Deliverable D5.1

Universal Node functional specification and use case requirements on data plane.

Dissemination level	PU
Version	1.0 Reviewed
Due date	31.01.2014
Version date	29.08.2014

This project is co-funded by the European Union

Document information

Authors

Editor – EHU

Contributors – Hagen Woesner (BISDN), Andreas Roos (DT), Mario Kind (DT), Eduardo Jacob (EHU), Jokin Garay (EHU), Jon Matias (EHU), Gergely Pongracz (ETH), Robert Szabo (ETH), David Verbeiren (INTEL), George Agapiou (OTE), Fulvio Risso (POLITO), Guido Marchetto (POLITO).

Reviewers - András Császár (ETH), Wouter Tavernier (IMINDS)

Coordinator

Dr. András Császár

Ericsson Magyarország Kommunikációs Rendszerek Kft. (ETH) AB

KONYVES KALMAN KORUT 11 B EP

1097 BUDAPEST

HUNGARY

Fax: +36 (1) 437-7467

Email: andras.csaszar@ericsson.com

Project funding

7th Framework Programme

FP7-ICT-2013-11

Collaborative project

Grant Agreement No. 619609

Revision and history chart

Version	Date	Comment
0.1	16.01.2014	First draft based on content from UNIFY WP5 Wiki
0.2	20.01.2014	Revised version with comments
0.3	21.01.2014	Figures included
0.4	24.01.2014	New ToC and Included requirements derived from other WPs Contribution r334 from DT added
0.5	27.01.2014	First proposal for functional specification
0.6	28.01.2014	Contribution to 2.4 and old sections/changes removed
0.7	30.01.2014	Contributions to 2.4, 2.6, 3.1 and 3.2
0.8	31.01.2014	Contributions to 2.1, 2.3, 2.5 and 2.7, updated to new template, minor format corrections
0.9	03.02.2014	Contributions to summary, sections 1.1, 1.2, 2.1, 2.2, 2.4, 2.6, 3, 4.2 and 4.4
0.10	05.02.2014	Contributions to 1, 2.2, 2.6,2.7, 4.3, reorganization of 2.1/2.3/2.4/2.5
0.11	06.02.2014	Changes agreed on review meeting and reorganization of previous chapter 4 (Selected Use Case Requirements)
0.12	10.02.2014	Included comments from Robert/David/Gergely, section 1 completed, contributions to section 2 use case, added introductions to section 4 requirements and subsections, new approach for section 5 functional specification, included references
0.13	11.02.2014	Changes agreed on review meeting
0.14	12.02.2014	Additional content in section 1, 2 and 5
0.15	12.02.2014	Previous changes accepted, sections 1 to 4 (except 4.5) closed
0.16	14.02.2014	Introduction to section 4.3, updates to section 4.5 and 5, references included, reordering of requirements according to level, minor format and edition changes
0.17	18.02.2014	Final changes before review process
0.18	26.02.2014	Consolidated comments from review process
0.19	10.03.2014	Final version
1.0	29.08.2014	Replaced references to non-public MS2.1

Table of contents

Executive summary

The main objective for WP5 is the design, implementation and performance evaluation of a Universal Node prototype, which aims to be applicable in current communication networks where a large variety of services are deployed on standard hardware.

As a first step towards the objective, this deliverable gathers the requirements for the Universal Node data plane, derived from the input of all the Work Packages, as will be collected in deliverable D2.1 "Use Cases and Initial Architecture", and structured according to their impact on the key aspects of the data plane: on the one hand network virtualization and resource sharing, on the other hand switching and traffic steering. Support aspects as configuration, performance, monitoring and security are also covered.

Secondly, this deliverable introduces the first and high-level approach to the functional specification of the data plane, based on three main modules focused on management, control and execution.

