>Figure 82: Resultant communication paths</p>
Result	EDAE chooses the path, which minimizes the constraint violation and minimizes the bandwidth.
Test Case Result	Partially Achieved, to be tested with more data

Table 62: EDAE_GENETIC_3 unit test description

Test Case ID	EDAE_GENETIC_3	Component	EDAE
Description	Compares the execution time of the genetic algorithm for different number of generations with respect to the greed search algorithm.		
Req ID	FR-UR-03, FR-UR-04, FR-UR-05, FR-UR-06, FR-UR-07, FR-UR-08, FR-UR-09, FR-UR-10, FR-UR-11, FR-UR-12, FR-UR-13, FR-UR-14, FR-UR-15	Priority	Medium
Prepared by	CERTH	Tested by	CERTH
Pre-condition(s)	None		
Test steps			
1	Create a topology with 11 switches and eight hosts		

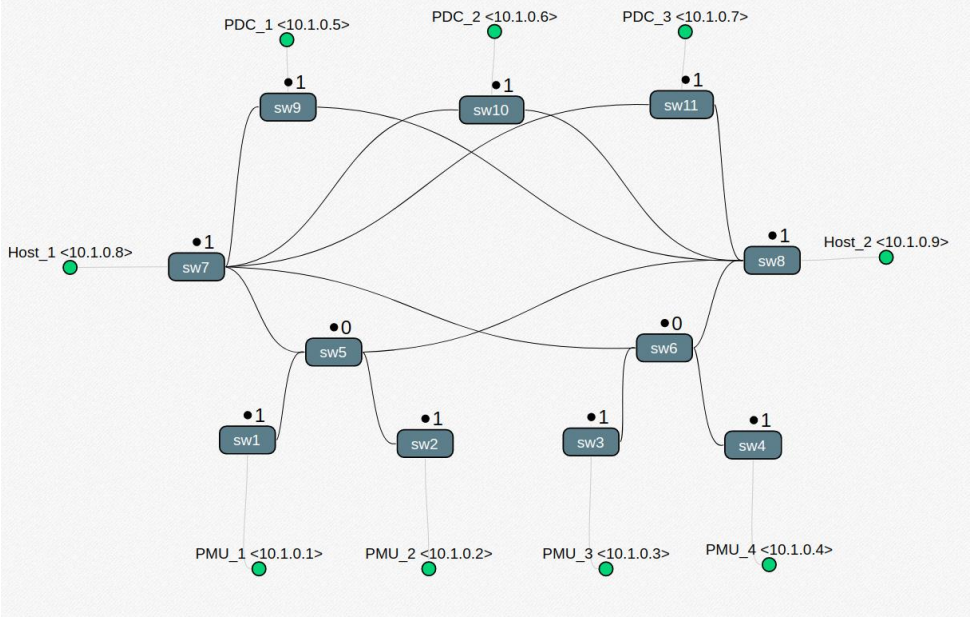
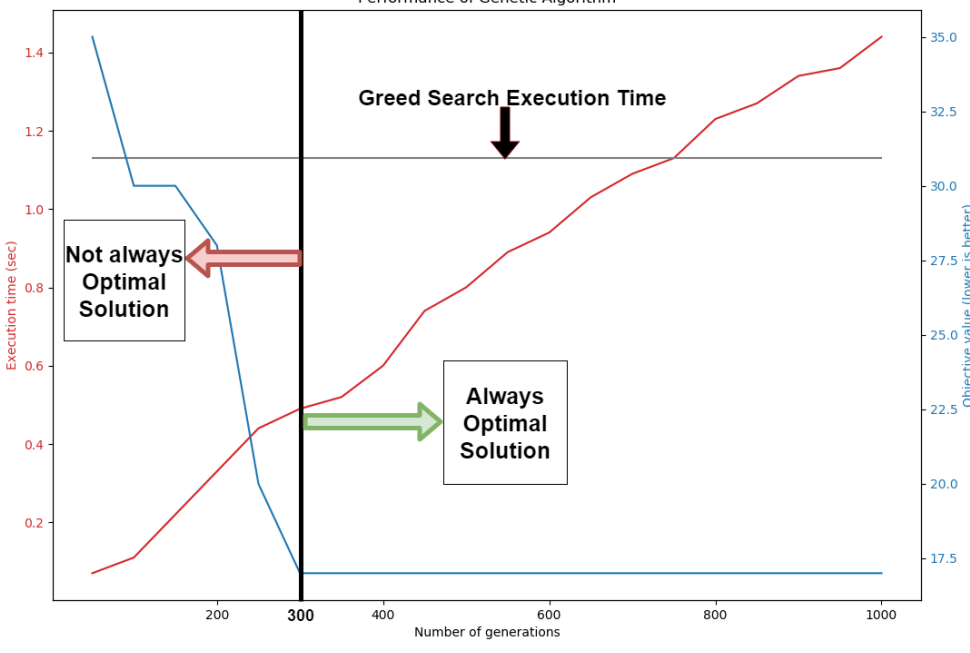
2	Run the genetic algorithm ten times for various number of generations 50 generations to 1000 with a step of ten
3	For each configuration average the results
Input data	 <p style="text-align: center;"><i>Figure 83: Topology</i></p>
Result	<p style="text-align: center;">Performance of Genetic Algorithm</p>  <p style="text-align: center;"><i>Figure 84: Execution results</i></p>
Test Case Result	Partially achieved, to be tested with more data

Table 63: EDAA_receiver_transmitter_1 unit test

Test Case ID	EDAE_receiver_transmitter_1	Component	EDAE															
Description	Optimal receiver-transmitter allocation by minimizing the latency of the network with EDAE’s Bandwidth constrained MILP																	
Req ID	FR-GR-6	Priority	Medium															
Prepared by	CERTH	Tested by	CERTH															
Pre-condition(s)	Receivers (PDCs) and transmitters (PMUs) can be connected through the SDN network.																	
Test steps																		
1	Optimal available PDCs-PMUs allocation by minimizing the latency of the network																	
2	Run the MILP algorithm with the objective to minimize the latency of the network																	
3	Obtain a list with the new PDC and PMU connections																	
Input data	The list of the PDCs, PMUs, their maximum bandwidth and their data transfer rate at the moment PDC(s) is disconnected from the SDN-network																	
	PMU			PMU-1			PMU-2			PMU-3			PMU-4					
	Bandwidth Capacity			40000			40000			40000			40000					
	Data			1000			3000			10000			10000					
	PDCs			A		B		A		B		A		B		A		B
Latency			4.56		3.45		3.45		2.65		9.24		5.01		2.95		7.33	
PDC			PDC-A			PDC-B												
Bandwidth Capacity			40000			40000												
Result	Network before the attack:																	
	<div><div><div><div>PDCA <10.1.0.5></div><div>•1</div><div>sw5</div></div><div><div>PDCB <10.1.0.6></div><div>•1</div><div>sw6</div></div><div><div><div>PDCC <10.1.0.7></div><div>•1</div><div>sw7</div></div><div>•1</div></div></div><div><div><div>sw1</div><div>•1</div></div><div><div>sw2</div><div>•1</div></div><div><div>sw3</div><div>•1</div></div><div><div>sw4</div><div>•1</div></div></div><div><div>PMU1 <10.1.0.1></div><div>•1</div></div><div><div>PMU2 <10.1.0.2></div><div>•1</div></div><div><div>PMU3 <10.1.0.3></div><div>•1</div></div><div><div>PMU4 <10.1.0.4></div><div>•1</div></div></div> <div><div><div>sw5</div><div>•1</div></div><div><div>sw6</div><div>•1</div></div><div><div>sw7</div><div>•1</div></div></div> <div><div>sw1</div><div>•1</div></div> <div><div>sw2</div><div>•1</div></div> <div><div>sw3</div><div>•1</div></div> <div><div>sw4</div><div>•1</div></div> <div><div>PMU1 <10.1.0.1></div><div>•1</div></div> <div><div>PMU2 <10.1.0.2></div><div>•1</div></div> <div><div>PMU3 <10.1.0.3></div><div>•1</div></div> <div><div>PMU4 <10.1.0.4></div><div>•1</div></div> <div><div>PDCA <10.1.0.5></div><div>•1</div></div> <div><div>PDCB <10.1.0.6></div><div>•1</div></div> <div><div>PDCC <10.1.0.7></div><div>•1</div></div> <div><div>sw5</div><div>•1</div></div> <div><div>sw6</div><div>•1</div></div> <div><div>sw7</div><div>•1</div></div> <div><div>sw1</div><div>•1</div></div> <div><div>sw2</div><div>•1</div></div> <div><div>sw3</div><div>•1</div></div> <div><div>sw4</div><div>•1</div></div> <div><div>PMU1 <10.1.0.1></div><div>•1</div></div> <div><div>PMU2 <10.1.0.2></div><div>•1</div></div> <div><div>PMU3 <10.1.0.3></div><div>•1</div></div> <div><div>PMU4 <10.1.0.4></div><div>•1</div></div> <div><div>PDCA <10.1.0.5></div><div>•1</div></div> <div><div>PDCB <10.1.0.6></div><div>•1</div></div> <div><div>PDCC <10.1.0.7></div><div>•1</div></div> <div><div>sw5</div><div>•1</div></div> <div><div>sw6</div><div>•1</div></div> <div><div>sw7</div><div>•1</div></div> <div><div>sw1</div><div>•1</div></div> <div><div>sw2</div><div>•1</div></div> <div><div>sw3</div><div>•1</div></div> <div><div>sw4</div><div>•1</div></div> <div><div>PMU1 <10.1.0.1></div><div>•1</div></div> <div><div>PMU2 <10.1.0.2></div><div>•1</div></div> <div><div>PMU3 <10.1.0.3></div><div>•1</div></div> <div><div>PMU4 <10.1.0.4></div><div>•1</div></div> <div><div>PDCA <10.1.0.5></div><div>•1</div></div> <div><div>PDCB <10.1.0.6></div><div>•1</div></div> <div><div>PDCC <10.1.0.7></div><div>•1</div></div> <div><div>sw5</div><div>•1</div></div> <div><div>sw6</div><div>•1</div></div> <div><div>sw7</div><div>•1</div></div> <div><div>sw1</div><div>•1</div></div> <div><div>sw2</div><div>•1</div></div> <div><div>sw3</div><div>•1</div></div> <div><div>sw4</div><div>•1</div></div> <div><div>PMU1 <10.1.0.1></div><div>•1</div></div> <div><div>PMU2 <10.1.0.2></div><div>•1</div></div> <div><div>PMU3 <10.1.0.3></div><div>•1</div></div> <div><div>PMU4 <10.1.0.4></div><div>•1</div></div> <div><div>PDCA <10.1.0.5></div><div>•1</div></div> <div><div>PDCB <10.1.0.6></div><div>•1</div></div> <div><div>PDCC <10.1.0.7></div><div>•1</div></div> <div><div>sw5</div><div>•1</div></div> <div><div>sw6</div><div>•1</div></div> <div><div>sw7</div><div>•1</div></div> <div><div>sw1</div><div>•1</div></div> <div><div>sw2</div><div>•1</div></div> <div><div>sw3</div><div>•1</div></div> <div><div>sw4</div><div>•1</div></div> <div><div>PMU1 <10.1.0.1></div><div>•1</div></div> <div><div>PMU2 <10.1.0.2></div><div>•1</div></div> <div><div>PMU3 <10.1.0.3></div><div>•1</div></div> <div><div>PMU4 <10.1.0.4></div><div>•1</div></div> <div><div>PDCA <10.1.0.5></div><div>•1</div></div> <div><div>PDCB <10.1.0.6></div><div>•1</div></div> <div><div>PDCC <10.1.0.7></div><div>•1</div></div> <div><div>sw5</div><div>•1</div></div> <div><div>sw6</div><div>•1</div></div> <div><div>sw7</div><div>•1</div></div> <div><div>sw1</div><div>•1</div></div> <div><div>sw2</div><div>•1</div></div> <div><div>sw3</div><div>•1</div></div> <div><div>sw4</div><div>•1</div></div> <div><div>PMU1 <10.1.0.1></div><div>•1</div></div> <div><div>PMU2 <10.1.0.2></div><div>•1</div></div> <div><div>PMU3 <10.1.0.3></div><div>•1</div></div> <div><div>PMU4 <10.1.0.4></div><div>•1</div></div> <div><div>PDCA <10.1.0.5></div><div>•1</div></div> <div><div>PDCB <10.1.0.6></div><div>•1</div></div> <div><div>PDCC <10.1.0.7></div><div>•1</div></div> <div><div>sw5</div><div>•1</div></div> <div><div>sw6</div><div>•1</div></div> <div><div>sw7</div><div>•1</div></div> <div><div>sw1</div><div>•1</div></div> <div><div>sw2</div><div>•1</div></div> <div><div>sw3</div><div>•1</div></div> <div><div>sw4</div><div>•1</div></div> <div><div>PMU1 <10.1.0.1></div><div>•1</div></div> <div><div>PMU2 <10.1.0.2></div><div>•1</div></div> <div><div>PMU3 <10.1.0.3></div><div>•1</div></div> <div><div>PMU4 <10.1.0.4></div><div>•1</div></div> <div><div>PDCA <10.1.0.5></div><div>•1</div></div> <div><div>PDCB <10.1.0.6></div><div>•1</div></div> <div><div>PDCC <10.1.0.7></div><div>•1</div></div> <div><div>sw5</div><div>•1</div></div> <div><div>sw6</div><div>•1</div></div> <div><div>sw7</div><div>•1</div></div> <div><div>sw1</div><div>•1</div></div> <div><div>sw2</div><div>•1</div></div> <div><div>sw3</div><div>•1</div></div> <div><div>sw4</div><div>•1</div></div> <div><div>PMU1 <10.1.0.1></div><div>•1</div></div> <div><div>PMU2 <10.1.0.2></div><div>•1</div></div> <div><div>PMU3 <10.1.0.3></div><div>•1</div></div> <div><div>PMU4 <10.1.0.4></div><div>•1</div></div> <div><div>PDCA <10.1.0.5></div><div>•1</div></div> <div><div>PDCB <10.1.0.6></div><div>•1</div></div> <div><div>PDCC <10.1.0.7></div><div>•1</div></div> <div><div>sw5</div><div>•1</div></div> <div><div>sw6</div><div>•1</div></div> <div><div>sw7</div><div>•1</div></div> <div><div>sw1</div><div>•1</div></div> <div><div>sw2</div><div>•1</div></div> <div><div>sw3</div><div>•1</div></div> <div><div>sw4</div><div>•1</div></div> <div><div>PMU1 <10.1.0.1></div><div>•1</div></div> <div><div>PMU2 <10.1.0.2></div><div>•1</div></div> <div><div>PMU3 <10.1.0.3></div><div>•1</div></div> <div><div>PMU4 <10.1.0.4></div><div>•1</div></div> <div><div>PDCA <10.1.0.5></div><div>•1</div></div> <div><div>PDCB <10.1.0.6></div><div>•1</div></div> <div><div>PDCC <10.1.0.7></div><div>•1</div></div> <div><div>sw5</div><div>•1</div></div> <div><div>sw6</div><div>•1</div></div> <div><div>sw7</div><div>•1</div></div> <div><div>sw1</div><div>•1</div></div> <div><div>sw2</div><div>•1</div></div> <div><div>sw3</div><div>•1</div></div> <div><div>sw4</div><div>•1</div></div> <div><div>PMU1 <10.1.0.1></div><div>•1</div></div> <div><div>PMU2 <10.1.0.2></div><div>•1</div></div> <div><div>PMU3 <10.1.0.3></div><div>•1</div></div> <div><div>PMU4 <10.1.0.4></div><div>•1</div></div> <div><div>PDCA <10.1.0.5></div><div>•1</div></div> <div><div>PDCB <10.1.0.6></div><div>•1</div></div> <div><div>PDCC <10.1.0.7></div><div>•1</div></div> <div><div>sw5</div><div>•1</div></div> <div><div>sw6</div><div>•1</div></div> <div><div>sw7</div><div>•1</div></div> <div><div>sw1</div><div>•1</div></div> <div><div>sw2</div><div>•1</div></div> <div><div>sw3</div><div>•1</div></div> <div><div>sw4</div><div>•1</div></div> <div><div>PMU1 <10.1.0.1></div><div>•1</div></div> <div><div>PMU2 <10.1.0.2></div><div>•1</div></div> <div><div>PMU3 <10.1.0.3></div><div>•1</div></div> <div><div>PMU4 <10.1.0.4></div><div>•1</div></div> <div><div>PDCA <10.1.0.5></div><div>•1</div></div> <div><div>PDCB <10.1.0.6></div><div>•1</div></div> <div><div>PDCC <10.1.0.7></div><div>•1</div></div> <div><div>sw5</div><div>•1</div></div> <div><div>sw6</div><div>•1</div></div> <div><div>sw7</div><div>•1</div></div> <div><div>sw1</div><div>•1</div></div> <div><div>sw2</div><div>•1</div></div> <div><div>sw3</div><div>•1</div></div> <div><div>sw4</div><div>•1</div></div> <div><div>PMU1 <10.1.0.1></div><div>•1</div></div> <div><div>PMU2 <10.1.0.2></div><div>•1</div></div> <div><div>PMU3 <10.1.0.3></div><div>•1</div></div> <div><div>PMU4 <10.1.0.4></div><div>•1</div></div> <div><div>PDCA <10.1.0.5></div><div>•1</div></div> <div><div>PDCB <10.1.0.6></div><div>•1</div></div> <div><div>PDCC <10.1.0.7></div><div>•1</div></div> <div><div>sw5</div><div>•1</div></div> <div><div>sw6</div><div>•1</div></div> <div><div>sw7</div><div>•1</div></div> <div><div>sw1</div><div>•1</div></div> <div><div>sw2</div><div>•1</div></div> <div><div>sw3</div><div>•1</div></div> <div><div>sw4</div><div>•1</div></div> <div><div>PMU1 <10.1.0.1></div><div>•1</div></div> <div><div>PMU2 <10.1.0.2></div><div>•1</div></div> <div><div>PMU3 <10.1.0.3></div><div>•1</div></div> <div><div>PMU4 <10.1.0.4></div><div>•1</div></div> <div><div>PDCA <10.1.0.5></div><div>•1</div></div> <div><div>PDCB <10.1.0.6></div><div>•1</div></div> <div><div>PDCC <10.1.0.7></div><div>•1</div></div> <div><div>sw5</div><div>•1</div></div> <div><div>sw6</div><div>•1</div></div> <div><div>sw7</div><div>•1</div></div> <div><div>sw1</div><div>•1</div></div> <div><div>sw2</div><div>•1</div></div> <div><div>sw3</div><div>•1</div></div> <div><div>sw4</div><div>•1</div></div> <div><div>PMU1 <10.1.0.1></div><div>•1</div></div> <div><div>PMU2 <10.1.0.2></div><div>•1</div></div> <div><div>PMU3 <10.1.0.3></div><div>•1</div></div> <div><div>PMU4 <10.1.0.4></div><div>•1</div></div> <div><div>PDCA <10.1.0.5></div><div>•1</div></div> <div><div>PDCB <10.1.0.6></div><div>•1</div></div> <div><div>PDCC <10.1.0.7></div><div>•1</div></div> <div><div>sw5</div><div>•1</div></div> <div><div>sw6</div><div>•1</div></div> <div><div>sw7</div><div>•1</div></div> <div><div>sw1</div><div>•1</div></div> <div><div>sw2</div><div>•1</div></div> <div><div>sw3</div><div>•1</div></div> <div><div>sw4</div><div>•1</div></div> <div><div>PMU1 <10.1.0.1></div><div>•1</div></div> <div><div>PMU2 <10.1.0.2></div><div>•1</div></div> <div><div>PMU3 <10.1.0.3></div><div>•1</div></div> <div><div>PMU4 <10.1.0.4></div><div>•1</div></div> <div><div>PDCA <10.1.0.5></div><div>•1</div></div> <div><div>PDCB <10.1.0.6></div><div>•1</div></div> <div><div>PDCC <10.1.0.7></div><div>•1</div></div> <div><div>sw5</div><div>•1</div></div> <div><div>sw6</div><div>•1</div></div> <div><div>sw7</div><div>•1</div></div> <div><div>sw1</div><div>•1</div></div> <div><div>sw2</div><div>•1</div></div> <div><div>sw3</div><div>•1</div></div> <div><div>sw4</div><div>•1</div></div> <div><div>PMU1 <10.1.0.1></div><div>•1</div></div> <div><div>PMU2 <10.1.0.2></div><div>•1</div></div> <div><div>PMU3 <10.1.0.3></div><div>•1</div></div> <div><div>PMU4 <10.1.0.4></div><div>•1</div></div> <div><div>PDCA <10.1.0.5></div><div>•1</div></div> <div><div>PDCB <10.1.0.6></div><div>•1</div></div> <div><div>PDCC <10.1.0.7></div><div>•1</div></div> <div><div>sw5</div><div>•1</div></div> <div><div>sw6</div><div>•1</div></div> <div><div>sw7</div><div>•1</div></div> <div><div>sw1</div><div>•1</div></div> <div><div>sw2</div><div>•1</div></div> <div><div>sw3</div><div>•1</div></div> <div><div>sw4</div><div>•1</div></div> <div><div>PMU1 <10.1.0.1></div><div>•1</div></div> <div><div>PMU2 <10.1.0.2></div><div>•1</div></div> <div><div>PMU3 <10.1.0.3></div><div>•1</div></div> <div><div>PMU4 <10.1.0.4></div><div>•1</div></div> <div><div>PDCA <10.1.0.5></div><div>•1</div></div> <div><div>PDCB <10.1.0.6></div><div>•1</div></div> <div><div>PDCC <10.1.0.7></div><div>•1</div></div> <div><div>sw5</div><div>•1</div></div> <div><div>sw6</div><div>•1</div></div> <div><div>sw7</div><div>•1</div></div> <div><div>sw1</div><div>•1</div></div> <div><div>sw2</div><div>•1</div></div> <div><div>sw3</div><div>•1</div></div> <div><div>sw4</div><div>•1</div></div> <div><div>PMU1 <10.1.0.1></div><div>•1</div></div> <div><div>PMU2 <10.1.0.2></div><div>•1</div></div> <div><div>PMU3 <10.1.0.3></div><div>•1</div></div> <div><div>PMU4 <10.1.0.4></div><div>•1</div></div> <div><div>PDCA <10.1.0.5></div><div>•1</div></div> <div><div>PDCB <10.1.0.6></div><div>•1</div></div> <div><div>PDCC <10.1.0.7></div><div>•1</div></div> <div><div>sw5</div><div>•1</div></div> <div><div>sw6</div><div>•1</div></div> <div><div>sw7</div><div>•1</div></div> <div><div>sw1</div><div>•1</div></div> <div><div>sw2</div><div>•1</div></div> <div><div>sw3</div><div>•1</div></div> <div><div>sw4</div><div>•1</div></div> <div><div>PMU1 <10.1.0.1></div><div>•1</div></div> <div><div>PMU2 <10.1.0.2></div><div>•1</div></div> <div><div>PMU3 <10.1.0.3></div><div>•1</div></div> <div><div>PMU4 <10.1.0.4></div><div>•1</div></div> <div><div>PDCA <10.1.0.5></div><div>•1</div></div> <div><div>PDCB <10.1.0.6></div><div>•1</div></div> <div><div>PDCC <10.1.0.7></div><div>•1</div></div> <div><div>sw5</div><div>•1</div></div> <div><div>sw6</div><div>•1</div></div> <div><div>sw7</div><div>•1</div></div> <div><div>sw1</div><div>•1</div></div> <div><div>sw2</div><div>•1</div></div> <div><div>sw3</div><div>•1</div></div> <div><div>sw4</div><div>•1</div></div> <div><div>PMU1 <10.1.0.1></div><div>•1</div></div> <div><div>PMU2 <10.1.0.2></div><div>•1</div></div> <div><div>PMU3 <10.1.0.3></div><div>•1</div></div> <div><div>PMU4 <10.1.0.4></div><div>•1</div></div> <div><div>PDCA <10.1.0.5></div><div>•1</div></div> <div><div>PDCB <10.1.0.6></div><div>•1</div></div> <div><div>PDCC <10.1.0.7></div><div>•1</div></div> <div><div>sw5</div><div>•1</div></div> <div><div>sw6</div><div>•1</div></div> <div><div>sw7</div><div>•1</div></div> <div><div>sw1</div><div>•1</div></div> <div><div>sw2</div><div>•1</div></div> <div><div>sw3</div><div>•1</div></div> <div><div>sw4</div><div>•1</div></div> <div><div>PMU1 <10.1.0.1></div><div>•1</div></div> <div><div>PMU2 <10.1.0.2></div><div>•1</div></div> <div><div>PMU3 <10.1.0.3></div><div>•1</div></div> <div><div>PMU4 <10.1.0.4></div><div>•1</div></div> <div><div>PDCA <10.1.0.5></div><div>•1</div></div> <div><div>PDCB <10.1.0.6></div><div>•1</div></div> <div><div>PDCC <10.1.0.7></div><div>•1</div></div> <div><div>sw5</div><div>•1</div></div> <div><div>sw6</div><div>•1</div></div> <div><div>sw7</div><div>•1</div></div> <div><div>sw1</div><div>•1</div></div> <div><div>sw2</div><div>•1</div></div> <div><div>sw3</div><div>•1</div></div> <div><div>sw4</div><div>•1</div></div> <div><div>PMU1 <10.1.0.1></div><div>•1</div></div> <div><div>PMU2 <10.1.0.2></div><div>•1</div></div> <div><div>PMU3 <10.1.0.3></div><div>•1</div></div> <div><div>PMU4 <10.1.0.4></div><div>•1</div></div> <div><div>PDCA <10.1.0.5></div><div>•1</div></div> <div><div>PDCB <10.1.0.6></div><div>•1</div></div> <div><div>PDCC <10.1.0.7></div><div>•1</div></div> <div><div>sw5</div><div>•1</div></div> <div><div>sw6</div><div>•1</div></div> <div><div>sw7</div><div>•1</div></div> <div><div>sw1</div><div>•1</div></div> <div><div>sw2</div><div>•1</div></div> <div><div>sw3</div><div>•1</div></div> <div><div>sw4</div><div>•1</div></div> <div><div>PMU1 <10.1.0.1></div><div>•1</div></div> <div><div>PMU2 <10.1.0.2></div><div>•1</div></div> <div><div>PMU3 <10.1.0.3></div><div>•1</div></div> <div><div>PMU4 <10.1.0.4></div><div>•1</div></div> <div><div>PDCA <10.1.0.5></div><div>•1</div></div> <div><div>PDCB <10.1.0.6></div><div>•1</div></div> <div><div>PDCC <10.1.0.7></div><div>•1</div></div> <div><div>sw5</div><div>•1</div></div> <div><div>sw6</div><div>•1</div></div> <div><div>sw7</div><div>•1</div></div> <div><div>sw1</div><div>•1</div></div> <div><div>sw2</div><div>•1</div></div> <div><div>sw3</div><div>•1</div></div> <div><div>sw4</div><div>•1</div></div> <div><div>PMU1 <10.1.0.1></div><div>•1</div></div> <div><div>PMU2 <10.1.0.2></div><div>•1</div></div> <div><div>PMU3 <10.1.0.3></div><div>•1</div></div> <div><div>PMU4 <10.1.0.4></div><div>•1</div></div> <div><div>PDCA <10.1.0.5></div><div>•1</div></div> <div><div>PDCB <10.1.0.6></div><div>•1</div></div> <div><div>PDCC <10.1.0.7></div><div>•1</div></div> <div><div>sw5</div><div>•1</div></div> <div><div>sw6</div><div>•1</div></div> <div><div>sw7</div><div>•1</div></div> <div><div>sw1</div><div>•1</div></div> <div><div>sw2</div><div>•1</div></div> <div><div>sw3</div><div>•1</div></div> <div><div>sw4</div><div>•1</div></div> <div><div>PMU1 <10.1.0.1></div><div>•1</div></div> <div><div>PMU2 <10.1.0.2></div><div>•1</div></div> <div><div>PMU3 <10.1.0.3></div><div>•1</div></div> <div><div>PMU4 <10.1.0.4></div><div>•1</div></div> <div><div>PDCA <10.1.0.5></div><div>•1</div></div> <div><div>PDCB <10.1.0.6></div><div>•1</div></div> <div><div>PDCC <10.1.0.7></div><div>•1</div></div> <div><div>sw5</div><div>•1</div></div> <div><div>sw6</div><div>•1</div></div> <div><div>sw7</div><div>•1</div></div> <div><div>sw1</div><div>•1</div></div> <div><div>sw2</div><div>•1</div></div> <div><div>sw3</div><div>•1</div></div> <div><div>sw4</div><div>•1</div></div> <div><div>PMU1 <10.1.0.1></div><div>•1</div></div> <div><div>PMU2 <10.1.0.2></div><div>•1</div></div> <div><div>PMU3 <10.1.0.3></div><div>•1</div></div> <div><div>PMU4 <10.1.0.4></div><div>•1</div></div> <div><div>PDCA <10.1.0.5></div><div>•1</div></div> <div><div>PDCB <10.1.0.6></div><div>•1</div></div> <div><div>PDCC <10.1.0.7></div><div>•1</div></div> <div><div>sw5</div><div>•1</div></div> <div><div>sw6</div><div>•1</div></div> <div><div>sw7</div><div>•1</div></div> <div><div>sw1</div><div>•1</div></div> <div><div>sw2</div><div>•1</div></div> <div><div>sw3</div><div>•1</div></div> <div><div>sw4</div><div>•1</div></div> <div><div>PMU1 <10.1.0.1></div><div>•1</div></div> <div><div>PMU2 <10.1.0.2></div><div>•1</div></div> <div><div>PMU3 <10.1.0.3></div><div>•1</div></div> <div><div>PMU4 <10.1.0.4></div><div>•1</div></div> <div><div>PDCA <10.1.0.5></div><div>•1</div></div> <div><div>PDCB <10.1.0.6></div><div>•1</div></div> <div><div>PDCC <10.1.0.7></div><div>•1</div></div> <div><div>sw5</div><div>•1</div></div> <div><div>sw6</div><div>•1</div></div> <div><div>sw7</div><div>•1</div></div> <div><div>sw1</div><div>•1</div></div> <div><div>sw2</div><div>•1</div></div> <div><div>sw3</div><div>•1</div></div> <div><div>sw4</div><div>•1</div></div> <div><div>PMU1 <10.1.0.1></div><div>•1</div></div> <div><div>PMU2 <10.1.0.2></div><div>•1</div></div> <div><div>PMU3 <10.1.0.3></div><div>•1</div></div> <div><div>PMU4 <10.1.0.4></div><div>•1</div></div> <div><div>PDCA <10.1.0.5></div><div>•1</div></div> <div><div>PDCB <10.1.0.6></div><div>•1</div></div> <div><div>PDCC <10.1.0.7></div><div>•1</div></div> <div><div>sw5</div><div>•1</div></div> <div><div>sw6</div><div>•1</div></div> <div><div>sw7</div><div>•1</div></div> <div><div>sw1</div><div>•1</div></div> <div><div>sw2</div><div>•1</div></div> <div><div>sw3</div><div>•1</div></div> <div><div>sw4</div><div>•1</div></div> <div><div>PMU1 <10.1.0.1></div><div>•1</div></div> <div><div>PMU2 <10.1.0.2></div><div>•1</div></div> <div><div>PMU3 <10.1.0.3></div><div>•1</div></div> <div><div>PMU4 <10.1.0.4></div><div>•1</div></div> <div><div>PDCA <10.1.0.5></div><div>•1</div></div> <div><div>PDCB <10.1.0.6></div><div>•1</div></div> <div><div>PDCC <10.1.0.7></div><div>•1</div></div> <div><div>sw5</div><div>•1</div></div> <div><div>sw6</div><div>•1</div></div> <div><div>sw7</div><div>•1</div></div> <div><div>sw1</div><div>•1</div></div> <div><div>sw2</div><div>•1</div></div> <div><div>sw3</div><div>•1</div></div> <div><div>sw4</div><div>•1</div></div> <div><div>PMU1 <10.1.0.1></div><div>•1</div></div> <div><div>PMU2 <10.1.0.2></div><div>•1</div></div> <div><div>PMU3 <10.1.0.3></div><div>•1</div></div> <div><div>PMU4 <10.1.0.4></div><div>•1</div></div> <div><div>PDCA <10.1.0.5></div><div>•1</div></div> <div><div>PDCB <10.1.0.6></div><div>•1</div></div> <div><div>PDCC <10.1.0.7></div><div>•1</div></div> <div><div>sw5</div><div>•1</div></div> <div><div>sw6</div><div>•1</div></div> <div><div>sw7</div><div>•1</div></div> <div><div>sw1</div><div>•1</div></div> <div><div>sw2</div><div>•1</div></div> <div><div>sw3</div><div>•1</div></div> <div><div>sw4</div><div>•1</div></div> <div><div>PMU1 <10.1.0.1></div><div>•1</div></div> <div><div>PMU2 <10.1.0.2></div><div>•1</div></div> <div><div>PMU3 <10.1.0.3></div><div>•1</div></div> <div><div>PMU4 <10.1.0.4></div><div>•1</div></div> <div><div>PDCA <10.1.0.5></div><div>•1</div></div> <div><div>PDCB <10.1.0.6></div><div>•1</div></div> <div><div>PDCC <10.1.0.7></div><div>•1</div></div> <div><div>sw5</div><div>•1</div></div> <div><div>sw6</div><div>•1</div></div> <div><div>sw7</div><div>•1</div></div> <div><div>sw1</div><div>•1</div></div> <div><div>sw2</div><div>•1</div></div> <div><div>sw3</div><div>•1</div></div> <div><div>sw4</div><div>•1</div></div> <div><div>PMU1 <10.1.0.1></div><div>•1</div></div> <div><div>PMU2 <10.1.0.2></div><div>•1</div></div>																	

