



SEMIoTICS

Deliverable D3.1: Software defined programmability for IoT Devices (first draft)

Deliverable release date	Initial 01.03.2019, revised 25.11.2019
Authors	1. Ermin Sakic, Arne Bröring (SAG) 2. Eftychia Lakka, Othonas Soultatos, Nikolaos Petroulakis (FORTH) 3. Jordi Serra, Luis Sanabria-Russo (CTTC) 4. Philip Wright (ENG) 5. Prodromos Vasileios Mekikis (IQUADRAT)
Responsible person	Ermin Sakic (SAG)
Reviewed by	Nikolaos Petroulakis (FORTH), Ermin Sakic, Vivek Kulkarni (SAG), Luis Sanabria-Russo, Jordi Serra (CTTC), Konstantinos Fysarakis (STS), Philip Wright (ENG), Kostas Ramantas (IQUADRAT)
Approved by	PTC Members (Vivek Kulkarni, Nikolaos Petroulakis, Ermin Sakic, Mirko Falchetto, Domenico Presenza, Verikoukis Christos) PCC Members (Vivek Kulkarni, Nikolaos Petroulakis, Verikoukis Christos, Georgios Spanoudakis, Domenico Presenza, Danilo Pau, Joachim Posegga, Darek Dober, Kostas Ramantas, Ulrich Hansen)
Status of the Document	Final
Version	1.0 revised
Dissemination level	Public

Table of Contents

1	Introduction.....	4
1.1	SEMIoTICS approach – Networking Aspects.....	5
1.2	Logical link with WP2 beneficiaries – T2.3.....	5
1.3	Logical link with WP3 beneficiaries – T3.3, T3.4.....	6
1.4	Logical link with WP4 beneficiaries – T4.1.....	6
1.5	PERT chart of SEMIoTICS	7
2	SEMIoTICS networking Infrastructure Requirements	8
2.1	Control Plane Requirements	8
2.2	Data Plane Requirements	11
2.3	Use Case-Specific Scenario Requirements	11
3	SEMIoTICS Network Controller Architecture	14
3.1	Role of the SDN Controller in SEMIoTICS Architecture	14
3.2	SDN Controller Function Blocks	24
4	Detailed SDN Controller Function Descriptions	29
4.1	Control / Data Plane Bootstrapping Manager	29
4.2	Pattern Engine	29
4.3	VTN Manager	30
4.4	Path Manager	31
4.5	Resource Manager	31
4.6	Security Manager	32
4.7	High-Availability Controller Clustering	32
4.8	SFC Manager	34
4.9	NFV MANO/VIM Connector.....	34
5	Interactions with Data-Plane Devices	37
5.1	OpenFlow for Control Plane Interactions	37
5.2	OVSDB/NETCONF/YANG for Management Plane Interactions	37
6	Evaluation of wireless field technologies	39
6.1	IEEE 802.11 Wireless LAN	41
6.2	IEEE 802.15.4	41
6.3	IEEE 802.15.4k.....	41
6.4	WiMAX	42
6.5	Xbee PRO 868.....	42
6.6	RF Mesh / multihop.....	43
6.7	LPWAN.....	43
6.8	Satellite Communication	44
6.9	Public Cellular NB (GPRS, GSM): MHz & BB (HSPA, LTE)	44
7	SDN in Context of SEMIoTICS Use Cases	45

8	Next Steps and Conclusions	47
9	References	49

Acronyms Table

Acronym	Definition
ASIC	Application-Specific Integrated Circuit
CPU	Central Processing Unit
DoS	Denial of Service
DPDK	Data Plane Developer's Kit
FPGA	Field-Programmable Gate Array
IoT	Internet of Things
IIoT	Industrial Internet of Things
KVM	Kernel-based Virtual Machine
LXD	Linux Containers
JSON	JavaScript Object Notation
FW	Firmware
LWM2M	Lightweight Machine-to-M
mW	Milliwatts
M2M	Machine-to-M
MCU	Micro Controller Unit
MQTT	Message Queuing Telemetry Transport
NETCONF	Network Configuration Protocol
NFV	Network functions virtualization
NFVO	NFV orchestrator
OFCONF	OpenFlow Configuration
OVSDB	Open vSwitch Database Management Protocol
ODL	OpenDaylight
POP	Point of Presence
QoS	Quality of Service
RO	Resource Orchestrator
SDN	Software-Defined Networking
SoS	System of System
SARA	Socially Assistive Robotic Solution for Mild Cognitive Impairment or mild Alzheimer's disease
SEMIOTICS	Smart End-to-end Massive IoT Interoperability, Connectivity and Security
SFC	Service Function Chaining
SLAM	Simultaneous Localization and Mapping
SPDI	Security, Privacy, Dependability, and Interoperability
SSC	SEMIOTICS SDN Controller
SSD	Solid State Disk
UC	Use Case
VIM	Virtualized Infrastructure Manager
VLAN	Virtual Local Area Network
VM	Virtual Machine
VNF	Virtual Network Function
vSwitch	Virtual Switch
VTN	Virtual Tenant Network
WoT	Web of Things

1 INTRODUCTION

The SEMIoTICS SDN Controller (SSC) is the centralized network intelligence responsible for mapping of Virtual Tenant Networks (VTNs) and Application Services (i.e., “connectivity patterns”) onto the underlying physical network topology. The SSC is furthermore aware of the network devices’ capabilities and resources.

The SSC is a typical example of a Network Operating System (NOS) to which the forwarding devices offload essential management and control tasks (i.e., path finding) to effectively decrease the data plane complexity and thus simplify the forwarding device costs. Contrary to specialized NOSs, which define the control plane of network devices and run independently of other NOS instances in the network, the SSC centralizes the NOS functions and provides a single logical entry point to the control plane.

The SSC aims to allow for centralized control plane manipulation of devices. It does so by being able to intelligently configure the forwarding tables (filters and FIBs) according to a pattern specification and the included SPDI and QoS properties and constraints. For the purpose of fault tolerance and high availability, it is capable of enforcing redundant paths, with respect to the up-to-date status of network topology. The global view of the topology and reservation states is leveraged in order to solve tasks where deployment of custom-tailored distributed protocol solutions might result in additional overhead and added complexity in the network. For example, centralized constraint-based routing based on actual network load allows for a higher degree of bandwidth utilization for guaranteed QoS, than the protocol-based approaches which perform routing and reservation in sequence. Configuration of forwarding databases, queue and ingress policing are some additional management tasks which the SSC is responsible for.

In this deliverable, we propose the SSC architecture that encompasses the various functions required to fulfil the network-related requirements defined in D2.3. To this end, in this deliverable:

- We present the high-level workflow of interactions between higher-layer instances of the SEMIoTICS architecture (Pattern Orchestrator) and the SSC.
- We reiterate on the network-specific requirements, required by the various use cases.
- We propose and discuss each of the controller functions necessary to realize the SEMIoTICS approach.
- We discuss the existing interfaces between the SSC and the data-plane devices.
- We compare the proposed architecture to the two state-of-art architectures, the generic OpenDaylight architecture and the project VirtuWind-specific SDN Controller solution.
- We analyze existing wireless communication technologies that could be used for communication with the IoT devices in SEMIoTICS use cases requiring radio access network.

The SEMIoTICS SDN Controller (SSC) is the centralized highly-available network control entity responsible of establishing isolated and QoS-enabling network services in both physical and virtual network topologies (i.e., in site-local and backend networks). To this end, SSC is enabled with an interface to parse SPDI pattern instance requests and is capable of evaluating and transforming the requests into network-specific configurations. To isolate the accessibility between end-points belonging to different tenants, SSC is capable of establishing Virtual Tenant Networks, i.e., a virtualized topology on top of the shared physical substrate.

1.1 SEMIoTICS approach – Networking Aspects

In the SEMIoTICS approach, network operators and end-points, via architectural patterns or directly via the controller's northbound interface, specify the required properties, derive network-related patterns or poll the network for statistics through abstracted and simplified northbound interfaces (NBIs) of the SSC. The SSC abstracts away the details of network configuration and the corresponding interface details by internalizing the decision making and enforcing the computed decisions upon network devices without user involvement in the process, thus supporting the semi-autonomous operation of the SEMIoTICS framework.

For the specified network-related patterns, the SSC executes the appropriate admission control and routing algorithms and automatically takes appropriate low-level actions to enforce the service requirements in the network. The SSC serves each communication service by manipulating the paths and allocating resources associated per service based on an abstract model of the network and its offered resources. The resulting configuration is then mapped to the realization method offered by each node, that allows for configuration of a forwarding entry per service, including the associated resources.

The higher-level network management systems thus gain access to exposed functions (e.g. rule specification, overlay and tenant configuration) and monitoring APIs without needing to internally implement the technology-specific network logic - device-compatible southbound interfacing. Thus, the burden of network configuration is moved from the management application to a centralized network controller.

The southbound interface technology plugins allow for interface-specific enforcement of technology-agnostic network functions. This is achieved by mapping of abstracted and generally-valid configurations to low-level device- and technology-specific configurations. Thus, an administrator polling the forwarding database of a network device need not require knowing the exact technology at hand (e.g. a FIB status or OpenFlow [1] flow table database) in order to fetch the needed information.

Furthermore, the SSC is able to execute its core functions, e.g. topology database population, constrained path computation etc. in a generic manner, and map the results to any controlled network device using the corresponding technology plugin.

In case of an OpenFlow device, the OpenFlow technology plugin takes care of translation of generic controller computation results to a configuration compliant with the standardized OpenFlow data model. In addition, the non-OpenFlow data objects can be configured using a single flexible management protocol (e.g. NETCONF [2]) thus supporting standardized and extended information models, modeled using the IETF YANG language (e.g. [3]). Since not all switch features may have a NETCONF/YANG Module definition in near future, it may be necessary to also leverage the support for SNMP and MIB definitions in the SSC realization for SEMIoTICS use cases.

1.2 Logical link with WP2 beneficiaries – T2.3

The use case definitions in D2.2, as well as the extracted use case requirements described in D2.3 affect the WP3 tasks dealing with SDN and NFV, especially the ones dealing with networking requirements. Namely, these include:

- Network level requirements at the control plane level.
- Network level requirements at the data plane level.
- Domain specific networking requirements.

Thus, they provide an input for definition of controller functions in T3.1. We reiterate the networking-related requirements in Section 2. Furthermore, this deliverable provides the architectural input for the networking-related components described in D2.4, with each of the presented SDN Controller components being directly mapped to the architectural components of the SDN/NFV layer.

1.3 Logical link with WP3 beneficiaries – T3.3, T3.4

Task 3.2 discusses the role of the NFV-related management infrastructure and its interaction with the SDN Controller. Its output, i.e. deliverable D3.2, details the requirements of the exchange of Service Function Chains-related information between the MANO and the Controller, for the purpose of enabling the dynamic interconnection of the VIM-instantiated services at network layer at runtime.

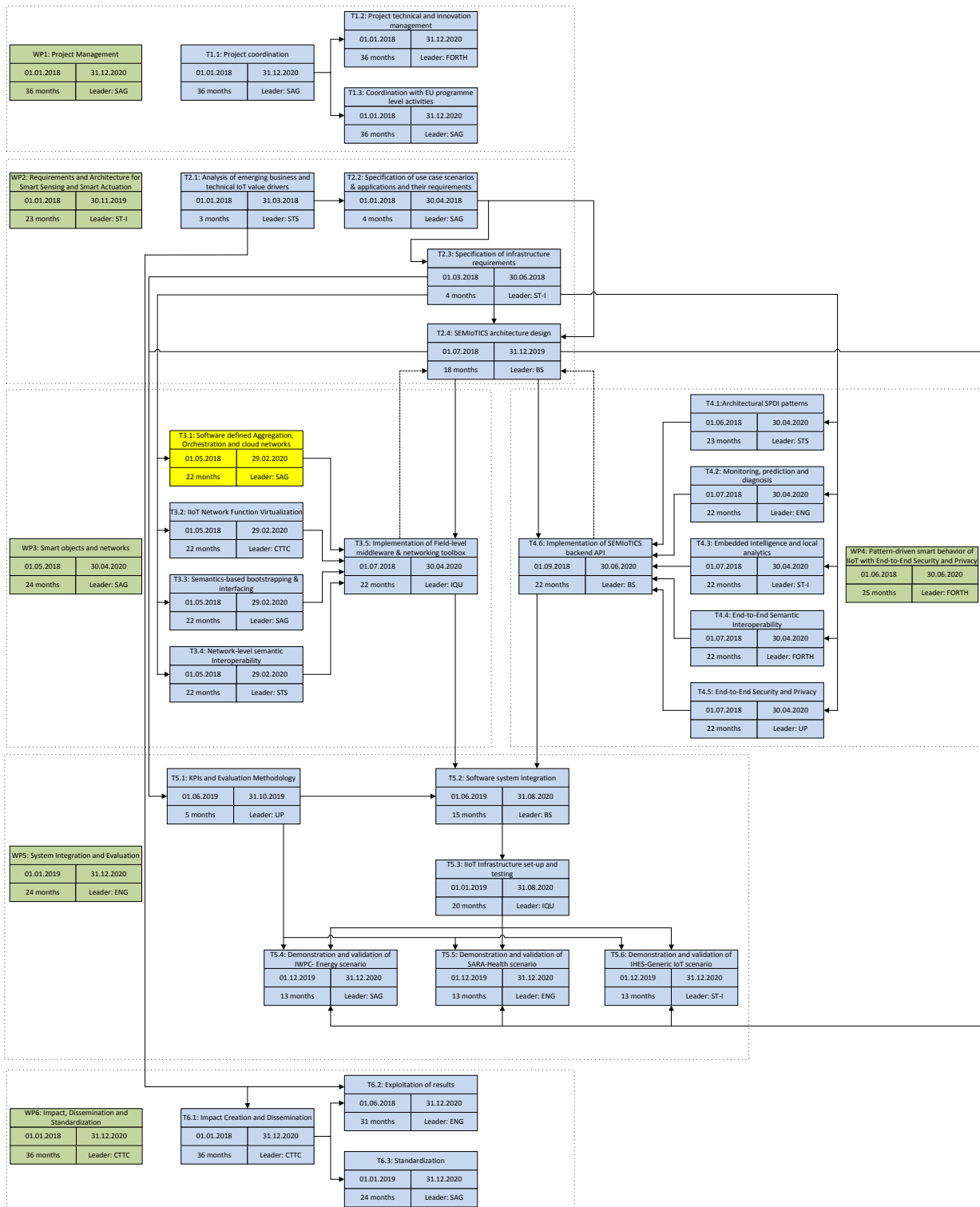
Task 3.4 and the corresponding deliverable (D3.4) defines the initial design and specification of the network programming interfaces that enable the development, optimization and adaptation properties required for the SEMIoTICS framework to support the deployment of network services from all SEMIoTICS layers and its seamless interaction with IoT Applications, as specified by SPDI patterns. T3.3 and T3.4 furthermore detail the application relations, i.e., the field and cross-domain connectivity (IIoT Gateway to Backend) relevant for service profiling in the SSC, and the associated semantic descriptions facilitating interoperability.

Task 3.5 is an overarching integration/implementation task that aims to describe the integration path of the SSC in the generic SEMIoTICS middleware, as well as its implementation, comprising the networking toolkit.

