SDN Controller Requirements for Next Generation Telco PaaS

Kentis, Angelos Mimidis; Soler, José; Fernando, Díaz; Aurora, Ramos; Olivier, Choisy; Aravinthan, Gopalasingham

Published in:
Proceedings of 22nd Conference on Innovation in Clouds, Internet and Networks

Link to article, DOI:
10.1109/ICIN.2019.8685842

Publication date:
2019

Document Version
Peer reviewed version

SDN Controller Requirements for Next Generation Telco PaaS

Mimidis Angelos, Soler José
DTU Fotonik
Lyngby, Denmark
agmimi@fotonik.dtu.dk
joss@fotonik.dtu.dk

Díaz Fernando, Ramos Aurora
ATOS
Madrid, Spain
fernando.diaz@atos.net
aurora.ramos@atos.net

Choisy Olivier
BCOM
Cesson-Sévigné, France
olivier.choisy@b-com.com

Aravinthan, Gopalasingham
Nokia Bell Labs France
Paris, France
gopalasingham.aravinthan@nokia-bell-labs.com

Abstract—**This paper presents an analysis of requirements to Software Defined Network (SDN) Controllers (SDNC), from the point of view of the Telco's Next Generation Platform as a Service (NGPaaS) functional and architectural needs. Numerous requirements have been identified, which can be grouped in 4 categories: domain-based, modularity-based, related to the SDN and Network Function Virtualization (NFV) integration and finally policy-based. These requirements are later mapped into the ONOS SDN controller, with the scope of identifying possible gaps and points of improvement.**

Keywords—PaaS, 5G, Telco-grade, Next-Generation, SDN, NFV, ONOS

I. INTRODUCTION

The Platform as a Service (PaaS) model [1] has facilitated the provision and management of services and applications in cloud infrastructures. In the rest of the paper, any reference to platforms, services or applications, is within the context of a Telco environment.

Building on the success of the PaaS Service model the Next Generation PaaS (NGPaaS) [2] project has proposed a framework that can build or deploy and manage Telco related PaaS. These Telco PaaS can either follow the Management and Orchestration (MANO) reference specification from ETSI [3], or follow their own (e.g. CORD [4]).

However building a robust, scalable and flexible NGPaaS that can meet the needs of 5G services, imposes requirements to all the functional elements of the NGPaaS architecture [5]. NGPaaS adheres to the design rules of componentization and build-ship-run. This allows the design and deployment of platforms and services in a build-to-order basis. The architecture of NGPaaS consists of a multitude of components (Operation Support System (OSS), Virtual Infrastructure Manager (VIM), Software Defined Network Controller (SDNC), etc). However the mentioned requirements cannot be generic to all of these components, and must be identified in a per component basis. This is mainly due to the inherently different role that each of these elements plays within the context of NGPaaS.

An SDNC is a core component of any PaaS implementation, since it is directly controlling the associated network infrastructure. Thus, this paper focuses on identifying a list of desired features and requirements that an SDNC must meet, in order for the SDNC to be considered suitable for NGPaaS.

To ensure that this survey of requirements is thorough and well organized, the requirements have been categorized into different groups. Each of these groups looks at the SDNC from a different point of view, hence enhancing the scope of the survey. The identified groups are the following:

- **Domain (D)**: Requirements within this group deal with how one or more SDNCs can effectively control a multi-domain network environment.
- **Modularity (M)**: The focus here is on identifying requirements that will facilitate a modular SDNC architecture.
- **SDN/NFV integration (SN)**: SDN and Network Function Virtualization (NFV) are two tightly woven paradigms within NGPaaS. This group of requirements focuses on facilitating their integration.
- **Policy enforcement (P)**: For an efficient management over a diverse network infrastructure (like that of NGPaaS), it is important to allow for the definition of high-level policies that do not depend on specific technologies and protocols.

While analyzing the different groups of requirements some overlapping was unavoidable, hence some requirements might be analyzed multiple times. However, this is done under a different scope each time. In addition, some requirements were analyzed deeper, due to their higher relevance to NGPaaS. To facilitate cross-referencing of requirements across the document, each requirement is also associated with a short unique identification tag (e.g. D1, M2 etc). Each of these requirements is finally checked against the features offered by the ONOS SDNC [6], in an attempt to evaluate its conformance with the NGPaaS platform.

In the last years there has been a lot of research [27], [28], [29] in relation to the evaluation of the different available SDNCs. However so far the approach was limited to feature and performance based comparisons, focused on the current network requirements and with no specific application field. As previously stated the approach in the present paper is different, as it focuses on identifying requirements for next generation Telco network infrastructure.