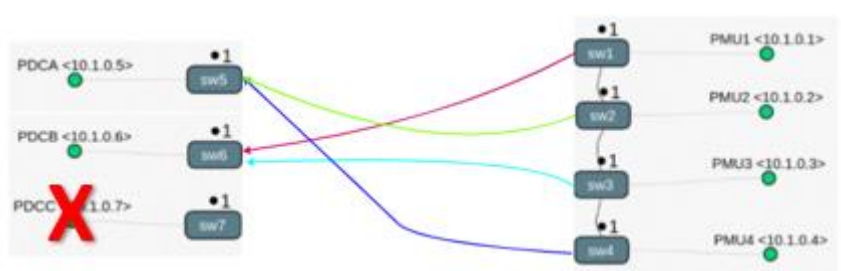
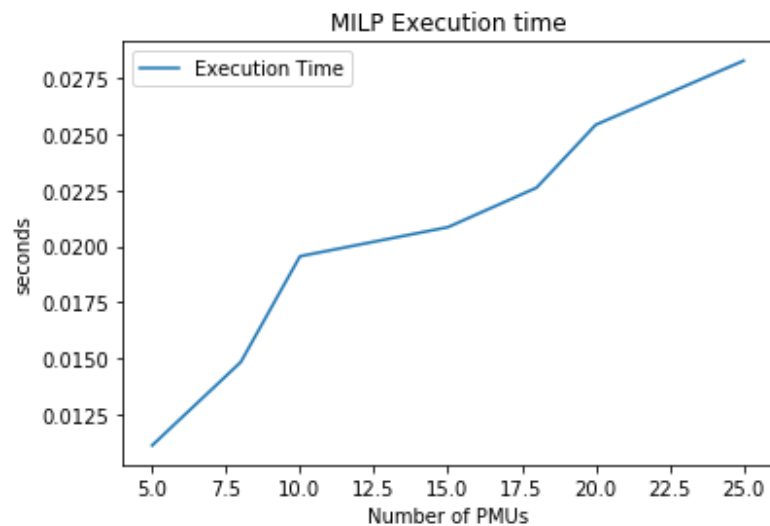
	<pre> Status: Optimal Route_1_A = 0.0 Route_1_B = 1000.0 Route_2_A = 3000.0 Route_2_B = 0.0 Route_3_A = 0.0 Route_3_B = 10000.0 Route_4_A = 10000.0 Route_4_B = 0.0 Which PMU should connect to PDC = [{'1': 'B'}, {'2': 'A'}, {'3': 'B'}, {'4': 'A'}] Total Cost of Transportation = 91000.0 </pre> <p>As it can be seen from the status parameter of the pulp package, the solution of the CBC solver is the optimal</p> <p>Network after the attack:</p> 
Test Case Result	Achieved, to be tested in pilots

Table 64: EDAE_receiver_transmitter_2 unit test

Test Case ID	EDAЕ_receiver_transmitter_2	Component	EDAЕ
Description	Compare the execution time of the bandwidth constrained algorithm for different size of receiver - transmitter of sets		
Req ID	FR-GR-6	Priority	High
Prepared by	CERTH	Tested by	CERTH
Pre-condition(s)	Receivers (PDCs) and transmitters (PMUs) can be connected through the SDN network.		
Test steps			
1	Obtain the list of the PDCs, PMUs, their maximum bandwidth and their data transfer rate at the moment PDC(s) is disconnected from the SDN-network		
2	Run the MILP algorithm with the objective to minimize the latency of the network		
3	Obtain a list with the new PDC and PMU connections and the execution time		
Input data	The list of the PDCs, PMUs, their maximum bandwidth and their data transfer at the moment PDC(s) is disconnected from the SDN-network for different sets of PMUs and PDCs		
Result	Input sets (PMUs)	Execution Time of MILP (seconds)	
	5	0.011109	

8	0.014834
10	0.019557
15	0.020854
18	0.022619
20	0.025427
25	0.028288



Number of PDCs	Execution Time of MILP (seconds)
4	0.020854
6	0.027695
8	0.028079
10	0.039315
12	0.040966
14	0.046971

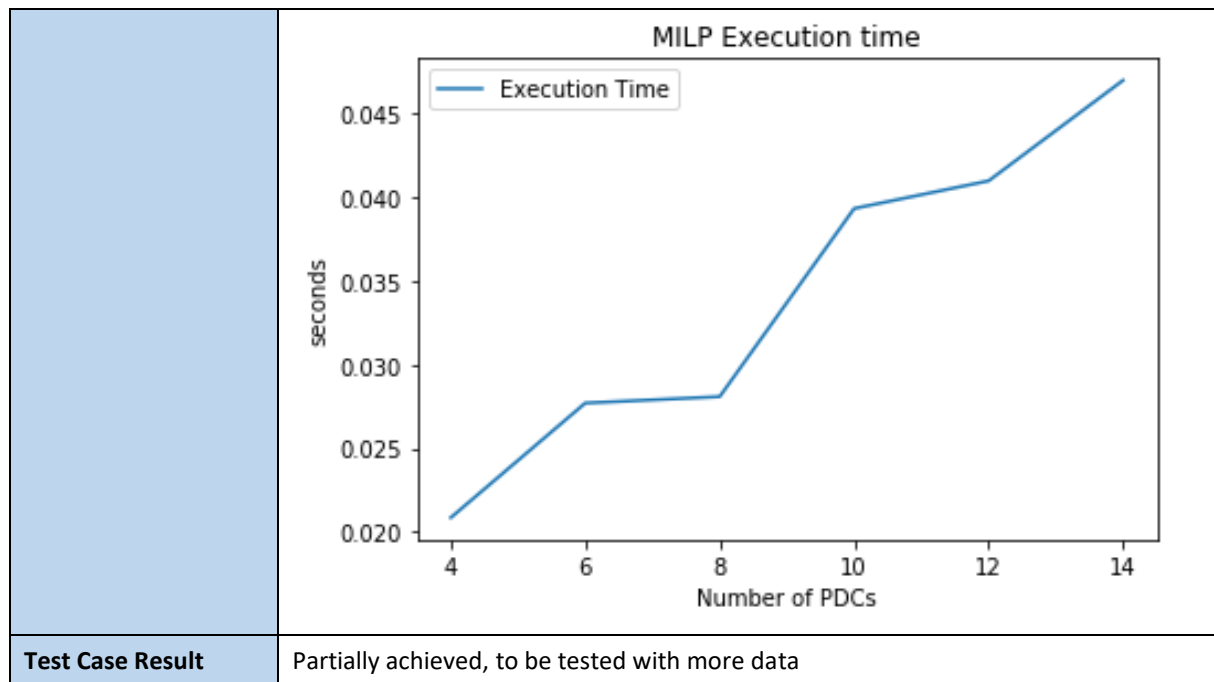


Table 65: EDAE_receiver_transmitter_3 unit test

Test Case ID	EDAE_receiver_transmitter_3	Component	EDAE						
Description	Optimal receiver-transmitter allocation by minimizing the latency of the network with EDAE’s transmitter constrained MILP								
Req ID	FR-GR-6		Priority			Medium			
Prepared by	CERTH		Tested by			CERTH			
Pre-condition(s)	Receivers (PDCs) and transmitters (PMUs) can be connected through the SDN network.								
Test steps									
1	Optimal available PDCs-PMUs allocation by minimizing the latency of the network								
2	Run the MILP algorithm with the objective to minimize the latency of the network								
3	Obtain a list with the new PDC and PMU connections								
Input data	The list of the PDCs, PMUs at the moment PDC(s) is disconnected from the SDN-network								
	PMU	PMU-1		PMU-2		PMU-3		PMU-4	
	PDCs	A	B	A	B	A	B	A	B
	Latency	4.56	3.45	3.45	2.65	9.24	5.01	2.95	7.33
	PDC	PDC-A		PDC-B					

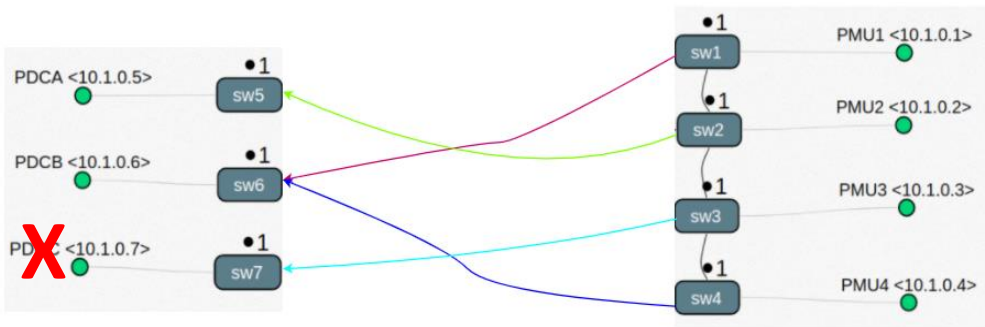
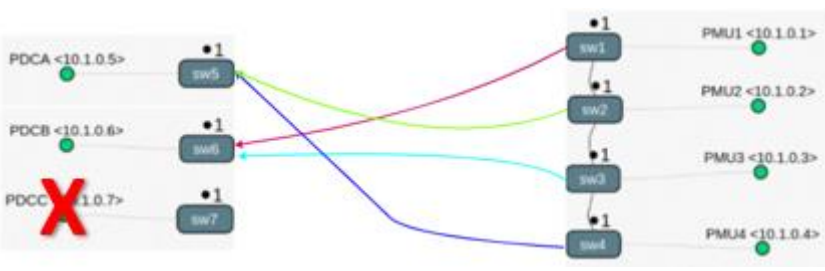
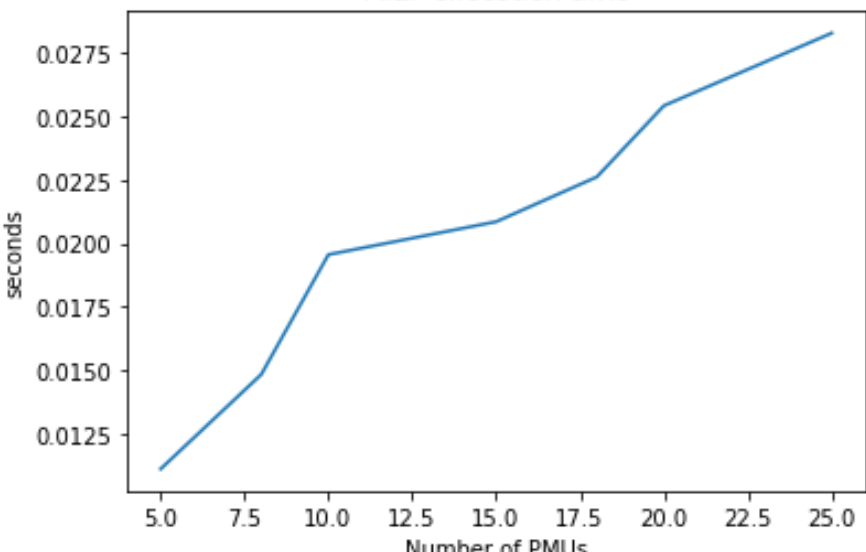
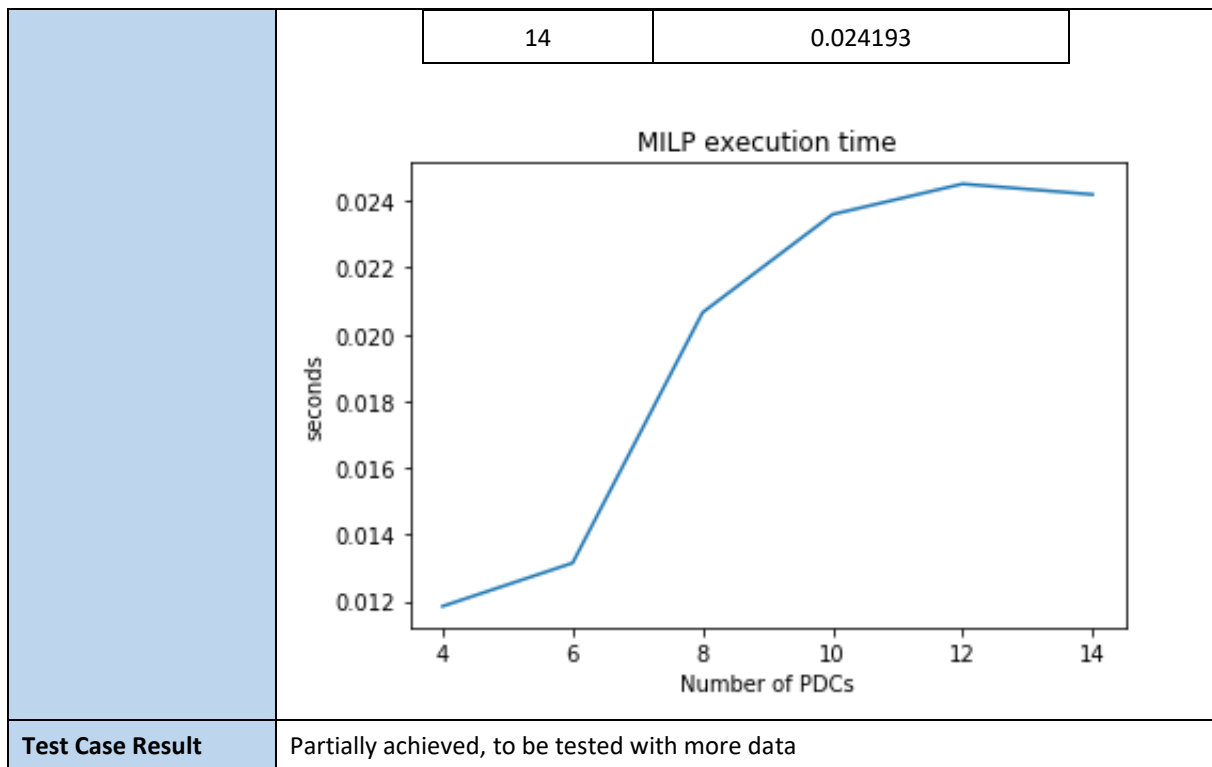
	PMU Capacity 20 20
Result	<p>Network before the attack:</p>  <p>Result of the transmitter constrained MILP algorithm:</p> <pre> Status: Optimal The choosen combinations are out of a total of 8: 1 B 2 B 3 B 4 A </pre> <p>As it can be seen from the status parameter of the pulp package, the solution of the CBC solver is the optimal</p> <p>Network after the attack:</p> 
Test Case Result	Partially Achieved, to be tested with more data

Table 66: EDAA_receiver_transmitter_4

Test Case ID	EDAA_receiver_transmitter_4	Component	EDAA
Description	Compare the execution time of the transmitter constrained algorithm for different size of transmitter - receiver of sets		
Req ID	FR-GR-6	Priority	Medium
Prepared by	CERTH	Tested by	CERTH
Pre-condition(s)	Receivers (PDCs) and transmitters (PMUs) can be connected through the SDN network.		