1.4 Logical link with WP4 beneficiaries – T4.1

WP4 tackles the pattern-related approach examined by the project. This includes, among others, the development of patterns for orchestration of smart objects with guaranteed properties. Apart from security, privacy, dependability and interoperability-related (SPDI) patterns, we have observed the need for another category describing the connectivity-related pattern definition. Connectivity patterns are needed to express the need for enabling basic and fault-tolerant application relationship. While connectivity-related patterns will be defined in T3.4, T4.1 presents the according pattern language and syntax, as well as the representative pattern definitions. WP4 will implement the Pattern Orchestrator component, that directly interacts with the SSC (and more specifically, the pattern engine component of the SSC) for the purpose of enabling the network connectivity between the specified application end-points.

1.5 PERT chart of SEMIoTICS



Please note that the PERT chart is kept on task level for better readability.

2 SEMIoTICS NETWORKING INFRASTRUCTURE REQUIREMENTS

The following subsections summarize the SDN-related requirements as defined in T2.3. The requirements can be abstracted into the main set of features of our controller:

- The SSC should enable flexible end-to-end QoS-enabled connectivity at any level of the infrastructure;
- The SSC should provide for a scalable control plane operation (i.e., the SSC should provide support for a large number of end-devices);
- Fault-tolerant SSC operation (including the defense mechanisms for enabling Byzantine Fault Tolerance) – the controller must be resistant against availability issues and Byzantine attacks for arbitrary amount of failures F ;
- The SSC must abstract away the details of network control using the means of a northbound interface;
- The SSC must enable a secure access to all exposed interfaces of the controller;
- The SSC should support OpenFlow 1.3-based interactions with either virtualized or physical data-plane components;
- The SSC should expose an interface to the external VIM for the purpose of SFC specification and the corresponding functions for enforcement of the same;
- The SSC should provide for an automated establishment of basic infrastructure services requiring network connectivity (initial bootstrapping).

More details on the above are provided in the Tables below, aggregated as Control Plane, Data Plane and Domain-specific requirements. The Reference column links to the section providing more details on the method to handling the described requirement - often a component introduced in the SDN controller necessary to provide for the required functionality.

2.1 Control Plane Requirements

The following table portrays the control plane requirements to be fulfilled mainly by the SDN controller solution.

TABLE 1: NETWORK CONTROL PLANE REQUIREMENTS

Req-ID	Functional	Description	Req. level	Addressed in / Addressed by component
R.GP.1	Yes	End-to-end connectivity between the heterogeneous IoT devices (at the field level) and the heterogeneous IoT Platforms (at the backend cloud level)	MUST	Section 3.1.1 Enabling components: Pattern Engine, VTN Manager, Path Manager, Resource Manager, SFC Manager, SDN Data-Plane Devices

R.GP.2	Yes	Scalable infrastructure due to the fast-paced growth of IoT devices	MUST	Section 3.1.1.1 Enabling components: SDN Data-Plane Devices, overall SSC solution
R.GP.3	Yes	High adaptation capability to accommodate different QoS connectivity needs (e.g. low latency, reliable communication)	MUST	Section 4.4 Enabling components: Pattern Engine, VTN Manager, Path Manager , Resource Manager
R.GP.4	Yes	Detection of events requiring a QoS change and triggering network reconfiguration need by SPDI pattern	MUST	Section 4.2.1 Enabling components: Pattern Engine
R.GP.5	Yes	Interaction between SDN controller and IoT backend cloud through a dedicated interface (called northbound software interface)	MUST	Section 4.2.1 Enabling components: Pattern Engine
R.GP.6	Yes	Interaction between SDN controller and network nodes (e.g. switches, routers or IoT Gateways) through dedicated interface (called southbound software interface)	MUST	Section 4.5 Enabling components: Resource Manager
R.GP.7	Yes	SDN controller giving feedback for a future generation of SPDI patterns to avoid using the same pattern in case of failure	MUST	Section 4.2.1 Enabling components: Pattern Engine
R.S.2	Yes	Authentication and authorization of the stakeholders MUST be enforced by the Network controller, e.g. through access and role-based lists for different levels of function granularities (overlay, customized access to service, QoS manipulation, etc.)	MUST	Section 4.6 Enabling components: Security Manager

R.S.7	Yes	The negotiation interface of the SDN Controller SHALL be secure against network-based attacks	SHALL	Section 4.6 Enabling components: Security Manager
R.NL.8 / R.BC.12	Yes	The VIM and Virtual Network frameworks must support Interfaces that enable VM tenant networking	MUST	Section 4.3.1 Enabling components: Virtual Tenant Network Manager
R.NL.9 / R.BC.13	Yes	Interface between the VIM and the SDN controller to allow VTN	MUST	Section 4.3.1 Enabling components: Virtual Tenant Network Manager
R.NL.1	Yes	Controller Node requirement: At least 6 CPU cores and 32 GB RAM	SHOULD	Trivial hardware requirement (i.e., not specifically discussed further in text).
R.NL.2	Yes	Controller Node requirement: At least 2 Network interfaces	SHOULD	Trivial hardware requirement (i.e., not specifically discussed further in text).
R.NL.3	Yes	Controller Node Requirement: Linux OS	MUST	Trivial software platform requirement (i.e., not specifically discussed further in text).
R.NL.4	Yes	Controller Node Requirement: Solid State Disk (SSD) of at least 1 TB	SHOULD	Trivial hardware requirement (i.e., not specifically discussed further in text).

				further in text).
--	--	--	--	-------------------

2.2 Data Plane Requirements

The following table portrays the data plane requirement to be fulfilled by the SDN switches.

TABLE 2: NETWORK DATA PLANE REQUIREMENTS

Req-ID	Functional	Description	Req. level	Addressed in / Addressed by component
R.NL.7 / R.BC.10	Yes	Virtual Physical SDN Switch requirement: Support for OpenFlow v1.3 protocol or greater	SHOULD	Section 4.5 Enabling components: Trivial software platform requirement for switch implementation. OpenFlow v1.3 supported by Resource Manager.

2.3 Use Case-Specific Scenario Requirements

The following table portrays the use case-specific requirements to be fulfilled by the SDN components.

TABLE 3: DOMAIN-SPECIFIC SCENARIO REQUIREMENTS

Req-ID	Functional	Description	Req. level	Addressed in / Addressed by component
R.UC1.1	Yes	Automatic establishment of networking setup MUST be performed to establish end-to-end connectivity between different stakeholders	MUST	Section 3.1.1 Enabling components: Pattern Engine, VTN Manager, Path Manager, Resource Manager, SFC Manager, SDN Data-Plane Devices
R.UC1.3	Yes	There MUST be enabled the definition of network QoS on application-level and automated translation into SDN controller configurations.	MUST	Section 4.4 Enabling components: Pattern Engine, Path Manager, Resource Manager
R.UC1.4	Yes	Network resource isolation MUST be performed for guaranteed Service properties – i.e. reliability, delay and bandwidth constraints.	MUST	Section 4.4 Enabling components: VTN Manager, Path Manager
R.UC1.5	Yes	Fail-over and highly available network management SHALL be	SHALL	Section 4.4 and 4.7

		performed in the face of either controller or data-plane failures.		Enabling components: Path Manager, Clustering Manager
R.UC1.6	Yes	Decisions made by unreliable, i.e. faulty or malicious SDN controllers, SHALL be identified and excluded.	SHALL	Section 4.7.1 Enabling components: Clustering Manager
R.UC1.7	Yes	The operation of the SDN control SHALL be scalable to cater for a massive IoT device integration and large-scale request handling in the SDN controller(s) using a (near-) optimal IoT client – SDN controller assignment procedure.	SHALL	Section 4.7.1 Enabling components: Overall SSC solution is scalable. However, due to Clustering being the largest scalability bottleneck, we focus on enhancing its design in forthcoming Deliverable D3.7.
R.UC2.3	Yes	The SEMIoTICS platform SHOULD guarantee proper connectivity between the various components of the SARA distributed application. The SARA solution is a distributed application not only because it uses different cloud services (e.g. AREAS Cloud services, AI services) from different remote computational nodes, but also because the SARA application logic itself is distributed across various edge nodes (SARA Hubs). The following diagram shows the components of the SARA system:	SHOULD	Section 3.1.1 Enabling components: Pattern Engine, VTN Manager, Path Manager, Resource Manager, SFC Manager, SDN Data-Plane Devices
R.UC2.15	No	The SEMIoTICS platform SHOULD provide low latency connectivity between the SARA hubs and cloud services (i.e. AREAS cloud services and AI services) to allow offloading of near real-time computation intensive tasks to the cloud. Examples include: • the robotic assistant (RA) employing AI services to analyse Patient's speech (audio) and body language (video) to identify significant events – e.g. "Patient requests an escort", "Patient asks where his glasses are" • the robotic rollator (RR) exploiting AI Services to analyse Patient's gait and posture to identify significant events – e.g. "Patient has fallen".	SHOULD	Section 4.4 Enabling components: Path Manager

		<ul style="list-style-type: none"> • mobile robotic Devices (RA/RR) exploiting cloud resources for simultaneous localization and mapping (SLAM) <p>Therefore, SARA hubs need to send with minimal delay:</p> <ul style="list-style-type: none"> • raw range data (e.g. from Lidar sensors) to identify proximal objects/objects, • real-time audio stream for speech analysis, • and real-time raw video stream (object/people recognition, gesture recognition, posture analysis). 		
R.UC2.17	No	The SEMIoTICS connectivity SHOULD support real time exchange of raw sensor data among sensors/actuators and SARA Hubs.	SHOULD	<p>Section 4.4</p> <p>Enabling components: Path Manager</p>

3 SEMIoTICS NETWORK CONTROLLER ARCHITECTURE

The architecture proposed by SEMIoTICS relies on a set of virtualized services that allow the secure and reliable transport of sensor data from field devices towards their final destination for processing or storage. Such final destinations are subject to applications' requirements, e.g.: bounded latency; are satisfied by "smart" path planning and embedding of network paths so that the application requirements are fulfilled. The virtualization dimension also includes dynamically and securely instantiating Virtual Network Functions (VNFs) at appropriate locations in the SEMIoTICS architecture (e.g.: at IoT Gateways, network layer, or the backend cloud), so to fulfil the low latency and storage requirements of the application. In light of such a dynamic network environment, the need for a centralized control of the whole network (physical and virtual) is paramount.

Similarly, obliging to industrial applications' requirements in terms of a deterministic upper bound end-to-end latency, high reliability of the application flows, is simplified through the centralization of the network view and serialization of configuration changes decisions. SDN Controller thus must internalize the path finding algorithms and configuration capabilities to serve the flows.

In an SDN framework, the SEMIoTICS SDN Controller (SSC) employs a set of Southbound interfaces (SBI) for configuring network devices (such as NETCONF or OF-CONFIG), and to control such devices' forwarding table (using protocols like OpenFlow). As network devices boot-up, they reach for the controller in order to register the devices for configuration and forwarding table modifications. After this process is finished, the SSC establishes a complete, centralized view/control of the network, whose topology may be visualized employing the SSC's northbound interfaces (usually RESTful APIs).

Physical Network Functions (PNFs) in the SEMIoTICS architecture, such as the IIoT Gateway, as well as the switches and routers at the Field and Network layers, are equipped with compute and storage resources, which, in combination with the Backend-Cloud's resources, are managed and exposed by the Virtual Infrastructure Manager (VIM). These devices and their resources may be used for the spin-off of VNF (like virtual switches, routers, firewalls, load balancers, processing or storage endpoints), and are (virtually) connected together in the form of VNF Forwarding Graphs (VNF-FG) to offer network services. Such connections are achieved by virtual network overlays (or Virtual Tenant Networks (VTN)) built by the SSC on top of the underlying physical network infrastructure. In order to provide a network service, the VIM and the SSC share a common VNF-FG. As the VIM spins-off VNFs (e.g., virtual switches) and specifies how these should be connected together, the SSC modifies the forwarding table of the appropriate VNFs in order to ensure secure and reliable communication of the whole VNF-FG.

The remainder of the section details the role of the SSC in the SEMIoTICS architecture, as well as its functionality exposed to the NFV Management and Orchestration framework.

3.1 Role of the SDN Controller in SEMIoTICS Architecture

Mainly driven by the explosion of the cloud technologies, networking is again returning to a more centralized model. For decades, network protocols such as Open Shortest Path First (OSPF), Border Gateway Protocol (BGP), Routing Information Protocol (RIP), or Spanning Tree Protocol (STP) have worked on top of the premise that "no entity has a complete view of the network". This fact significantly stirred the design of networks and protocols.

Software Defined Networking (SDN) seeks to disaggregate the control and data planes¹. In this approach, a central control plane, i.e. SDN Controller, is endowed with a complete view of the network. This attribute

¹ Which previously resided inside network devices. Cross compatibility is ensured by following standardized protocols.

dramatically changes the way networks are now conceived, transitioning to SDN networks composed of a single SDN Controller² and many bare-metal SDN-enabled forwarding devices.

This section describes the role of the SSC in the SEMIoTICS architecture, including its physical location in the SEMIoTICS topology, and how it achieves data plane control through the control plane.

3.1.1 SDN CONTROLLER POSITIONING IN THE SEMIoTICS ARCHITECTURE

The SDN components in the context of SEMIoTICS use-cases include the control and data plane counterparts, comprising:

- The SEMIoTICS SDN Controller (SSC) as the main point of logic computation and data planes rules configuration
- Data Plane components, i.e. physical or virtual switches with an interface exposing APIs for controller-based reconfiguration.

The deployment of the SSC is expected in the Network Layer, i.e., at the site-level backbone. Thus, the SSC manages the ingress and egress connections stemming at field level. The data-plane devices, i.e. the switches are available both at Network and Field Layer, with a particular subset of devices available in either Network Layer only (serving as a gateway for external connections), or both in field and network layer, for setups requiring more than a single Layer 2 domain and thus having higher scalability requirements.

While topology-agnostic, the SSC is expected to operate on a multi-ring network or partial mesh topologies, offering support for disjoint path establishment and thus fault-tolerance for critical services in the face of data-plane failures. A topology depicted in Figure 1 depicts an exemplary fully-meshed network layer, tolerant against both link and switch failures, also highlighting data and control plane connections.

Summarized, the controller manages the ingress and egress connections stemming at field level, targeting both other field level devices as destinations or remote sinks located in cloud / backend layer. By providing the generic and QoS-constrained connectivity, controller addresses the R.GP.1, R.UC1.1 and R.UC2.3 requirements. In the remainder of the document we discuss the individual development plan for functions of the controller, comprising the capabilities necessary to enable the connectivity requirement.

3.1.1.1 ACHIEVING GLOBAL CONTROL OF THE NETWORK FROM A CENTRALISED SDN CONTROLLER

There are two main ways in which an SSC manages the underlying fabric namely, directly or via network overlays [4]. The former consists of the SSC directly communicating with SDN-enabled switches via southbound interfaces (SBI) such as OpenFlow [1], NETCONF [2], OVSDB, OpFlex, and others. These SBIs allow the controller privileged access to the devices' forwarding tables, as well as other device configurations per se. This method is the one used by most popular off-the-shelf controllers i.e.: ODL; and its variation SSC which is to be developed in SEMIoTICS. The latter method uses encapsulation protocols (e.g.: VXLAN, NVGRE, IPSEC) on top of conventional networks³ to build the desired network topologies. Even though it is debatable whether overlays are SDN or not, they are indeed software defined⁴.

² High-Availability (HA) clusters may host a single logical SDN Controller instance with multiple dislocated replicated instances.

³ By conventional it is meant that there exists L2 or L3 connectivity among components of the network.