The paper is structured as follows; Section II lists the domain-based requirements, section III lists the requirements from the perspective of modularity, section IV identifies the requirements related to the SDN/NFV

integration and section V analyzes the requirements related to policy enforcement in SDN. These requirements are then mapped over the features offered by the ONOS SDNC, in section VI. Finally the paper concludes in section VII.

II. Domain-Based Considerations

An NGPaaS will be hosting Virtual Network Functions (VNFs) from different vendors, and may, depending on the use case, comprise a cloud federation. Due to this distributed nature of the deployed VNFs, an SDN architecture that supports multi-controller support must be designed.

A. Identified Requirements

To support multi-controller network control, SDNCs that will be part of NGPaaS must fulfil the following requirements:

- **Distribution (*D1*)**: Several controller instances (of the same or different vendor) shall be deployable, so that each instance can manage different domains of the network. The functionalities/capabilities of each instance can be based on different criteria (administrative, functional, geographical separation). For the domain-based consideration distribution is the most important requirement, with the remaining (scalability, latency, etc.) being highly dependent on distribution.
- **Scalability (*D2*)**: The possibility to scale, in or out, the number of controller instances involved in the network-control operations. This decision shall be based on network performance considerations.
- **Latency (*D3*)**: The capability to guarantee the expected network latency by either migrating a controller instance (closer to the devices under its control), or adapting the network devices under the controller's control. The latter can be achieved by performance adaptation for both the data flows and the control flows.
- **High Availability (*D4*)**: The network infrastructure shall be able to withstand failures from one or more controller instances, without performance degradation.
- **Consistency (*D5*)**: The overall consistency of the network managed by the controller instances must be guaranteed at all times. This includes the necessary mechanisms to address possible errors (e.g., recovery or rollback) and to direct commands to the right controller instance, to achieve the expected network configuration. It is assumed that the different controller instances may not be autonomous. An orchestrator function or properties from the controller instances must be able to guarantee complete consistency of the system and shall be able to trigger consistent and adaptive changes depending on certain events (e.g. errors).
- **Dynamicity (*D6*)**: The different controller instances must be configurable independently in an ad-hoc manner, so they can adapt as the network state changes.
- **Slicing (*D7*)**: The different distributed controller instances shall be able to comply with slice constraints for operations sent to the network's resources.

B. Distributed SDNCs: Architecture

Currently, there is no formalized architecture for controller distribution and neither for the required protocols for inter-controller communication. Several architectures have been proposed; for example, [7] [8] for flat architecture and [9] for hierarchical architecture. These proposals can have either a horizontal or vertical hierarchy (Figure 1).

A network domain can be considered as a set of network devices which are under the responsibility of an operator. In this context, controller distribution is useful in both intra-domain scenarios and inter-domain scenarios. In an intra-domain scenario, the network devices might be organized according to different criteria as different administrative, geographical sub-domains. For the inter-domain scenario, a distribution of controller instances may be present to provide the different requirements for each individual domain. Each domain can be managed by a dedicated controller to take care of the specifics of each domain and to comply with access rights. A distribution like this may be managed by one, or more, inter-domain controllers as depicted in Figure 2, so as to guarantee that consistent rules are applied in each domain.

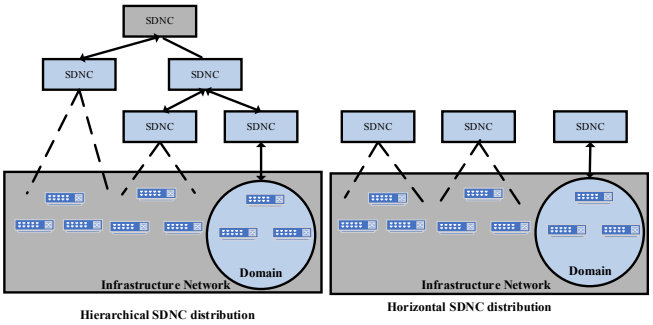

Figure 1: Horizontal vs Vertical SDNC distribution

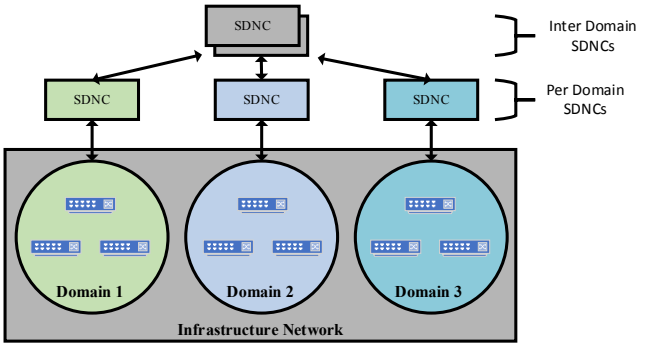

Figure 2: Inter domain controller

C. Distributed SDNCs: some key challenges

One of the key challenges for controller distribution is the consistency property. To address this issue, the following questions must be answered:

- How can the correct application of a network configuration rule be assured on the system, as a whole?
- How should controller failovers be managed?
- How to reconfigure the network and keep the information at the controllers updated?