Test steps																		
1	Obtain the list of the PDCs, PMUs at the moment PDC(s) is disconnected from the SDN-network																	
2	Run the MILP algorithm with the objective to minimize the latency of the network																	
3	Obtain a list with the new PDC and PMU connections and the execution time																	
Input data		The list of the PDCs, PMUs at the moment PDC(s) is disconnected from the SDN-network for different sets of PMUs and PDCs																
Result	<table><tr><th>Input sets (PMUs)</th><th>Execution Time of MILP (seconds)</th></tr><tr><td>5</td><td>0.007932</td></tr><tr><td>8</td><td>0.009495</td></tr><tr><td>10</td><td>0.010889</td></tr><tr><td>15</td><td>0.011641</td></tr><tr><td>18</td><td>0.012874</td></tr><tr><td>20</td><td>0.015206</td></tr><tr><td>25</td><td>0.01486</td></tr></table>		Input sets (PMUs)	Execution Time of MILP (seconds)	5	0.007932	8	0.009495	10	0.010889	15	0.011641	18	0.012874	20	0.015206	25	0.01486
	Input sets (PMUs)	Execution Time of MILP (seconds)																
	5	0.007932																
	8	0.009495																
	10	0.010889																
	15	0.011641																
	18	0.012874																
	20	0.015206																
	25	0.01486																
	<p>MILP execution time</p> 																	
<table><tr><th>Number of PDCs</th><th>Execution Time of MILP (seconds)</th></tr><tr><td>4</td><td>0.011855</td></tr><tr><td>6</td><td>0.013151</td></tr><tr><td>8</td><td>0.020657</td></tr><tr><td>10</td><td>0.023597</td></tr><tr><td>12</td><td>0.02451</td></tr></table>		Number of PDCs	Execution Time of MILP (seconds)	4	0.011855	6	0.013151	8	0.020657	10	0.023597	12	0.02451					
Number of PDCs	Execution Time of MILP (seconds)																	
4	0.011855																	
6	0.013151																	
8	0.020657																	
10	0.023597																	
12	0.02451																	



9.4 EDAE workflow

9.4.1 Technical environment

Following the previous examples Table 67 details the list of software and versions used for EDAE workflow implementation and testing.

Table 67: List of software and version used for EDAE workflow implementation and testing

Base software	Version
Python	3.7
Docker	Any
Apache Airflow	1.10.9
PostgreSQL	11
RabbitMQ	3.7.28

9.4.2 Unit tests

Table 68 to Table 72 describe 5 unit tests for EDAE workflow.

Table 68: EDAE_workflow1 unit test description

Test Case ID	EDAE_workflow1	Component	EDAE_workflow
Description	Test the data management from interfaces in order to give it to EDAE engine in the required format.		
Req ID		Priority	
Prepared by	IREC	Tested by	IREC
Pre-condition(s)	S-RAF sends an incident		



Test steps	
1	The workflow is raised manually simulating the reception of an S-RAF incident.
2	The workflow gets data (an example of output) from S-RAF interface.
3	The workflow gets data (an example of output) from AIDB interface.
4	The workflow gets data (an example of output) from SDN-C interface.
5	From AIDB data the workflow will extract PMUs and PDCs
6	The workflow will group all the information in one json to be send to EDAE Rest API in a post request.
Input data	http://172.20.0.7:8000/EDAE/aidb/gridmodel/testcase http://172.20.0.6:8000/sdncontroller/testcase http://172.20.0.5:8000/sraf/incident/testcase See json details in annex 13.413.4.1
Result	As a summary: <pre>{ "Get_Sraf_Info": { "ASSETS": [], "PATHS": [] }, "Grid_Model": [], "PDCs": [], "PMUs": [], "Get_SDN_Controller_Info": [] }</pre> See json details in annex 13.4
Test Case Result	Achieved in local environments. To be tested in pilot.

Table 69: EDAE_workflow2 unit test description

Test Case ID	EDAE_workflow2	Component	EDAE_workflow
Description	Test the data management from EDAE engine to EDAE-Dashboard.		
Req ID		Priority	
Prepared by	IREC	Tested by	IREC
Pre-condition(s)	EDAE engine gives a proposal that requires to be approved by the EDAE-Dashboard.		
Test steps			
1	The workflow gets the EDAE engine result.		
2	The workflow generates a proposal ID to identify this information to be evaluated.		
3	The workflow notify EDAE-Dashboard the information by publishing in a RabbitMQ this information grouped in one json		
Input data	<div>The result of EDAE engine. As a summary:</div> <pre>{ "APPROVAL_REQUIRED":true, "PDC": [], "PMU": [], "PATH": []}</pre> <div>See json details in annex 13.4</div>		
Result	As a summary (the peak corresponds to the message sent):		

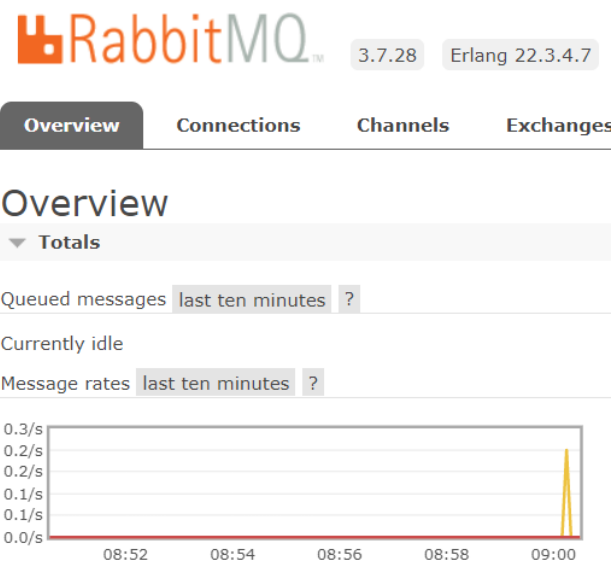
	 <p>Overview</p> <p>Queued messages last ten minutes ?</p> <p>Currently idle</p> <p>Message rates last ten minutes ?</p> <p>0.3/s 0.2/s 0.2/s 0.1/s 0.1/s 0.0/s</p> <p>08:52 08:54 08:56 08:58 09:00</p> <p>Output summary:</p> <pre>{ "PROPOSAL_ID": 53427606, "DATA": { "APPROVAL_REQUIRED": true, "PDC": [], "PMU": [], "PATH": [] } }</pre> <p>See json details in annex 13.4</p>
Test Case Result	Achieved in local environments. To be tested in pilot.

Table 70: EDAE_workflow3 unit test description

Test Case ID	EDAE_workflow3	Component	EDAE
Description	Test the data management from EDAE engine to implementation.		
Req ID		Priority	
Prepared by	IREC	Tested by	IREC
Pre-condition(s)	EDAE engine gives a result that DOES NOT requires to be approved.		
Test steps			
1	The workflow gets the EDAE engine result.		
2	The workflow generates a proposal ID to identify this information to be evaluated. It's been marked as "implemented".		
3	The workflow call SDN Controller interface to send the changes proposed by EDAE engine.		
Input data	The result of EDAE engine. As a summary:		

	<pre>{ "APPROVAL_REQUIRED": false, "PDC": [], "PMU": [], "PATH": [] }</pre> <p>See json details in annex 13.4</p>
Result	The result is a call to the SDN Controller API in order to apply the changes proposed by the EDAE engine.
Test Case Result	To be tested in pilot.

Table 71: EDAE_workflow4 unit test description

Test Case ID	EDAE_workflow4	Component	EDAE_workflow
Description	Test the data management from EDAE-Dashboard to implementation.		
Req ID		Priority	
Prepared by	IREC	Tested by	IREC
Pre-condition(s)	EDAE-Dashboard sends information regarding the proposal with a result “approved”.		
Test steps			
1	The workflow gets from RabbitMQ the information from EDAE-Dashboard where we know that the proposal was approved.		
2	The workflow updates the proposal stored in the database with the result of the approval process.		
3	The workflow call SDN Controller interface to send the changes contained in the proposal.		
Input data	The result of EDA-Dashboard approval process. <pre>{ "PROPOSAL_ID" : 53427606, "RESULT" : "APPROVED" }</pre>		
Result	The result is a call to the SDN Controller API in order to apply the changes proposed by the EDAE engine.		
Test Case Result	To be tested in pilot.		

Table 72: EDAE_workflow5 unit test description

Test Case ID	EDAE_workflow5	Component	EDAE_workflow
Description	Test the data management from EDAE-Dashboard.		
Req ID		Priority	
Prepared by	IREC	Tested by	IREC
Pre-condition(s)	EDAE-Dashboard sends information regarding the proposal with the result “rejected”.		
Test steps			
1	The workflow gets from RabbitMQ the information from EDAE-Dashboard where we know that the proposal was rejected.		
2	The workflow updates the proposal stored in the database with the result of the approval process.		
Input data	The result of EDAE-Dashboard approval process.		

	<pre>{ "PROPOSAL_ID" : 53427606, "RESULT" : "REJECTED" }</pre>
Result	Nothing external. Just the database update.
Test Case Result	Achieved in local environments. To be tested in pilot.

9.5 EDAE-Dashboard

9.5.1 Technical environment

To develop and test the EDAE dashboard, a real SDN network is replicated. For that end, a network simulator is employed, Mininet, to emulate the real behaviour of a SDN network. A custom SDN network is emulated to test the EDAE dashboard. Additionally, RYU is used as SDN controller. The topological information is obtained from the AIDB, through the AIDB API. The SDN-network information is collected from the SDN-C, in this case RYU. The generated information, in terms of statistics and network performance metrics, is stored in two different databases, RedisDB and MongoDB. The network simulator, Mininet, and the SDN-C, RYU, run on Ubuntu 16.04.6 LTS operating system while the databases run on Windows operating system.

Table 73: List of software used for implementation and testing

Base software	Version
Mininet	2.2.2
Ryu	OpenFlow 1.3
Mongo DB	4.2
REDIS	6.0.6
Python	3.7

In the followings, the description of the unit tests carried out to validate the EDAE-Dashboard performance is presented, together with the obtained results.

9.5.2 Unit tests

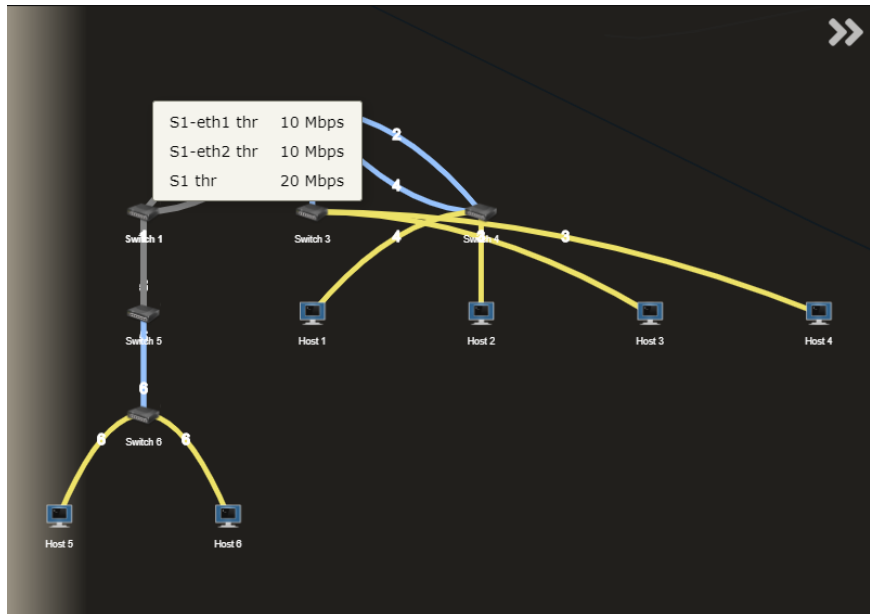
The unit tests are devoted to validating the EDAE-Dashboard implementation. In Table 74 a summary of the unit tests that have been carried out is presented.

Table 74: EDAE-Dashboard Unit tests summary

Unit test ID	Description
EDAE_DASH_001	Current SDN-network state representation
EDAE_DASH_002	EDAE proposal representation

Table 75: EDAE_DASH_001 - Current SDN network state representation

Test Case ID	EDAE_DASH_001	Component	EDAE Dashboard
--------------	---------------	-----------	----------------

Description	Representation of the current state of the SDN network in terms of host, switches and links. Besides, the network performance metrics for the switches and links weights are represented.		
Req ID		Priority	
Prepared by	AYESA	Tested by	AYESA
Pre-condition(s)	<ul style="list-style-type: none">AIDB API provides the topological information.A simple SDN-network is deployed in Mininet and a RYU controller and its REST API are also deployed.Redis and Mongo databases are deployed and accessible.		
Test steps			
1	The tester accesses the EDAE-Dashboard user interface.		
2	The tester can see the representation of the actual SDN-network status in terms of host, switches, and links. The links between nodes are different depending on the nodes type: Switch – switch: blue Switch – host: yellow		
3	The tester accesses the network performance metrics of each switch through a pop-up when cursor is placed over a switch.		
4	The tester accesses the network performance metrics of each host through a pop-up when cursor is placed over a host.		
Input data			
Result		Switch information	
			
		Host information	

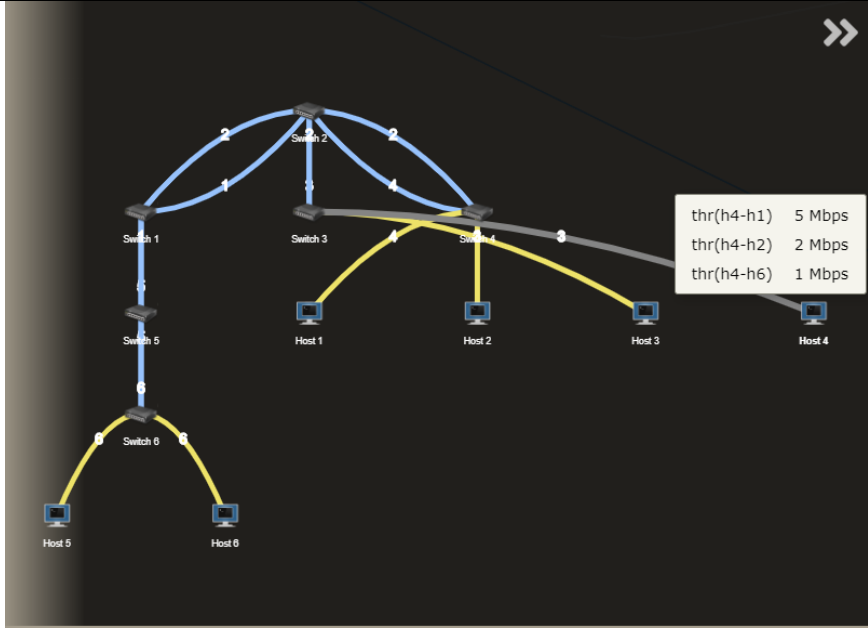
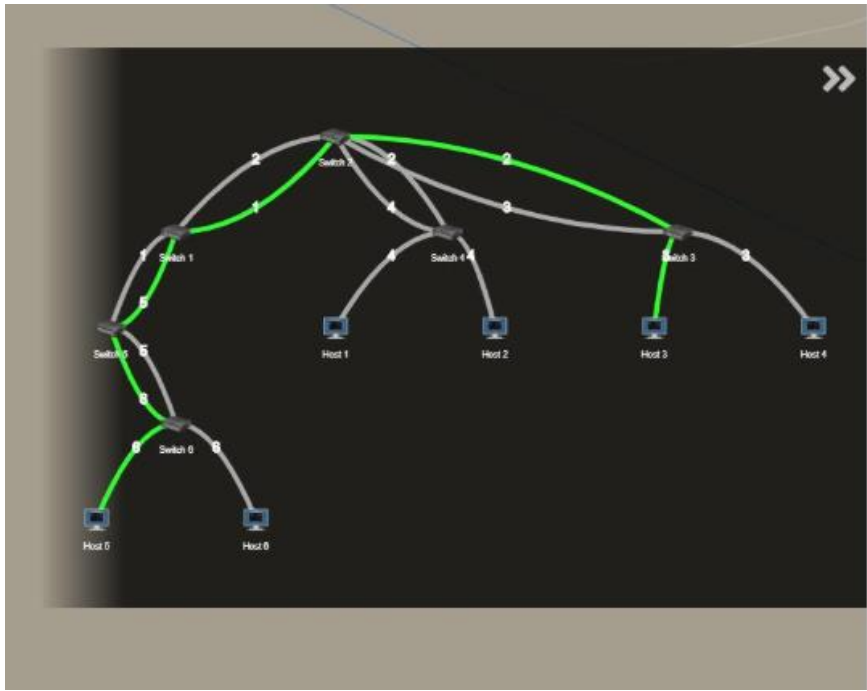
	
Test Case Result	Achieved, to be tested in Pilot.

Table 76: EDAE_DASH_002 – EDAE proposal representation

Test Case ID	EDAE_DASH_002	Component	EDAE Dashboard
Description	Representation of the EDAE proposals in terms of topological and routing routes changes.		
Req ID		Priority	
Prepared by	AYESA	Tested by	AYESA
Pre-condition(s)	<ul style="list-style-type: none">AIDB API provides the topological information.A simple SDN-network is deployed in Mininet and a RYU controller and its REST API are also deployed.Redis and Mongo databases are deployed and accessible.EDAE- tool provides the network proposal.		
Test steps			
1	The tester accesses the EDAE-Dashboard user interface.		
2	The tester can see the representation of the EDAE proposals over the SDN-network topology.		
3	The tester accesses the path between two hosts when clicking over the hosts. The proposed path is highlighted in green and the rest of the paths, between other hosts, are painted in grey.		
Input data			

Result	 <p>In this test, it is represented the path between host5 and host3</p>
Test Case Result	Achieved, to be tested in Pilot.

9.6 Assets inventory database

9.6.1 Technical environment

The unit test readiness has required the deployment of the different databases as well as the microservices in a developing environment. Unit tests have been focused on API invocations.

Table 77: Base software and version used for AIDB unit testing

Base software	Version
Java	OpenJDK 14
PostgreSQL	v9.4.8
Mongo DB	v4.0.16
Neo4J	v3.5.17
REDIS	v4.0.5
LDAP	LDAP
AIDB microservice	AIDB microservice
Postman as API client	

9.6.2 Unit tests

The unit tests are devoted to check out the AIDB implementation. Additionally, these tests are a way to a better understanding and appraisal of the AIDB functionality and, in the API cases, easy the integration develop of other components. A unit test can involve more than one API method and

even other unit test, therewith we manage a good test coverage thank to for example an asset creation is validated using a query method which has to return the expected values.

Table 78: AIDB Unit test summary

Unit test ID	Methods
AIDB_001 SDN Asset creation	Asset Create () AssetQuery ()
AIDB_002 SDN Asset update	AssetUpdate () AssetQuery ()
AIDB_003 SDN topology attribute update	AssetUpdate () AssetQuery ()
AIDB_004 SDN topology update	TopologyQuery() AssetUpdate () AssetQuery ()
AIDB_005 Grid definition registration	
AIDB_006 SDN-Grid relationship creation	TopologyQuery() AssetUpdate () AssetQuery ()

Table 79: AIDB_001 SDN Asset creation - Unit Test

Test Case ID	AIDB_001	Component	AIDB
Description	A complete SDN architecture made up of SDN Switches, Hosts, and an SDN controller, and their relationships are registered into the database.		
Req ID		Priority	
Prepared by		Tested by	
Pre-condition(s)	<ul style="list-style-type: none">• The API User has to be registered in the LDAP with valid credentials.• The API User has to be bound to a Use case, i.e., UC0.• The database partition devoted to the chosen Use Case has to be clean.		
Test steps			
1	Invoke the AIDB Create API using the Use case definition depicted in the Figure 73: AIDB - SDN asset inventory example.		
2	Invoke the AIDB Query API to get the entire asset information		
Input data			
Result			
Test Case Result	{Not achieved, Achieved, To be tested in Pilot}		

Table 80: AIDB_002 SDN Asset update - Unit Test

Test Case ID	AIDB_002	Component	AIDB
Description	A part of the asset information is updated without topology impact.		

Req ID		Priority	
Prepared by		Tested by	
Pre-condition(s)	<ul style="list-style-type: none">Having launched the AIDB_001 SDN Asset creation		
Test steps			
1	Invoke the AIDB Update API indicating a modification in the description field of some assets.		
2	Invoke the AIDB Query API to get the asset information.		
Input data			
Result			
Test Case Result	{Not achieved, Achieved, To be tested in Pilot}		

Table 81: AIDB_003 SDN topology attribute update - Unit Test

Test Case ID	AIDB_003	Component	AIDB
Description	A part of the information associated to the SDN topology is updated.		
Req ID		Priority	
Prepared by		Tested by	
Pre-condition(s)	<ul style="list-style-type: none">Launch after having tested the AIDB_001 SDN Asset creation.		
Test steps			
1	Invoke the AIDB Update API indicating a modification in the description field of some assets.		
2	Invoke the AIDB Query API to get the entire asset information.		
Input data			
Result			
Test Case Result	{Not achieved, Achieved, To be tested in Pilot}		

Table 82: AIDB_004 SDN topology update - Unit Test

Test Case ID	AIDB_004	Component	AIDB
Description	A part of SDN topology is updated with a new relationship and deleting another.		
Req ID		Priority	
Prepared by		Tested by	
Pre-condition(s)	<ul style="list-style-type: none">Launch after having tested the AIDB_001 SDN Asset creation.Select a target host and his related SDN switch to foresee what relationship will be deleted.		
Test steps			



1	Invoke the AIDB Topology query, from a source host towards the target host (selected for test case). After the invocation the query will indicate a path from source to the target host.
2	Invoke the AIDB Update API deleting the current relationship between the target host and its immediate SDN switch, as well, adding a new relationship that binds the target host to other SDN Switch.
3	Invoke the AIDB Query API to get the entire asset information. The query result should not contain the deleted relationship, and the new relationship has to appear.
4	Repeat the invocation of the first step. Now, the obtained path has to reflex the new topology, achieving the target host through the other SDN switch.
Input data	
Result	
Test Case Result	{Not achieved, Achieved, To be tested in Pilot}

Table 83: AIDB_005 Grid definition registration - Unit Test

Test Case ID	AIDB_005	Component	AIDB
Description	The AIDB registers Grid Asset references regarding a given Grid definition in <i>pandapower</i> format. Grid Assets are created into the AIDB with his correct asset type. Next, the same Grid definition file can be retrieved lately.		
Req ID		Priority	
Prepared by		Tested by	
Pre-condition(s)	The API User has to be registered in the LDAP with valid credentials. The API User has to be bound to a Use case, i.e., UC0.		
Test steps			
1	Invoke the <i>GridModelUpdate</i> using the pandapower file corresponding to the Grid model example included in the Annexes.		
2	Invoke the AIDB Query API to get the entire asset information. The query result should contain all of Grid Assets included in the input Grid definition file.		
3	Invoke the <i>GridModelQuery()</i> . The AIDB will return the same definition file used in the first step.		
Input data			
Result			
Test Case Result	{Not achieved, Achieved, To be tested in Pilot}		

Table 84: AIDB_006 SDN-Grid relationship creation - Unit Test

Test Case ID	AIDB_006	Component	AIDB
--------------	----------	-----------	------

Description	The AIDB registers Grid Asset references regarding a given Grid definition in pandapower format. Grid Assets are created into the AIDB with his correct asset type. Next, the same Grid definition file can be retrieved lately.		
Req ID		Priority	
Prepared by		Tested by	
Pre-condition(s)	<ul style="list-style-type: none">Having launched the test AIDB_001 and AIDB_005.		
Test steps			
1	Invoke the AIDB Update API creating three relationships, the first one between the “Host 1” and the “Diesel generator 1” (Diesel generator 1 monitoring is able), the second between the “Diesel generator 1” and the “Host 1” (This way allows the control operations for the Diesel generator 1), finally between and the “Diesel generator 2” and “Host 2” (Diesel generator 2 only is able to monitoring.		
2	Invoke the AIDB Query API to get the entire asset information. The query result should contain the relationships have just been created.		
3	Invoke the AIDB Topology query, indicating the Switch 1 as source asset, and selecting border flag.		
4	Invoke the AIDB Topology query, indicating the Host 1 as source asset, and selecting border flag.		
	Repeat steps 3 and 4 with downstream ‘No’		
5	Repeat steps 3 and 4 with Switch 2 and Host 2		
Input data			
Result			
Test Case Result	{Not achieved, Achieved, To be tested in Pilot}		

10 Innovation Summary

An innovative method for self-healing applied to smart grids is presented. This method is taking advantage on the reconfiguration of information flows that offers the SDN-technology. A specific application called EDAAE was developed for that purpose. It increases the resilience on smart grids by covering the following points:

- Restoring power system observability and controllability by reconfiguring the communication network as quickly as possible, also considering the latency and the bandwidth on the communication links.
- Maximizing the system observability and controllability by recovering as many communication endpoints as possible (e.g., all disconnected PMUs).
- Disconnecting and isolating from the network the assets that are under attack or malfunctioning and could compromise the behaviour of the other assets.
- Automating the network self-healing actions based on the results from a risk assessment [D3.5] that evaluates the network assets together with the power system elements.