⁴ In fact, Virtualized Infrastructure Managers (VIM, e.g.: OpenStack) employ an SDN Controller and overlays to provide private tenant networks.

Backend/Cloud

Network

Field

Data plane
 Control plane

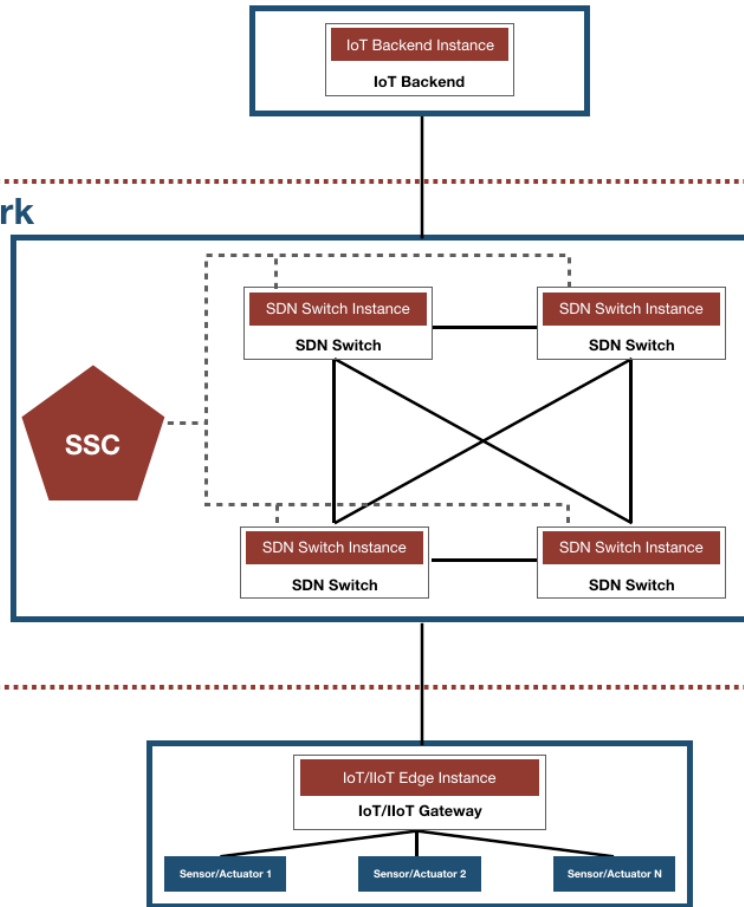


FIGURE 1: INSTANCE OF THE SEMIoTICS ARCHITECTURE DEPLOYMENT VIEW

Regardless of the SDN Controller vendor, there are core features that must be supported by an industrial variant of an SDN Controller. A non-exhaustive description of these is provided below and [4]:

- Data plane programmability: change the way flows are forwarded, apply filters, or dynamically changing packet headers. Via northbound interfaces control information concentrated at the SDN Controller could be accessed/changed by SDN Applications. Such applications may then use such information to apply templates that effectively change the network configuration, e.g.: satisfy bandwidth constraints, QoS enforcement, forward through least expensive paths, and so on.
- Southbound protocol support: the most widely used southbound protocol is OpenFlow. An SDN Controller should be able to interact with OpenFlow agents (or other southbound protocol) in SDN-enabled forwarding devices in order to control the different actions to be performed while forwarding.
 - As mentioned previously, OpenFlow is not the only protocol supported by ODL as SBI. NETCONF, OVSDB and SNMP are some of the other alternatives [5, 4].

- **External API support:** to ensure it could be used within various cloud orchestration environments, e.g.: OpenStack. Through well specified APIs, network policies exchange is realized, allowing the control of the networking resources of a virtual infrastructure.
- **Centralization:** allowing administrators a complete view of the network. It should also support discovery protocols so new network devices are registered and bootstrapped if necessary.
- **Performance:** as network devices rely on the SDN Controller for handling incoming flows not contained in the current flow table, controllers should ensure such requests are handled as fast as possible, otherwise the SDN Controller is prone to become a network bottleneck.
- **High Availability and Scalability:** the ability to work as a cluster allows an SDN Controller to expand its performance and availability by adding more controller nodes and load balancers.
- **Security:** as the functioning of the network depends on the SDN Controller, it should be capable of authenticating/authorizing members of the network while performing intrusion detection and prevention, e.g.: secure southbound channels, encryption.

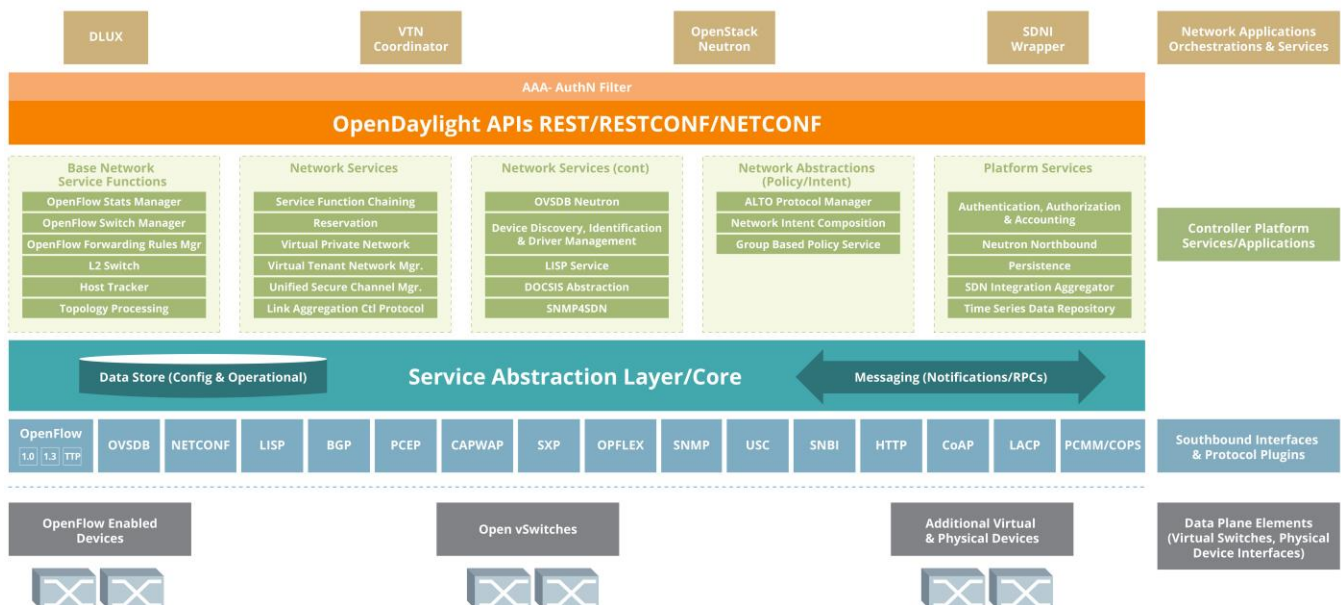


FIGURE 2: OPENDAYLIGHT SDN CONTROLLER PLATFORM [5]

Modern SDN Controllers, such as OpenDaylight (ODL), are built following a modular architecture (see Figure 2). Each independent module could leverage services exposed by other modules in order to provide composite functionality. Focusing on the configuration of the data plane, the following details the most relevant southbound protocol modules implemented by ODL (or Southbound Interfaces and Protocol Plugins, as referred to at the bottom of Figure 2), which must also exist in SEMIoTICS SDN Controller:

- **NETCONF:** ODL supports communication with NETCONF servers at SDN-enabled forwarding devices. Additionally, this module may work as a NETCONF server itself in order to expose Control plane information to external entities, e.g.: SDN applications.

- **OpenFlow:** Enables the SDN Controller to configure the flow table of OpenFlow enabled SDN devices remotely. OpenFlow Library allows the SDN Controller to listen for OpenFlow messages (spawns a daemon), as well of supporting multiple OpenFlow versions.
- **OVSDB:** It is used for managing⁵ OVS-enabled switches (physical or virtual). OVS [6] is an opensource software that implements virtual switching that is interoperable with almost all popular hypervisors. OVS uses OpenFlow to perform message forwarding in the control plane for both virtual and physical ports. OpenDaylight's OVSDB southbound plugin manages OVS devices supporting OVSDB schema and the OVSDB management protocol,
- **SNMP4SDN:** This module allows ODL to interact with SNMP-enabled switches. It uses SNMP methods for writing forwarding as well as configuration information into the devices.

Previously mentioned SBI establish communication through the control plane (see Figure 1). One module or a combination of these are required for effective SDN-enabled device configuration in SEMIoTICS. In the remainder of the document, we assume the OpenFlow-connector as the only fully required protocol in SEMIoTICS use cases.

Modern SDN controllers (i.e., ODL) are equipped with a east-westbound connectivity, allowing for for multi-controller-based interaction when establishing connections across multiple administrative domains. By separating network control into multiple administrative domains, higher scalability of the overall system can be supported, due to individual controllers experiencing a lower load on average, than when controlling all devices in a single domain. By extending an OpenDaylight based platform, SSC will remain compatible with the east-westbound implementations and thus enable the scalability requirement R.GP.2.

3.1.1.2 ACTING UPON NETWORK STATE AND SERVING NETWORK STATE INFORMATION

To showcase the internal state handling and algorithms internalized in the centralized controller, we list below the scenario of pre-configuration, network bootstrapping and a trivial connectivity pattern evaluation and its enforcement; these are also depicted in the Figure 3.

Startup Phase: The user pre-configures the controller and forwarding devices, i.e., using a configuration file, with the set of IP addresses and executes the boot up procedure. Additional, security related actions are executed by the user, including the roll-out of public certificates to switches associated with the allowed controller instances. Optionally, the devices are capable of executing the DHCP/DHCPv6/SLAAC bindings and autonomously receiving and applying a discovered IP address, instead of the manual configuration.

Default Bootstrapping Phase: SSC must discover the devices in its network, i.e., the switches using the OpenFlow discovery protocol, SNMP LLDP MIB crawling or similar bootstrapping mechanism. Following the discovery and establishment of control session, the controller listens for incoming packets from end-devices (hosts), e.g., the IIoT Gateway, SCADA application etc. and updates its host database with the corresponding attachment points behind which the hosts are located. The SSC then proceeds to install the required flow rules, so to enable basic infrastructural services, i.e., a network connection between the IIoT Gateway and backend, in order to provide for a possibility of IIoT Gateway to report its status and the capabilities of its field devices (i.e., sensors and actuators) in the Thing Directory in the Backend (please refer to Deliverable D3.3).

Runtime Phase: SSC is capable of accepting pattern rules specified as a set of properties, encompassed in invariants, that describe the intent which is to be fulfilled by the underlying data substrate. The SSC accordingly processes the patterns posted as REST requests, in Drools format, at its northbound interface and validates the viability of its enforcement in the internal modules. For example, a controller might be requested with a pattern necessitating and end-to-end flow embedding in a higher-than-best-effort service class, i.e., with specific bandwidth and delay requirements.

⁵ View, create, modify, and delete.

To adhere to the rule specification, the controller internally evaluates the topology state for a path that would fulfill the named criteria, i.e., using a combination of routing algorithms, designed to consider a set of constraints (including delay and bandwidth) [7]. If the connectivity is possible given the current amount of resources, the controller propagates the enforcement request for individual flow rules to the lower-level southbound interface (i.e., the OpenFlow plugin). The OpenFlow plugin is then in charge of sending the according flow modification messages to the corresponding data plane switches.

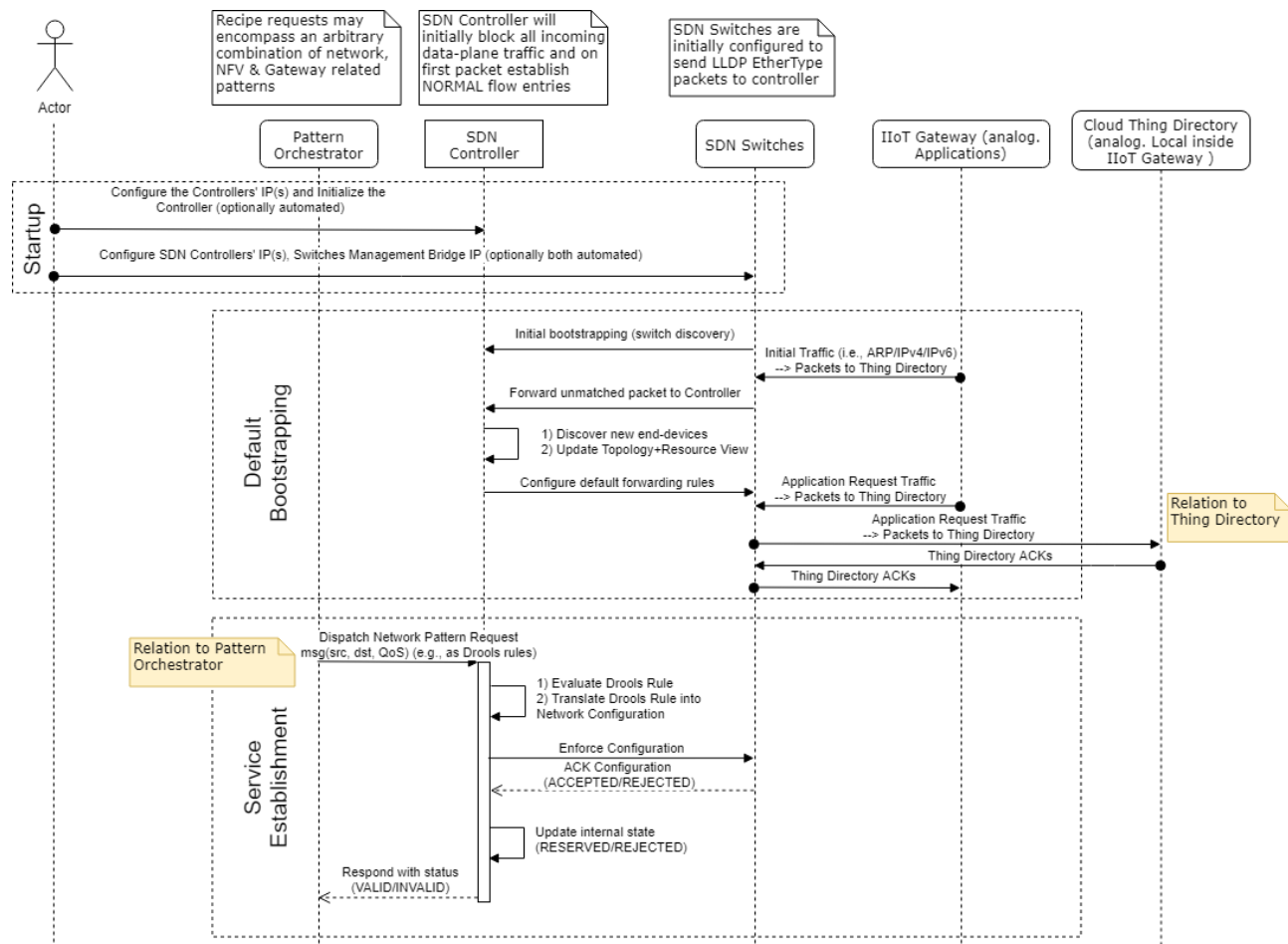


FIGURE 3: THE ABSTRACTED WORKFLOW OF BASIC SYSTEM INITIALIZATION, BOOTSTRAPPING AND PATTERN ENFORCEMENT IN THE SEMIoTICS SDN CONTROLLER

3.1.2 PATTERN ENFORCEMENT USING A LOCAL PATTERN ENGINE

One of the examined capabilities of Pattern Enforcement could be used to find or verify suitable paths in order to pre-plan and reserve paths with respect to the SPDI properties. After evaluating the available network components and flows, SPDI patterns could be executed to realize the verification process where new paths can be inserted (i.e. add the required flow rules), removed (i.e. delete flow rules) or modified (post flow rules) in the controller and consequently in the programmable switches. The patterns in the controller can be used not only for the verification of network paths but also at runtime i.e., following a network link failure or when a SPDI property is not guaranteed to reconstruct or restore required SPDI properties when such properties have been violated.