These functions and the controller orchestration can all be handled by the top-most controller in a hierarchical architecture.

Another major challenge is the choice of controller placement, depending on the traffic constraints (e.g., Quality of Service (QoS) or latency) and the priorities of these constraints. A number of questions can also help to address this issue:

- Is the distribution dynamic or static?
- Does the controller distribution evolve as the platform changes or does it depend on Key Performance Indicators (KPIs)?
- Should this be handled by the top orchestrator in the hierarchy?

In the context of NGPaaS, the distribution of controllers is important for the implementation of the NGPaaS platform. The role of the SDNCs will be to create the connectivity paths between different deployed VNFs.

III. Modularity-Based Considerations

There are many different SDNC frameworks with diverse sets of implementation models, programming languages, features and architectural designs. Despite this abundance of choices, it is still difficult to make an informed SDNC selection, since they all compete with each other in terms of features, usability, extension, etc. Coming down to the implementation, an SDNC is just a set of software functions working together to achieve a specific functionality. Most of current SDNCs are based on a modular architecture. However, current SDNCs are still monolithic as they do not support features that will optimize them for a specific deployment scenario.

A modular SDNC framework shall be based on the build-ship-run approach (***M1***). This ensures that the SDNCs are deployed with a minimum footprint, whilst still satisfying the requirements (QoS/Quality of Experience (QoE)) of their use case (e.g. slices in 5G) and network domain (e.g. Radio Access Network (RAN), Core, Fronthaul etc.). A built-to-order SDNC framework can choose dynamically its software modules depending on the use case. This eliminates the deployment of duplicated or unused functionalities that can be found in monolithic SDNCs. Such an approach will also improve control-plane performance and as a result impact the QoS/QoE for each use-case. The design of such an SDNC framework is illustrated in Figure 3. There, a *Built-to-order* module is used to deploy an SDNC for the management of a Radio Access Network.

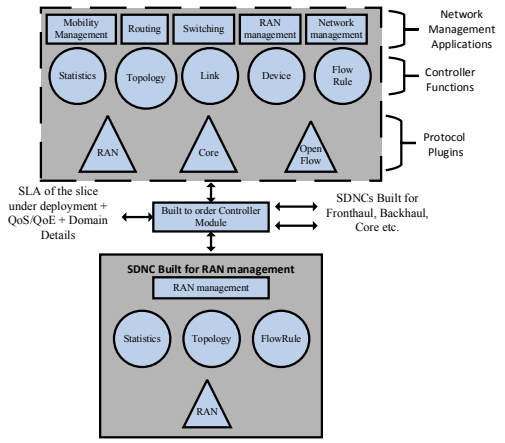

Figure 3: Built-to-Order SDNC Frameworks

To do so the *Built-to-order* module utilizes a repository of available SDNC functions. This framework has been designed with the 12 cloud factors [11] in mind, in order to satisfy a number of known challenges (low footprint, resiliency, etc).

IV. SDN/NFV Integration Considerations

According to ETSI [12], the success of SDN/NFV integration depends on the fulfilment of a list of requirements. The scope of this section is to list these requirements and analyze them from the perspective of the SDNC.

*A. Virtual Network Function Components interconnection with SDN (**SN1**)*

A VNF might be composed of several Virtual Network Function Components (VNFCs), thus an SDNC could have a double role. First it should be responsible for establishing links between VNFCs and second it might support traffic management functions (routing, QoS) between VNFCs.

*B. VNF Interconnection with SDN (**SN2**)*

A network service might comprise a source, a destination, and a set of intermediary VNFs. This complex VNF chain can appear in different scenarios, such as:

- Chaining based on a network service designed based on a VNF Forwarding Graph (VNF-FG). This is a static chain, applied when the network service is instantiated.
- Chaining based on customer policy/service. This form of VNF chaining is more dynamic and flexible and is determined at service instantiation time. It can be based on a variety of factors such as policies, network/compute/storage conditions and customer based policies.
- Chaining based on VNF processing. This form of chaining is even a more dynamic, since it can be modified ad-hoc (after instantiation time) based on processing provided VNFs. An example for this case is load balancing traffic across VNFs. Here elasticity is provided by adding or removing VNFC or VNF instances in a VNF chain.