EDAAE is using a multi-objective genetic algorithm and of the MILP algorithm. The genetic algorithm manages the network path reconstruction and is activated when a security constrain is violated. The MILP calculates the optimal connection of PMU to a PDC, when a PDC is disconnected (packet loss percentage above certain threshold).

EDAAE results are drawn in an independent dashboard called EDAAE-Dashboard, which monitor the current state of the SDN-Network, graphically represents the EDAAE outputs, allows the user interaction to validate the changes on the network and presents some statistics from all the represented nodes. An interesting feature is that the operator can accept or reject the proposed changes totally or partially, which means that the operator could accept only part of the EDAAE proposal.

A customized SDN-Controller is developed using Ryu framework, and specifically some improvements to the original implementation were introduced:

- The API REST commands that retrieve and update the configuration of SND-switch using open Flow protocol (Ofctl_rest) had been upgraded to use OpenFlow version 1.3.
- The API REST commands to retrieve the network topology (rest_topology) had been improved to support inputs and outputs in decimal format (datapath IDs and port numbers) and to achieve compliance between rest_topology and ofctl_rest.
- The API REST commands to retrieve the network topology (rest_topology) integrate information regarding the delay monitoring expansion.
- Security features:
 - TLS secured by a nginx proxy server in each SDN-C
 - Basic http authentication to authorize each client to access NBI

The SDN-C is complemented with a SDN-Dashboard developed to provide a user-friendly interface to monitor and manage the network. As a novelty this interface is decoupled form the SDN-C and allows the connection of multiple SDN-Cs using REST-based NBI which in turn provides a fault-tolerant capability of the SDN-Control layer.

Finally, a cross component is designed to store ICT network and grid assets. This is AIDB, which relates grid assets towards the SDN Hosts. This component is crucial for the whole SDN-microSENSE framework as it is usable not only for SDN-SELF but also S-RAF and XL-SIEM.

Although the technical achievements, there is room for improvement and some of the features that should be considered in future tools versions are:

- Synchronization of EPES real grid assets with AIDB. In order to collect the real-time data from devices.
- Inventory all SDN assets automatically, without human intervention by the mean an AIDB daemon.
- EDAE is connected with one SDN-C, include the possibility to make decisions in a coordinated way through more than one SDN-C in the same network.
- Explore other algorithms, as machine learning, for EDAE to learn from the decisions taken by the user.

A publication is produced due to the content of this deliverable:

- Toni Cantero-Gubert, Alba Colet, Pol Paradell, and J. L. Domínguez-García. 2020. *“Building a testing environment for SDN networks analysis for electrical grid applications”*. In Proceedings of the 15th International Conference on Availability, Reliability and Security (ARES '20). Association for Computing Machinery, New York, NY, USA, Article 112, 1–6. DOI:<https://doi.org/10.1145/3407023.3409230>

11 Conclusions

This deliverable explained in detail the components involved in the network management process in case of a failure or a cyber-attack in the EPES.

EDAE component is the brain of the process and it has been designed to maximize the observability and the QoS of the communication network. EDAE is not only rearranging logical connection in the communication network but also deciding the data paths based on the risks of the assets that are present in the electrical grid and information of the network switches.

As seen in section 2, neither traditional self-healing mechanisms nor those strategies used in smart grids does not offer such a complete reaction as the combination of EDAE and SDN technology is able to provide. The innovations of this network management process is highlighted in section 9.

Functional and non-functional requirements listed in section 3 has been addressed through the description of the component and the unit testing.

The reader will be able to allocate the components described in this document in the overall architecture of SDN-microSENSE project and to see the interdependencies and integration of those components: AIDB, S-RAF, EDAE and SDN-C. Special mention is done to the user graphic interface of EDAE and SDN-C: EDAE-Dashboard and SDN-Dashboard.

Even though the actual interfaces and integration points could be updated in the context of WP7, they are explained in detail in section 5 to 7 in this deliverable.

From the requirements, following the design and implementation, the last part of this document is the unit testing proposed for all components and its processes. It demonstrates the success of the implementation and pointed out the limitations of the design.

12 References

- [1] M. H. Rehmani, A. Davy, B. Jennings and C. Assi, "Software defined networks-based smart grid communication: A comprehensive survey," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 3, pp. 2637-2670, 2019.
- [2] E. A. Leal and J. F. Botero, "Software defined power substations: An architecture for network communications and its control plane," in *2016 8th IEEE Latin-American Conference on Communications (LATINCOM)*, 2016.
- [3] A. Cahn, J. Hoyos, M. Hulse and E. Keller, "Software-defined energy communication networks: From substation automation to future smart grids," in *2013 IEEE International conference on smart grid communications (SmartGridComm)*, 2013.
- [4] A. Depari, A. Flammini, M. Lavarini and E. Sisinni, "Enhanced software defined meter for smart utility networks," *Computer Standards & Interfaces*, vol. 48, pp. 160-172, 2016.
- [5] Z. Zhou, J. Gong, Y. He and Y. Zhang, "Software defined machine-to-machine communication for smart energy management," *IEEE Communications Magazine*, vol. 55, no. 10, pp. 52-60, 2017.
- [6] Y. Xin, I. Baldine, J. Chase, T. Beyene, B. Parkhurst and A. Chakraborty, "Virtual smart grid architecture and control framework," in *2011 IEEE International Conference on Smart Grid Communications (SmartGridComm)*, 2011.
- [7] P. Manso, J. Moura and C. Serrão, "SDN-Based Intrusion Detection System for Early Detection and Mitigation of DDoS Attacks," *Information*, vol. 10, no. 3, p. 106, 2019.
- [8] P. Radoglou-Grammatikis, P. Sarigiannidis, G. Efstathiopoulos, P.-A. Karypidis and A. Sarigiannidis, "Diderot: An intrusion detection and prevention system for dnp3-based scada systems," in *Proceedings of the 15th International Conference on Availability, Reliability and Security*, 2020.
- [9] R. K. Das, F. H. Pohrmen, A. K. Maji and G. Saha, "FT-SDN: A Fault-Tolerant Distributed Architecture for Software Defined Network," *WIRELESS PERSONAL COMMUNICATIONS*, 2020.
- [10] F. Botelho, A. Bessani, F. M. Ramos and P. Ferreira, "On the design of practical fault-tolerant SDN controllers," in *2014 third European workshop on software defined networks*, Lisbon, IEEE, 2014, pp. 73-78.
- [11] A. Bucchiarone, H. Muccini and P. Pelliccione, "Architecting Fault-tolerant component-based systems: from requirements to testing," *Electronic Notes in Theoretical Computer Science*, vol. 168, pp. 77-90, 2007.
- [12] A. Rehman, R. L. Aguiar and J. P. Barraca, "Fault-Tolerance in the Scope of Software-Defined Networking (SDN)," *IEEE Access*, vol. 7, pp. 124474-124490, 2019.

- [13] J. Chen, J. Chen, F. Xu, M. Yin and W. Zhang, "When software defined networks meet fault tolerance: A survey," in *International conference on algorithms and architectures for parallel processing*, 2015.
- [14] A. Rehman, R. L. Aguiar and J. P. Barraca, "A proposal for faulttolerant and self-healing hybrid sdn control network," in *Simpósio de Informática*, Aveiro, Portugal, 2017.
- [15] A. Modarresi, S. Gangadhar and J. P. Sterbenz, "A framework for improving network resilience using SDN and fog nodes," *9th International Workshop on Resilient Networks Design and Modeling (RNDM)*, Alghero, vol. doi: 10.1109/RNDM.2017.8093036., pp. 1-7, 2017.
- [16] X. Zhang, K. Wei, L. Guo, W. Hou and J. Wu, "SDN-based Resilience Solutions for Smart Grids," *2016 International Conference on Software Networking (ICSN)*, vol. doi: 10.1109/ICSN.2016.7501931, pp. 1-5, 2016.
- [17] C. C. Machado, L. Z. Granville and A. Schaeffer-Filho, "ANSwer: Combining NFV and SDN features for network resilience strategies," *IEEE Symposium on Computers and Communication (ISCC)*, vol. doi: 10.1109/ISCC.2016.7543771, pp. 391-396, 2016.
- [18] M. Zadsar, M. R. Haghifam and S. M. M. Larimi, "Approach for self-healing resilient operation of active distribution network with microgrid," *IET Generation, Transmission & Distribution*, vol. 11, no. 18, pp. 4633-1643, 2017.
- [19] C. Colson, M. Nehrir and R. Gunderson, "Distributed multi-agent microgrids: a decentralized approach to resilient power system self-healing," in *2011 4th International Symposium on Resilient Control Systems*, Boise, ID, USA, 2011.
- [20] IEEE Std 1547.4-2011, "IEEE Guide for Design, Operation, and Integration of Distributed Resource Island Systems with Electric Power Systems," 2011 July 20. [Online]. Available: <https://ieeexplore.ieee.org/servlet/opac?punumber=5960749>.
- [21] Z. Wang and J. Wang, "Self-Healing Resilient Distribution Systems Based on Sectionalization Into Microgrids," *IEEE Transactions on Power Systems*, vol. 30, no. 6, pp. 3139-3149, 2015.
- [22] ENTSO-E, "Operational Reserve Ad-hoc Team Report Final Version," [Online]. Available: https://eepublicdownloads.azureedge.net/clean-documents/pre2015/resources/LCFR/2012-06-14_SOC-AhT-OR_Report_final_V9-3.pdf. [Accessed 23 September 2020].
- [23] C. Speckamp, SOPTIM, 26 10 2016. [Online]. Available: <https://www.soptim.de/de/blog/detail/Software-von-SOPTIM-managt-zuverlaessig-Markt-fuer-Minutenreserveleistung-13T/>. [Accessed 17 09 2020].
- [24] NREL, "Operating Reserves and Variable Generation," 2011. [Online]. Available: <https://www.nrel.gov/docs/fy11osti/51978.pdf>. [Accessed 20 November 2020].

- [25] European Commission, “Commission Regulation (EU) 2017/2196 - Network Code on Electricity Emergency and Restoration,” 24 November 2017. [Online]. Available: <https://eur-lex.europa.eu/eli/reg/2017/2196>. [Accessed 20 November 2020].
- [26] 50Hertz, Amprion, TenneT, TransnetBW, „Systemschutzplan der vier deutschen Übertragungsnetzbetreiber,” 14 December 2018. [Online]. Available: <https://www.netztransparenz.de/portals/1/Content/EU-Network-Codes/ER-Verordnung/Systemschutzplan%20der%20%C3%9CNB%20-%20Hauptdokument.pdf>. [Zugriff am 20 November 2020].
- [27] VDE FNN, “VDE-AR-N4142 - Automatic remedial measures to prevent system breakdowns,” 2020.
- [28] A. Gómez-Expósito, A. J. Conejo and C. Cañizares, Electric energy systems: analysis and operation, CRC press, 2018.
- [29] J. Machowski, Z. Lubosny, J. W. Bialek and J. R. Bumby, Power Systems Dynamics: Stability and Control, John Wiley & Sons, 2020.
- [30] European Commission, Smart Grid Mandate M/490 , [Online]. Available: <https://ec.europa.eu/growth/tools-databases/mandates/index.cfm?fuseaction=search.detail&id=475#>. [Accessed 2020 September 23].
- [31] EPRI, “Common Functions for Smart Inverters: 4th Edition,” Palo Alto, CA, 3002008217, 2016.
- [32] Belectric GmbH, “Solar PV-BatteryHybrid System – Test Report,” 2019. [Online]. Available: <https://www.nationalgrideso.com/document/140311/download>. [Accessed 18 November 2020].
- [33] K. Stein, M. Tun, M. Matsuura and R. Rocheleau, “Characterization of a Fast Battery Energy StorageSystem for Primary Frequency Response,” *Energies*, 2018.
- [34] S. Sproul, “Do you know the difference between Virtual Inertia and Fast Frequency Response?,” *pv magazine Australia*, [Online]. Available: <https://www.pv-magazine-australia.com/2020/04/08/do-you-know-the-difference-between-virtual-inertia-and-fast-frequency-response/>. [Accessed 18 November 2020].
- [35] WSW Wuppertaler Stadtwerke GmbH, „Demand Side Management vs. Demand Response,” [Online]. Available: <https://www.wsw-online.de/happy-power-hour/wissensbereich/demand-side-management-vs-demand-response/>. [Zugriff am 18 November 2020].
- [36] NERC, “Demand Response Availability Data System (DADS): Phase I & II Final Report,” [Online]. Available: <https://www.nerc.com/pa/RAPA/dads/DADSPHaseII/DADS%20Phase%20I%20and%20II%20Report.pdf>. [Accessed 18 November 2020].

-
- [37] IEA, "Demand Response," 2020. [Online]. Available: <https://www.iea.org/reports/demand-response>. [Accessed 18 November 2020].
- [38] A. Zangeneh and M. Moradzadeh, "Self-healing: Definition, Requirements, Challenges and Methods.," in *Microgrid Architectures, Control and Protection Methods*, Springer, 2020.
- [39] M. Roos, P. Nguyen, J. Morren and J. Slootweg, "Sectionalizing Distribution Networks Concept and System Framework," Eindhoven, 2018.
- [40] X. Gao and X. Ai, "The Application of Self-Healing Technology in Smart Grid," in *2011 Asia-Pacific Power and Energy Engineering Conference*, Wuhan, 2011.
- [41] H. Lin, "Self-Healing Attack-Resilient PMU Network for Power System Operation," *IEEE Transactions on Smart Grid* vol. 9 no. 3, pp. 1551-1565, May 2018.
- [42] C. De Las Muñecas and A. Tirados, "Intrusion Detection Systems in SDN-based Self-Healing PMU Networks," Department of Computer Science, Illinois Institute of Technology, Illinois, 2016.
- [43] SDN-microSENSE consortium, "Deliverable D2.3: Platform Specifications and Architecture," European Commission, 2020.
- [44] [Online]. Available: https://ryu.readthedocs.io/en/latest/app/ofctl_rest.html.
- [45] [Online]. Available: https://github.com/faucetsdn/ryu/blob/master/ryu/app/rest_topology.py.
- [46] J. Deacon, "Model-View-Controller (MVC) Architecture," Computer Systems Development, Consulting and Training, 2009.
- [47] [Online]. Available: <https://github.com/python-zk/kazoo>.
- [48] [Online]. Available: <https://github.com/martimy/flowmanager>.
- [49] K. Papachristou, T.-I. Theodorou, S. Papadopoulos, A. Protogerou, A. Drosou and D. Tzovaras, "Routing policy verification for enhanced energy quality of service and security monitoring in IoT," in *Proceedings of the 23rd Pan-Hellenic Conference on Informatics*, Nicosia, Cyprus, 2019.
- [50] T. A. Q. Pham, Y. Hadjadj-Aoul and A. Outtagarts, "Deep reinforcement learning based qos-aware routing in knowledge-defined networking," in *International Conference on Heterogeneous Networking for Quality, Reliability, Security and Robustness*, 2018.
- [51] M. Rezaee and M. H. Y. Moghaddam, "SDN-based quality of service networking for wide area measurement system," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 5, pp. 3018-3028, 2019.
- [52] J. B. Hong, S. Yoon, H. Lim and D. S. Kim, "Optimal network reconfiguration for software defined networks using shuffle-based online MTD," in *2017 IEEE 36th Symposium on Reliable Distributed Systems (SRDS)*, 2017.

- [53] M. Wang, J. Liu, J. Mao, H. Cheng, J. Chen and C. Qi, "RouteGuardian: Constructing secure routing paths in software-defined networking," *Tsinghua Science and Technology*, vol. 22, no. 4, pp. 400-412, 2017.
- [54] Y. Qu, X. Liu, D. Jin, Y. Hong and C. Chen, "Enabling a Resilient and Self-healing PMU Infrastructure Using Centralized Network Control," in *Proceedings of the 2018 ACM International Workshop on Security in Software Defined Networks & Network Function Virtualization*, 2018.
- [55] T. P. Lillicrap, *Continuous control with deep reinforcement learning*, 2015: arXiv e-prints.
- [56] A. Mambrini and D. Izzo, "Pade: A parallel algorithm based on the moea/d framework and the island model," in *International Conference on Parallel Problem Solving from Nature*, 2014.
- [57] D. Izzo, M. Rucinski and F. Biscani, "The Generalized Island Model," in *Parallel Architectures and Bioinspired Algorithms*, Springer, 2012, pp. 151-169.
- [58] P. T. A. Quang, J.-M. Sanner, C. Morin and Y. Hadjadj-Aoul, "Multi-objective multi-constrained QoS Routing in large-scale networks: A genetic algorithm approach," in *International conference on smart communications in network technologies (SaCoNeT)*, 2018.
- [59] IEEE SA - The IEEE Standards Association, "60255-118-1-2018 - IEEE/IEC International Standard - Measuring relays and protection equipment - Part 118-1: Synchrophasor for power systems - Measurements," [Online]. Available: <https://standards.ieee.org/standard/60255-118-1-2018.html>. [Accessed 29 September 2020].
- [60] IEEE SA - The IEEE Standards Association, "C37.118.2-2011 - IEEE Standard for Synchrophasor Data Transfer for Power Systems," [Online]. Available: https://standards.ieee.org/standard/C37_118_2-2011.html. [Accessed 29 September 2020].
- [61] IEEE SA - The IEEE Standards Association, "C37.247-2019 - IEEE Standard for Phasor Data Concentrators for Power Systems," [Online]. Available: https://standards.ieee.org/standard/C37_247-2019.html. [Accessed 29 September 2020].
- [62] J. W. Guck, A. Van Bemten, M. Reisslein and W. Kellerer, "Unicast QoS routing algorithms for SDN: A comprehensive survey and performance evaluation," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 1, pp. 388-415, 2017.
- [63] "Daoquan Li, Xue Wang, Yingnan Jin, & Haoxin Liu (2020). Research on QoS routing method based on NSGAII in SDNJournal of Physics: Conference Series, 1656, 012027."
- [64] S. Kumar, M. K. Soni and D. K. Jain, "Requirements and challenges of PMUs communication in WAMS environment," *Far East Journal of Electronics and Communications*, vol. 13, no. 2, pp. 121-135, 2014.
- [65] W. Tao, M. Ma, C. Fang, W. Xie, M. Ding, D. Xu and Y. Shi, "Design and Application of a Distribution Network Phasor Data Concentrator," *Applied Sciences*, vol. 10, no. 8, p. 2942, 2020.

- [66] P. Kansal and A. Bose, "Bandwidth and Latency Requirements for Smart Transmission Grid Applications," *IEEE Transactions on Smart Grid*, vol. 3, no. 3, pp. 1344-1352, 2012.
- [67] The Office of the General Counsel, [Online]. Available: <https://www.energy.gov/gc/downloads/communications-requirements-smart-grid-technologies>.
- [68] <https://www.spp.org/documents/55158/spp%20pmu%20communication%20handbook%20v1.0.pdf>. [Online].
- [69] a. [https://www.ciscopress.com/articles/article.asp?p=357102#:~:text=One-way%20latency%20\(mouth%20to](https://www.ciscopress.com/articles/article.asp?p=357102#:~:text=One-way%20latency%20(mouth%20to). [Online].
- [70] https://www.arubanetworks.com/assets/ds/DS_2530SwitchSeries.pdf. [Online].
- [71] K. Deb, A. Pratap, S. Agarwal and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE transactions on evolutionary computation*, vol. 6, no. 2, pp. 182-197, 2002.
- [72] C. Hewicker, O. Werner and H. Ziegler, "Qualitative Analysis of Cross-Border Exchange of Balancing Energy and Operational Reserves between Netherlands and Belgium," KEMA Consulting GmbH, Bonn, 2013.
- [73] [Online]. Available: <https://www.iana.org/assignments/ieee-802-numbers/ieee-802-numbers.xhtml>.
- [74] [Online]. Available: <https://www.iana.org/assignments/protocol-numbers/protocol-numbers.xhtml>.
- [75] [Online]. Available: <https://www.iana.org/assignments/icmp-parameters/icmp-parameters.xml>.
- [76] [Online]. Available: <https://www.iana.org/assignments/icmp-parameters/icmp-parameters.xml>.
- [77] [Online]. Available: <https://www.iana.org/assignments/arp-parameters/arp-parameters.xhtml>.