3.1.3 NETWORK OVERLAYS SUPPORTING NFV: SDN AND NFV FROM THE SDN CONTROLLER'S PERSPECTIVE

VNFs are seen as virtual counterparts of physical switches, routers, firewalls, load balancers, IDS, IPS, and so on. As opposed to conventional Physical Network Functions (PNF), VNFs can potentially bring significant reductions in the total cost of the infrastructure, mainly because from a set of fairly generic hardware (i.e.: data center) many VNF can be spawned, used and terminated; effectively creating customized network services when needed, and freeing their used resources when terminated.

As hinted previously, infrastructure flexibility is one of the most relevant features provided by Network Functions Virtualization (NFV) [8] (either at the compute, storage or networking level), and network overlays play a crucial role in network virtualization. Through overlays, the SDN Controller is able to create different network topologies for each project, dubbed VTNs, which are effectively isolated from each other.

Specifically, in purely virtualized networking environments (i.e., in Backend layer networking), we rely on instantiation of network services, including the VTNs, as well as the actual end-point VNFs using a single interface, i.e. a single Network Service Description (NSd files). This is in contrast to field layers of infrastructure, that may have to rely on controlling physical SDN infrastructure, as well as establishing communication flows between non-virtualized end-points, such as programmable logical controllers and field sensors. There, specification of virtual tenant networks is explicit, at the level of VTN Manager SSC's component.

ODL (the reference SDN Controller for SEMIoTICS) is equipped with a VTN module for interfacing with Virtualized Infrastructure Managers (VIM) such as OpenStack [5]. The ODL's VTN module is a policy manager that registers any tenant resource in the VIM via ODL's ML2 plugin⁶, so any tenant configuration modification at the VIM is reflected in ODL, too. That is, by analyzing the information gathered for each tenant (network topologies, VNFs, MAC, IPv4 addresses, and so forth), VTN is able to replicate such policy⁷ using the VIM's exposed networking APIs and ODL's SBIs.

Service Function Chains (SFC) are a particular case for policy enforcement. SFC specifies the order of PNFs/VNFs packets should flow through in order to provide a Network Service (NS). ODL is equipped with an SFC module which feeds from control information (including VTN received from the VIM) and builds flow policies to handle the sequence of VNFs a specific flow should go through. It includes the following components:

- Classifier: selects flows for traversing a specific SFC based on a flow match policy.
- Service chain: refers to the list of devices the matched flow should traverse.
- Service path: the actual VNF instances traversed.
- Service overlay: a topology created to visualize the service path.
- Metadata: information passed between service functions.

The SFC is also able to receive SFC designs/recipes via an exposed northbound API.

⁶ The ML2 plugin was created for ODL-OpenStack integration. It passes all OpenStack's Neutron API calls to ODL's VTN manager via REST calls [4].

⁷ I.e. What nodes should be able to communicate with which ones.

3.1.3.1 ACHIEVING DATA TRANSPORT LEVERAGING VIRTUAL NETWORK FUNCTIONS

NFV and SDN are the key enablers of interconnected virtual functions. The virtualized application elements for each UC are conceived as VNFs instantiable by the Virtual Infrastructure Manager (VIM) on top of a NFV infrastructure. Networking for both these virtual and physical application functions is provided in the form of overlays handled by the SDN Controller, while VNF instantiation and lifecycle management⁸ is handled by the NFV Orchestrator. **Table 4 provides an overview of requirements relying specifically on interaction or depending on SDN/NFV function calls and discusses our method of handling the requirement.**

TABLE 4: UC2 NETWORK REQUIREMENTS (SARA HEALTHCARE)

Req-ID	General Description	Specific NFV/SDN Requirements	Reference / Enabled by
R.UC2.1	The SEMIoTICS platform SHOULD support time- and safety-critical requirements by allowing SARA application logic to be deployed on resource-constrained edge gateways (e.g. smartphones, vehicles, mobile robots). SEMIoTICS platform functionalities SHOULD be locally available even in case of failure of communication with the SEMIoTICS cloud nodes.	IIoT devices MUST support virtualization and MUST be reachable by NFV MANO components. Furthermore, the topology supporting the UC MUST be expressed in form of a NFV MANO-compatible network service via the corresponding descriptors.	While the first requirement on generic connectivity is discussed throughout this document, the fulfilment and the format of the network service specification (using NSd – networking service descriptors) is detailed in Deliverable D3.2.
R.UC2.2	The SEMIoTICS platform SHOULD support the SARA solution to manage the trade-off between different requirements (e.g. reliability, power consumption, latency, fault-tolerance) by allowing both SARA application logic and platform features to be distributed over a cluster of gateways (SARA Hubs).	VNF network overlays MUST exist among SARA Hubs.	The fulfilment of this requirement is enabled by Virtual Tenant Network instantiation, as described in Section 4.3.1 and to more extent in Deliverable D3.2. Enabling components: VTN Manager.
R.UC2.3	The SEMIoTICS platform SHOULD guarantee proper connectivity between the various components of the SARA distributed application.	VNF network overlays MUST exist among SARA Hubs. Achieving the network overlays is to be done by use of Virtual Tenant Network instantiation using the VTN	The fulfilment of this requirement is enabled by Virtual Tenant Network instantiation, as described in Section 4.3.1.

⁸ Including scaling in or out.

	<p>The SARA solution is a distributed application not only because it uses different cloud services (e.g. AREAS Cloud services, AI services) from different remote computational nodes, but also because the SARA application logic itself is distributed across various edge nodes (SARA Hubs).</p>	<p>Manager component of the SDN Controller.</p>	<p>Enabling components: VTN Manager.</p>
R.UC2.5	<p>The SEMIoTICS platform should allow the SARA solution to discover the IoT devices that are registered in the system. IoT devices deployed by the SARA solution are expected to register themselves into the system using various standard protocols (e.g. LwM2M, MQTT, Bluetooth LE, ZigBee, etc.).</p>	<p>SARA components handling the registration of devices, and supported IoT devices MUST be located in the same VTN.</p>	<p>The fulfilment of this requirement is enabled by allocating the SARA devices in a single Virtual Tenant Network.</p> <p>Enabling components: VTN Manager.</p>
R.UC2.15	<p>The SEMIoTICS platform SHOULD provide low latency connectivity between the SARA hubs and cloud services (i.e. AREAS cloud services and AI services) to allow offloading of near real-time computation intensive tasks to the cloud. Examples include:</p> <ul style="list-style-type: none"> the robotic assistant (RA) employing AI services to analyse Patient's speech (audio) and body language (video) to identify significant events – e.g. "Patient requests an escort", "Patient asks where his glasses are" the robotic rollator (RR) exploiting AI 	<p>The SDN Controller SHOULD be able to modify the path followed by packets from this UC VTN if necessary. Such modifications seek to reduce end-to-end delay and comply with UC constraints.</p>	<p>The fulfilment of this requirement is enabled by path computation that is aware of QoS constraints on the resulting identified paths, as discussed in more detail in Section 4.4.</p> <p>Enabling components: Path Manager, SFC Manager.</p>

	<p>Services to analyse Patient's gait and posture to identify significant events – e.g. "Patient has fallen".</p> <ul style="list-style-type: none"> mobile robotic Devices (RA/RR) exploiting cloud resources for simultaneous localization and mapping (SLAM) <p>Therefore, SARA hubs need to send with minimal delay:</p> <ul style="list-style-type: none"> raw range data (e.g. from Lidar sensors) to identify proximal objects/objects, real-time audio stream for speech analysis, and real-time raw video stream (object/people recognition, gesture recognition, posture analysis). 		
--	---	--	--

TABLE 5 UC3 NETWORK REQUIREMENTS (AI SENSOR FOR EVENT DETECTION)

Req-ID	Description	Specific NFV/SDN Requirements	Reference / Enabled by
R.UC3.6	MCU into IoT Sensing unit should be associated with a high-level tool for automatic generation of optimized code to support pre trained neural networks.	There MUST be network connectivity among components (UC components may reside in the same VTN). Such components may be VNFs on top of IIoT Gateways, or PNFs.	The fulfilment of this requirement is granted by providing generic connectivity property by the networking infrastructure, as discussed in Section 3.1.
R.UC3.7	MCU IoT Sensing unit shall be able to send change detection and signal local changes / anomalies to IoT Sensing gateway.		<p>Enabling components: Pattern Engine, VTN Manager, Path Manager, Resource Manager, SFC Manager, SDN Data-Plane Devices</p>

R.UC3.14	The specific M2M protocol adopted on UC3 is based on MQTT. A MQTT broker service will be available to dispatch messages between the coordinating Sensing gateway and its associated Sensing units.	An MQTT broker MUST exist within this VTN in order to support MQTT.	The fulfilment of this requirement is enabled by interconnecting an MQTT broker with other sensing devices in a single Virtual Tenant Network. Enabling components: VTN Manager
----------	--	--	---

SDN/NFV in context of Backend Layer Networking: Deployment of interactions between NFV/SDN in the backend relies on Network Service descriptors (NSd⁹) [9]. NSd descriptors are static text files that describe the desired network topology, VM images to use, resources to be used, scaling factors, event triggers, KPIs, physical location of the instantiation, among others. Later on, NSd are onboarded onto the NFV Orchestrator (NFVO), which in turn translates them into API calls to the VIM in order to instantiate a copy of said virtual network service.

As opposed to a traditional PNF network deployment, typical for the flexibility provided by NFV/SDN admits fast deployment modifications. This attribute allows the Network Service (or UC in this context) to scale in/out¹⁰ according to the load and subject to NFVI resources availability, as well as constant monitoring and modification of forwarding paths according to UC constraints.

3.1.4 MANAGING VIM NETWORKS WITH AN EXTERNAL SDN CONTROLLER

VIMs often incorporate their own SDN Controller, e.g.: Neutron in OpenStack; but there are a number of benefits associated with delegating the management of the network to another software. These range from fault tolerance through service isolation, or simply additional functionality.

Most popular VIMs are composed of different isolated projects that work together. For instance, OpenStack Nova [10] handles compute, Cinder [11] storage, Neutron [12] networking, and so on. It is also possible to keep each project on a separate physical node (or cluster of nodes) in order to provide resilience against network or resource outage/saturation.

The SEMIoTICS SDN Controller falls within this model. That is, a stand-alone external SDN Controller which uses the ML2 northbound plugin [13] in order to relay/retrieve network information to/from VIMs [4]. Furthermore, it uses the OpenFlow and OVSDb southbound plugins to act upon the virtual network devices (e.g.: OVS) running on compute hosts within the VIM domain, as well as the underlying physical network connecting compute nodes to the network. This characteristic allows for a central network policy enforcement entity, where virtual and physical networks may be jointly optimized to provide UCs with the necessary network performance.

3.2 SDN Controller Function Blocks

In the following section we provide a brief overview of the internal architecture of the SEMIoTICS SDN Controller and its differentiation to related state-of-the-art activities, such as the controller developed in the context of the VirtuWind project.

3.2.1 SEMIoTICS SDN CONTROLLER FUNCTION BLOCKS: COMPONENT DIAGRAM

In the network control approach in SEMIoTICS, we leverage the centralized control plane and programmability offered by the SDN paradigm and exploit the flexibility of NFV. Each authorized application or tenant that needs

⁹ Even though NSd are in turn composed by other descriptors (for VNFs, virtual links, among others), here the generic NSd term is preferred.

¹⁰ E.g.: increase or reduce the compute/storage resource of a VM.

connectivity is served one or multiple related communication services. Specification of application recipes ensures the composition of application relations at global level. Connectivity patterns stem from a combination of different connectivity and QoS service requirements, ranging from E2E-delay and bandwidth requirements, to different path protection schemes (e.g. duplication or fast-failover).

As in previous related projects from the industrial domain (i.e., VirtuWind [14]) each request is mapped to a unique network tenant. To ensure isolation when competing for a limited set of network resources, in an industrial network, each tenant must be served a guaranteed pool of resources (e.g., in scope of its Virtual Tenant Network).

Figure 4 depicts the key architecture blocks of our SDN approach, a brief description of which can be found below:

- 1) **Pattern-based northbound interface of the SSC:** Allows the network operator to specify and collect information about enforced patterns. Similarly, it allows for specification of pattern rules and thus the orchestration of network from perspective of an automated service, such as the Pattern Orchestrator (please see D4.1) or pattern-enabled end-devices.
- 2) **The set of core SSC components:** They enable the decision-making related to pattern enforcement, reference storage and pattern eviction at runtime. The core components are furthermore in charge of collecting, maintaining and modifying the network configuration according to the presented set of pattern rules.
- 3) **An exposed interface for NFV Management and Orchestration (MANO) interactions:** MANO components (i.e., the VIM or VNF Manager) may interact with the SEMIoTICS SDN Controller so to interconnect the virtual network functions (VNFs) and thus enable the required service function chains (SFCs) interconnection at network layer;
- 4) **The set of southbound interfaces providing connectivity between the SSC and network devices.**
- 5) **Network Devices:** Which expose the set of standardized interfaces so to allow for enforcement of network configurations (i.e., the packet matching rules composed of filters and actions).