*C. SDN across multiple VIMs (**SN3**)*

It can be the case that SDN control shall span across domains managed by different VIMs. For this multi-VIM use case, two scenarios can be considered:

- VIMs placed in the same NFV Infrastructure Point of Presence (NFVI-PoP)
- VIMs placed in different NFVI-PoPs.

These scenario are associated with the following challenges:

- Crossing administrative or organizational boundaries can impose requirements at various levels (e.g. security, connectivity, information hiding, and Service Level Agreement (SLA) fulfilment). This can constrain the choice of possible solutions.
- The network protocols used in a NFVI-PoP and in the Wide Area Network (WAN) have very different characteristics.
- Provisioning of network services across VIMs is also challenging due to requirements such as low latency.

- Setup of paths between newly instantiated VNF and existing Physical Network Functions (PNFs)/VNFs over the WAN, must be dynamic.
- The connection and the guarantee of SLAs for VNF scaling over multiple NFVI-PoPs, must be ensured.
- Virtual links must be extended out of the NFVI-PoP and into the WAN.
- The transport network shall be reconfigurable, e.g. in the case of natural disasters or network congestion.

D. *Definition of the SDNC hierarchy (**SN4**)*

As already discussed in section II, it might be sometimes necessary to consider a multi-controller deployment scenario. This section focuses, on how the SDNC hierarchy is related to the SDN/NFV integration. The following scenarios for SDNC hierarchy are considered herein:

- SDNC hierarchy designed for distributed performance, scalability and reliability for both multi and single layer transport networks.
- SDNC hierarchy designed for distributed, cross service provider (cross-SP) or cross-domain services.
- SDNC hierarchy for Network as a Service (NaaS).
- SDNC hierarchy for multi-domain, transport network fast fault recovery.

The hierarchy of an SDNC can be implemented within/across different functional blocks of the ETSI NFV architecture; hence the hierarchy of SDNCs should be analyzed in different scenarios:

- Inside the NFV Infrastructure (NFVI).
- In a VNF.
- Across functional blocks.
- Below the WAN Infrastructure Manager (WIM) (while the WIM is not part of the ETSI framework, this can occur when layers/regions of multiple trust domains occur in the WAN).

E. *Placement of SDN Controller in a virtualized environment (**SN5**)*

A virtualized SDNC can be deployed either as a single component or as a collection of multiple components. So a virtualized SDNC can be: Either a single VNF or a VNFC (which composes the different sub-elements of the SDNC).

F. *Implementation of VNF-FG (**SN6**)*

Based on [3], a descriptor for a VNF-FG contains a list of virtual links that need to be established between its VNFs and PNFs. This will help forming a Network Connection Topology (NCT) and may contain one or more Network Forwarding Path (NFP) elements. These NFPs associate traffic flows (matching certain criteria) to forwarding paths (the sequence of network functions to be traversed), within an NCT. SDN can play a role for implementing both the NCT and the NFP. SDNCs involved in the NFP configuration might be different from those involved in setting up the underlying NCT. The SDNC shall also be able to provision virtual address spaces for VNF-FGs.

G. *SDNC interface with orchestration blocks (**SN7**)*

The SDNC in an NGPaaS instance will interface with the NFV Management and Orchestrator components. This will facilitate the NFV Orchestrator (NFVO) to pass initialization flows to the SDNC, when new VNFs are instantiated. The SDNC shall also collect network performance metrics and report these metrics via an Application-Controller Plane Interface (ACPI) to the Orchestrator.

H. *Impact on multi-tenancy (**SN8**)*

The SDNC must program the network elements (both physical and virtual), in order to meet the requirements of each specific VNF service. This also ensures proper network slicing, connectivity and isolation in a multi-tenant environment. This way, the network can be properly sliced and shared amongst virtual networks, each of these networks dedicated to a specific VNF service. The SDNC will also guarantee isolation among different virtual network resources, which are created to serve the requested services.

I. *Impact on QoS (**SN9**)*

The SDNC shall allow infrastructure connectivity services to specify parameters related to performance. The SDNC shall also provide the mechanisms required to support QoS control (e.g. by processing virtual networks deployment based on specific topology and QoS requirements).

J. *Flow setup time (**SN10**)*

The SDNC shall aim to minimize the time required to setting up flows, thus maximizing the number per unit of time.