13 Annexes

13.1 EDAE Interface details

Example of a preliminary version of the output S-RAF generates:

```
{
  "ASSETS":[
    {
      "name":"SDN-Switch",
      "type":"h",
      "category":"NetworkComponent/Switch",
      "gdpr":false,
      "assetId":1,
      "riskassessmentId":1714,
      "businesspartnerId":1,
      "risklevel":"M",
      "threats":[
        {
          "threat":"DEFAULT-1",
          "name":"DEFAULT",
          "level":"VH",
          "vulnerabilities":[
            {
              "vulnerability":"CVE-2019-1890",
              "level":"L",
              "impact":"M",
              "risklevel":"M",
              "privacyfunctionalimpactscore":0,
              "privacyimpactscorescore":0,
              "privacydatatypes":["M"]
            }
          ],
          "privacydatatypescore":0,
          "privacyscore":0,
          "privavyriskscore":0,
          "cvssscore":3,
          "cvss":"L"
        }
      ],
      "result":"M"
    }
  ],
  "cumulativeRiskLevel":"M",
  "contributingirls":["M"]
},
"controls":[]
],
"links":[
  {
    "type":"CONNECTED_TO",
    "assetId":3
  },
  {
    "type":"CONNECTED_TO",
    "assetId":5
  }
],
"businessValue":"VH",
"tags":{
```

```

    }
  },
  {
    "name":"SDN-enabled RTU",
    "category":"Business Service/Infrastructure Service",
    "gdpr":false,
    "assetId":2,
    "riskassessmentId":1714,
    "businesspartnerId":1,
    "risklevel":"H",
    "threats":[
      {
        "threat":"DEFAULT-1",
        "name":"DEFAULT",
        "level":"VH",
        "vulnerabilities":[
          {
            "vulnerability":"CVE-2019-14931",
            "level":"VL",
            "impact":"VH",
            "risklevel":"H",
            "privacyfunctionalimpactscore":0,
            "privacyimpactscorescore":0,
            "privacydatatypes":["H"]
          }
        ],
        "privacydatatypescore":0,
        "privacyscore":0,
        "privavyriskscore":0,
        "cvssscore":6,
        "cvss":"H"
      }
    ],
    "result":"H"
  }
],
"cumulativeRiskLevel":"H",
"contributingirls":["H"]
},
"controls":[]
],
"links":[
  {
    "type":"CONNECTED_TO",
    "assetId":1
  }
],
"businessValue":"VH",
"tags":{
  "attackpath":"source"
}
},
{
  "name":"Programmable Logic Controller 1",
  "type":"h",
  "category":"Computer/Server",
  "gdpr":false,
  "assetId":3,
  "riskassessmentId":1714,
```





```

"businesspartnerId":1,
"risklevel":"H",
"threats":[
{
  "threat":"DEFAULT-1",
  "name":"DEFAULT",
  "level":"VH",
  "vulnerabilities":[
    {
      "vulnerability":"CVE-2017-6031",
      "level":"VH",
      "impact":"M",
      "risklevel":"H",
      "description":"A Header Injection
issue was discovered in Certec EDV GmbH at vise
scada prior to Version 3.0. An \"improper
neutralization of HTTP headers for scripting
syntax\" issue has been identified, which may allow
remote code execution.",
      "privacyfunctionalimpactscore":0,
      "privacyimpactscorescore":0,
      "privacydatatypes":[]
    },
    {
      "privacydatatypescore":0,
      "privacyscore":0,
      "privavyriskscore":0,
      "cvssscore":6.8,
      "cvss":"H"
    }
  ],
  "privacydatatypescore":0,
  "privacyscore":0,
  "privavyriskscore":0,
  "cvssscore":6.8,
  "cvss":"H"
},
{
  "vulnerability":"CVE-2013-2780",
  "level":"VH",
  "impact":"L",
  "risklevel":"M",
  "privacyfunctionalimpactscore":0,
  "privacyimpactscorescore":0,
  "privacydatatypes":[]
},
{
  "privacydatatypescore":0,
  "privacyscore":0,
  "privavyriskscore":0,
  "cvssscore":7.8,
  "cvss":"H"
},
{
  "vulnerability":"CVE-2017-6033",
  "level":"VH",
  "impact":"M",
  "risklevel":"H",
  "description":"A DLL Hijacking issue
was discovered in Schneider Electric Interactive
Graphical SCADA System (IGSS) Software, Version
12 and previous versions. The software will execute
a malicious file if it is named the same as a
legitimate file and placed in a location that is
earlier in the search path.",
  "privacyfunctionalimpactscore":0,

```

```

"privacyimpactscorescore":0,
"privacydatatypes":[]
},
{
  "privacydatatypescore":0,
  "privacyscore":0,
  "privavyriskscore":0,
  "cvssscore":6.8,
  "cvss":"H"
},
{
  "result":"H"
},
{
  "cumulativeRiskLevel":"H",
  "contributinggirls":[]
},
{
  "controls":[]
},
{
  "links":[]
},
{
  "type":"USED_BY",
  "assetId":4
},
{
  "businessValue":"VH",
  "tags":{
  }
},
{
  "name":"Administrator",
  "category":"Organizational/Personnel",
  "gdpr":false,
  "assetId":4,
  "riskassessmentId":1714,
  "businesspartnerId":1,
  "risklevel":"",
  "threats":[]
},
{
  "cumulativeRiskLevel":"",
  "contributinggirls":[]
},
{
  "controls":[]
},
{
  "links":[]
},
{
  "businessValue":"H",
  "tags":{
    "attackpath":"destination"
  }
},
{

```




```

{
  "name": "Programmable Logic Controller 2",
  "type": "h",
  "category": "Computer/Server",
  "gdpr": false,
  "assetId": 5,
  "riskassessmentId": 1714,
  "businesspartnerId": 1,
  "risklevel": "H",
  "threats": [
    {
      "threat": "DEFAULT-1",
      "name": "DEFAULT",
      "level": "VH",
      "vulnerabilities": [
        {
          "vulnerability": "CVE-2013-2780",
          "level": "VH",
          "impact": "L",
          "risklevel": "M",
          "privacyfunctionalimpactscore": 0,
          "privacyimpactscorescore": 0,
          "privacydatatypes": [
            ],
          "privacydatatypescore": 0,
          "privacyscore": 0,
          "privavyriskscore": 0,
          "cvssscore": 7.8,
          "cvss": "H"
        }
      ],
      "vulnerability": "CVE-2017-6033",
      "level": "VH",
      "impact": "M",
      "risklevel": "H",
      "description": "A DLL Hijacking issue
was discovered in Schneider Electric Interactive
Graphical SCADA System (IGSS) Software, Version
12 and previous versions. The software will execute
a malicious file if it is named the same as a
legitimate file and placed in a location that is
earlier in the search path.",
      "privacyfunctionalimpactscore": 0,
      "privacyimpactscorescore": 0,
      "privacydatatypes": [
        ],
      "privacydatatypescore": 0,
      "privacyscore": 0,
      "privavyriskscore": 0,
      "cvssscore": 6.8,
      "cvss": "H"
    }
  ],
  "vulnerability": "CVE-2017-6031",
  "level": "VH",
  "impact": "M",
  "risklevel": "H",

```

```

      "description": "A Header Injection
issue was discovered in Certec EDV GmbH atvise
scada prior to Version 3.0. An \"improper
neutralization of HTTP headers for scripting
syntax\" issue has been identified, which may allow
remote code execution.",
      "privacyfunctionalimpactscore": 0,
      "privacyimpactscorescore": 0,
      "privacydatatypes": [
        ],
      "privacydatatypescore": 0,
      "privacyscore": 0,
      "privavyriskscore": 0,
      "cvssscore": 6.8,
      "cvss": "H"
    }
  ],
  "result": "H"
},
],
"cumulativeRiskLevel": "H",
"contributingfactors": [
  "H",
  "M"
],
"controls": [
],
"links": [
  {
    "type": "CONNECTED_TO",
    "assetId": 6
  }
],
"businessValue": "VH",
"tags": {
}
},
{
  "name": "SCADA 1",
  "type": "h",
  "category": "Computer/Server",
  "gdpr": false,
  "assetId": 6,
  "riskassessmentId": 1714,
  "businesspartnerId": 1,
  "risklevel": "H",
  "threats": [
    {
      "threat": "DEFAULT-1",
      "name": "DEFAULT",
      "level": "VH",
      "vulnerabilities": [
        {
          "vulnerability": "CVE-2013-2780",
          "level": "VH",
          "impact": "L",

```




```

    "risklevel": "M",
    "privacyfunctionalimpactscore": 0,
    "privacyimpactscopescore": 0,
    "privacydatatypes": [
    ],
    "privacydatatypescore": 0,
    "privacyscore": 0,
    "privavyriskscore": 0,
    "cvssscore": 7.8,
    "cvss": "H"
  },
  {
    "vulnerability": "CVE-2017-6031",
    "level": "VH",
    "impact": "M",
    "risklevel": "H",
    "description": "A Header Injection
issue was discovered in Certec EDV GmbH atvise
scada prior to Version 3.0. An \"improper
neutralization of HTTP headers for scripting
syntax\" issue has been identified, which may allow
remote code execution.",
    "privacyfunctionalimpactscore": 0,
    "privacyimpactscopescore": 0,
    "privacydatatypes": [
    ],
    "privacydatatypescore": 0,
    "privacyscore": 0,
    "privavyriskscore": 0,
    "cvssscore": 6.8,
    "cvss": "H"
  },
  {
    "vulnerability": "CVE-2017-6033",
    "level": "VH",
    "impact": "M",
    "risklevel": "H",
    "description": "A DLL Hijacking issue
was discovered in Schneider Electric Interactive
Graphical SCADA System (IGSS) Software, Version
12 and previous versions. The software will execute
a malicious file if it is named the same as a
legitimate file and placed in a location that is
earlier in the search path.",
    "privacyfunctionalimpactscore": 0,
    "privacyimpactscopescore": 0,
    "privacydatatypes": [
    ],
    "privacydatatypescore": 0,
    "privacyscore": 0,
    "privavyriskscore": 0,
    "cvssscore": 6.8,
    "cvss": "H"
  }
],
"result": "H"

```

```

    },
    "cumulativeRiskLevel": "H",
    "contributinggirls": [
      "H",
      "M"
    ],
    "controls": [
    ],
    "links": [
    ],
    "businessValue": "VH",
    "tags": {
      "attackpath": "destination"
    }
  },
  "PATHS": [
    [
      {
        "id": "5f7ec5302af19b000193880f",
        "name": "SDN-enabled RTU",
        "category": "Business
Service/Infrastructure Service",
        "gdpr": false,
        "assetId": 2,
        "riskassessmentId": 1714,
        "businesspartnerId": 1,
        "risklevel": "H",
        "threats": [
          {
            "threat": "DEFAULT-1",
            "name": "DEFAULT",
            "level": "VH",
            "vulnerabilities": [
              {
                "vulnerability": "CVE-2019-
14931",
                "level": "VL",
                "impact": "VH",
                "risklevel": "H",
                "privacyfunctionalimpactscore": 0,
                "privacyimpactscopescore": 0,
                "privacydatatypes": [
                ],
                "privacydatatypescore": 0,
                "privacyscore": 0,
                "privavyriskscore": 0,
                "cvssscore": 6,
                "cvss": "H"
              }
            ],
            "result": "H"
          }
        ]
      }
    ]
  ],

```



```

    "cumulativeRiskLevel": "H",
    "contributinggirls": [
      "H"
    ],
    "controls": [
      {
        "type": "CONNECTED_TO",
        "assetId": 1
      }
    ],
    "businessValue": "VH",
    "tags": {
      "attackpath": "source"
    }
  },
  {
    "id": "5f7ec5302af19b000193880e",
    "name": "SDN-Switch",
    "type": "h",
    "category": "NetworkComponent/Switch",
    "gdpr": false,
    "assetId": 1,
    "riskassessmentId": 1714,
    "businesspartnerId": 1,
    "risklevel": "M",
    "threats": [
      {
        "threat": "DEFAULT-1",
        "name": "DEFAULT",
        "level": "VH",
        "vulnerabilities": [
          {
            "vulnerability": "CVE-2019-1890",
            "level": "L",
            "impact": "M",
            "risklevel": "M",
          }
        ]
      }
    ],
    "privacyfunctionalimpactscore": 0,
    "privacyimpactscorescore": 0,
    "privacydatatypes": [
      "L"
    ],
    "privacydatatypescore": 0,
    "privacyscore": 0,
    "privavyriskscore": 0,
    "cvssscore": 3,
    "cvss": "L"
  },
  {
    "result": "M"
  }
],
"cumulativeRiskLevel": "M",
"contributinggirls": [
  "M"
],

```

```

    "controls": [
      {
        "type": "CONNECTED_TO",
        "assetId": 3
      },
      {
        "type": "CONNECTED_TO",
        "assetId": 5
      }
    ],
    "businessValue": "VH",
    "tags": {
      "attackpath": "source"
    }
  },
  {
    "id": "5f7ec5302af19b0001938810",
    "name": "Programmable Logic Controller",
    "type": "h",
    "category": "Computer/Server",
    "gdpr": false,
    "assetId": 3,
    "riskassessmentId": 1714,
    "businesspartnerId": 1,
    "risklevel": "H",
    "threats": [
      {
        "threat": "DEFAULT-1",
        "name": "DEFAULT",
        "level": "VH",
        "vulnerabilities": [
          {
            "vulnerability": "CVE-2017-6031",
            "level": "VH",
            "impact": "M",
            "risklevel": "H",
            "description": "A Header Injection issue was discovered in Certec EDV GmbH atvise scada prior to Version 3.0. An \"improper neutralization of HTTP headers for scripting syntax\" issue has been identified, which may allow remote code execution."
          }
        ]
      }
    ],
    "privacyfunctionalimpactscore": 0,
    "privacyimpactscorescore": 0,
    "privacydatatypes": [
      "L"
    ],
    "privacydatatypescore": 0,
    "privacyscore": 0,
    "privavyriskscore": 0,
    "cvssscore": 6.8,
    "cvss": "H"
  },
  {
    "result": "M"
  }
],

```



```

        "vulnerability": "CVE-2013-2780",
        "level": "VH",
        "impact": "L",
        "risklevel": "M",

        "privacyfunctionalimpactscore": 0,
        "privacyimpactscorescore": 0,
        "privacydatatypes": [

        ],
        "privacydatatypescore": 0,
        "privacyscore": 0,
        "privavyriskscore": 0,
        "cvssscore": 7.8,
        "cvss": "H"
    },
    {
        "vulnerability": "CVE-2017-6033",
        "level": "VH",
        "impact": "M",
        "risklevel": "H",
        "description": "A DLL Hijacking
issue was discovered in Schneider Electric
Interactive Graphical SCADA System (IGSS)
Software, Version 12 and previous versions. The
software will execute a malicious file if it is named
the same as a legitimate file and placed in a
location that is earlier in the search path.",

        "privacyfunctionalimpactscore": 0,
        "privacyimpactscorescore": 0,
        "privacydatatypes": [

        ],
        "privacydatatypescore": 0,
        "privacyscore": 0,
        "privavyriskscore": 0,
        "cvssscore": 6.8,
        "cvss": "H"
    }
    ],
    "result": "H"
}
],
"cumulativeRiskLevel": "H",
"contributingirls": [
    "H",
    "M"
],
"controls": [

],
"links": [
    {
        "type": "USED_BY",
        "assetId": 4
    }
],
"businessValue": "VH",

```

```

        "tags": {
        }
    },
    {
        "id": "5f7ec5302af19b0001938811",
        "name": "Administrator",
        "category": "Organizational/Personnel",
        "gdpr": false,
        "assetId": 4,
        "riskassessmentId": 1714,
        "businesspartnerId": 1,
        "risklevel": "",
        "threats": [

        ],
        "cumulativeRiskLevel": "",
        "contributingirls": [

        ],
        "controls": [

        ],
        "links": [

        ],
        "businessValue": "H",
        "tags": {
            "attackpath": "destination"
        }
    }
]
}

```

Figure 85: Example of a preliminary version of SRAF.json

The following table is an example of the .json file it can be retrieved from the AIDB corresponding in the previous figure:

```

[
{
    "invExternalId": "SDN_CONTROLLER_UCO",
    "invExternalName": "SDN_CONTROLLER_UCO",
    "invClass": "ELEMENT",
    "invNetLevel": "SDN_CONTROLLER_LEVEL",
    "invAssetType": "SDN Controller",
    "invState": "CONN_E",
    "invFather": "10.0.0.3",
    "invAttributeValues": [
    {
        "attribute": "sdnDriver",
        "value": "Northbound"
    },
    {
        "attribute": "sdnEndpoint",
        "value": "sdnEndpoint1"
    }
    ]
}
]

```

```
,
"relationships":null
},
{
  "invExternalId":"0000000000000001",
  "invExternalName":"0000000000000001",
  "invClass":"ELEMENT",
  "invNetLevel":"SDN_SWITCH_LEVEL",
  "invAssetType":"SDN_SWITCH",
  "invState":"CONN_E",
  "invFather":null,
  "invAttributeValues":[
    {
      "attribute":"InvDescription",
      "value":null
    },
    {
      "attribute":"sdnManufacturer",
      "value":"Nicira, Inc."
    },
    {
      "attribute":"sdnSwDesc",
      "value":"2.3.90"
    }
  ],
  "relationships":[
    {
      "relationshipId":"1",
      "externalIdA":"0000000000000001",
      "externalIdB":"0000000000000002",
      "enable":true,
      "weight":1.0,
      "capacity":1.0
    },
    {
      "relationshipId":"2",
      "externalIdA":"0000000000000002",
      "externalIdB":"0000000000000001",
      "enable":true,
      "weight":1.0,
      "capacity":1.0
    },
    {
      "relationshipId":"3",
      "externalIdA":"0000000000000001",
      "externalIdB":"0000000000000003",
      "enable":true,
      "weight":1.0,
      "capacity":1.0
    },
    {
      "relationshipId":"4",
      "externalIdA":"0000000000000003",
      "externalIdB":"0000000000000001",
      "enable":true,
      "weight":1.0,
      "capacity":1.0
    }
  ],
  {
    "invExternalId":"0000000000000002",
    "invExternalName":"0000000000000002",
    "invClass":"ELEMENT",
    "invNetLevel":"SDN_SWITCH_LEVEL",
    "invAssetType":"SDN_SWITCH",
    "invState":"CONN_E",
```

```
"invFather":null,
"invAttributeValues":[
  {
    "attribute":"InvDescription",
    "value":null
  },
  {
    "attribute":"sdnManufacturer",
    "value":"Nicira, Inc."
  },
  {
    "attribute":"sdnSwDesc",
    "value":"2.3.90"
  }
],
"relationships":[
  {
    "relationshipId":"1",
    "externalIdA":"0000000000000001",
    "externalIdB":"0000000000000002",
    "enable":true,
    "weight":1.0,
    "capacity":1.0
  },
  {
    "relationshipId":"2",
    "externalIdA":"0000000000000002",
    "externalIdB":"0000000000000001",
    "enable":true,
    "weight":1.0,
    "capacity":1.0
  },
  {
    "relationshipId":"5",
    "externalIdA":"0000000000000002",
    "externalIdB":"0000000000000003",
    "enable":true,
    "weight":1.0,
    "capacity":1.0
  },
  {
    "relationshipId":"6",
    "externalIdA":"0000000000000003",
    "externalIdB":"0000000000000002",
    "enable":true,
    "weight":1.0,
    "capacity":1.0
  }
]
},
{
  "invExternalId":"0000000000000003",
  "invExternalName":"0000000000000003",
  "invClass":"ELEMENT",
  "invNetLevel":"SDN_SWITCH_LEVEL",
  "invAssetType":"SDN_SWITCH",
  "invState":"CONN_E",
  "invFather":null,
  "invAttributeValues":[
    {
      "attribute":"InvDescription",
      "value":null
    },
    {
      "attribute":"sdnManufacturer",
      "value":"Nicira, Inc."
    }
  ],
```

```
{
  "attribute": "sdnSwDesc",
  "value": "2.3.90"
},
"relationships": [
  {
    "relationshipId": "4",
    "externalIdA": "0000000000000003",
    "externalIdB": "0000000000000001",
    "enable": true,
    "weight": 1.0,
    "capacity": 1.0
  },
  {
    "relationshipId": "3",
    "externalIdA": "0000000000000001",
    "externalIdB": "0000000000000003",
    "enable": true,
    "weight": 1.0,
    "capacity": 1.0
  },
  {
    "relationshipId": "6",
    "externalIdA": "0000000000000003",
    "externalIdB": "0000000000000002",
    "enable": true,
    "weight": 1.0,
    "capacity": 1.0
  },
  {
    "relationshipId": "5",
    "externalIdA": "0000000000000002",
    "externalIdB": "0000000000000003",
    "enable": true,
    "weight": 1.0,
    "capacity": 1.0
  }
]
{
  "invExternalId": "10.0.0.1",
  "invExternalName": "PDC_1",
  "invClass": "ELEMENT",
  "invNetLevel": "SDN_HOST_LEVEL",
  "invAssetType": "PDC",
  "invState": "CONN_E",
  "invFather": null,
  "invAttributeValues": [
    {
      "attribute": "InvDescription",
      "value": null
    },
    {
      "attribute": "sdnIP",
      "value": "10.0.0.1"
    }
  ],
  "relationships": [
    {
      "relationshipId": "7",
      "externalIdA": "10.0.0.1",
      "externalIdB": "0000000000000001",
      "enable": true,
      "weight": 1.0,
      "capacity": 1.0
    }
  ],
}
```

```
{
  "relationshipId": "8",
  "externalIdA": "0000000000000001",
  "externalIdB": "10.0.0.1",
  "enable": true,
  "weight": 1.0,
  "capacity": 1.0
}
]
{
  "invExternalId": "10.0.0.2",
  "invExternalName": "PDC_2",
  "invClass": "ELEMENT",
  "invNetLevel": "SDN_HOST_LEVEL",
  "invAssetType": "PDC",
  "invState": "CONN_E",
  "invFather": null,
  "invAttributeValues": [
    {
      "attribute": "InvDescription",
      "value": null
    },
    {
      "attribute": "sdnIP",
      "value": "10.0.0.2"
    }
  ],
  "relationships": [
    {
      "relationshipId": "9",
      "externalIdA": "10.0.0.2",
      "externalIdB": "0000000000000001",
      "enable": true,
      "weight": 1.0,
      "capacity": 1.0
    }
  ],
  {
    "relationshipId": "10",
    "externalIdA": "0000000000000001",
    "externalIdB": "10.0.0.2",
    "enable": true,
    "weight": 1.0,
    "capacity": 1.0
  }
],
{
  "invExternalId": "10.0.0.11",
  "invExternalName": "PMU_1",
  "invClass": "ELEMENT",
  "invNetLevel": "SDN_HOST_LEVEL",
  "invAssetType": "PMU",
  "invState": "CONN_E",
  "invFather": "10.0.0.1",
  "invAttributeValues": [
    {
      "attribute": "InvDescription",
      "value": null
    },
    {
      "attribute": "sdnIP",
      "value": "10.0.0.11"
    }
  ],
  "relationships": [
    {

```

```

"relationshipId": "11",
"externalIdA": "10.0.0.11",
"externalIdB": "0000000000000003",
"enable": true,
"weight": 1.0,
    "capacity": 1.0
},
{
  "relationshipId": "12",
  "externalIdA": "0000000000000003",
  "externalIdB": "10.0.0.11",
  "enable": true,
  "weight": 1.0,
    "capacity": 1.0
}
},
{
  "invExternalId": "10.0.0.12",
  "invExternalName": "PMU_2",
  "invClass": "ELEMENT",
  "invNetLevel": "SDN_HOST_LEVEL",
  "invAssetType": "PMU",
  "invState": "CONN_E",
  "invFather": "10.0.0.1",
  "invAttributeValues": [
    {
      "attribute": "InvDescription",
      "value": null
    },
    {
      "attribute": "sdnIP",
      "value": "10.0.0.12"
    }
  ],
  "relationships": [
    {
      "relationshipId": "13",
      "externalIdA": "10.0.0.12",
      "externalIdB": "0000000000000003",
      "enable": true,
      "weight": 1.0,
        "capacity": 1.0
    },
    {
      "relationshipId": "14",
      "externalIdA": "0000000000000003",
      "externalIdB": "10.0.0.12",
      "enable": true,
      "weight": 1.0,
        "capacity": 1.0
    }
  ],
  "invExternalId": "10.0.0.13",
  "invExternalName": "PMU_3",
  "invClass": "ELEMENT",
  "invNetLevel": "SDN_HOST_LEVEL",
  "invAssetType": "PMU",
  "invState": "CONN_E",
  "invFather": "10.0.0.2",
  "invAttributeValues": [
    {
      "attribute": "InvDescription",
      "value": null
    }
  ],

```

```

{
  "attribute": "sdnIP",
  "value": "10.0.0.13"
},
{
  "relationships": [
    {
      "relationshipId": "15",
      "externalIdA": "10.0.0.13",
      "externalIdB": "0000000000000002",
      "enable": true,
      "weight": 1.0,
        "capacity": 1.0
    },
    {
      "relationshipId": "16",
      "externalIdA": "0000000000000002",
      "externalIdB": "10.0.0.13",
      "enable": true,
      "weight": 1.0,
        "capacity": 1.0
    }
  ],
  "invExternalId": "10.0.0.14",
  "invExternalName": "PMU_4",
  "invClass": "ELEMENT",
  "invNetLevel": "SDN_HOST_LEVEL",
  "invAssetType": "PMU",
  "invState": "CONN_E",
  "invFather": "10.0.0.2",
  "invAttributeValues": [
    {
      "attribute": "InvDescription",
      "value": null
    },
    {
      "attribute": "sdnIP",
      "value": "10.0.0.14"
    }
  ],
  "relationships": [
    {
      "relationshipId": "17",
      "externalIdA": "10.0.0.14",
      "externalIdB": "0000000000000002",
      "enable": true,
      "weight": 1.0,
        "capacity": 1.0
    },
    {
      "relationshipId": "18",
      "externalIdA": "0000000000000002",
      "externalIdB": "10.0.0.14",
      "enable": true,
      "weight": 1.0,
        "capacity": 1.0
    }
  ],
  "invExternalId": "10.0.0.3",
  "invExternalName": "SDN_Controller",
  "invClass": "ELEMENT",
  "invNetLevel": "SDN_HOST_LEVEL",
  "invAssetType": "SDN_Controller",

```



```
"invState": "CONN_E",
"invFather": null,
"invAttributeValues": [
  {
    "attribute": "InvDescription",
    "value": null
  },
  {
    "attribute": "sdnIP",
    "value": "10.0.0.3"
  }
],
"relationships": [
  {
    "relationshipId": "19",
    "externalIdA": "10.0.0.3",
    "externalIdB": "0000000000000001",
    "enable": true,
    "weight": 1.0,
    "capacity": 1.0
  },
  {
    "relationshipId": "20",
    "externalIdA": "0000000000000001",
    "externalIdB": "10.0.0.3",
    "enable": true,
    "weight": 1.0,
    "capacity": 1.0
  }
]
}
```

13.2 Grid model example A

Figure 86 contains the single line schema of a grid example. In this, 4 buses, 4 Loads, 2 diesel generators, a PV generator, and several lines have been depicted. Below, their properties and possible values have been included.