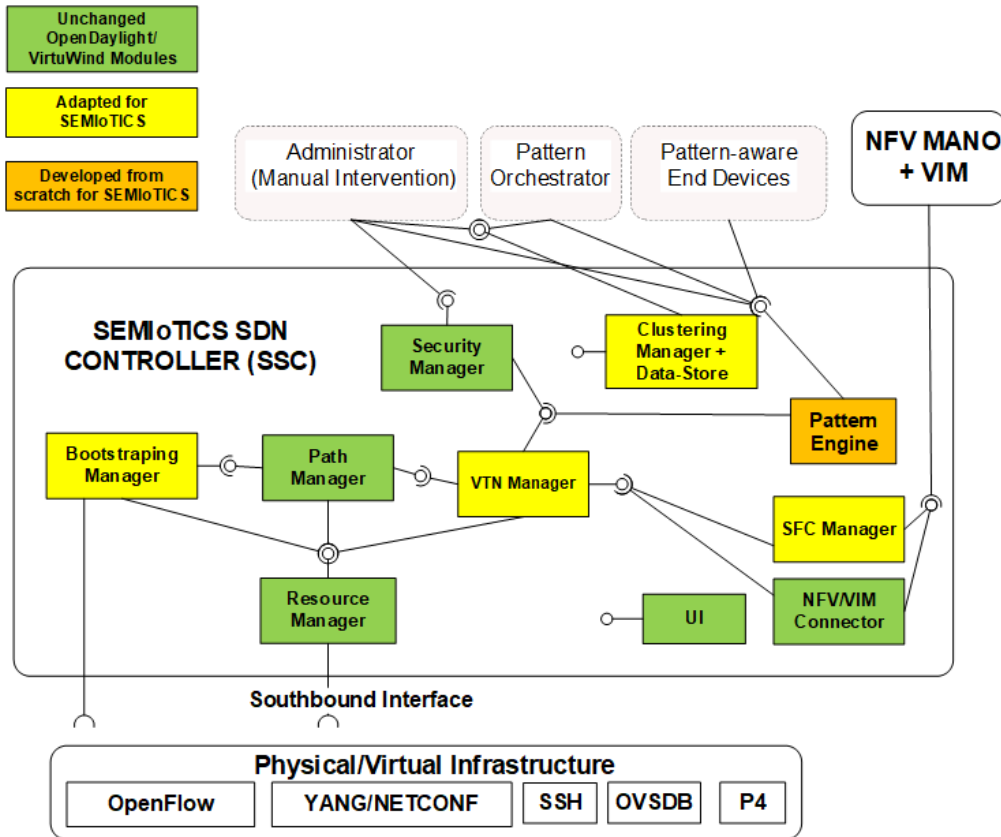


FIGURE 4: SEMIoTICS SDN ARCHITECTURE

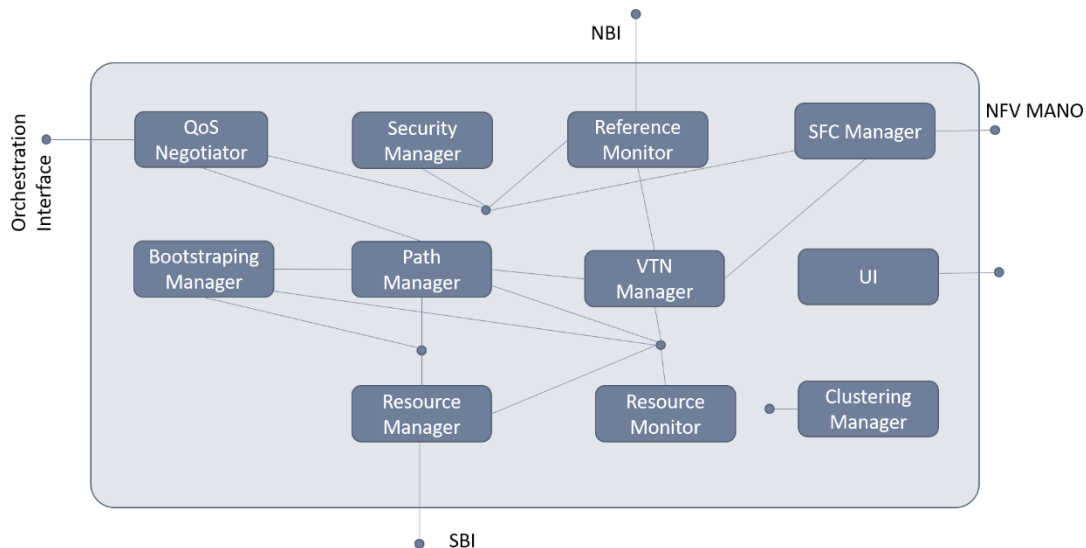


FIGURE 5: COMPONENT OVERVIEW OF THE VIRTUWIND SDN CONTROLLER

3.2.2 COMPATIBILITY WITH STATE-OF-ART SDN CONTROLLER ARCHITECTURES

In the following sub-sections, we discuss the compatibility and extensions of the SEMIoTICS controller architecture with the state-of-art distributions controllers proposed in the ODL and VirtuWind projects.

OpenDaylight and VirtuWind are the only SDN controller platforms fulfilling a larger subset of proposed SEMIoTICS system and networking requirements, hence they were selected for the reference implementation in SEMIoTICS.

3.2.2.1 COMPATIBILITY WITH VIRTUWIND SDN CONTROLLER ARCHITECTURE

As we have previously mentioned, the SSC is responsible for the orchestration of field- and network-level switching devices and is implemented on top of the revised SDN controller ODL, which was published as a result of the VirtuWind project. SEMIoTICS employs many of the VirtuWind's built-in SDN components for the initial implementation. For example, VirtuWind's Path Manager is reused for QoS-based path identification for realizing the end-to-end connectivity between the connecting partners. On the other hand, some off-the-shelf modules from ODL platform, such as its OpenFlow implementation (OpenFlowPlugin) is leveraged to enable interaction with the OpenFlow enabled SDN switches, but also to utilize and extend its data-store implementation and data models for storage of the SDN controller information.

In Figure 5 the components of the VirtuWind SDN controller are presented. By comparing Figure 5 with Figure 4, it can be observed that many of the components in both architectures are common. In both cases, each authorized application or tenant that needs connectivity requests an appropriate VTN, and later issues communication service flow requests to the system. The isolation of tenants is administered using the VTN north-bound interface of the SDN Controller, while the service requirements define a communication service interface as per tenant's intent specification. Each application service is mapped to a unique VTN and a network tenant.

The main difference between the two architectures is that, in SEMIoTICS, the north-bound interface is generic-pattern-based, allowing the network operator to specify and collect information about enforced patterns, while allowing for specification of pattern rules and thus the orchestration of network from the perspective of an automated service, hence the exclusion of Reference Monitor component in our architecture.

On the other hand, due to the QoS-constrained nature of power control systems, the north-bound interface in VirtuWind is specialized in QoS metrics. For instance, the flow requests are a combination of different connectivity and QoS service requirements, ranging from E2E-delay and bandwidth requirements, to different path protection schemes (e.g. duplication or fast-failover).

To that end, instead of a pattern orchestrator, there is a QoS orchestrator that is specialized for setting up a QoS-enabled end-to-end connectivity service via multiple network operator domains. Moreover, a QoS negotiator is responsible for the communication between the SDN Controller and the QoS Orchestrator and the translation of orchestrator's requests to domain-specific actions.

3.2.2.2 COMPATIBILITY WITH OPENDAYLIGHT MODULAR CONTROLLER ARCHITECTURE

OpenDaylight (ODL)¹¹ is an SDN controller that lets the user to programmably manage OpenFlow capable switches. ODL is a large open-source project with a number of features and compatible northbound applications. OpenStack, as described in the previous sections, is a cloud orchestration platform that can work independently without ODL, but with the SDN controller provides the user more programmatic control over the infrastructure; hence, improving the scope for automation.

In SEMIoTICS, one of the advantages of using an ODL-based implementation is the compatibility with OpenFlow, NETCONF/YANG and OVSDB protocols, supported by numerous of network vendors. Similarly, ODL already internalizes particular functions useful for the SSC UCs, including topology discovery and graph population, capability discovery and JAVA-to-REST API bindings (using the YangTools [15] project).

¹¹ <https://www.opendaylight.org>

As a modular framework, ODL allows the developers and the users to:

- Install the protocols and services needed on-demand.
- Combine multiple services and protocols to solve more complex problems when needed.
- Develop custom functionality for extending the existing platform to support the specialized use cases.

The modular development pursued by the ODL project furthermore allows us a relatively straightforward extension, so to include the modules necessary for industrial use-cases, e.g., to enable flexible QoS, pattern-based service instantiation as well as the multi-tenant operation, which are pillars of the SEMIoTICS solution realization.

4 DETAILED SDN CONTROLLER FUNCTION DESCRIPTIONS

4.1 Control / Data Plane Bootstrapping Manager

Implementation Status: To be reused from VirtuWind project and updated afterwards. In SEMIoTICS, the Bootstrapping Manager is updated with extensions to provide for an automated deployment of basic infrastructural network services.

Industrial SDN networks require a highly-available control plane. The control plane may require an in-band or out-of-band control plane realization depending on the exact use case. In-band control plane revolves around reusing the physical SDN data plane to host the control plane flows - i.e., the control traffic exchanged between the SDN Controller and switches shares the same network as the application flows. In contrast to in-band, the out-of-band-control plane relies on exclusive physical links for interconnection of the controller and managed switches. The wind park Use Case 1 assumes an in-band deployment, so to minimize the Capital Expenditures (CAPEX) related to out-of-band cabling requirements. By means of an automated network bootstrapping procedure, the SSC guarantees a robust and resilient control plane configuration at network runtime.

The robustness to *controller failures* is ensured by bootstrapping a multi-controller state replication design as described in Section 4.7.

To handle the *data plane failures*, and their effect on the control plane flows, redundant control flow embedding can be leveraged. While recent works propose slower, restoration-based techniques in industrial scenarios, industrial scenarios typically use 1+1 protection by duplicating controller-to-controller and controller-to-switch TCP-based flows on maximally disjoint paths, thereby ensuring zero packet loss for control flows, at the expense of doubled bandwidth requirements per control flow connection. Since these typically have low bandwidth requirements, we do not consider it a crucial drawback in our approach.

Enabling point-to-point connectivity in OpenDaylight and reference VirtuWind implementations requires manual or scripted specification of end-points to be interconnected by the network flows. This would lead to a large manual effort complexity due to a high number of infrastructural services necessary by SEMIoTICS components in the field and backend layer that require such connectivity. To this end, we will extend the Bootstrapping Manager from VirtuWind project, to support an automated instantiation of network services for infrastructural network flows (e.g., Thing Directory synchronization, IoT Gateway <-> Router <-> Internet flows) as required by the SEMIoTICS use cases to minimize the scenario deployment efforts.

4.2 Pattern Engine

Implementation Status: To be developed from scratch for purposes of SEMIoTICS use cases.

4.2.1 PATTERN ENGINE DESCRIPTION

Pattern Engine (PE) module enables the capability to insert, modify, execute and retract patterns at design or at runtime in the SDN controller. PE can be based on a rule engine which is able to express design patterns as production rules. Enabling reasoning, driven by production rules, appeared to be an efficient way to represent SEMIoTICS patterns. For that reason, a rule engine is required to support backward and forward chaining inference and verification. Drools [16] rule engine appeared to be a suitable solution to support design patterns by applying and extending the Rete algorithm [17]. More specifically, since Drools rule engine is based on Maven, it can support the integration of all required dependencies with the ODL controller, as well as the integration of the entities that interact with the controller to run Drools at design and at runtime. Finally, PE integrates different subcomponents required by the rule engine such as the knowledge base, the core engine and the compiler.

Pattern Engine is able to detect invalid rule configurations by means of component observations – i.e., connectivity-related patterns requiring flow installation will evaluate to False on failure of intermediate network elements (detected by subscribing to network topology events). Thus, the event detection requirement R.GP.4. will be fulfilled by the monitoring component implemented in Pattern Engine. The Pattern Engine additionally exposes a bidirectional interface towards backend, i.e., the Pattern Orchestrator component. On each status change of an active pattern instance, the remote Pattern Orchestrator is notified, so that additional reconfiguration steps can be partaken there. This adheres to the Requirement R.GP.5 and R.GP.7.

The final implementation of the Pattern Engine will be provided in future Deliverable D3.7.

4.2.2 PATTERN SPECIFICATION NBI

To support insertion, modification and deletion of facts and rules in the knowledge base by administrators or users, suitable northbound interfaces (YANG APIs and the respective REST APIs) are implemented. Finally, a number of different YANG interfaces is implemented to interact with the different components including also network components such as switches, service functions and end-hosts, active links and statistics from the controller as required by the pattern rules.

4.3 VTN Manager

Implementation Status: To be reused from VirtuWind project. If modified for the purposes of SEMIoTICS use cases, an according update will be presented in D3.7.

4.3.1 SPECIFICATION OF VIRTUAL TENANT NETWORKS

VTN Manager is a component of the that provides for a multi-tenancy functionality. It should realize logical slices (“virtual tenant networks”) for per-application mapping and enforcement of isolation of the tenant networks in the infrastructure. VTN Manager’s APIs in SEMIoTICS is planned to leverage the logical VTN primitives (VTN definition, vInterface definition, port-mappings, flow-filters and flow-conditions) and exploit YANG/RESTCONF as modelling language and transport protocol. VTN Manager must thus allow for creation of tenant networks and translation of pattern requests into path-request calls to Path Manager in scope of its VTN. VTN Manager stores all resulting data structures containing information about reservations and established VTNs in the centralized data store.

VTN Manager isolates individual tenants of the network at Layer 2 level, i.e., unmapped participants deployed in different tenant networks will be unable to interact with each other. Thus, VTN Manager provides network-level security end-point interactions. The VTNs can be provisioned by means of a YANG-modeled REST interface, thus providing for R.NL.8 and R.NL.9 requirement support.

4.3.2 AN NBI FOR A MULTI-TENANT CAPABLE NETWORK SERVICE ORCHESTRATION

This SEMIoTICS module provides a north-bound interface API (NBI API). It translates the application's request into low-level APIs of underlying components (VTN Manager and Security Manager). It provides RESTCONF RPC calls modelled in YANG. Additionally, it verifies the applications authenticity using credentials issued by Security Manager, also it logs application's requests and responses in the distributed data store.

4.4 Path Manager

Implementation Status: To be reused from VirtuWind project. If modified for the purposes of SEMIoTICS use cases, an according update will be presented in D3.7.

For the guaranteed industrial QoS, i.e. the bandwidth provisioning, flow isolation and worst-case delay estimation, we plan to leverage the Path Manager component. The related project VirtuWind [14] has previously proposed using network calculus, a deterministic mathematical modeling framework for communication networks to enable delay-bound, bandwidth-guaranteeing end-to-end network service [7]. Instead of basing its routing decision on a reactive control loop of network observations, Path Manager provides for real-time constraints by mechanisms for admission (and rejection) of flows. Namely, by maintaining an accurate model of the network state and service embeddings in the control plane, Path Manager ensures per-flow isolation and worst-case guarantees at all times.

Using the Path Manager, SSC will enable path computation under consideration of different QoS connectivity needs, including the requirement on low latency and reliable communication, thus supporting the platform design requirements across multiple Use Cases: R.GP.3, R.UC1.3, R.UC2.15 and R.UC2.17. The computation of paths is done according to isolation properties, i.e., the individual new reservations will not affect existing flows, thus supporting requirement R.UC1.4. Additionally, Path Manager supports the computation of maximally redundant paths, required to provide resilience in the face of network failures and thus enabled fulfillment of requirement R.UC1.5.

4.5 Resource Manager

Implementation Status: To be reused from VirtuWind project. If modified for the purposes of SEMIoTICS use cases, an according update will be presented in D3.7.

Resource Manager is responsible for configuration management and network control tasks, i.e. embedding of L2/L3 OpenFlow flow rules into the network. Resource Manager provides for embedding of: i) real-time flows which require dedicated per-queue flow assignments; ii) best effort flows, without queue considerations; and iii) the meter structures for policing purposes.

Resource Manager is capable of interacting with the infrastructure (switches, routers) etc. using one or multiple of the following interfaces: OpenFlow/OVSDB/SSH. This adheres to the requirement R.GP.6. The SSC controller will be capable of interacting both with virtual and physical OpenFlow switches. Open vSwitch is an exemplary production-ready virtual switch software (R.NL.7), that is to be used in SEMIoTICS Backend Layer for enabling the virtualization infrastructure.

4.6 Security Manager

Implementation Status: To be reused from VirtuWind project. If modified for the purposes of SEMIoTICS use cases, an according update will be presented in D3.7.