Finally, the deliverable details the specific requirements of a selected use case on the data plane and also defines the expected performance parameters. The virtual BNG is the use case specifically selected to demonstrate the initial achievements of WP5. It poses as main challenge for the Universal Node the handling of a high packet throughput and also covers interactions with the orchestration layer thus contributing to the design of the Integrated Prototype.

This choice of a rather traditional use case is motivated by the fact that it will allow techno-economic comparisons between an implementation based on the UNIFY approach and existing traditional implementations. The Universal Node should however be capable of handling more forward looking use cases such as those that will be developed by WP2. Here only a high level example is given: a network app store. This is a use case reflecting the change in networking where the strict borders that exist today between providers and users start to become blurred:

- More and more user apps will run in the cloud requiring higher levels of customization. Even user specific packet processing code can be deployed on the universal nodes (user networking apps).
- Some of these apps or functions need to run with minimized delay, close to the users (application migration, flexible deployment).
- User devices could take part in the global resource cloud (dynamic change in user-provider role).

1 Introduction

Today, rigid network control and infrastructure limit the flexibility of service creation. The UNIFY consortium envisions full network and service virtualization to enable rich and flexible services and operational efficiency. We do research on, develop and evaluate the means to orchestrate, verify and observe end-to-end service delivery from home and enterprise networks through aggregation and core networks to data centres. One central point to our unified production environment is the ability to operate virtual network functions throughout the whole network. Therefore, we design, build and evaluate means to host such virtual network functions over commodity hardware based nodes.

To be able to design a flexible, easily programmable network, we have to prove that programmability in packet processing does not incur high performance penalty. We argue that today's Ethernet chips are already at a level of complexity where the additional overhead of programmability is relatively low. This realization may have serious impact to the unfolding of SDN. If the price of programmability is indeed low, more programmable chips may prove to be a cost-effective solution for a wide-range of task eventually paving the way towards SDN. The Universal Node (UN) work package aims to prove this assumption by designing a commodity processor based packet processor node that is capable of high-performance forwarding and also capable of running high-complexity packet processing programs in a virtualized environment [1].

This technical approach will open up wider possibilities not only for reducing OPEX and CAPEX but also for building up new services or reengineering existing ones with a far more flexible approach: even more this is accelerating the innovation cycles and enabling faster time to market, allowing for a higher degree of customization and, last but not least, improving Telecom economics in terms of operations.

The starting point for the definition of the Universal Node has been the set of requirements identified by all the Work Packages and that will be documented in D2.1 "Use Cases and Initial Architecture" along with various Use Cases considered for UNIFY. Those requirements have been analyzed to determine whether they had an impact on the data plane architecture/interface. Those which did have been collected in Appendix A and from them the specific requirements for the data plane have been extracted. Also one specific use-case, presented in section 3 has been selected to complete the initial requirement assessment and design.

At this stage, the reference architecture is an ongoing task in WP2. Figure 1.1 highlights the UN-specific aspects of the architecture, or in other words the "big picture" from the UN's perspective.

The UN combines both packet forwarding and packet processing in a single box. The Network Controller (e.g. OpenFlow controller) is responsible for controlling the packet forwarding and traffic steering between the external interfaces of the UN and the packet processing applications executing on the UN. The Resource Controller (e.g. OpenStack) controls the processing resources of the UN (CPU, memory, storage) and their

allocation to the applications. Moreover, if the UN runs specific packet processing applications, like Deep Packet Inspection for example, this specific application will most likely have a specific control interface (e.g. Application Specific Control).