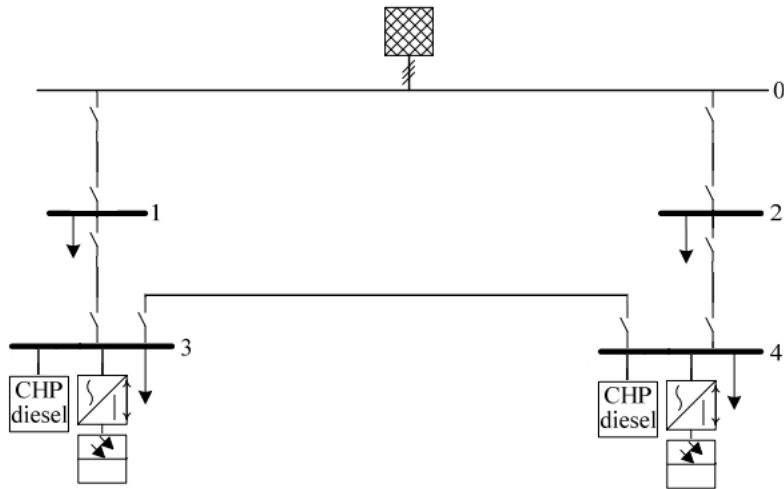


Figure 86: AIDB Grid model example (single line)

name	vn_kv	type	zone	in_service
Bus 0	20.0	b	SimpleMV	True
Bus 1	20.0	b	SimpleMV	True
Bus 2	20.0	b	SimpleMV	True
Bus 3	20.0	b	SimpleMV	True
Bus 4	20.0	b	SimpleMV	True

Figure 87: AIDB - Grid Bus model

name	std_type	hv_bus	lv_bus	sn_mva	vn_hv_kv	vn_lv_kv	vk_perce	vkr_perce	pfe_kw	io_perce	shift_degr	tap_side	tap_neutr	tap_min	tap_max	tap_step_	tap_step_	tap_pos	tap_phas	parallel	df	in_service
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

Figure 88: AIDB - Grid Trafo data model

Notice, the example does not have transformers.

Name	std_type	from_bus	to_bus	length_km	r_ohm_per_km	x_ohm_per_km	c_nf_per_km	g_us_per_km	max_i_ka	df	parallel	type	in_service
Line 0-1	CABLE_CIGRE_MV	0	1	3.0	0.501	0.716	151.1749	0.0	0.145	1.0	1	cs	True
Line 0-2	CABLE_CIGRE_MV	0	2	3.0	0.501	0.716	151.1749	0.0	0.145	1.0	1	cs	True
Line 1-3	CABLE_CIGRE_MV	1	3	2.0	0.501	0.716	151.1749	0.0	0.145	1.0	1	cs	True
Line 2-4	CABLE_CIGRE_MV	2	4	2.0	0.501	0.716	151.1749	0.0	0.145	1.0	1	cs	True





Line 3-4	CABLE_CIGRE_MV	3	4	4.0	0.501	0.716	151.1749	0.0	0.145	1.0	1	cs	True
-----------------	----------------	---	---	-----	-------	-------	----------	-----	-------	-----	---	----	------

Figure 89: AIDB - Grid Line data model

Name	bus	p_mw	q_mvar	const_z_percent	const_i_percent	sn_mva	scaling	in_service	type
Load R1	1	0.4999962	0.1253108	0.0	0.0	0.51546	1.0	True	None
Load R2	2	0.4999962	0.1253108	0.0	0.0	0.51546	1.0	True	None
Load R3	3	0.4999962	0.1253108	0.0	0.0	0.51546	1.0	True	None
Load R4	4	0.4999962	0.1253108	0.0	0.0	0.51546	1.0	True	None

Figure 90: AIDB - Grid Load data model

Name	bus	vm_pu	va_degree	in_service	s_sc_max_mva	s_sc_min_mva	rx_min	rx_max
None	0	1.03	0.0	True	5000.0	5000.0	0.1	0.1

Figure 91: AIDB - Grid external grid data model

name	bus	p_mw	vm_pu	sn_mva	min_q_mvar	max_q_mvar	scaling	slack	in_service	type	min_p_mw	max_p_mw
Diesel generator 1	3	0.5	1.0	nan	-0.5	0.5	1.0	False	True	sync	0.0	0.5
Diesel generator 2	4	0.5	1.0	nan	-0.5	0.5	1.0	False	True	sync	0.0	0.5

Figure 92: AIDB - Grid Generator data model

name	bus	p_mw	q_mvar	sn_mva	scaling	in_service	type	current_source	min_p_mw	max_p_mw	min_q_mvar	max_q_mvar	controllable
PV 3	3	1.0	1.0	nan	1.0	True	PV1	True	0.0	5.0	-5.0	5.0	True
PV 4	4	1.0	1.0	5.0	1.0	True	PV2	True	0.0	5.0	-5.0	5.0	True

Figure 93: AIDB - Grid Sgenerator data model

13.3 AIDB unit test results

Due to the high detail level and the large amount of information, inputs and test results have been included below:

AIDB_001 SDN Asset creation – Input data	AIDB_001 SDN Asset creation – Input data
<pre>[{ "invExternalId": "SDN_CONTROLLER_UC0", "invExternalName": "SDN_CONTROLLER_UC0", "invClass": "ELEMENT", "invNetLevel": "CONTROLLER", "invAssetType": "SDN Controller", "invState": null, "invFather": null, "invAttributeValues": [{ "attribute": "sdnDriver", "value": "Northbound" }, { "attribute": "sdnEndpoint", "value": "sdnEndpoint" }], "relationships": null }, { "invExternalId": "0000000000000001", "invExternalName": "0000000000000001", "invClass": "ELEMENT", "invNetLevel": "SWITCH", "invAssetType": "Switch", "invState": null, "invFather": null, "invAttributeValues": [{ "attribute": "InvDescription", "value": null }, { "attribute": "sdnManufacturer", "value": "Nicira, Inc." }, { "attribute": "sdnSwDesc", "value": "2.3.90" }], "relationships": [{ "externalIdA": "0000000000000001", "externalIdB": "0000000000000002", "enable": true, "weight": 1.0, "bandwidth": 2.0, "maxLatency": 1.0 }, { "externalIdA": "0000000000000001", "externalIdB": "0000000000000003", "enable": true, "weight": 1.0, "bandwidth": 2.0, "maxLatency": 1.0 }] }]</pre>	<pre> "maxLatency": 1.0 }, { "externalIdA": "0000000000000001", "externalIdB": "0000000000000004", "enable": true, "weight": 1.0, "bandwidth": 2.0, "maxLatency": 1.0 }, { "externalIdA": "0000000000000002", "externalIdB": "0000000000000001", "enable": true, "weight": 1.0, "bandwidth": 2.0, "maxLatency": 1.0 }, { "externalIdA": "0000000000000003", "externalIdB": "0000000000000001", "enable": true, "weight": 1.0, "bandwidth": 2.0, "maxLatency": 1.0 }, { "externalIdA": "0000000000000004", "externalIdB": "0000000000000001", "enable": true, "weight": 1.0, "bandwidth": 2.0, "maxLatency": 1.0 }], { "invExternalId": "0000000000000002", "invExternalName": "0000000000000002", "invClass": "ELEMENT", "invNetLevel": "SWITCH", "invAssetType": "Switch", "invState": null, "invFather": null, "invAttributeValues": [{ "attribute": "InvDescription", "value": null }, { "attribute": "sdnManufacturer", "value": "Nicira, Inc." }, { "attribute": "sdnSwDesc", "value": "2.3.90" }] },]</pre>

AIDB_001 SDN Asset creation – Input data

```
"relationships": [
  {
    "externalIdA": "0000000000000002",
    "externalIdB": "0000000000000001",
    "enable": true,
    "weight": 1.0,
    "bandwidth": 2.0,
    "maxLatency": 1.0
  },
  {
    "externalIdA": "0000000000000002",
    "externalIdB": "0000000000000004",
    "enable": true,
    "weight": 1.0,
    "bandwidth": 2.0,
    "maxLatency": 1.0
  },
  {
    "externalIdA": "0000000000000002",
    "externalIdB": "0000000000000005",
    "enable": true,
    "weight": 1.0,
    "bandwidth": 2.0,
    "maxLatency": 1.0
  },
  {
    "externalIdA": "0000000000000001",
    "externalIdB": "0000000000000002",
    "enable": true,
    "weight": 1.0,
    "bandwidth": 2.0,
    "maxLatency": 1.0
  },
  {
    "externalIdA": "0000000000000004",
    "externalIdB": "0000000000000002",
    "enable": true,
    "weight": 1.0,
    "bandwidth": 2.0,
    "maxLatency": 1.0
  },
  {
    "externalIdA": "0000000000000005",
    "externalIdB": "0000000000000002",
    "enable": true,
    "weight": 1.0,
    "bandwidth": 2.0,
    "maxLatency": 1.0
  }
],
{
  "invExternalId": "0000000000000003",
  "invExternalName": "0000000000000003",
  "invClass": "ELEMENT",
  "invNetLevel": "SWITCH",
  "invAssetType": "Switch",
  "invState": null,
  "invFather": null,
  "invAttributeValues": [
    {
      "attribute": "InvDescription",
      "value": null
    },
    {
      "attribute": "sdnManufacturer",
      "value": "Nicira, Inc."
    }
  ]
}
```

AIDB_001 SDN Asset creation – Input data

```
{
  "attribute": "sdnSwDesc",
  "value": "2.3.90"
},
"relationships": [
  {
    "externalIdA": "0000000000000003",
    "externalIdB": "0000000000000001",
    "enable": true,
    "weight": 1.0,
    "bandwidth": 2.0,
    "maxLatency": 1.0
  },
  {
    "externalIdA": "0000000000000003",
    "externalIdB": "0000000000000004",
    "enable": true,
    "weight": 1.0,
    "bandwidth": 2.0,
    "maxLatency": 1.0
  },
  {
    "externalIdA": "0000000000000003",
    "externalIdB": "10.0.0.1",
    "enable": true,
    "weight": 1.0,
    "bandwidth": 2.0,
    "maxLatency": 1.0
  },
  {
    "externalIdA": "0000000000000003",
    "externalIdB": "10.0.0.3",
    "enable": true,
    "weight": 1.0,
    "bandwidth": 2.0,
    "maxLatency": 1.0
  },
  {
    "externalIdA": "0000000000000001",
    "externalIdB": "0000000000000003",
    "enable": true,
    "weight": 1.0,
    "bandwidth": 2.0,
    "maxLatency": 1.0
  },
  {
    "externalIdA": "0000000000000004",
    "externalIdB": "0000000000000003",
    "enable": true,
    "weight": 1.0,
    "bandwidth": 2.0,
    "maxLatency": 1.0
  },
  {
    "externalIdA": "10.0.0.1",
    "externalIdB": "0000000000000003",
    "enable": true,
    "weight": 1.0,
    "bandwidth": 2.0,
    "maxLatency": 1.0
  },
  {
    "externalIdA": "10.0.0.3",
    "externalIdB": "0000000000000003",
    "enable": true,
    "weight": 1.0,
    "bandwidth": 2.0,
    "maxLatency": 1.0
  }
]
```



AIDB_001 SDN Asset creation – Input data

```
    "maxLatency": 1.0
  }
],
{
  "invExternalId": "0000000000000004",
  "invExternalName": "0000000000000004",
  "invClass": "ELEMENT",
  "invNetLevel": "SWITCH",
  "invAssetType": "Switch",
  "invState": null,
  "invFather": null,
  "invAttributeValues": [
    {
      "attribute": "InvDescription",
      "value": null
    },
    {
      "attribute": "sdnManufacturer",
      "value": "Nicira, Inc."
    },
    {
      "attribute": "sdnSwDesc",
      "value": "2.3.90"
    }
  ],
  "relationships": [
    {
      "externalIdA": "0000000000000004",
      "externalIdB": "0000000000000001",
      "enable": true,
      "weight": 1.0,
      "bandwidth": 2.0,
      "maxLatency": 1.0
    },
    {
      "externalIdA": "0000000000000004",
      "externalIdB": "0000000000000002",
      "enable": true,
      "weight": 1.0,
      "bandwidth": 2.0,
      "maxLatency": 1.0
    },
    {
      "externalIdA": "0000000000000004",
      "externalIdB": "0000000000000003",
      "enable": true,
      "weight": 1.0,
      "bandwidth": 2.0,
      "maxLatency": 1.0
    },
    {
      "externalIdA": "0000000000000004",
      "externalIdB": "0000000000000005",
      "enable": true,
      "weight": 1.0,
      "bandwidth": 2.0,
      "maxLatency": 1.0
    },
    {
      "externalIdA": "0000000000000004",
      "externalIdB": "10.0.0.2",
      "enable": true,
      "weight": 1.0,
      "bandwidth": 2.0,
      "maxLatency": 1.0
    }
  ],
}
```

AIDB_001 SDN Asset creation – Input data

```
  "externalIdA": "0000000000000001",
  "externalIdB": "0000000000000004",
  "enable": true,
  "weight": 1.0,
  "bandwidth": 2.0,
  "maxLatency": 1.0
},
{
  "externalIdA": "0000000000000002",
  "externalIdB": "0000000000000004",
  "enable": true,
  "weight": 1.0,
  "bandwidth": 2.0,
  "maxLatency": 1.0
},
{
  "externalIdA": "0000000000000003",
  "externalIdB": "0000000000000004",
  "enable": true,
  "weight": 1.0,
  "bandwidth": 2.0,
  "maxLatency": 1.0
},
{
  "externalIdA": "0000000000000005",
  "externalIdB": "0000000000000004",
  "enable": true,
  "weight": 1.0,
  "bandwidth": 2.0,
  "maxLatency": 1.0
},
{
  "externalIdA": "10.0.0.2",
  "externalIdB": "0000000000000004",
  "enable": true,
  "weight": 1.0,
  "bandwidth": 2.0,
  "maxLatency": 1.0
}
],
{
  "invExternalId": "0000000000000005",
  "invExternalName": "0000000000000005",
  "invClass": "ELEMENT",
  "invNetLevel": "SWITCH",
  "invAssetType": "Switch",
  "invState": null,
  "invFather": null,
  "invAttributeValues": [
    {
      "attribute": "InvDescription",
      "value": null
    },
    {
      "attribute": "sdnManufacturer",
      "value": "Nicira, Inc."
    },
    {
      "attribute": "sdnSwDesc",
      "value": "2.3.90"
    }
  ],
  "relationships": [
    {
      "externalIdA": "0000000000000005",
      "externalIdB": "0000000000000002",
      "enable": true,
```

AIDB_001 SDN Asset creation – Input data

```

    "weight": 1.0,
    "bandwidth": 2.0,
    "maxLatency": 1.0
  },
  {
    "externalIdA": "0000000000000005",
    "externalIdB": "0000000000000004",
    "enable": true,
    "weight": 1.0,
    "bandwidth": 2.0,
    "maxLatency": 1.0
  },
  {
    "externalIdA": "0000000000000002",
    "externalIdB": "0000000000000005",
    "enable": true,
    "weight": 1.0,
    "bandwidth": 2.0,
    "maxLatency": 1.0
  },
  {
    "externalIdA": "0000000000000004",
    "externalIdB": "0000000000000005",
    "enable": true,
    "weight": 1.0,
    "bandwidth": 2.0,
    "maxLatency": 1.0
  }
],
{
  "invExternalId": "10.0.0.1",
  "invExternalName": "10.0.0.1",
  "invClass": "ELEMENT",
  "invNetLevel": "HOST",
  "invAssetType": "Host",
  "invState": null,
  "invFather": null,
  "invAttributeValues": [
    {
      "attribute": "InvDescription",
      "value": null
    },
    {
      "attribute": "sdnIP",
      "value": null
    }
  ],
  "relationships": [
    {
      "externalIdA": "10.0.0.1",
      "externalIdB": "0000000000000003",
      "enable": true,
      "weight": 1.0,
      "bandwidth": 2.0,
      "maxLatency": 1.0
    },
    {
      "externalIdA": "0000000000000003",
      "externalIdB": "10.0.0.1",
      "enable": true,
      "weight": 1.0,
      "bandwidth": 2.0,
      "maxLatency": 1.0
    }
  ]
},
{

```

AIDB_001 SDN Asset creation – Input data

```

  "invExternalId": "10.0.0.2",
  "invExternalName": "10.0.0.2",
  "invClass": "ELEMENT",
  "invNetLevel": "HOST",
  "invAssetType": "Host",
  "invState": null,
  "invFather": null,
  "invAttributeValues": [
    {
      "attribute": "InvDescription",
      "value": null
    },
    {
      "attribute": "sdnIP",
      "value": null
    }
  ],
  "relationships": [
    {
      "externalIdA": "10.0.0.2",
      "externalIdB": "0000000000000004",
      "enable": true,
      "weight": 1.0,
      "bandwidth": 2.0,
      "maxLatency": 1.0
    },
    {
      "externalIdA": "0000000000000004",
      "externalIdB": "10.0.0.2",
      "enable": true,
      "weight": 1.0,
      "bandwidth": 2.0,
      "maxLatency": 1.0
    }
  ]
},
{
  "invExternalId": "10.0.0.3",
  "invExternalName": "10.0.0.3",
  "invClass": "ELEMENT",
  "invNetLevel": "HOST",
  "invAssetType": "Host",
  "invState": null,
  "invFather": null,
  "invAttributeValues": [
    {
      "attribute": "InvDescription",
      "value": null
    },
    {
      "attribute": "sdnIP",
      "value": null
    }
  ],
  "relationships": [
    {
      "externalIdA": "10.0.0.3",
      "externalIdB": "0000000000000003",
      "enable": true,
      "weight": 1.0,
      "bandwidth": 2.0,
      "maxLatency": 1.0
    },
    {
      "externalIdA": "0000000000000003",
      "externalIdB": "10.0.0.3",
      "enable": true,
      "weight": 1.0,

```

AIDB_001 SDN Asset creation – Input data

```

    "bandwidth": 2.0,
    "maxLatency": 1.0
  }
}
]

```

AIDB_001 SDN Asset creation – Result

```

[
  {
    "invExternalId": "SDN_CONTROLLER_UC0",
    "invExternalName": "SDN_CONTROLLER_UC0",
    "invClass": "ELEMENT",
    "invNetLevel": "CONTROLLER",
    "invAssetType": "SDN Controller",
    "invState": null,
    "invFather": null,
    "invAttributeValues": [
      {
        "attribute": "sdnDriver",
        "value": "Northbound"
      },
      {
        "attribute": "sdnEndpoint",
        "value": "sdnEndpoint"
      }
    ],
    "relationships": null
  },
  {
    "invExternalId": "0000000000000001",
    "invExternalName": "0000000000000001",
    "invClass": "ELEMENT",
    "invNetLevel": "SWITCH",
    "invAssetType": "Switch",
    "invState": null,
    "invFather": null,
    "invAttributeValues": [
      {
        "attribute": "InvDescription",
        "value": null
      },
      {
        "attribute": "sdnManufacturer",
        "value": "Nicira, Inc."
      },
      {
        "attribute": "sdnSwDesc",
        "value": "2.3.90"
      }
    ],
    "relationships": [
      {
        "relationshipId": "0000000000000001_0000000000000002",
        "externalIdA": "0000000000000001",
        "externalIdB": "0000000000000002",
        "enable": true,
        "weight": 1.0,
        "bandwidth": 2.0,
        "maxLatency": 1.0
      },
      {
        "relationshipId": "0000000000000001_0000000000000003",
        "externalIdA": "0000000000000001",
        "externalIdB": "0000000000000003",
        "enable": true,
        "weight": 1.0,
        "bandwidth": 2.0,
        "maxLatency": 1.0
      },
      {
        "relationshipId": "0000000000000001_0000000000000004",
        "externalIdA": "0000000000000001",
        "externalIdB": "0000000000000004",
        "enable": true,
        "weight": 1.0,
        "bandwidth": 2.0,
        "maxLatency": 1.0
      }
    ]
  }
]

```

AIDB_001 SDN Asset creation – Result

```

"relationshipId": "0000000000000001_0000000000000003",
  "externalIdA": "0000000000000001",
  "externalIdB": "0000000000000003",
  "enable": true,
  "weight": 1.0,
  "bandwidth": 2.0,
  "maxLatency": 1.0
},
{
  "relationshipId": "0000000000000001_0000000000000004",
  "externalIdA": "0000000000000001",
  "externalIdB": "0000000000000004",
  "enable": true,
  "weight": 1.0,
  "bandwidth": 2.0,
  "maxLatency": 1.0
},
{
  "relationshipId": "0000000000000002_0000000000000001",
  "externalIdA": "0000000000000002",
  "externalIdB": "0000000000000001",
  "enable": true,
  "weight": 1.0,
  "bandwidth": 2.0,
  "maxLatency": 1.0
},
{
  "relationshipId": "0000000000000003_0000000000000001",
  "externalIdA": "0000000000000003",
  "externalIdB": "0000000000000001",
  "enable": true,
  "weight": 1.0,
  "bandwidth": 2.0,
  "maxLatency": 1.0
},
{
  "relationshipId": "0000000000000004_0000000000000001",
  "externalIdA": "0000000000000004",
  "externalIdB": "0000000000000001",
  "enable": true,
  "weight": 1.0,
  "bandwidth": 2.0,
  "maxLatency": 1.0
}
],
{
  "invExternalId": "0000000000000002",
  "invExternalName": "0000000000000002",
  "invClass": "ELEMENT",
  "invNetLevel": "SWITCH",
  "invAssetType": "Switch",
  "invState": null,
  "invFather": null,
  "invAttributeValues": [
    {
      "attribute": "InvDescription",
      "value": null
    }
  ]
}
]

```

AIDB_001 SDN Asset creation – Result

```

    },
    {
      "attribute": "sdnManufacturer",
      "value": "Nicira, Inc."
    },
    {
      "attribute": "sdnSwDesc",
      "value": "2.3.90"
    }
  ],
  "relationships": [
    {
      "relationshipId": "000000000000002_0000000000000001",
      "externalIdA": "000000000000002",
      "externalIdB": "000000000000001",
      "enable": true,
      "weight": 1.0,
      "bandwidth": 2.0,
      "maxLatency": 1.0
    },
    {
      "relationshipId": "000000000000002_0000000000000004",
      "externalIdA": "000000000000002",
      "externalIdB": "000000000000004",
      "enable": true,
      "weight": 1.0,
      "bandwidth": 2.0,
      "maxLatency": 1.0
    },
    {
      "relationshipId": "000000000000002_0000000000000005",
      "externalIdA": "000000000000002",
      "externalIdB": "000000000000005",
      "enable": true,
      "weight": 1.0,
      "bandwidth": 2.0,
      "maxLatency": 1.0
    },
    {
      "relationshipId": "000000000000001_0000000000000002",
      "externalIdA": "000000000000001",
      "externalIdB": "000000000000002",
      "enable": true,
      "weight": 1.0,
      "bandwidth": 2.0,
      "maxLatency": 1.0
    },
    {
      "relationshipId": "000000000000004_0000000000000002",
      "externalIdA": "000000000000004",
      "externalIdB": "000000000000002",
      "enable": true,
      "weight": 1.0,
      "bandwidth": 2.0,
      "maxLatency": 1.0
    }
  ]
}