The SEMIoTICS security management system includes the Security Manager (SM) component which offers support for authentication and accounting services. SM should realize the authentication and accounting services to the rest of the SDN Controller as well as the users and applications that interact with the controller. With respect to authentication, the SM exposes interfaces for the administration of local SDN Controller accounts. The necessary methods for C.R.U.D (Create, Read, Update, Delete) Users, Roles and Domains are developed by the SM exposed them to other controller components as well. Moreover, the SM provides authentication capabilities based on credentials stored by exposing a method which has local credentials as input and its output is an authentication token. Also, it exposes a token validation technique which can be used by the other controller components. It verifies both the validation of the provided token and the bearer of the token if it is the one who he claims to be. Additional APIs are exposed for applications to present their credentials. If these credentials prove valid, the SM can issue an authentication token to the requesting party. The token can then be presented to the Pattern NBI when attempting to interact with the SDN Controller. The Pattern NBI can be responsible for transferring these tokens to the SM internally for validation, so the former can then proceed to evaluate the request (i.e., if it is allowed based on the active policies). Depending on the use case, we can distinguish two scenarios:

- User/application authentication based on a local set of entered policies / users
- User/application authentication based on an external set of entered policies / users, i.e., using an

Every interface to the SDN controller is protected by authentication/authorization mechanisms. Interfaces relevant for SEMIoTICS, including the Security Manager, VTN Manager and Pattern Engine northbound interfaces, are protected by HTTPS digest authentication, thus supporting the R.S.7 requirement. To protect and isolate access to particular internal APIs of the controller, Security Manager enables role-based definition of authorities granted access to the service, thus fulfilling the requirement R.S.2.

external LDAP server or similar. In the case of distributed authentication, the SM must present the tokens to the external server for validation.

4.7 High-Availability Controller Clustering

Implementation Status: To be reused from VirtuWind project. The component will be extended to (a) support Byzantine Fault Tolerance operation in SEMIoTICS; (b) to provide a more scalable operation in Byzantine Fault Tolerance mode. An according update will be presented in D3.7.

The issue of the controller's single point of failure is resolved by means of state replication and fail-over to one of the backup controllers on failure.

Centralized controller state registry: The Clustering module may handle the controller relationship per-data state in the distributed data-store. The controller state as well as the up-to-date network information can be collected in a single registry shard that is replicated across multiple controller instances.

Strong (SC) primitives for update ordering: Components that have stringent requirements on the data state staleness, such as the Path Manager which makes critical routing and resource reservation decisions, may require serialized updates across all instances of the replicated controller. By serializing the individual updates, we ensure that no data-store updates are applied without having first observed the previous history of the updates made to that state. Such components make use of the controller state distribution based on SC primitives (e.g., using distributed RAFT [18] consensus).

By providing for hot-standby failover in case of controller failures/faults, the HA clustering component will enable the industrial-grade high-availability requirement R.UC1.5. The final design of the high availability controller clustering solution for the SDN control plane will be presented in D3.7.

4.7.1 ENABLING BYZANTINE FAULT TOLERANT SDN CONTROL PLANE OPERATION

Distributed SDN were introduced as a way to improve the scalability of the control plane and mitigate the problem of single-point-of-failure. However, control plane correctness may be endangered by malicious controllers that enforce incorrect configurations, i.e., as a result of adversary attack or corrupt controller instance state [19] [20].

We have recently proposed MORPH [20], a framework that enables a Byzantine Fault Tolerant (BFT) SDN control plane, by allowing for runtime detection of malicious controllers and their dynamic exclusion from the system configuration, and thus the SEMIoTICS requirements R.UC1.5, R.UC1.6 and R.S.7. MORPH framework's function is based on the working principle that the switch processes all control packets originating from all the controllers in their administration domain, before deciding to apply the configuration.

To cater for tolerating unreliable controller decisions, made by faulty/malicious SDN controllers, we introduce the above Byzantine Fault Tolerance extensions, which extends the existing Clustering solution proposed and implemented by OpenDaylight and VirtuWind project. Thus, we are able to cover the requirement for reliable control plane, made in R.UC1.6. The final design of this module will be presented in D3.7.

In the network phase where no malicious or unavailable controllers are identified, such approach results in a high system footprint and limited scalability, which is an additional requirement of the SEMIoTICS project (refer to requirements R.UC1.7 and R.GP.2). Moreover, the current data plane implementations of SDN-enabled switches do not have the necessary capabilities to process multiple control packets before applying them. If the packet comparison is executed in software, the deployment of a Byzantine Fault Tolerant SDN Control plane would clearly result in a highly loaded switch control plane.

To provide for a higher scalability of the proposed solution and thus fulfillment of R.GP.2 and R.UC1.7, an alternative solution by offloading some of the tasks from the control plane of the switches to their data plane. In order to do so, the switches' data plane is programmed using the P4 language binding, which is a special-purpose language that allows for programming of packet forwarding planes. We aim to decrease the system footprint and enable a higher number of total control flows deployable in the SDN. The final design of the high scalability solution for the control plane will be presented in D3.7.

4.8 SFC Manager

Implementation Status: SFC Manager will be reused from VirtuWind project and subsequently adapted as required to support the interactions with the Pattern Engine of the SDN Controller, so to support orchestrated SFC deployment in Use Case 2 scenario (ref. Section 7).

The SFC Manager (SFCM) component of OpenDaylight is able to handle service function chaining of network functions. An SFC determines an abstract set of service functions and their ordering constraints that should be applied to packets and/or frames selected as a result of classification. Additionally, the implied order could not be a linear progression, due to the fact that the architecture allows for nodes which copy to more than one branch; also, the architecture allows for cases where there is flexibility in the order in which services need to be applied.

In the SEMIoTICS cases, service instances may include Firewall, IDS, DPI, and HoneyPot. These services may be the physical appliances or virtual machines running in network function virtualization infrastructures. They may be composed of one or multiple instances. At the Management and Control planes, the SFCM is responsible for administrating the services chains, i.e., for mapping the operator's/tenant's/ application's requirements into service chains. SEMIoTICS has identified the benefit of incorporation the NFV and SDN world in order to bring the best of both worlds related to network setup, configuration and management together. For that reason, the SFCM will be enhanced to handle the interactions between the SDN controller and the MANO, in order to receive networking information about instantiated VMs, as well as to provide information about possible paths fulfilling the requirements of the SFC.

4.8.1 AN NBI ENABLING SFC ORCHESTRATION

The SFC Manager should expose a number of NBI interfaces that various components can use to provide and receive information about service chains. Information such as SFC required, which tenants want to use them, which destinations are being accessed, what applications the traffic pertains to and about the service instances of the network functions can also be supported. That includes the exposure of both an administration interface through the controller's NBI which is also used by NFV MANO, and a login interface for the applications through the development of suitable YANG and RESTful interfaces.

4.9 NFV MANO/VIM Connector

Implementation Status: To be reused from OpenDaylight project. In particular, the ML2 connector seems the good contender for implementation of this component in Use Case 2. If modified for the purposes of SEMIoTICS use cases, an according update will be presented in D3.7.

Within ETSI's standardized NFV Architecture (see Figure 6), the Management and Orchestration (MANO) components are those responsible for hardware resources abstraction (VIM), VNF lifecycle management (VNFM), and orchestrator (NFVO). Each one of these expose services (or functions) through well-defined interface abstractions (usually REST APIs), which in turn are used by other MANO components (e.g.: for instantiating a Network Service), or external elements (e.g.: for gathering information about VNFs).

The SEMIoTICS SDN Controller is an external component to the reference NFV MANO framework of Figure 6. That is, the management of virtual network resources (e.g.: VTN, VNF), and the control of the underlying physical network are tasks handled by the SEMIoTICS SDN Controller. This brings benefits in terms of outage/saturation resilience, primarily due to the isolation of network services to separate hosts. But also allows for joint optimization of both overlay and physical network paths/resources, which could help satisfy SEMIoTICS's UC requirements/constraints.

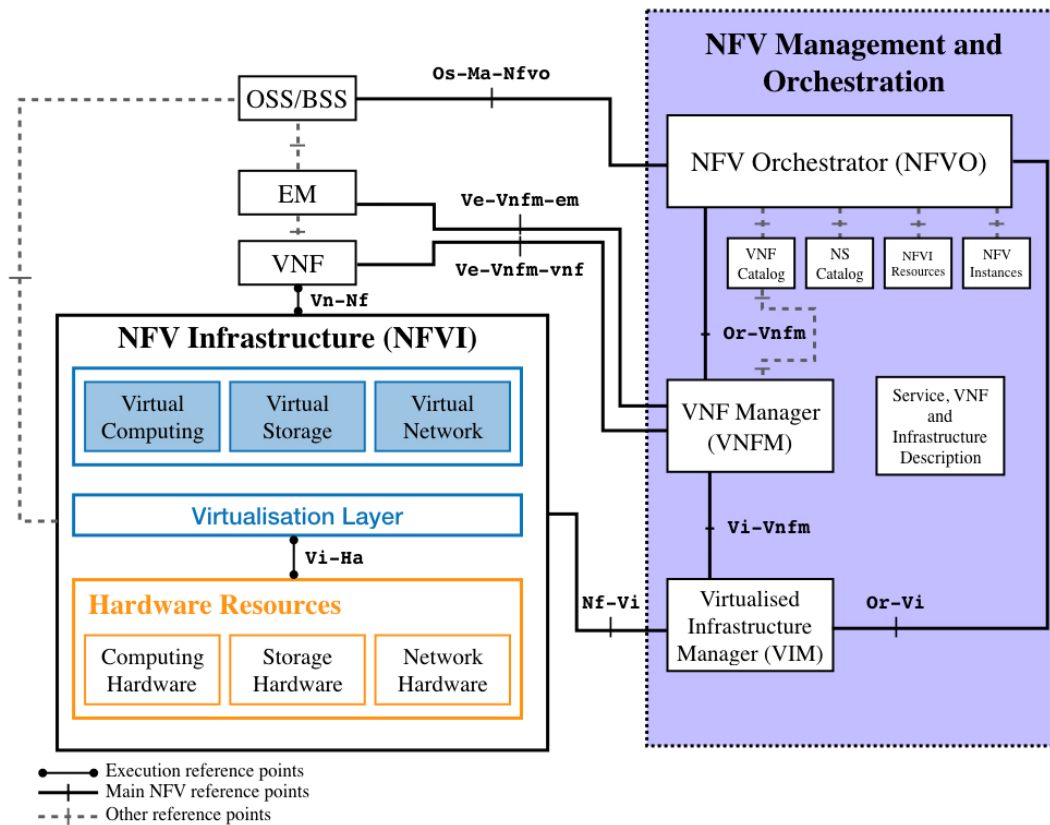


FIGURE 6: FUNCTIONAL BLOCKS OF AN NFV PLATFORM [8]

For the instantiation of an end-to-end service, the NFVO requests a “Network as a Service” through the corresponding exposed interfaces at the VIM (realized through the **Or-Vi** reference point shown in Figure 6). Then, through well-defined interfaces (e.g.: ML2 plugin [21]) the VIM exchanges the tenant’s network policy with the SDN Controller (realized through the **Nf-Vi** reference point shown in Figure 6), which in turn may request the instantiation of VNFs and other virtual network resources to complete the network service instantiation.

Other relevant information exchange occurring through relevant NFV MANO reference points are [9]:

- **Or-Vi**
 Orchestrator-VIM communication reference point. It is used for:
 - Resource reservation and/or allocation requests by the Orchestrator.
 - Virtualized hardware resource configuration and state information exchange.
 The **Os-Ma-Nfvo** reference point (see Figure 6) can be used by OSS/BSS (or other entity such as SEMIoTICS global pattern orchestrator) to gather information of the NFVI and trigger the creation/modification of a NS; but is the **Or-Vi** reference point the one that enables direct communication between MANO and VIM in order to realize such service by allocating resources from the infrastructure.
- **Nf-Vi**
 This reference point is used for NFVI-VIM communication. Particularly:
 - Assignment of virtualized resources after an allocation request.
 - Forwarding of virtualized resources state information.
 - Hardware resources configuration, information exchange and events capture.
 - Information exchange with external SDN Controllers.
- **Os-Ma-Nfvo**

It realizes Operations Support System/Business Support System (OSS/BSS)-NFV Management and Orchestration communication. It is used for:

- Request for network service lifecycle management.
- Requests for VNF lifecycle management.
- Forwarding of NFV related state information.
- Policy management exchanges.
- Data analytics exchanges.
- Forwarding of NFV related accounting and usage records.
- NFVI capacity and inventory information exchanges.

It is valid to assume the use of this reference point to software other than OSS/BSS. That is, any authorized software external to NFV could use this reference point for gathering information of the physical/virtualized infrastructure, as well as signaling the intention to create a network service via the NFVO.

5 INTERACTIONS WITH DATA-PLANE DEVICES

5.1 OpenFlow for Control Plane Interactions

SDN decouples data and control planes, concentrating control information of the whole network at the SDN Controller. In this paradigm shift, network devices obey to whatever forwarding instructions the SDN Controller dictates. For instance, when a packet arrives at an SDN-enabled forwarding device, it checks if there are any matching entries in its local forwarding table. If it fails to find one, it queries the SDN Controller through the control plane and waits for a reply containing the instructions on how to handle such flow.

The most widely used protocol for forwarding table modifications and SDN Controller-forwarding device communication, is Open Network Foundation (ONF) OpenFlow (OF) [1]. It works as a standard interface SDN Controllers use to interact with the forwarding table of network devices.

OF spurs from an initiative to promote innovation in networking (originally aimed to campus networks), where there was almost no practical way of experimenting and testing new protocols on networks with real traffic. Instead of hoping for hardware vendors to provide an open and programmable interface to their routers and switches (which may threaten companies with undesired competition), OpenFlow exploits a common set of functions related to flow tables, simply requiring a minimum set of actions¹².

An SDN-enabled forwarding device (or VNF) relies on the SDN Controller for updating its flow table and to determine all forwarding decisions. This is done exclusively using protocols such as OpenFlow messages through a separate control channel. The following summarizes relevant terminology related to OpenFlow:

- Open flow tables: a collection of flow rules, each one composed of:
 - Match fields: specific packet header information that can be used to filter flows. A flow is said to match a rule if its constituent packets have the same characteristics as specified on the match fields of the rule.
 - Actions: to be executed once a flow is matched.
 - Statistics: e.g.: keep track of the number of packets and bytes that matched a flow rule.
- A secure channel towards the Controller: often referred to as the control channel.
- OpenFlow Agent: running the OpenFlow protocol on SDN-enabled devices, thus realizing the communication between the SDN Controller and network devices' flow tables.

The most recent OpenFlow version, for example, defines up to 45 matching fields per flow rule [22]. That is, a particular forwarding decision taken at the Controller can be based on a wider criterion than that employed in traditional routing and switching; for instance, using the source and destination transport layer port numbers, protocol type, or a combination. Furthermore, OpenFlow network devices can be queried by the SSC in order to retrieve physical ports state information, or flow table statistics; widening even more the criteria upon which forwarding strategies could be defined.

The SEMIoTICS SDN Controller (SSC) is equipped with an OpenFlow module, which acts as Southbound Interface (SBI) for configuring OF-enabled devices' (physical or virtual) forwarding decisions.

5.2 OVSD/NETCONF/YANG for Management Plane Interactions

From SEMIoTICS SDN Controller's perspective, there are southbound interfaces specifically used for device configuration [4].

¹² E.g.: **Forward** packets to a given port, **Encapsulate and forward** first packet in a new flow to Controller, or **Drop**. Nevertheless, other actions are available, like: pushing new VLAN tags, TTL reduction, set a header field, among others.

NETCONF [2] is an XML-based protocol used as a southbound interface for remote configuration or communication with (physical) devices. Additionally, it could be employed as a northbound interface of the SDN Controller in the form of NETCONF servers to:

- Spawn, reconfigure or destroy SDN Controller's modules or applications.
- Read, write remote procedure calls (RPC, or functions provided by SEMIoTICS SDN Controller).