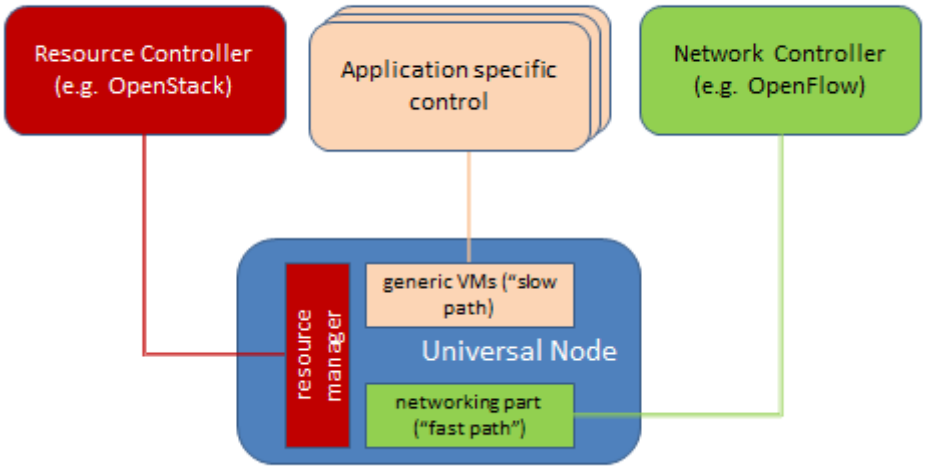

Figure 1.1: Universal Node related to external components

Regardless of the final definition of the reference architecture, for practical reasons, this deliverable considers two basic roles to be played in relation to the UN: the UN owner and the UN user. The UN owner has the responsibility of managing the physical node and making the resources available to the UN users. On the other hand, the UN user is the consumer of the resources exposed by the UN owner. The UN user controls the forwarding and processing of packets through the interfaces exposed by the UN owner.

The architecture of the Universal Node is also an ongoing task at this stage in WP5. Figure 1.2 shows the current status of this architecture and makes it possible to contextualize the target components of this deliverable.

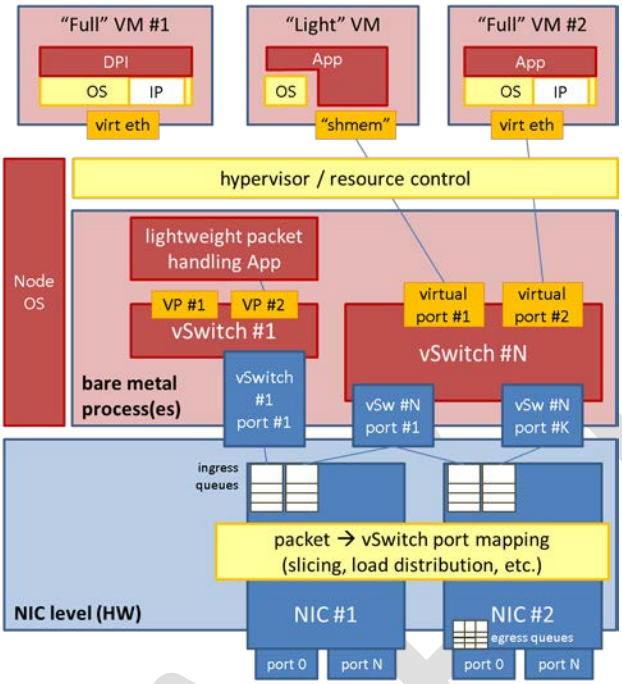

Figure 1.2: Current Universal Node architecture

The universal node contains the following functional blocks:

- Network Interface Card (NIC) level or in other words the I/O layer: this is typically a network interface card that supports limited functionality in hardware. After L1 media handling, it has some mechanism to distribute packets/frames among the ingress queues that are handled by packet processors. These packet processors are typically virtual switches, but direct mapping of a VM's application is also possible. This mechanism can be used for network slicing or some basic (e.g. random or static hash based) load distribution – this way the NIC hardware can offload the CPU cores
- Bare metal process(es): this is the performance-critical part of the UN. Packets enter the node from the I/O layer with high speed and are copied to the memory space of a CPU. Typically this is some internal memory due to performance reasons. In the bare metal environment the resources of a CPU core are fully accessible. This way only coarse grain resource control is possible (granularity is one CPU core), and the extensive use of external memory – since it is a shared and relatively slow resource – is not recommended.
    a. Typically the virtual switch instances will run in this environment, which is responsible for fast packet switching between physical interfaces and virtual ports. The virtual switch can also be used as a platform for deploying high-performance packet processing functions where the limited function set of the switch is enough (e.g. IP routing, stateless firewall). There will always be at least one virtual switch in the Universal Node, set up during the bootstrapping process, and depending on

the final requirements imposed by architecture decisions from WP2 and WP3, it could be possible to instantiate more virtual switches .