```

AIDB_001 SDN Asset creation – Result

```

"relationshipId": "000000000000005_0000000000000002",
  "externalIdA": "000000000000005",
  "externalIdB": "000000000000002",
  "enable": true,
  "weight": 1.0,
  "bandwidth": 2.0,
  "maxLatency": 1.0
},
{
  "invExternalId": "000000000000003",
  "invExternalName": "000000000000003",
  "invClass": "ELEMENT",
  "invNetLevel": "SWITCH",
  "invAssetType": "Switch",
  "invState": null,
  "invFather": null,
  "invAttributeValues": [
    {
      "attribute": "InvDescription",
      "value": null
    },
    {
      "attribute": "sdnManufacturer",
      "value": "Nicira, Inc."
    },
    {
      "attribute": "sdnSwDesc",
      "value": "2.3.90"
    }
  ],
  "relationships": [
    {
      "relationshipId": "000000000000003_0000000000000001",
      "externalIdA": "000000000000003",
      "externalIdB": "000000000000001",
      "enable": true,
      "weight": 1.0,
      "bandwidth": 2.0,
      "maxLatency": 1.0
    },
    {
      "relationshipId": "000000000000003_0000000000000004",
      "externalIdA": "000000000000003",
      "externalIdB": "000000000000004",
      "enable": true,
      "weight": 1.0,
      "bandwidth": 2.0,
      "maxLatency": 1.0
    },
    {
      "relationshipId": "000000000000003_10.0.0.1",
      "externalIdA": "000000000000003",
      "externalIdB": "10.0.0.1",
      "enable": true,
      "weight": 1.0,
      "bandwidth": 2.0,
      "maxLatency": 1.0
    }
  ]
}

```



AIDB_001 SDN Asset creation – Result

```
"relationshipId": "000000000000003_10.0.0.3",
  "externalIdA": "000000000000003",
  "externalIdB": "10.0.0.3",
  "enable": true,
  "weight": 1.0,
  "bandwidth": 2.0,
  "maxLatency": 1.0
},
{
  "relationshipId": "000000000000001_0000000000
00003",
  "externalIdA": "000000000000001",
  "externalIdB": "000000000000003",
  "enable": true,
  "weight": 1.0,
  "bandwidth": 2.0,
  "maxLatency": 1.0
},
{
  "relationshipId": "000000000000004_0000000000
00003",
  "externalIdA": "000000000000004",
  "externalIdB": "000000000000003",
  "enable": true,
  "weight": 1.0,
  "bandwidth": 2.0,
  "maxLatency": 1.0
},
{
  "relationshipId": "10.0.0.1_000000000000003",
  "externalIdA": "10.0.0.1",
  "externalIdB": "000000000000003",
  "enable": true,
  "weight": 1.0,
  "bandwidth": 2.0,
  "maxLatency": 1.0
},
{
  "relationshipId": "10.0.0.3_000000000000003",
  "externalIdA": "10.0.0.3",
  "externalIdB": "000000000000003",
  "enable": true,
  "weight": 1.0,
  "bandwidth": 2.0,
  "maxLatency": 1.0
}
}
{
  "invExternalId": "000000000000004",
  "invExternalName": "000000000000004",
  "invClass": "ELEMENT",
  "invNetLevel": "SWITCH",
  "invAssetType": "Switch",
  "invState": null,
  "invFather": null,
  "invAttributeValues": [
    {
      "attribute": "InvDescription",
      "value": null
    },
    {
      "attribute": "sdnManufacturer",
```

AIDB_001 SDN Asset creation – Result

```
"value": "Nicira, Inc."
},
{
  "attribute": "sdnSwDesc",
  "value": "2.3.90"
}
],
"relationships": [
{
  "relationshipId": "000000000000004_0000000000
00001",
  "externalIdA": "000000000000004",
  "externalIdB": "000000000000001",
  "enable": true,
  "weight": 1.0,
  "bandwidth": 2.0,
  "maxLatency": 1.0
},
{
  "relationshipId": "000000000000004_0000000000
00002",
  "externalIdA": "000000000000004",
  "externalIdB": "000000000000002",
  "enable": true,
  "weight": 1.0,
  "bandwidth": 2.0,
  "maxLatency": 1.0
},
{
  "relationshipId": "000000000000004_0000000000
00003",
  "externalIdA": "000000000000004",
  "externalIdB": "000000000000003",
  "enable": true,
  "weight": 1.0,
  "bandwidth": 2.0,
  "maxLatency": 1.0
},
{
  "relationshipId": "000000000000004_0000000000
00005",
  "externalIdA": "000000000000004",
  "externalIdB": "000000000000005",
  "enable": true,
  "weight": 1.0,
  "bandwidth": 2.0,
  "maxLatency": 1.0
},
{
  "relationshipId": "000000000000004_10.0.0.2",
  "externalIdA": "000000000000004",
  "externalIdB": "10.0.0.2",
  "enable": true,
  "weight": 1.0,
  "bandwidth": 2.0,
  "maxLatency": 1.0
},
{
  "relationshipId": "000000000000001_0000000000
00004",
  "externalIdA": "000000000000001",
  "externalIdB": "000000000000004",
```


AIDB_001 SDN Asset creation – Result

```

    "enable": true,
    "weight": 1.0,
    "bandwidth": 2.0,
    "maxLatency": 1.0
  },
  {
    "relationshipId": "00000000000002_0000000000
0004",
    "externalIdA": "000000000000002",
    "externalIdB": "000000000000004",
    "enable": true,
    "weight": 1.0,
    "bandwidth": 2.0,
    "maxLatency": 1.0
  },
  {
    "relationshipId": "00000000000003_0000000000
0004",
    "externalIdA": "000000000000003",
    "externalIdB": "000000000000004",
    "enable": true,
    "weight": 1.0,
    "bandwidth": 2.0,
    "maxLatency": 1.0
  },
  {
    "relationshipId": "00000000000005_0000000000
0004",
    "externalIdA": "000000000000005",
    "externalIdB": "000000000000004",
    "enable": true,
    "weight": 1.0,
    "bandwidth": 2.0,
    "maxLatency": 1.0
  },
  {
    "relationshipId": "10.0.0.2_000000000000004",
    "externalIdA": "10.0.0.2",
    "externalIdB": "000000000000004",
    "enable": true,
    "weight": 1.0,
    "bandwidth": 2.0,
    "maxLatency": 1.0
  }
],
{
  "invExternalId": "000000000000005",
  "invExternalName": "000000000000005",
  "invClass": "ELEMENT",
  "invNetLevel": "SWITCH",
  "invAssetType": "Switch",
  "invState": null,
  "invFather": null,
  "invAttributeValues": [
    {
      "attribute": "InvDescription",
      "value": null
    },
    {
      "attribute": "sdnManufacturer",
      "value": "Nicira, Inc."
    }
  ],
  {

```

AIDB_001 SDN Asset creation – Result

```

    "attribute": "sdnSwDesc",
    "value": "2.3.90"
  },
  {
    "relationships": [
      {
        "relationshipId": "00000000000005_0000000000
0002",
        "externalIdA": "000000000000005",
        "externalIdB": "000000000000002",
        "enable": true,
        "weight": 1.0,
        "bandwidth": 2.0,
        "maxLatency": 1.0
      },
      {
        "relationshipId": "00000000000005_0000000000
0004",
        "externalIdA": "000000000000005",
        "externalIdB": "000000000000004",
        "enable": true,
        "weight": 1.0,
        "bandwidth": 2.0,
        "maxLatency": 1.0
      },
      {
        "relationshipId": "00000000000002_0000000000
0005",
        "externalIdA": "000000000000002",
        "externalIdB": "000000000000005",
        "enable": true,
        "weight": 1.0,
        "bandwidth": 2.0,
        "maxLatency": 1.0
      },
      {
        "relationshipId": "00000000000004_0000000000
0005",
        "externalIdA": "000000000000004",
        "externalIdB": "000000000000005",
        "enable": true,
        "weight": 1.0,
        "bandwidth": 2.0,
        "maxLatency": 1.0
      }
    ]
  },
  {
    "invExternalId": "10.0.0.1",
    "invExternalName": "10.0.0.1",
    "invClass": "ELEMENT",
    "invNetLevel": "HOST",
    "invAssetType": "Host",
    "invState": null,
    "invFather": null,
    "invAttributeValues": [
      {
        "attribute": "InvDescription",
        "value": null
      },
      {
        "attribute": "sdnIP",
        "value": "10.0.0.1"
      }
    ]
  }
]

```

AIDB_001 SDN Asset creation – Result

```
[
  "relationships": [
    {
      "relationshipId": "10.0.0.1_0000000000000003",
      "externalIdA": "10.0.0.1",
      "externalIdB": "0000000000000003",
      "enable": true,
      "weight": 1.0,
      "bandwidth": 2.0,
      "maxLatency": 1.0
    },
    {
      "relationshipId": "0000000000000003_10.0.0.1",
      "externalIdA": "0000000000000003",
      "externalIdB": "10.0.0.1",
      "enable": true,
      "weight": 1.0,
      "bandwidth": 2.0,
      "maxLatency": 1.0
    }
  ],
  {
    "invExternalId": "10.0.0.2",
    "invExternalName": "10.0.0.2",
    "invClass": "ELEMENT",
    "invNetLevel": "HOST",
    "invAssetType": "Host",
    "invState": null,
    "invFather": null,
    "invAttributeValues": [
      {
        "attribute": "InvDescription",
        "value": null
      },
      {
        "attribute": "sdnIP",
        "value": "10.0.0.2"
      }
    ]
  },
  "relationships": [
    {
      "relationshipId": "10.0.0.2_0000000000000004",
      "externalIdA": "10.0.0.2",
      "externalIdB": "0000000000000004",
      "enable": true,
      "weight": 1.0,
      "bandwidth": 2.0,
      "maxLatency": 1.0
    },
    {
      "relationshipId": "0000000000000004_10.0.0.2",
      "externalIdA": "0000000000000004",
      "externalIdB": "10.0.0.2",
      "enable": true,
      "weight": 1.0,
      "bandwidth": 2.0,
      "maxLatency": 1.0
    }
  ],
  {
    "invExternalId": "10.0.0.3",
    "invExternalName": "10.0.0.3",
    "invClass": "ELEMENT",
    "invNetLevel": "HOST",
    "invAssetType": "Host",
    "invState": null,
    "invFather": null,
    "invAttributeValues": [
      {
        "attribute": "InvDescription",
        "value": null
      },
      {
        "attribute": "sdnIP",
        "value": "10.0.0.3"
      }
    ]
  },
  "relationships": [
    {
      "relationshipId": "10.0.0.3_0000000000000003",
      "externalIdA": "10.0.0.3",
      "externalIdB": "0000000000000003",
      "enable": true,
      "weight": 1.0,
      "bandwidth": 2.0,
      "maxLatency": 1.0
    },
    {
      "relationshipId": "0000000000000003_10.0.0.3",
      "externalIdA": "0000000000000003",
      "externalIdB": "10.0.0.3",
      "enable": true,
      "weight": 1.0,
      "bandwidth": 2.0,
      "maxLatency": 1.0
    }
  ]
}
```

AIDB_001 SDN Asset creation – Result

```
{
  "invClass": "ELEMENT",
  "invNetLevel": "HOST",
  "invAssetType": "Host",
  "invState": null,
  "invFather": null,
  "invAttributeValues": [
    {
      "attribute": "InvDescription",
      "value": null
    },
    {
      "attribute": "sdnIP",
      "value": "10.0.0.3"
    }
  ]
},
"relationships": [
  {
    "relationshipId": "10.0.0.3_0000000000000003",
    "externalIdA": "10.0.0.3",
    "externalIdB": "0000000000000003",
    "enable": true,
    "weight": 1.0,
    "bandwidth": 2.0,
    "maxLatency": 1.0
  },
  {
    "relationshipId": "0000000000000003_10.0.0.3",
    "externalIdA": "0000000000000003",
    "externalIdB": "10.0.0.3",
    "enable": true,
    "weight": 1.0,
    "bandwidth": 2.0,
    "maxLatency": 1.0
  }
]
}
```

AIDB_002 SDN Asset update – Input data

```
[
  {
    "invExternalId": "10.0.0.1",
    "invExternalName": "10.0.0.1",
    "invClass": "ELEMENT",
    "invNetLevel": "HOST",
    "invAssetType": "Host",
    "invState": "CONN_E",
    "invFather": null,
    "invAttributeValues": [
      {
        "attribute": "InvDescription",
        "value": "SCADA"
      }
    ]
  }
]
```

Notice both invState and InvDescription attribute values have been indicated in the Input and the result reflexes these updates.

AIDB_002 SDN Asset update – Result

```
[
```

AIDB_002 SDN Asset update – Result

```
{
  "invExternalId": "10.0.0.1",
  "invExternalName": "10.0.0.1",
  "invClass": "ELEMENT",
  "invNetLevel": "HOST",
  "invAssetType": "Host",
  "invState": "CONN_E",
  "invFather": null,
  "invAttributeValues": [
    {
      "attribute": "InvDescription",
      "value": "SCADA"
    },
    {
      "attribute": "sdnIP",
      "value": "10.0.0.1"
    }
  ],
  "relationships": [
    {
      "relationshipId": "10.0.0.1_0000000000000003",
      "externalIdA": "10.0.0.1",
      "externalIdB": "0000000000000003",
      "enable": true,
      "weight": 1.0,
      "bandwidth": 2.0,
      "maxLatency": 1.0
    },
    {
      "relationshipId": "0000000000000003_10.0.0.1",
      "externalIdA": "0000000000000003",
      "externalIdB": "10.0.0.1",
      "enable": true,
      "weight": 1.0,
      "bandwidth": 2.0,
      "maxLatency": 1.0
    }
  ]
}
```

AIDB_003 SDN topology attribute update – Input data

```
[
  {
    "invExternalId": "10.0.0.1",
    "invExternalName": "10.0.0.1",
    "invClass": "ELEMENT",
    "invNetLevel": "HOST",
    "invAssetType": "Host",
    "invState": "CONN_E",
    "invFather": null,
    "relationships": [
      {
        "relationshipId": "0000000000000003_10.0.0.1",
        "externalIdA": "0000000000000003",
        "externalIdB": "10.0.0.1",
        "enable": true,
        "weight": 7.0,
        "bandwidth": 2.0,
        "maxLatency": 1.0
      }
    ]
  }
]
```

AIDB_003 SDN topology attribute update – Input data

```
}
]
```

AIDB_003 SDN topology attribute update – Result

```
[
  {
    "invExternalId": "10.0.0.1",
    "invExternalName": "10.0.0.1",
    "invClass": "ELEMENT",
    "invNetLevel": "HOST",
    "invAssetType": "Host",
    "invState": "CONN_E",
    "invFather": null,
    "invAttributeValues": [
      {
        "attribute": "InvDescription",
        "value": "SCADA"
      },
      {
        "attribute": "sdnIP",
        "value": "10.0.0.1"
      }
    ],
    "relationships": [
      {
        "relationshipId": "10.0.0.1_0000000000000003",
        "externalIdA": "10.0.0.1",
        "externalIdB": "0000000000000003",
        "enable": true,
        "weight": 1.0,
        "bandwidth": 2.0,
        "maxLatency": 1.0
      },
      {
        "relationshipId": "0000000000000003_10.0.0.1",
        "externalIdA": "0000000000000003",
        "externalIdB": "10.0.0.1",
        "enable": true,
        "weight": 7.0,
        "bandwidth": 2.0,
        "maxLatency": 1.0
      }
    ]
  }
]
```

AIDB_004 SDN topology update – Input data

```
[
  {
    "invExternalId": "10.0.0.3",
    "invExternalName": "10.0.0.3",
    "invClass": "ELEMENT",
    "invNetLevel": "HOST",
    "invAssetType": "Host",
    "invState": null,
    "invFather": null,
    "invAttributeValues": [
      {

```

AIDB_004 SDN topology update
– Input data

```

    "attribute": "InvDescription",
    "value": null
  },
  {
    "attribute": "sdnIP",
    "value": "10.0.0.3"
  }
],
"relationships": [
  {
    "relationshipId": "10.0.0.3_0000000000000003",
    "externalIdA": null,
    "externalIdB": null,
  },
  {
    "relationshipId": "0000000000000003_10.0.0.3",
    "externalIdA": null,
    "externalIdB": null,
  },
  {
    "externalIdA": "10.0.0.3",
    "externalIdB": "0000000000000001",
    "enable": true,
    "weight": 1.0,
    "bandwidth": 2.0,
    "maxLatency": 1.0
  },
  {
    "externalIdA": "0000000000000001",
    "externalIdB": "10.0.0.3",
    "enable": true,
    "weight": 1.0,
    "bandwidth": 2.0,
    "maxLatency": 1.0
  }
]
}
]

```

Notice how the 10.0.0.3 is unplugged from the Switch3 and is plugged to the Switch1.

AIDB_004 SDN topology update
– Result

```

[
  {
    "invExternalId": "10.0.0.3",
    "invExternalName": "10.0.0.3",
    "invClass": "ELEMENT",
    "invNetLevel": "HOST",
    "invAssetType": "Host",
    "invState": null,
    "invFather": null,
    "invAttributeValues": [
      {
        "attribute": "InvDescription",
        "value": null
      },
      {
        "attribute": "sdnIP",

```

```

    "value": "10.0.0.3"
  }
],
"relationships": [
  {
    "relationshipId": "10.0.0.3_0000000000000001",
    "externalIdA": "10.0.0.3",
    "externalIdB": "0000000000000001",
    "enable": true,
    "weight": 1.0,
    "bandwidth": 2.0,
    "maxLatency": 1.0
  },
  {
    "relationshipId": "0000000000000001_10.0.0.3",
    "externalIdA": "0000000000000001",
    "externalIdB": "10.0.0.3",
    "enable": true,
    "weight": 1.0,
    "bandwidth": 2.0,
    "maxLatency": 1.0
  }
]
}
]

```

Due to the enormous length of input and results the AIDB_005 Grid definition registration input and results have been included in the document.

AIDB_006 SDN-Grid relationship creation - Input

```

[
  {
    "invExternalId": "10.0.0.3",
    "invExternalName": "10.0.0.3",
    "invClass": "ELEMENT",
    "invNetLevel": "HOST",
    "invAssetType": "Host",
    "invState": null,
    "invFather": null,
    "relationships": [
      {
        "externalIdA": "10.0.0.3",
        "externalIdB": "Diesel generator 1",
        "enable": true,
      },
      {
        "externalIdA": "Diesel generator 1",
        "externalIdB": "10.0.0.3",
        "enable": true,
      }
    ]
  }
]

```

13.4 EDAE unit testing details

13.4.1 Inputs for the workflow

From AIDB to EDAE process

```
[
{
  "invExternalId": "SDN_CONTROLLER_UC0",
  "invExternalName": "SDN_CONTROLLER_UC0",
  "invClass": "ELEMENT",
  "invNetLevel": "SDN_CONTROLLER_LEVEL",
  "invAssetType": "SDN Controller",
  "invState": "CONN_E",
  "invFather": "10.0.0.1",
  "invAttributeValues": [
    {
      "attribute": "sdnDriver",
      "value": "Northbound"
    },
    {
      "attribute": "sdnEndpoint",
      "value": "sdnEndpoint1"
    }
  ],
  "relationships": "None"
},
{
  "invExternalId": "0000000000000001",
  "invExternalName": "0000000000000001",
  "invClass": "ELEMENT",
  "invNetLevel": "SDN_SWITCH_LEVEL",
  "invAssetType": "SDN_SWITCH",
  "invState": "CONN_E",
  "invFather": "None",
  "invAttributeValues": [
    {
      "attribute": "InvDescription",
      "value": "None"
    },
    {
      "attribute": "sdnManufacturer",
      "value": "Nicira, Inc."
    },
    {
      "attribute": "sdnSwDesc",
      "value": "2.3.90"
    }
  ],
  "relationships": [
    {
      "relationshipId": "1",
      "externalIdA": "0000000000000001",
      "externalIdB": "0000000000000002",
      "enable": true,
      "weight": 1.0,
      "capacity": 1.0
    },
    {
      "relationshipId": "2",
      "externalIdA": "0000000000000002",
      "externalIdB": "0000000000000001",
      "enable": true,
      "weight": 1.0,
      "capacity": 1.0
    }
  ]
}
]
```

```
    "relationshipId": "3",
    "externalIdA": "0000000000000001",
    "externalIdB": "0000000000000003",
    "enable": true,
    "weight": 1.0,
    "capacity": 1.0
  },
  {
    "relationshipId": "4",
    "externalIdA": "0000000000000003",
    "externalIdB": "0000000000000001",
    "enable": true,
    "weight": 1.0,
    "capacity": 1.0
  }
],
{
  "invExternalId": "0000000000000002",
  "invExternalName": "0000000000000002",
  "invClass": "ELEMENT",
  "invNetLevel": "SDN_SWITCH_LEVEL",
  "invAssetType": "SDN_SWITCH",
  "invState": "CONN_E",
  "invFather": "None",
  "invAttributeValues": [
    {
      "attribute": "InvDescription",
      "value": "None"
    },
    {
      "attribute": "sdnManufacturer",
      "value": "Nicira, Inc."
    },
    {
      "attribute": "sdnSwDesc",
      "value": "2.3.90"
    }
  ],
  "relationships": [
    {
      "relationshipId": "1",
      "externalIdA": "0000000000000001",
      "externalIdB": "0000000000000002",
      "enable": true,
      "weight": 1.0,
      "capacity": 1.0
    },
    {
      "relationshipId": "2",
      "externalIdA": "0000000000000002",
      "externalIdB": "0000000000000001",
      "enable": true,
      "weight": 1.0,
      "capacity": 1.0
    },
    {
      "relationshipId": "5",
      "externalIdA": "0000000000000002",
      "externalIdB": "0000000000000003",
      "enable": true,
      "weight": 1.0,
      "capacity": 1.0
    },
    {
      "relationshipId": "6",
      "externalIdA": "0000000000000003",

```

```

    "externalIdB": "0000000000000002",
    "enable": true,
    "weight": 1.0,
    "capacity": 1.0
  }
},
{
  "invExternalId": "0000000000000003",
  "invExternalName": "0000000000000003",
  "invClass": "ELEMENT",
  "invNetLevel": "SDN_SWITCH_LEVEL",
  "invAssetType": "SDN_SWITCH",
  "invState": "CONN_E",
  "invFather": "None",
  "invAttributeValues": [
    {
      "attribute": "InvDescription",
      "value": "None"
    },
    {
      "attribute": "sdnManufacturer",
      "value": "Nicira, Inc."
    },
    {
      "attribute": "sdnSwDesc",
      "value": "2.3.90"
    }
  ],
  "relationships": [
    {
      "relationshipId": "4",
      "externalIdA": "0000000000000003",
      "externalIdB": "0000000000000001",
      "enable": true,
      "weight": 1.0,
      "capacity": 1.0
    },
    {
      "relationshipId": "3",
      "externalIdA": "0000000000000001",
      "externalIdB": "0000000000000003",
      "enable": true,
      "weight": 1.0,
      "capacity": 1.0
    },
    {
      "relationshipId": "6",
      "externalIdA": "0000000000000003",
      "externalIdB": "0000000000000002",
      "enable": true,
      "weight": 1.0,
      "capacity": 1.0
    },
    {
      "relationshipId": "5",
      "externalIdA": "0000000000000002",
      "externalIdB": "0000000000000003",
      "enable": true,
      "weight": 1.0,
      "capacity": 1.0
    }
  ],
  {
    "invExternalId": "10.0.0.1",
    "invExternalName": "10.0.0.1",

```

```

    "invClass": "ELEMENT",
    "invNetLevel": "SDN_HOST_LEVEL",
    "invAssetType": "PDC",
    "invState": "CONN_E",
    "invFather": "None",
    "invAttributeValues": [
      {
        "attribute": "InvDescription",
        "value": "None"
      },
      {
        "attribute": "sdnIP",
        "value": "10.0.0.1"
      }
    ],
    "relationships": [
      {
        "relationshipId": "7",
        "externalIdA": "10.0.0.1",
        "externalIdB": "0000000000000001",
        "enable": true,
        "weight": 1.0,
        "capacity": 1.0
      },
      {
        "relationshipId": "8",
        "externalIdA": "0000000000000001",
        "externalIdB": "10.0.0.1",
        "enable": true,
        "weight": 1.0,
        "capacity": 1.0
      },
      {
        "relationshipId": "80",
        "externalIdA": "10.0.0.3",
        "externalIdB": "10.0.0.3",
        "enable": true,
        "weight": 1.0,
        "capacity": 1.0
      }
    ],
    {
      "invExternalId": "10.0.0.2",
      "invExternalName": "10.0.0.2",
      "invClass": "ELEMENT",
      "invNetLevel": "SDN_HOST_LEVEL",
      "invAssetType": "PDC",
      "invState": "CONN_E",
      "invFather": "None",
      "invAttributeValues": [
        {
          "attribute": "InvDescription",
          "value": "None"
        },
        {
          "attribute": "sdnIP",
          "value": "10.0.0.2"
        }
      ],
      "relationships": [
        {
          "relationshipId": "9",
          "externalIdA": "10.0.0.2",
          "externalIdB": "0000000000000002",
          "enable": true,
          "weight": 1.0,