OVSDB [23] is another popular southbound protocol used for managing Open vSwitch-enabled (OVS) physical or virtual switches. It enables the SDN Controller to view, create, modify, and delete OVS objects such as bridges, tunnels, and ports. Reference SDN Controllers, such as ODL, implement an OVSDB module which, among other things, offers:

- Network virtualization.
- Southbound plugin for configuring OVS devices.
- Library for encode/decode OVSDB protocol.
- REST interfaces for configuring OVS devices.

The semantic modelling and data organization of NETCONF configuration data, notifications, and SDN Controller's RPCs are developed using YANG. It directly maps XML, allowing fast prototyping and modification of configurations. It supports several programming language-like statements such as leaf (single value, no children), leaf-lists (one value, no children, multiple instances), containers (no value, holds related children), must, list (like hashes or dictionaries), and more. Furthermore, it contains built-in types like integrals, strings, binary data, bit fields, references, or other. All of these features allow the deployment/consumption of SDN Controller applications from YANG models guaranteeing API contracts.

6 EVALUATION OF WIRELESS FIELD TECHNOLOGIES

In this section, we analyze existing wireless communication technologies that could be used for communication with the IoT devices in the different SEMIoTICS use cases defined in D2.2.

While the wireless technologies, i.e., radio access points are not to be managed through SDN, we nevertheless give an overview of technologies deployed in the prototypical Use Case deployments, due to D3.1/D3.7 being the only network-focused deliverables in the project.

Relation to Use Cases:

Use Case 1 does not rely on wireless technologies and instead assumes a completely cabled setup, typical for industrial wind park deployments. An LTE router deployment is available for providing internet connectivity in the setup in non-wired environment (but only for the purpose of demonstrator showcase).

Use Case 2 leverages IEEE 802.11 Wireless LAN and IEEE 802.15.4 in the field layer, in particular:

- The four SARA hubs (i.e. User Mobile Phone, Robotic Rollator, Robotic Assistant and Home Gateway) communicate via IEEE 802.11 Wireless LAN
- The User Mobile Phone uses also cellular connectivity (LTE/HSPA depending on availability, to communicate with remote call center)
- The Home Gateway communicates with appliances and home automation via 802.15.4 ZigBee

Use Case 3 field devices, i.e., the IHES Sensing nodes are connected to the IoT Gateway hosting the IHES supervisor service using 802.11 Wireless LAN. From IoT gateway on, the connection is assumed to be cabled one. The final use case demonstrator will be connected to relevant web services in mobility exploiting a HSPA/LTE router connection (but this is required only to allow the demo more portable and does not present a design constraint).

The selected technologies were chosen due to their applicability in the shown use cases and availability in commodity hardware. Nevertheless, the deployment of SEMIoTICS architecture, and in particular the gateway solution is not limited to combining with these wireless technologies only. Hence, for completeness, we next provide the description of alternative wireless technologies applicable to SEMIoTICS field layer:

Wireless Technology	Description	Applicable for use in Use Cases
IEEE 802.11 Wireless LAN	Frequency bands: <ul style="list-style-type: none"> ▪ 900 MHz (802.11ah) ▪ 2.4 GHz (802.11b/g/n/ax) ▪ 3.65 GHz (802.11y) ▪ 5.0 GHz (802.11j) WLAN ▪ 5 GHz or 5.8 GHz (802.11a/h/j/n/ac/ax) ▪ 5.9 GHz (802.11p) ▪ 60 GHz (802.11ad/ay) Data rates: 1 Mbps up to a theoretical limit of 1Gbps (802.11ac)	Use Case 1 (802.11ac/b/g/n for backend communication and 802.11ah for long-range communication, i.e., to / between offshore turbines). Use Case 2 (as discussed) Use Case 3 (as discussed)
IEEE 802.15.4	Frequency bands: 868/915/2450 MHz Data Rate: 250kbps (WirelessHART)	Use Case 1: N/A Use Case 2: Home Gateway to Home appliances Use Case 3: IHES Sensing nodes to IHES supervisor

WiMAX	<p>Frequency bands: 2.5 and 3.5 GHz (requires license), 5.8 GHz (license-free)</p> <p>Data rates: Up to 1Gbps for fixed stations</p>	<p>Use Case 1: As an alternative to LTE deployment for internet/backend connectivity.</p> <p>Use Case 2: As an alternative to LTE deployment for remote call center connectivity.</p> <p>Use Case 3: As an alternative to LTE deployment for mobility web services connectivity.</p>
Xbee PRO	<p>Frequency bands: 868/915 MHz</p> <p>Data rates: 250kbps</p>	<p>Use Case 1: N/A</p> <p>Use Case 2: Home Gateway to Home appliances</p> <p>Use Case 3: IHES Sensing nodes to IHES supervisor</p>
LoRaWAN	<p>Frequency bands: 433/868/915 MHz</p> <p>Data rates: Up to 50kbps</p>	<p>Use Case 1: N/A</p> <p>Use Case 2: Home Gateway to Home appliances</p> <p>Use Case 3: IHES Sensing nodes to IHES supervisor</p>
SIGFOX	<p>Frequency band: 868 MHz</p> <p>Data rate: Up to 100bps</p>	<p>Use Case 1: N/A</p> <p>Use Case 2: N/A</p> <p>Use Case 3: N/A</p>
Narrowband-IoT	<p>Frequency bands: 1800/900/800 MHz</p> <p>Data rate: 250kbps</p>	<p>Use Case 1: As an alternative to LTE deployment for remote call center connectivity.</p> <p>Use Case 2: Home Gateway to Home appliances or/and as an alternative to LTE deployment for remote call center connectivity.</p> <p>Use Case 3: IHES Sensing nodes to IHES supervisor or/and as an alternative to LTE deployment for remote call center connectivity.</p>
Public Cellular (HSPA & LTE)	<p>Frequency bands: 850/900/1800/1900/2900 MHz</p> <p>Data rate (LTE): ~1200 Mbit downstream, ~225 Mbit upstream</p>	<p>Use Case 1: As described above.</p> <p>Use Case 2: As described above.</p> <p>Use Case 3: As described above.</p>
Satellite Communications	<p>Frequency bands: 137 MHz – 150 MHz or 1,6 GHz</p> <p>Data rate: 2400bps (Iridium), 15 kbps upstream / 60kbps downstream (ACeS)</p>	<p>Use Case 1: Alternatively, for interconnection of local site to backend in case of off-shore / rural deployments. Limited applicability due to reliability/availability constraints.</p> <p>Use Case 2: N/A</p> <p>Use Case 3: Alternatively, for interconnection of IoT Gateway to backend in case of off-shore / rural deployments. Limited applicability</p>

		due to reliability/availability constraints.
--	--	--

6.1 IEEE 802.11 Wireless LAN

The IEEE 802.11 Wireless LAN (WLAN) is one of the most prevalent wireless communication standards. The technology is mature and stable, and mass production has driven hardware module prices down. WLAN is not only limited to the 2.4 GHz and the 5 GHz ISM bands, as changes in the firmware can allow the usage of neighboring frequencies. One example of this is the case with the 5.9 GHz spectrum reserved for communication on Intelligent Transport Systems. Some countries allow additional frequencies, as is the case of the IEEE 802.11j amendment, which allows the operation of WLAN in the 4.9 GHz to 5 GHz band. Yet another example is the IEEE 802.11y amendment, which enables high power data transfer equipment to operate on the 3650 to 3700 MHz band in the United States of America.

An additional approach, which is not limited to WLAN or to unlicensed frequency bands, is remixing the analog signal from a standard device into another frequency using analog radio equipment. The new signal will have the same bandwidth of the original signal but will have a different center frequency.

The data rate of WLAN goes from 1 Mbps up to a theoretical limit of 480 Mbps, depending on the modulation used. The current draft of the IEEE 802.11ac aims to achieve data rates over 1 Gbps. As a general rule, the larger the range, the slower the data rate that can be achieved; an exception is on data rates below 12 Mbps, in which the faster orthogonal frequency-division multiplexing (OFDM) modulated data can be transmitted farther than the slowed direct sequence spread spectrum (DSSS) modulation.

Several installations exist worldwide using the 2.4 GHz band, which achieve more than 1 km of range. However, most, if not all of them, have a direct line-of-sight, bridging a building and a mountain installation using high-gain directional antennas. For the first scenario as defined above, the limited range will not allow using the 2.4 GHz band. For the second scenario, the communication range of WLAN is sufficient to fulfill the distance as well as the data rate.

In the 5 GHz band, there is the possibility of sending up to 1000 mW of power (+30 dBm, 1 Watt), which can easily exceed one kilometer. This sending power is permitted in the USA when using the 5.725 GHz – 5.875 MHz ISM band without any further restrictions. In the USA and Europe, the 5.250 MHz - 5.725 MHz permits this sending power only when employing dynamic frequency selection (DFS) so that any potential interference with airport radar equipment operating in the vicinity can be detected and avoided. It should be mentioned that the specific frequency requirements as defined by the Federal Communications Commission are undergoing an overhaul; the exact changes haven't been defined yet.

6.2 IEEE 802.15.4

The IEEE 802.15.4 is a standard for low-rate personal area networks (PAN), which defines the physical layer and the media access control. The basic IEEE 802.15.4 is the foundation of ZigBee and WirelessHART and uses mainly three frequency bands: 868.0-868.6MHz (Europe), 902-928 MHz (North America), 2400-2483.5 MHz (Worldwide). There are additional frequency bands for specific countries, for example 950-956 MHz for Japan (IEEE 802.15.4d) and 314-316 MHz, 430-434 MHz, 779-787 MHz for China (IEEE 802.15.4c).

WirelessHART has been specifically designed for industrial wireless sensor networks. Siemens has products available for the WirelessHART [24] standard in the SITRANS product line, including temperature and pressure sensors. WirelessHART allows for the use of multihop technology, effectively increasing the range of coverage. There is a maximum hop count of five and the base data rate of 250 kbps reduces with the number of hops.

6.3 IEEE 802.15.4k

In general, the channel modulation used by the basic IEEE 802.15.4 combined with the permitted radiated power in Europe of 100 mW (in the USA 200 mW) in the 2.4GHz band makes it a challenge to achieve 3 km of range. Hence, there are many amendments from which to choose from, and one of them, IEEE 802.15.4k, has proven to communicate at such distances.

The IEEE 802.15.4k Low Energy Critical Infrastructure Monitoring (LECIIM) achieves the required range with commercial off-the-shelf hardware. Using an extremely slow, direct sequence spread spectrum modulation, the transmitters are in operation for extended periods of time, before the receiver can understand the signal. The company onRamp Wireless is, at the moment of writing this report, the only manufacturer of IEEE 802.15.4k LECIM hardware. Their hardware achieves a range of several kilometers and their endpoints can work on batteries. Their radio modules have the advantage of working on the 2.4GHz ISM frequency band and have a channel bandwidth of 1MHz. The center frequency can be determined by the user in a 1 MHz raster.

The radio modules have -141dBm receiver sensitivity, providing for a very large link budget, which will allow for large transmission ranges. The typical transmission power in Europe is 10mW EIRP. The modules use 600mW of power while transmitting and 350mW while receiving. Most of the times the modules will be in deep sleep using just 15µW. Using very low duty cycles of a few messages per day, such a radio module can operate for 10 years on battery power.

Access points use 15W-20W of power and have to be always on. Due to the duty cycle constraints and the use of a very slow 'direct sequence spread spectrum' (DSSS), any communication will have a delay of up to 45 seconds.

The OnRamp system uses a star topology, and every access point will require a 128 kb/s uplink to a network control center, which is managed by OnRamp Wireless. This presents an important limitation as the communications system cannot provide a simple point-to-point or point-to-multipoint connection. Endpoints cannot communicate with each other directly and the endpoints cannot communicate to each other over an access point directly, as all communications have to be managed by the centralized equipment.

6.4 WiMAX

The IEEE 802.16 Worldwide Interoperability for Microwave Access (WiMAX) communications standard concentrates on metropolitan area networks and in its latest version can provide connection speeds of up to 1 Gbps. Channel bandwidth depends on the specific profile used and goes between 1.25 MHz and 20 MHz. Specified to work in just about any frequency between 2 to 66 GHz, hardware is available for the following spectrum profiles: 700 MHz, 1.5 GHz, 1.9 GHz, 2.3 GHz, 2.5 GHz, 2.8 GHz, 3.3 GHz, 3.5 GHz, 3.7 GHz, 4.9 GHz, 5.2 GHz, 5.4 GHz, 5.8 GHz and 5.9 GHz. Communications range can run up to 50 km using the lowest data rate and very high sending power. Independent tests confirm that WiMAX is more energy efficient than GSM, UMTS and LTE mobile communication when compared under similar conditions (data rate, communications range, and ambient temperature).

The major disadvantages are similar to mobile communication. They require a licensed band and additional backend components to support simple point-to-point communication. The requirement for a licensed band means that a frequency must be acquired from the government authorities, in each country in which operation is desired. This is typically linked with very high costs.

6.5 Xbee PRO 868

Instead of using the 2.4 GHz frequency band, going to a lower frequency will help achieve the required range for the same amount of power. This led to the usage of the 868 MHz unlicensed band, e.g., utilized by the XBee-PRO 868, from the company Digi International, which uses a proprietary protocol. The data sheet quotes a distance of 40km, which has also been proven under line-of-sight conditions (+25dBm sending power, +2.1dBi antenna gain). Independent tests show good results for 10.6km (+14dBi Antenna, Line-of-Sight, sending telemetry data from a model airplane). The same data sheet quotes a range of 550m for indoor and

urban environments. As the range with line-of-sight is very good, an alternative for non-line-of-sight would be to position one or two nodes as repeaters to forward the information.

The XBee product is not compatible with IEEE 802.15.4, using a proprietary protocol instead. The 868MHz frequency band is an SDR band in Europe, providing operation free of charge. The equivalent frequency for Region 1 (Americas) is 915MHz, which use is also free of charge.

6.6 RF Mesh / multihop

RFmesh, also called multihop, is a networking concept in which nodes that are member of a network can relay the information to other nodes, so that data can be propagated along the network. Using multihop, the basic communications range can be extended several times, at the cost of an increase in latency and power usage, and the decrease of data throughput rate. An important advantage is the self-healing mechanisms that include the communication protocols used. If a network node fails, the traffic is routed around the problem.

Many of the technologies analyzed in this document allow for a multihop communication. One common example is WLAN, which in the IEEE 802.11s amendment to the standard supports multihop. Another example is the IEEE 802.15.4. Together, these two represent almost all the available RF mesh devices available on the market. They work in the ISM frequency bands that permit for a license-free operation in any region in the world.

6.7 LPWAN

The low-power wide-area network (LPWAN) is a type of wireless communication technology. LPWAN is wide area network designed to allow long range communication at a low bit rate among remote and energy constrained objects. It is an umbrella for different specific technologies, as described below.

6.7.1 LORAWAN

Long Range Wide Area Network (LoRaWAN) is a communications standard in the 868/915MHz SRD frequency band. Due to the relatively low frequency, relatively high sending power (20dBm) and good receiver sensitivity (-148dBm), devices have been reported to achieve a typical range of 2km (with a theoretical maximum range of 16km).

LoRaWAN is the network on which LoRa operates and can be used by remote and unconnected industries. LoRaWAN is a media access control (MAC) layer protocol. It is a network layer protocol for managing communication between LPWAN gateways and end-node devices as a routing protocol, maintained by the LoRa Alliance. Under the network architecture demanded by LoRa, each client device requires a logical connection to one or more LoRa gateways to achieve a connection; the protocol does not allow clients to talk to each other directly. The devices use less power than cellular mobile communication, and they have low data rates, which go from 0.3kbps up to 50 kbps.