K. *Control plane failures (**SN11**)*

The SDNC shall be able to withstand failures of the control plane (e.g., via SDNC redundancy) in a robust manner. This will ensure persistence of the network configuration

V. Policy-Based Considerations

In the present section a number of requirements for the SDNC and from the point of view of network policy enforcement are listed. A mind map of the complete set of these policy requirements is provided in Figure 4.

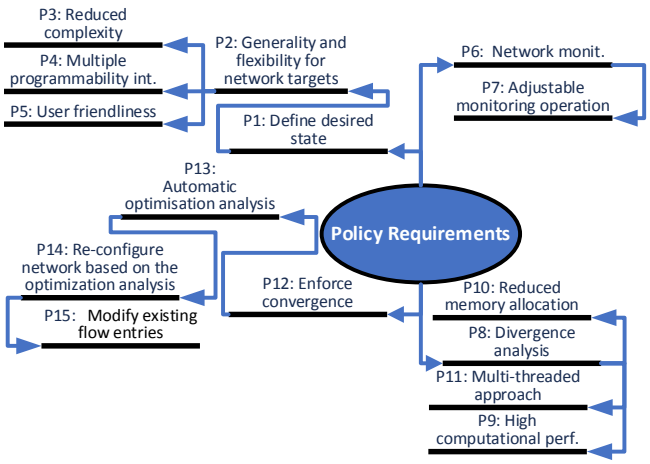

Figure 4: Requirements for network policy enforcement

A. *Definition of the desired state of the network*

Network policies are sets of conditions and constraints, used to define a desired network state in a generic way; the role of the SDNC shall be to translate these network policies into a specific network state (desired state), which is

implemented as a set of network intents (SDNC internal mechanisms that specify desired network behavior) (**P1**)
The SDNC shall also have the necessary flexibility to accommodate a variety of different network targets; this is because policies are expected to be generic/extensible (**P2**).
The policy framework shall have a low implementation complexity (**P3**). It should also consider a multitude of programmatic interfaces (**P4**). A final requirement is that it should be user friendly and should facilitate easy integration (**P5**).

B. Network Monitoring

For the SDNC to evaluate the alignment between the actual network state and the desired state (as expressed by the network policies), a monitoring channel between the SDNC and the network must exist (**P6**). Since, the monitoring requirements from the network, can vary based on the cases/policies, the monitoring functionality of the SDNC must be adjustable (**P7**).

C. Divergence analysis

After the actual state of the network is evaluated (by means of monitoring), the SDNC shall be able to perform a divergence analysis. This should be between the actual state of the network and desired state of the network (**P8**).
Divergence analysis is a processing-intensive and memory-intensive task, thus increasing the processing requirements for the SDNC. To mitigate this, the implementation shall have as low a computational (**P9**) and memory overhead (**P10**) as possible.
One way to balance the performance versus the processing requirements of the divergence analysis, is to use multi-threaded approach (e.g. when >1 network targets are evaluated, allocate multiple threads) (**P11**).

D. Enforce Convergence

After performing the divergence analysis, the SDNC shall perform the actions that will align the actual state of the network and the desired state of the network (**P12**).
The SDNC must be able to perform an optimization analysis based on a wide variety of different network parameters (**P13**) (**P9**, **P10** and **P11** included in case of processing-intensive and memory-intensive tasks). Finally to modify the network elements, the SDNC will require a configuration channel (**P14**, **P15**) to the network.

VI. MAPPING TO EXISTING SDNCS

In order to evaluate the alignment of the current state of the art in SDNCs, with the identified requirements listed in this paper, the two most prominent SDNCs where considered (ONOS and ODL). A gap analysis has been performed between the identified requirements and the features of each SDNC. However due to space limitations and since this work also resulted in a policy framework prototype for ONOS, only the results for the ONOS SDNC are presented herein. However, in [13] a complete and extended analysis is available.

A. ONOS

Based on the requirements for SDNCs outlined previously, this section provides a mapping of those requirements and the features to the ONOS SDNC, provided in the previous sections.

*1) Distribution (**D1**)*

ONOS instances can be deployed either as independent controllers or as a distributed cluster. This clustering capability, meets the requirement of D1. However this distributed architecture does not have a specific hierarchy.

*2) Scalability (**D2**)*

The ONOS developers define scalability as the ability to add compute resources and increase the control plane capacity, while keeping the control plane centralized [14]. This is a supported feature heavily based on the distribution capability; hence the requirements of D2 are also met by ONOS.

*3) Latency (**D3**)*

It is possible to create new ONOS instances on demand, and possible to change the mastership of each network device to specific ONOS instances. Hence by using a network manager it is possible to modify the network state, such that, for example, specific control plane latency requirements are met.
The same can be said for latency requirements in the data plane, since traffic can be rerouted through paths with different network characteristics.