```

```

"capacity": 1.0
},
{
  "relationshipId": "10",
  "externalIdA": "0000000000000002",
  "externalIdB": "10.0.0.2",
  "enable": true,
  "weight": 1.0,
  "capacity": 1.0
}
],
{
  "invExternalId": "10.0.0.3",
  "invExternalName": "10.0.0.3",
  "invClass": "ELEMENT",
  "invNetLevel": "SDN_HOST_LEVEL",
  "invAssetType": "PMU",
  "invState": "CONN_E",
  "invFather": "10.0.0.2",
  "invAttributeValues": [
    {
      "attribute": "InvDescription",
      "value": "None"
    },
    {
      "attribute": "sdnIP",
      "value": "10.0.0.3"
    }
  ]
},
"relationships": [
  {
    "relationshipId": "11",
    "externalIdA": "10.0.0.3",
    "externalIdB": "0000000000000003",
    "enable": true,
    "weight": 1.0,
    "capacity": 1.0
  },
  {
    "relationshipId": "12",
    "externalIdA": "0000000000000003",
    "externalIdB": "10.0.0.3",
    "enable": true,
    "weight": 1.0,
    "capacity": 1.0
  },
  {
    "relationshipId": "120",
    "externalIdA": "10.0.0.1",
    "externalIdB": "10.0.0.1",
    "enable": true,
    "weight": 1.0,
    "capacity": 1.0
  }
],
{
  "invExternalId": "10.0.0.4",
  "invExternalName": "10.0.0.4",
  "invClass": "ELEMENT",
  "invNetLevel": "SDN_HOST_LEVEL",
  "invAssetType": "Attacker",
  "invState": "CONN_E",
  "invFather": "None",
  "invAttributeValues": [

```

```

"attribute": "InvDescription",
"value": "None"
},
{
  "attribute": "sdnIP",
"value": "10.0.0.3"
}
],
"relationships": [
  {
    "relationshipId": "13",
    "externalIdA": "10.0.0.4",
    "externalIdB": "0000000000000001",
    "enable": true,
    "weight": 1.0,
    "capacity": 1.0
  },
  {
    "relationshipId": "14",
    "externalIdA": "0000000000000001",
    "externalIdB": "10.0.0.4",
    "enable": true,
    "weight": 1.0,
    "capacity": 1.0
  }
]
]

```

From S-RAF to EDAE process

```

{
  "ASSETS": [
    {
      "name": "SDN-Switch",
      "type": "h",
      "category": "NetworkComponent/Switch",
      "gdpr": false,
      "assetId": 1,
      "riskassessmentId": 1714,
      "businesspartnerId": 1,
      "risklevel": "M",
      "threats": [
        {
          "threat": "DEFAULT-1",
          "name": "DEFAULT",
          "level": "VH",
          "vulnerabilities": [
            {
              "vulnerability": "CVE-2019-1890",
              "level": "L",
              "impact": "M",
              "risklevel": "M",
              "privacyfunctionalimpactscore": 0,
              "privacyimpactscorescore": 0,
              "privacydatatypes": [
                ],
              "privacydatatypescore": 0,
              "privacyscore": 0,
              "privavyriskscore": 0,
              "cvssscore": 3,
              "cvss": "L"
            }
          ]
        }
      ],
    }
  ],

```



From S-RAF to EDAE process

```

    "result": "M"
  },
  ],
  "cumulativeRiskLevel": "M",
  "contributingRisks": [
    "M"
  ],
  "controls": [
  ],
  "links": [
    {
      "type": "CONNECTED_TO",
      "assetId": 3
    },
    {
      "type": "CONNECTED_TO",
      "assetId": 5
    }
  ],
  "businessValue": "VH",
  "tags": {
  }
},
{
  "name": "SDN-enabled RTU",
  "category": "Business Service/Infrastructure Service",
  "gdpr": false,
  "assetId": 2,
  "riskassessmentId": 1714,
  "businesspartnerId": 1,
  "risklevel": "H",
  "threats": [
    {
      "threat": "DEFAULT-1",
      "name": "DEFAULT",
      "level": "VH",
      "vulnerabilities": [
        {
          "vulnerability": "CVE-2019-14931",
          "level": "VL",
          "impact": "VH",
          "risklevel": "H",
          "privacyfunctionalimpactscore": 0,
          "privacyimpactscorescore": 0,
          "privacydatatypes": [
          ],
          "privacydatatypescore": 0,
          "privacyscore": 0,
          "privavyriskscore": 0,
          "cvssscore": 6,
          "cvss": "H"
        }
      ],
      "privacydatatypescore": 0,
      "privacyscore": 0,
      "privavyriskscore": 0,
      "cvssscore": 6,
      "cvss": "H"
    }
  ],
  "result": "H"
}
],
"cumulativeRiskLevel": "H",
"contributingRisks": [
  "H"
],
"controls": [
],

```

From S-RAF to EDAE process

```

    "links": [
      {
        "type": "CONNECTED_TO",
        "assetId": 1
      }
    ],
    "businessValue": "VH",
    "tags": {
      "attackpath": "source"
    }
  },
  {
    "name": "Programmable Logic Controller 1",
    "type": "h",
    "category": "Computer/Server",
    "gdpr": false,
    "assetId": 3,
    "riskassessmentId": 1714,
    "businesspartnerId": 1,
    "risklevel": "H",
    "threats": [
      {
        "threat": "DEFAULT-1",
        "name": "DEFAULT",
        "level": "VH",
        "vulnerabilities": [
          {
            "vulnerability": "CVE-2017-6031",
            "level": "VH",
            "impact": "M",
            "risklevel": "H",
            "description": "A Header Injection issue was discovered in Certec EDV GmbH atvise scada prior to Version 3.0. An \"improper neutralization of HTTP headers for scripting syntax\" issue has been identified, which may allow remote code execution.",
            "privacyfunctionalimpactscore": 0,
            "privacyimpactscorescore": 0,
            "privacydatatypes": [
            ],
            "privacydatatypescore": 0,
            "privacyscore": 0,
            "privavyriskscore": 0,
            "cvssscore": 6.8,
            "cvss": "H"
          }
        ],
        "privacydatatypescore": 0,
        "privacyscore": 0,
        "privavyriskscore": 0,
        "cvssscore": 7.8,
        "cvss": "H"
      },
      {
        "vulnerability": "CVE-2013-2780",
        "level": "VH",
        "impact": "L",
        "risklevel": "M",
        "privacyfunctionalimpactscore": 0,
        "privacyimpactscorescore": 0,
        "privacydatatypes": [
        ],
        "privacydatatypescore": 0,
        "privacyscore": 0,
        "privavyriskscore": 0,
        "cvssscore": 7.8,
        "cvss": "H"
      },
      {
        "vulnerability": "CVE-2017-6033",
        "level": "VH",

```


**From S-RAF to EDAA process**

```

        "impact": "M",
        "risklevel": "H",
        "description": "A DLL Hijacking issue was
discovered in Schneider Electric Interactive Graphical SCADA
System (IGSS) Software, Version 12 and previous versions. The
software will execute a malicious file if it is named the same as
a legitimate file and placed in a location that is earlier in the
search path.",
        "privacyfunctionalimpactscore": 0,
        "privacyimpactscorescore": 0,
        "privacydatatypes": [

        ],
        "privacydatatypescore": 0,
        "privacyscore": 0,
        "privavyriskscore": 0,
        "cvssscore": 6.8,
        "cvss": "H"
    }
    ],
    "result": "H"
}
],
"cumulativeRiskLevel": "H",
"contributingirls": [
    "H",
    "M"
],
"controls": [

],
"links": [
    {
        "type": "USED_BY",
        "assetId": 4
    }
],
"businessValue": "VH",
"tags": {

}
},
{
    "name": "Administrator",
    "category": "Organizational/Personnel",
    "gdpr": false,
    "assetId": 4,
    "riskassessmentId": 1714,
    "businesspartnerId": 1,
    "risklevel": "",
    "threats": [

    ],
    "cumulativeRiskLevel": "",
    "contributingirls": [

    ],
    "controls": [

    ],
    "links": [

    ],
    "businessValue": "H",
    "tags": {
        "attackpath": "destination"
    }
}

```

From S-RAF to EDAA process

```

    }
},
{
    "name": "Programmable Logic Controller 2",
    "type": "h",
    "category": "Computer/Server",
    "gdpr": false,
    "assetId": 5,
    "riskassessmentId": 1714,
    "businesspartnerId": 1,
    "risklevel": "H",
    "threats": [
        {
            "threat": "DEFAULT-1",
            "name": "DEFAULT",
            "level": "VH",
            "vulnerabilities": [
                {
                    "vulnerability": "CVE-2013-2780",
                    "level": "VH",
                    "impact": "L",
                    "risklevel": "M",
                    "privacyfunctionalimpactscore": 0,
                    "privacyimpactscorescore": 0,
                    "privacydatatypes": [

                    ],
                    "privacydatatypescore": 0,
                    "privacyscore": 0,
                    "privavyriskscore": 0,
                    "cvssscore": 7.8,
                    "cvss": "H"
                }
            ],
            "vulnerability": "CVE-2017-6033",
            "level": "VH",
            "impact": "M",
            "risklevel": "H",
            "description": "A DLL Hijacking issue was
discovered in Schneider Electric Interactive Graphical SCADA
System (IGSS) Software, Version 12 and previous versions. The
software will execute a malicious file if it is named the same as
a legitimate file and placed in a location that is earlier in the
search path.",
            "privacyfunctionalimpactscore": 0,
            "privacyimpactscorescore": 0,
            "privacydatatypes": [

            ],
            "privacydatatypescore": 0,
            "privacyscore": 0,
            "privavyriskscore": 0,
            "cvssscore": 6.8,
            "cvss": "H"
        }
    ],
    "vulnerability": "CVE-2017-6031",
    "level": "VH",
    "impact": "M",
    "risklevel": "H",
    "description": "A Header Injection issue was
discovered in Certec EDV GmbH atvise scada prior to Version
3.0. An \"improper neutralization of HTTP headers for scripting
syntax\" issue has been identified, which may allow remote
code execution.",
    "privacyfunctionalimpactscore": 0,

```

**From S-RAF to EDAE process**

```

    "privacyimpactscopescore":0,
    "privacydatatypes":[
      ],
    "privacydatatypescore":0,
    "privacyscore":0,
    "privavyriskscore":0,
    "cvssscore":6.8,
    "cvss":"H"
  }
},
"result":"H"
}
],
"cumulativeRiskLevel":"H",
"contributingirls":[
  "H",
  "M"
],
"controls":[
],
"links":[
  {
    "type":"CONNECTED_TO",
    "assetId":6
  }
],
"businessValue":"VH",
"tags":{
}
},
{
  "name":"SCADA 1",
  "type":"h",
  "category":"Computer/Server",
  "gdpr":false,
  "assetId":6,
  "riskassessmentId":1714,
  "businesspartnerId":1,
  "risklevel":"H",
  "threats":[
    {
      "threat":"DEFAULT-1",
      "name":"DEFAULT",
      "level":"VH",
      "vulnerabilities":{
        {
          "vulnerability":"CVE-2013-2780",
          "level":"VH",
          "impact":"L",
          "risklevel":"M",
          "privacyfunctionalimpactscore":0,
          "privacyimpactscopescore":0,
          "privacydatatypes":{
            ],
            "privacydatatypescore":0,
            "privacyscore":0,
            "privavyriskscore":0,
            "cvssscore":7.8,
            "cvss":"H"
          },
          {
            "vulnerability":"CVE-2017-6031",

```

From S-RAF to EDAE process

```

    "level":"VH",
    "impact":"M",
    "risklevel":"H",
    "description":"A Header Injection issue was
discovered in Certec EDV GmbH atvise scada prior to Version
3.0. An \"improper neutralization of HTTP headers for scripting
syntax\" issue has been identified, which may allow remote
code execution.",
    "privacyfunctionalimpactscore":0,
    "privacyimpactscopescore":0,
    "privacydatatypes":{
      ],
      "privacydatatypescore":0,
      "privacyscore":0,
      "privavyriskscore":0,
      "cvssscore":6.8,
      "cvss":"H"
    },
    {
      "vulnerability":"CVE-2017-6033",
      "level":"VH",
      "impact":"M",
      "risklevel":"H",
      "description":"A DLL Hijacking issue was
discovered in Schneider Electric Interactive Graphical SCADA
System (IGSS) Software, Version 12 and previous versions. The
software will execute a malicious file if it is named the same as
a legitimate file and placed in a location that is earlier in the
search path.",
      "privacyfunctionalimpactscore":0,
      "privacyimpactscopescore":0,
      "privacydatatypes":{
        ],
        "privacydatatypescore":0,
        "privacyscore":0,
        "privavyriskscore":0,
        "cvssscore":6.8,
        "cvss":"H"
      }
    },
    "result":"H"
  }
],
"cumulativeRiskLevel":"H",
"contributingirls":[
  "H",
  "M"
],
"controls":[
],
"links":[
],
"businessValue":"VH",
"tags":{
  "attackpath":"destination"
}
},
"PATHS":{
  {
    "id":"5f7ec5302af19b000193880f",

```

**From S-RAF to EDAE process**

```
"name": "SDN-enabled RTU",
"category": "Business Service/Infrastructure Service",
"gdpr": false,
"assetId": 2,
"riskassessmentId": 1714,
"businesspartnerId": 1,
"risklevel": "H",
"threats": [
  {
    "threat": "DEFAULT-1",
    "name": "DEFAULT",
    "level": "VH",
    "vulnerabilities": [
      {
        "vulnerability": "CVE-2019-14931",
        "level": "VL",
        "impact": "VH",
        "risklevel": "H",
        "privacyfunctionalimpactscore": 0,
        "privacyimpactscopescore": 0,
        "privacydatatypes": [
          ],
        "privacydatatypescore": 0,
        "privacyscore": 0,
        "privavyriskscore": 0,
        "cvssscore": 6,
        "cvss": "H"
      }
    ],
    "result": "H"
  }
],
"cumulativeRiskLevel": "H",
"contributingirls": [
  "H"
],
"controls": [
],
"links": [
  {
    "type": "CONNECTED_TO",
    "assetId": 1
  }
],
"businessValue": "VH",
"tags": {
  "attackpath": "source"
}
},
{
  "id": "5f7ec5302af19b000193880e",
  "name": "SDN-Switch",
  "type": "h",
  "category": "NetworkComponent/Switch",
  "gdpr": false,
  "assetId": 1,
  "riskassessmentId": 1714,
  "businesspartnerId": 1,
  "risklevel": "M",
  "threats": [
    {
      "threat": "DEFAULT-1",
      "name": "DEFAULT",
      "level": "VH",
```

From S-RAF to EDAE process

```
"vulnerabilities": [
  {
    "vulnerability": "CVE-2019-1890",
    "level": "L",
    "impact": "M",
    "risklevel": "M",
    "privacyfunctionalimpactscore": 0,
    "privacyimpactscopescore": 0,
    "privacydatatypes": [
      ],
    "privacydatatypescore": 0,
    "privacyscore": 0,
    "privavyriskscore": 0,
    "cvssscore": 3,
    "cvss": "L"
  }
],
"result": "M"
},
"cumulativeRiskLevel": "M",
"contributingirls": [
  "M"
],
"controls": [
],
"links": [
  {
    "type": "CONNECTED_TO",
    "assetId": 3
  },
  {
    "type": "CONNECTED_TO",
    "assetId": 5
  }
],
"businessValue": "VH",
"tags": {
}
},
{
  "id": "5f7ec5302af19b0001938810",
  "name": "Programmable Logic Controller 1",
  "type": "h",
  "category": "Computer/Server",
  "gdpr": false,
  "assetId": 3,
  "riskassessmentId": 1714,
  "businesspartnerId": 1,
  "risklevel": "H",
  "threats": [
    {
      "threat": "DEFAULT-1",
      "name": "DEFAULT",
      "level": "VH",
      "vulnerabilities": [
        {
          "vulnerability": "CVE-2017-6031",
          "level": "VH",
          "impact": "M",
          "risklevel": "H",
          "description": "A Header Injection issue was discovered in Certec EDV GmbH atvise scada prior to Version
```

From S-RAF to EDAE process

3.0. An "\improper neutralization of HTTP headers for scripting syntax\" issue has been identified, which may allow remote code execution.",

```
"privacyfunctionalimpactscore":0,
"privacyimpactscorescore":0,
"privacydatatypes":[
```

```
],
"privacydatatypescore":0,
"privacyscore":0,
"privavyriskscore":0,
"cvssscore":6.8,
"cvss":"H"
```

```
},
{
  "vulnerability":"CVE-2013-2780",
  "level":"VH",
  "impact":"L",
  "risklevel":"M",
  "privacyfunctionalimpactscore":0,
  "privacyimpactscorescore":0,
  "privacydatatypes":[
```

```
],
"privacydatatypescore":0,
"privacyscore":0,
"privavyriskscore":0,
"cvssscore":7.8,
"cvss":"H"
```

```
},
{
  "vulnerability":"CVE-2017-6033",
  "level":"VH",
  "impact":"M",
  "risklevel":"H",
  "description":"A DLL Hijacking issue was
```

discovered in Schneider Electric Interactive Graphical SCADA System (IGSS) Software, Version 12 and previous versions. The software will execute a malicious file if it is named the same as a legitimate file and placed in a location that is earlier in the search path.",

```
"privacyfunctionalimpactscore":0,
"privacyimpactscorescore":0,
"privacydatatypes":[
```

```
],
"privacydatatypescore":0,
"privacyscore":0,
"privavyriskscore":0,
"cvssscore":6.8,
"cvss":"H"
```

```
}
},
"result":"H"
```

```
}
},
"cumulativeRiskLevel":"H",
"contributinggirls":[
```

```
"H",
"M"
```

```
],
"controls":[
```

```
],
"links":[
{
```

From S-RAF to EDAE process

```
"type":"USED_BY",
"assetId":4
```

```
}
],
"businessValue":"VH",
"tags":{
```

```
}
},
{
  "id":"5f7ec5302af19b0001938811",
  "name":"Administrator",
  "category":"Organizational/Personnel",
  "gdpr":false,
  "assetId":4,
  "riskassessmentId":1714,
  "businesspartnerId":1,
  "risklevel":"",
  "threats":[
```

```
],
"cumulativeRiskLevel":"",
"contributinggirls":[
```

```
],
"controls":[
```

```
],
"links":[
```

```
],
"businessValue":"H",
"tags":{
  "attackpath":"destination"
}
```

```
}
]
}
```

From SDN-C to EDAE process

```
{
  "1": [
    {
      "port_no": 1,
      "rx_packets": 9,
      "tx_packets": 6,
      "rx_bytes": 738,
      "tx_bytes": 252,
      "rx_dropped": 0,
      "tx_dropped": 0,
      "rx_errors": 0,
      "tx_errors": 0,
      "rx_frame_err": 0,
      "rx_over_err": 0,
      "rx_crc_err": 0,
      "collisions": 0,
      "duration_sec": 12,
      "duration_nsec": 9.76e+08
    },
  ]
}
```

From EDAE process to SDN-C

```
{
  "dpid": 1,
  "cookie": 1,
  "cookie_mask": 1,
  "table_id": 0,
  "idle_timeout": 30,
  "hard_timeout": 30,
  "priority": 11111,
  "flags": 1,
  "match": {
    "in_port": 1
  },
  "actions": [
    {
      "type": "OUTPUT",
      "port": 2
    }
  ]
}' http://localhost:8080/stats/flowentry/add
```

```
    "PDC_2"
  ],
  {
    "path": [
      "PMU_4",
      "SW_2",
      "SW_1",
      "PDC_1"
    ]
  }
}
```

From EDAE process to EDAE-Dashbaord

```
{
  "APPROVAL_REQUIRED": true,
  "PDC": [
    {
      "ID": "PDC_1",
      "PMUS_con": 2,
      "PMUS": [
        "PMU_1",
        "PMU_4"
      ]
    },
    {
      "ID": "PDC_3",
      "PMUS_con": 3,
      "PMUS": [
        "PMU_1", "PMU_2",
        "PMU_4"
      ]
    }
  ],
  "PMU": [
    {
      "ID": "PMU_1",
      "PDC": [
        "PDC_1",
        "PDC_13"
      ],
      "traffic_demand": "10 Mbps"
    },
    {
      "ID": "PMU_2",
      "PDC": [
        "PDC_12"
      ],
      "traffic_demand": "20 Mbps"
    }
  ],
  "PATH": [
    {
      "path": [
        "PMU_1",
        "SW_3",
        "SW_2",
        "SW_1",

```

From EDAE process to EDAE-Dashboard

```
{
  "APPROVAL_REQUIRED":true,
  "PDC":[
    {
      "ID":"PDC_1",
      "PMUS_con":2,
      "PMUS":[
        "PMU_1",
        "PMU_4"
      ]
    },
    {
      "ID":"PDC_3",
      "PMUS_con":3,
      "PMUS":[
        "PMU_1", "PMU_2",
        "PMU_4"
      ]
    }
  ],
  "PMU":[
    {
      "ID":"PMU_1",
      "PDC":[
        "PDC_1",
        "PDC_13"
      ],
      "traffic_demand":"10 Mbps"
    },
    {
      "ID":"PMU_2",
      "PDC":[
        "PDC_12"
      ],
      "traffic_demand":"20 Mbps"
    }
  ],
  "PATH":[
    {
      "path":[
        "PMU_1",
        "SW_3",
        "SW_2",
        "SW_1",
        "PDC_2"
      ]
    },
    {
      "path":[
        "PMU_4",
        "SW_2",
        "SW_1",
        "PDC_1"
      ]
    }
  ]
}
```

13.4.2 Output of the workflow

From AIDB to EDAE process

Same as the input

From S-RAF to EDAE process

Same as the input

From SDN-C to EDAE process

Same as the input

From EDAE process to SDN-C

Same as the input

From EDAE process to EDAE-Dashbaord

```
{
  "PROPOSAL_ID":53427606,
  "DATA":{
    "APPROVAL_REQUIRED":true,
    "PDC":[
      {
        "ID":"PDC_1",
        "PMUS_con":2,
        "PMUS":[
          "PMU_1",
          "PMU_4"
        ]
      },
      {
        "ID":"PDC_3",
        "PMUS_con":3,
        "PMUS":[
          "PMU_1", "PMU_2",
          "PMU_4"
        ]
      }
    ],
    "PMU":[
      {
        "ID":"PMU_1",
        "PDC":[
          "PDC_1",
          "PDC_13"
        ],
        "traffic_demand":"10 Mbps"
      },
      {
        "ID":"PMU_2",
        "PDC":[
          "PDC_12"
        ],
        "traffic_demand":"20 Mbps"
      }
    ],
    "PATH":[
      {
        "path":[
          "PMU_1",
          "SW_3",
          "SW_2",
          "SW_1",
```

```
      "PDC_2"
    ]
  },
  {
    "path":[
      "PMU_4",
      "SW_2",
      "SW_1",
      "PDC_1"
    ]
  }
]
}
```

From EDAE to SDN-C

```
{
  "dpid": 1,
  "cookie": 1,
  "cookie_mask": 1,
  "table_id": 0,
  "idle_timeout": 30,
  "hard_timeout": 30,
  "priority": 11111,
  "flags": 1,
  "match":{
    "in_port":1
  },
  "actions":[
    {
      "type":"OUTPUT",
      "port": 2
    }
  ]
}' http://localhost:8080/stats/flowentry/add
```

13.5 SDN-C API details

This section is considered as confidential and it is found in another document.