The standard is driven by the LoRa Alliance, with Cisco, IBM, Semtech and Gemalto among its members.

6.7.2 SIGFOX

Sigfox uses the 868MHz/915MHz license-free frequency band to enable a business model similar to mobile network operators. As of October 2018, the Sigfox IoT network has covered a total of 4.2 million square kilometres in a total of 50 countries and is on track to reach 60 countries by the end of 2018¹³. Sigfox provides cost effective communication (1€ - 5€ per device per year) for devices which don't need a fast connection.

¹³ <https://www.electronicsworld.com/news/products/rf-microwave-optoelectronics/sigfox-iot-network-reaches-50-countries-2018-10/>

This system has a channel bandwidth of 200kHz and a slow connection speed (100bps). The communication latency is of 4s for a typical message, and 60s maximum. Moreover, it has to be taken into account that this technology allows to exchange 140 messages per day as a maximum which is expected to be an exclusion criterion for most scenarios even if low bandwidth is needed.

Many manufacturers of Sigfox clients can be found on the market, including Atmel and Texas Instruments.

Before Sigfox is selected as the preferred communications technology, special attention has to be paid to the network coverage in the area of use, as today (October 2015) it covers only one city in Germany (Munich).

6.7.3 NB-IOT

Narrowband IoT (NB-IoT) is developed by 3GPP to enable a wide range of cellular devices and services. It aims to address the needs of very low data rate devices (often powered by batteries) that need to connect to mobile networks. As a cellular standard, the goal of NB-IoT is to standardize IoT devices to be interoperable and more reliable. The 3GPP standard is currently in Release 13 (LTE Advanced Pro). It defines the device receiving bandwidth as 180 kHz with a download/upload rate of 250 kbits and a latency of 1.6s–10s.

6.8 Satellite Communication

An interesting option is using satellite data communication in the bands of 137 MHz – 150 MHz or 1,6 GHz. Some of the current satellite systems available allow usage of a simple and relatively small rod antenna. Examples are Iridium and Orbcomm in Low Earth Orbit (LEO), as well as Globalstar and Inmarsat in geosynchronous orbit (GEO). Modules for machine-to-machine communication (M2M) are available on the market.

Using satellite communication has the disadvantage of generating running costs. However, it has the advantage of not requiring any infrastructure for a long-range communication. Unfortunately, such systems have a significant latency when delivering the data. As an example, Orbcomm announces that 2% of all the data have a delay of over 15 minutes, while Iridium defines waiting times between 1-8 minutes as possible.

6.9 Public Cellular NB (GPRS, GSM): MHz & BB (HSPA, LTE)

Mobile communication allows the possibility of packet switched digital data communication using GPRS, EDGE, Evolved EDGE and HSPA. If the field components which require communication are all in range of a mobile communications network, then a very simple solution would be to provide the elements with communication over GSM/UMTS. These wireless technologies work in the bands of 850/900/1800/1900/2900 MHz.

As this is not always the case, an alternative is the setup of a local microcell. For a microcell, a low power cellular base station providing mobile communication coverage over a few kilometers, standard GSM/UMTS modules can be used. It should be noted that this microcell would require a connection to the backbone systems of the mobile network operator, as a cellular base station is going to require different resources located there, i.e., the home location register and the base station controller.

Different manufacturers offer such microcell products, for example Huawei and Alcatel-Lucent. The communications module can be obtained from many manufacturers, including Qualcomm, CSR and SIMCom. License to the appropriate frequency band will run by the mobile network operator.

7 SDN IN CONTEXT OF SEMIoTICS USE CASES

The SSC with the majority of its functions will be deployed in Use Cases 1 and 2 as the main network connectivity enabler. Both Use Case 1 and Use Case 2 will deploy complete SSC function set, i.e., including all modules of Figure 4, however, they will highlight individual features of the architecture, due to Use Cases encompassing a vast larger number of overall SEMIoTICS components, apart from networking.

Use Case 1:

SEMIOTICS Use Case 1 comprises multiple programmable logic controllers, IIoT gateway, network routers (internet gateway) and SCADA application instances, interacting for purpose of field level monitoring, actuation and reporting of observed system state values to MindSphere platform in the backend. All interactions require basic field-layer and internet network connectivity, enabled by the SDN controller in automated and reliable manner. Furthermore, field layer services, such as: (i) the wind park controller-to-turbine actuation; (ii) updating of threshold values by SCADA in the turbine controller, with goal of per-turbine frequency and voltage optimization in energy production, as well as; (iii) proactive/reactive actuation in case of failure (i.e., stopping the turbine in case of oil/grease detection and outlier sound samples); require QoS-constrained interaction between the observing instance and the actuator.

To this end, SSC enables QoS-constrained network flows, fulfilling the individual latency, bandwidth requirements of the service, and will demonstrate:

- (I) The initial automated bootstrapping of the network (focuses on Bootstrapping Manager)
- (II) Instantiation of Virtual Tenant Networks in the field-layer (focuses on using Security Manager for authentication/authorization in SSC's UI and the VTN Manager for VTN enforcement).
- (III) Providing best-effort flows for interconnecting infrastructural components in scope of the established VTN (focuses on interaction between Path Manager, Bootstrapping Manager, VTN Manager and Resource Manager components).
- (IV) Providing QoS-constrained flows for interconnecting critical components (i.e., Programmable Logic Controller-to-Monitoring App connection). This workflow will require interaction between Path Manager, Pattern Engine, VTN Manager and Resource Manager, but will also necessitate the reservation state updates in Clustering Manager).

Use Case 2:

The second use case of SEMIoTICS focuses on an ambient assisted living scenario. It encompasses a complex environment, requiring support for integration of heterogeneous devices and communication protocols, high degrees of interoperability and support for distributed services and applications (each with its own set of intrinsic requirements), while guaranteeing the safety of the patient and the security and privacy of her patient data. There is a significant motivation to leverage the flexibility provided by Service Function Chaining (SFC; as detailed in deliverable D3.2), to define specific service chains for each type of traffic in this scenario. With the functionality of each individual service functions in the chains being in line with the details presented in deliverable D3.2 (Section 2, in specific), we foresee the enablement of following SFCs:

- Chain 1 – Mobile Phone: Firewall -> Header Enrichment -> IDS -> Output
- Chain 2 – Robotic Rolator: Firewall -> IDS -> Load Balancer -> Output
- Chain 3 – Smart Home: Firewall -> IDS -> Output
- Chain 4 – Robot: Firewall -> Load Balancer -> Output
- Chain 5 - Malicious: Firewall -> Honeypot

SSC will enable the Service Function Chaining, i.e., the configuration of paths fulfilling the forwarding of matched traffic through the VNF chains, as required by the above SFCs. To this end, VIM will interact with the SFC Manager of the SSC to request the according translation of SFC graph into the resulting network

configuration and will showcase SSC's enablement of connectivity in backend layer using the VIM connector API of the SSC.

Use Case 3:

While in principle SDN could be deployed in Use Case 3 as well, we will focus on highlighting other components of the SEMIoTICS architecture except SDN there.

8 NEXT STEPS AND CONCLUSIONS

Next Steps:

As a core component, positioned between the SEMIoTICS backend/cloud (more specifically the Pattern Orchestrator) and physical forwarding data plane, the SSC is a complex piece of software. Thus, we will next initiate building and extending the SEMIoTICS controller based on the existing open-source and VirtuWind code base, instead of initiating an error-prone and costly from-the-scratch development. The overview of the existing implemented SSC components that are to be reused, and those that are to be developed from scratch / newly implemented is provided in Figure 4 and individual component sections in this document.

In the following, we summarize the most important development activities planned for the upcoming period:

Pattern Engine Implementation and adjusting of interacting components: To support pattern-model in specification of connectivity requirements, we will require and thus will develop and integrate the Pattern Engine controller component. Additional changes to other relevant modules of the controller will be made, so to support the Pattern Engine interactions. Additionally, to support the connectivity specific requirements and QoS properties, we will develop the support for patterns capable of requesting, enforcing and monitoring QoS-enabled connectivity.

Enabling scalability in failure-tolerant SSC: The SSC's Clustering Manager component will be developed / extended from existing OpenDaylight-based release, as necessary to support Byzantine Fault Tolerance approach to high availability of the controller instances. Indeed, as described in Section 4.7, current planned conceptual solution to providing for Byzantine Fault Tolerance comes with large overhead of replicated computation and replicated control flows necessary to support redundant decision-making by multiple controller instances. We will investigate, document and provide a Proof-of-Concept of Byzantine Fault Tolerant computation with focus on scalability aspects of the SSC solution.

Automated establishment of basic network services: Enabling point-to-point connectivity in OpenDaylight and reference VirtuWind implementations requires manual or scripted specification of end-points to be interconnected by the network flows. This would lead to a large manual effort complexity due to a high number of infrastructural services necessary by SEMIoTICS components in the field and backend layer that require such connectivity. To this end, we will implement an automated instantiation of network services for infrastructural network flows (e.g., Thing Directory synchronization, IoT Gateway <-> Router <-> Internet flows) as required by the SEMIoTICS use cases to minimize the scenario deployment efforts.

Final SSC component responsibilities: In the subsequent deliverable D3.7 (scheduled for M26), we will provide the final design of the SSC components. During the development phases of SEMIoTICS, if further gaps and differences to the presented component overview arise, then those will be documented in the Deliverable D3.7.

Conclusion:

The present deliverable provides an overview of the purpose, architecture placement and the design of the SEMIoTICS SDN Controller (SSC) solution. Furthermore, we shed light on the currently available wireless technologies for connecting (low-powered) sensor/actuator at large scale and long distance for industrial use cases. The SSC with the majority of its functions will be deployed in Use Cases 1 and 2 as the main network connectivity enabler. The mapping of selected wireless technologies on the individual Use Cases 1, 2 & 3 is presented in introduction of Section 6.

The SSC provides a mean to dynamically and flexibly provision, monitor and evict virtual tenant networks and network services at per-application granularity, both during the engineering phase (pre-planned services) and at runtime. The flexibility in deployment of services comes from the wide gamut of specifiable and guaranteed

properties associated with the provisionable network services, including but not limited to the end-to-end delay, bandwidth and redundancy properties.

This deliverable provides the insight on planned development efforts, new and modified components and the introduced deltas to existing state-of-the-art controller platforms based on OpenDaylight open-source controller solution.

9 REFERENCES

- [1] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker and J. Turner, "OpenFlow: enabling innovation in campus networks," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 2, pp. 69-74, 2008.
- [2] R. Enns, M. Bjorklund, J. Schoenwaelder and A. Bierman, "RFC 6241: Network Configuration Protocol (NETCONF)," June 2011. [Online]. Available: <http://www.rfc-editor.org/info/rfc6241>. [Accessed 23 October 2018].
- [3] M. Bjorklund, "YANG-a data modeling language for the network configuration protocol (NETCONF). No. RFC 6020.," 2010.
- [4] R. Toghraee, *Learning OpenDaylight: The art of deploying successful networks*, Birmingham: Packt Publishing Ltd., 2017.
- [5] OpenDaylight, "OpenDaylight Lithium," [Online]. Available: <https://www.opendaylight.org/what-we-do/current-release/lithium>. [Accessed January 2019].
- [6] T. L. Foundation, "Open vSwitch," [Online]. Available: <https://www.openvswitch.org/>. [Accessed January 2019].
- [7] J. W. Guck, A. Van Bemten and W. Kellerer, "DetServ: Network models for real-time QoS provisioning in SDN-based industrial environments," *IEEE Transactions on Network and Service Management* 14.4, pp. 1003-1017, 2017.
- [8] ETSI, "ETSI.org: Network Functions Virtualisation (NFV); Architectural Framework (ETSI GS NFV 002 V1.2.1)," February 2014. [Online]. Available: https://www.etsi.org/deliver/etsi_gs/NFV/001_099/002/01.02.01_60/gs_NFV002v010201p.pdf. [Accessed November 2018].
- [9] ETSI, "ETSI.org: Network Functions Virtualisation (NFV); Management and Orchestration (ETSI GS NFV-MAN 001)," December 2014. [Online]. Available: https://www.etsi.org/deliver/etsi_gs/NFV-MAN/001_099/001/01.01.01_60/gs_NFV-MAN001v010101p.pdf. [Accessed November 2018].
- [10] OpenStack, "Compute API," [Online]. Available: <https://developer.openstack.org/api-guide/compute/>.
- [11] OpenStack, "OpenStack Docs: Block Storage," [Online]. Available: <https://developer.openstack.org/api-ref/block-storage/v3/>.
- [12] OpenStack, "OpenStack Docs: Networking API v2," [Online]. Available: <https://developer.openstack.org/api-ref/network/v2/>.
- [13] OpenDaylight, "OpenStack and OpenDaylight," [Online]. Available: https://wiki.opendaylight.org/view/OpenStack_and_OpenDaylight.
- [14] T. Mahmoodi, V. Kulkarni, W. Kellerer, P. Mangan, F. Zeiger, S. Spirou, I. Askoxylakis, X. Vilajosana, H. J. Einsiedler and J. Quittek, "VirtuWind: virtual and programmable industrial network prototype deployed in operational wind park," *Transactions on Emerging Telecommunications Technologies*, no. 27-9, pp. 1281-1288, 2016.
- [15] "YangTools Project," [Online]. Available: <https://github.com/opendaylight/yangtools>.
- [16] "Drools Business Rules Management System (BRMS)," [Online]. Available: <https://www.drools.org>.
- [17] R. Charles Forgy, "A Fast Algorithm for the Many Pattern/Many Object Pattern Match Problem," *Artificial Intelligence*, vol. 19, p. 17-37, 1982.
- [18] H. Howard, M. Schwarzkopf, A. Madhavapeddy and J. Crowcroft, "Raft refloated: do we have consensus?," *ACM SIGOPS Operating Systems Review*, vol. 49, no. 1, pp. 12-21, 2015.
- [19] K. ElDefrawy and T. Kaczmarek, "Byzantine fault tolerant software-defined networking (SDN) controllers," *Computer Software and Applications Conference (COMPSAC), 2016 IEEE 40th Annual*, pp. 208-213, 2016.
- [20] E. Sakic, N. Djeriç and W. Kellerer, "MORPH: An Adaptive Framework for Efficient and Byzantine Fault-Tolerant SDN Control Plane," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 10, pp. 2158-2174, 2018.

- [21] OpenStack, "ML2 plug-in," [Online]. Available: <https://docs.openstack.org/newton/networking-guide/config-ml2.html>. [Accessed January 2019].
- [22] O. N. F. (ONF), "OpenFlow Switch Specification: version 1.5.0 (Protocol 0x06)," 2014. [Online]. Available: <https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-switch-v1.5.0.noipr.pdf>. [Accessed 2018].
- [23] B. Pfaff and B. David, "The open vSwitch database management protocol," *RFC 7047*, 2013.
- [24] J. Song, S. Han, A. Mok, D. Chen, M. Lucas, M. Nixon and W. Pratt, "WirelessHART: Applying wireless technology in real-time industrial process control," *IEEE real-time and embedded technology and applications symposium*, pp. 377-386, 2008.
- [25] OpenStack, "OpenStack Docs: Server concepts," [Online]. Available: https://developer.openstack.org/api-guide/compute/server_concepts.html.