*4) High Availability (**D4**)*

As ONOS developers state: “Distribution provides fault tolerance and resilience even when individual controller instances fail” [15]. To coordinate the nodes in the cluster, ONOS instances use Atomix [16], which is based on a consensus algorithm.

*5) Consistency (**D5**)*

ONOS fulfils this requirement based on the features of scalability and high availability. Since version 1.4, Data stores for Device, Link and Host information management etc. use an optimistic replication technique complemented by a background gossip protocol that provides consistency [17].

*6) Dynamicity (**D6**)*

By operating in a clustering mode, ONOS ensures that changes in the configuration of one of the instances will propagate to all instances of a cluster [17]. This provides consistency throughout the cluster, but does not allow for instances to be tailored to the network segment they control.

*7) Slicing (**D7**)*

There is no specific reference on ONOS documentation about supporting slicing out of the box. However, since ONOS is an extensible and modular framework, it is possible to build an ONOS-native application for slicing. At the moment there are multiple such attempts, each however built for specific use cases [4], [18], [19].

*8) Build-Ship-Run Approach (**M1**)*

This requirement relates to the ability to create custom versions of an SDNC and deploy them on demand in the infrastructure. Due to its modular architecture, ONOS allows for the creation of pre-packaged instances which when deployed come up preconfigured with pre-defined capabilities (loaded applications, configurations etc). However, as previously mentioned (under Dynamicity) a single cluster will have to comprise similar ONOS instances.

*9) Establish & manage links between VNFs (**SN1, SN2**)*

ONOS is described as the component that interconnects VNFs within the CORD platform [4]. In the CORD

implementation, ONOS is also capable of providing this chaining at service instantiation time.

*10) Policy-based chaining (**SN2**)*

In this case the SDNC must first receive policies from a controlling entity such as a MANO orchestrator and apply the necessary network actions. ONOS provides a REST API [20] that can be used for this purpose. However, since ONOS does not currently offer a dedicated policy framework, this feature will have to be implemented.

*11) VNF Processing-based Chaining (**SN2**)*

As long as the SDNC offers an ACPI, modification of VNF chains can be done by any authorized entity. VNFs could modify chains by contacting the NFVO/VIM instead of the SDNC.

*12) Load Balancing across VNFs (**SN2**)*

This functionality may depend on entities external to the SDNC, which provide VNF instantiation capabilities (e.g. the NFVO or the VNFM). Load balancing requests may be commanded by these entities through the same ACPI interface mentioned in 10) and 11). Hence, it shall be possible to program a customized traffic load balancing application in ONOS.

*13) Connectivity over WAN (**SN3**)*

For an intra-domain scenario, the infrastructure shall support the same Data-Controller Plane Interface (DCPI) protocols as the SDNC. For an inter-domain scenario several restrictions may apply, since different trust domains imply that the SDNC has no direct access to the network. In this case, the WAN Infrastructure Manager (WIM) must provide an interface which allocates connectivity resources. For most cases the NFVO is the entity contacting the WIM and, in turn, the WIM contacts the SDNC. So, although the SDNC is controlling the network devices, the NFVO is in charge of resource allocation.

*14) Virtual Link Extension over WAN (**SN3**)*

This feature is a subset of the previous requirement but implies the presence of slicing capabilities (D7). Hence, providing virtual links over WAN depends on the slicing capabilities of the underlying infrastructure.

*15) SDN Controller hierarchy (**SN4**)*

There is no hierarchy as such in an ONOS cluster, as explained in 1). However, this requirement follows the same demands as distribution (D1), scalability (D2) and high availability (D4) requirements. Distributed performance is achieved as a result of the distributed core and clustering capabilities. Reliability is linked with the high availability provided by ONOS and explained in D4.

*16) SDN controller as a single software image (**SN5**)*

Being able to ship as a single software image and run it in a virtualized environment is a requirement important for the SDN/NFV integration. ONOS supports deployment in both physical and virtual environments.

*17) Interface with the NFV Orchestrator (**SN7**)*

ONOS provides both a CLI and a RESTful interface to allow external entities (e.g. NFV orchestrator) to request network changes. This requirement also enables other requirements with regards to the SDN/NFV integration (e.g. VI.A.9, VI.A.10 and those requiring high-level orchestration).

*18) Multitenancy (**SN8**)*

In terms of slicing, this requirement is aligned with the description provided in 7). Isolation can be provided in a number of ways, for instance though VLAN tagging (a feature currently supported by ONOS.

*19) Mechanisms to support QoS (**SN9**)*

Currently, ONOS supports *Meters*, which can help to impose QoS. Queueing mechanisms may also help to impose QoS at the data plane, and ONOS can, through its support for NETCONF, allow for a diverse configuration on the network infrastructure.

*20) Minimize flow setup time (**SN10**)*

A single ONOS instance can install up to 700k local flow rules [21]. Results improve for ONOS clusters [21]. The time it takes to install these flow rules however is more related to the network devices and the control channel conditions, than it is to the SDNC.

*21) Robustness against control plane failures (**SN11**)*

This requirement is very closely related to 4). If resiliency is needed, redundancy can be provided by an ONOS cluster.

*22) Definition of the desired state of the network (**P1**)*

Currently, ONOS supports *connectivity intents* [22] but it will need a further extension in order to support a wider variety of functionalities and the mentioned policy-to-network state translation. At the moment ONOS does not provide a dedicated Policy Framework, however ongoing work for the NGPaaS project has extended ONOS with a network Policy Framework [23]

*23) Flexibility, programmability, configuration (**P2, P4, P14, P15**)*

Currently, ONOS supports a wide variety of DCPI protocols like OpenFlow and NETCONF [24]. ONOS also supports multiple device-specific drivers so it can seamlessly communicate with network devices (data plane switches) from different vendors. Finally the layered architecture of ONOS makes use of protocols and drivers transparent to the applications developed (technology agnostic).

However, with regards to ACPI protocols ONOS is relatively limited as it only supports the internal Java API and the REST API for external calls. However this is a trend amongst the majority of SDNCs and can be overcome by creating (use-case specific) protocol translation layers.

*24) Low-complexity/High performance (**P3, P9**)*

ONOS provides a well-defined and modular structure for its services and the applications. This facilitates the disaggregation of functionalities (micro-service approach), thus minimizing the dangers of code/functionality replication.

*25) User-friendliness (**P5**)*

As mentioned, ONOS does not provide a dedicated policy framework at the moment, however a simple and user-friendly policy framework for ONOS is proposed in [23].

*26) Monitoring Capabilities (**P6, P7,SN7**)*

As stated in [25] an ONOS-native application dedicated to metrics is available and can gather a wide range of statistics. In addition ONOS allows the collection of a number of network statistics and parameters, like switch-specific stats, events like link failures etc. If more explicit statistics are required, it should be possible to integrate ONOS with a networking monitoring suite like S-flow [26].

*27) Divergence Analysis (**P8**)*

Currently, *Intents* provide a high-level interface to define and enforce the desired network control. The *Intents* are compiled by the controller and this results in actions on network devices [22]. This requirement is also related to the one described in E.8. Although ONOS *Intents* are powerful, their scope is relatively limited.

*28) Support for multi-threading (**P11**)*

To address this requirement during divergence analysis, ONOS supports asynchronous intent compilation based on Java Future.

*29) Alignment between actual state and desired state (**P12**)*

To fulfill this requirement, ONOS *Intents* can be translated into DCPI instructions (e.g. OpenFlow) in order to comply with the intended state described in the policies [22].

*30) Optimization Analysis (**P13**)*

This refers to the ability of the controller to perform an analysis for the best result for state enforcement. This is related to P9, in order to keep a low computational footprint.

VII. Conclusion

This paper presents an analysis on the minimum requirements that an SDNC must meet in order to be suitable for NGPaaS. A multitude of requirements has been identified and analyzed, which have been grouped herein in 4 categories (domain, modularity, SDN/NFV integration and policy enforcement). Finally this list of requirements was cross-checked against the capabilities offered by the ONOS SDNC, in an attempt for a preliminary validation of its compliance to NGPaaS. The assessment is that, for the most part, ONOS meets the requirements for NGPaaS. However some extensions (e.g. policy framework, variance in capabilities within instances in an ONOS cluster), should be investigated for future releases.

VIII. Acknowledgment

This work has been performed in the framework of the NGPaaS project, funded by the European Commission under the Horizon 2020 and 5G-PPP Phase2 programs, under Grant Agreement No. 761 557 (http://ngpaas.eu).

The authors would like to thank Eder Ollora, for the preliminary work that led to this paper

References

[1] C. Röck and S. Kolb, “Nucleus – Unified Deployment and Management for Platform as a Service Nucleus – Unified Deployment and Management for Platform as a Service,” no. 100, 2016.

[2] M. Kentis, Angelos Mimidis; Ollora Zaballa, Eder; Soler, José; Bessem, S.; Roullet, Laurent; Van Rossem, S.; Pinneterre, S.; Paolino, M.; Raho, D.; Du, X.; Mariani, L.; Ramos, A.; Labrador, I.; Broadbent, A.; Zembra, “The Next Generation Platform as a Service Cloudifying Service Deployments in Telco- Operators Infrastructure,” in *Proceedings of the 25th International Conference on Telecommunications (ICT 2018)*, 2018.

[3] J. Quittek, P. Bauskar, T. BenMeriem, A. Bennett, M. Besson, and A. Et, “Network Functions Virtualisation (NFV); Management and Orchestration,” *Gs Nfv-Man 001 V1.1.1*, vol. 1, pp. 1–184, 2014.

[4] L. Peterson, “CORD : Central Office Re-Architected as a Datacenter,” no. November, 2015.

[5] S. Van Rossem *et al.*, “A Vision for the Next Generation Platform as a Sevice,” in *2018 IEEE 1st 5G World Forum*, vol. 1.

[6] On.Lab, “Introducing ONOS - a SDN network operating system for Service Providers”, Whitepaper, 2014

[7] M. Cello, Y. Xu, A. Walid, G. Wilfong, H. J. Chao, and M. Marchese, “BalCon: A distributed elastic SDN control via efficient switch migration,” *Proc. - 2017 IEEE Int. Conf. Cloud Eng. IC2E 2017*, pp. 40–50, 2017.

[8] K. Phemius, M. Bouet, and J. Leguay, “DISCO: Distributed SDN controllers in a multi-domain environment,” *IEEE/IFIP NOMS 2014 - IEEE/IFIP Netw. Oper. Manag. Symp. Manag. a Softw. Defin. World*, no. Vm, 2014.

[9] Y. Fu, J. Bi, K. Gao, Z. Chen, J. Wu, and B. Hao, “Orion: A hybrid hierarchical control plane of software-defined networking for large-scale networks,” *Proc. - Int. Conf. Netw. Protoc. ICNP*, pp. 569–576, 2014.

[10] R. Alimi, R. Penno, and Y. Yang, “RFC 7285: Application-Layer Traffic Optimization (ALTO) Deployment,” 2014.

[11] “Twelve factors.” [Online]. Available: https://12factor.net/.

[12] ETSI, “Network Functions Virtualisation (NFV); Ecosystem; Report on SDN Usage in NFV Architectural Framework,” *Etsi Gs Nfv-Eve 005 V1.1.1*, vol. 1, no. 12, pp. 1–125, 2015.

[13] “List of NGPaaS deliverables.” [Online]. Available: http://ngpaas.eu/project-outcomes/#deliverables.

[14] Open Networking Foundation, “Raising the bar on SDN control plane performance, scalability, and high availability,” 2015.

[15] “ONOS Distributed Operation.” [Online]. Available: https://wiki.onosproject.org/display/ONOS/Distributed+Operation.

[16] Atomix, ‘Fault-tolerant distributed coordination framework for Java 8,.’” [Online]. Available: http://atomix.io/atomix/.

[17] “ONOS Cluster Coordination.” [Online]. Available: https://wiki.onosproject.org/display/ONOS/Cluster+Coordination.

[18] Open Networking Foundation, “CORD: Network Edge Platform for Service Delivery,” 2018.

[19] Open Networking Foundation, “Mobile CORD (M-CORD) . Open Reference Solution for 5G,” 2018.

[20] “ONOS REST API.” [Online]. Available: https://wiki.onosproject.org/display/ONOS/Appendix+B%3A+REST+API.

[21] Open Networking Foundation, “SDN Control Plane Performance” July, 2017.

[22] “ONOS intent framework.” [Online]. Available: https://wiki.onosproject.org/display/ONOS/Intent+Framework.

[23] A. Mimidis, O. Zaballa, D. Version, and O. Zaballa, “Policy Framework for the Next Generation Platform as a Service,” 2018.

[24] “ONOS Southbound protocols.” [Online]. Available: https://wiki.onosproject.org/display/ONOS/Southbound+protocols.

[25] “API: Performance Test Related API.” [Online]. Available: https://wiki.onosproject.org/display/ONOS/API%3A+Performance+Test+Related+API.

[26] “sFlow.” [Online]. Available: https://sflow.org/.

[27] R. Khondoker et al. ,”Feature-based Comparison and Selection of Software Defined Networking (SDN) Controllers”, World Congress on Computer Applications and Information Systems (WCCAIS) ,2014

[28] O. Salman et al., “SDN controllers: A comparative study”, 18th Mediterranean Electrotechnical Conference (MELECON), 2016

[29] L. Mamushiane et al, “A comparative evaluation of the performance of popular SDN controllers”, 10th Wireless Days Conference (WD), 2018