b. It is possible to run some packet handling application ("black box" code) in bare metal mode as well. This is the typical use case for performance-critical applications that use features that cannot be implemented in a switch (e.g. NAT). In this case the application should come from a trusted source, since resource control is limited: typically these applications would run as built-in extra functions of the virtual switches, sharing memory and CPU (e.g. the API could be a callback function).

- Resource control / hypervisor: anything that needs better resource control would run on the top of the resource control layer.
- Full VM: this is a standard virtual machine that can be used to deploy packet processing code that requires normal operating system support for its networking tasks (e.g. IP stack, sockets). This is the "slow, but easy" way of deploying new services into an operator's network.
- Light VM: this is an optimized virtual machine that offers higher performance by introducing some limitations. The most typical limitation is that if we want to support "zero copy" operation on the incoming packets some support is required from the application (or the OS). That means that for example in this case the generic socket API cannot be used. This is suitable for applications that use some "shared memory friendly" approach (e.g. pcap or DPDK based applications) but require some resource control.
- Node OS: this is the operating system that is used as an interface towards the different controllers and can also be used to handle resource control / scheduling. Typically it can be seen as a Linux OS that processes controller requests by running controller daemons, modifies rules, starts/stops processes and virtual machines, etc.

2 State of the art

Currently there are several activities targeting the software defined networking area. The following items seem relevant for the Universal Node work as they focus on the main functional areas to be covered by the UN:

- Packet processing:
  - ClickOS [2] (EU FP7 Change project): a somewhat limited, but high-performance virtualization environment was created where tiny VMs are used to put together packet processing functions.
  - xDPd [3](BISDN): an open source OpenFlow [4] switch implementation designed to be platform independent.
  - Different variants of OVS: the "de-facto" virtual switch implementation in Linux. The original OVS has severe performance limitations [5], but there are several improved implementations, such as the DPDK based OVDK [6] switch.
- Control plane APIs:
  - OpenStack [7]: an open source resource (compute, storage) controller.
  - OpenFlow [4]: a standardized, generic API to control packet processing in a network node.
  - OpenDaylight [8]: an open source network controller containing the OpenFlow 1.0 and 1.3.x APIs.
- Network virtualization:
  - FlowVisor [9]: an open source network slicer that allows multiple tenants to share the same physical infrastructure, acting as a transparent proxy between OpenFlow switches and multiple OpenFlow controllers.
  - OpenVirteX [10]: a network virtualization platform providing topology and address virtualization.

3 Selected use case

The use cases selected within Work Package 2 to cover the various aspects in which the UNIFY project aims to have an impact will be listed in deliverable D2.1. Based on the currently envisioned overall architecture and the relatively high level at which the use cases are currently described at this early stage of the project, we believe that a Broadband Network Gateway (BNG) function implemented on the UN will prove to be a very valuable use case for the design and implementation of the UN prototype in WP5. This function is in fact present in many of the project use cases from WP2. It will introduce all requirements that apply to "basic" infrastructure virtualization (Use Case Group 1 in D2.1) as well as allow us to later evolve it towards a use case that benefits from more flexibility and more geographical distribution of the functional elements like in the "Virtual Residential Gateway – BNG" use case (Use Case Group 2 in D2.1). Finally, the functional elements making up the BNG could also be spread between multiple operators as is the case in the Use Case Group 3.

This choice of a rather traditional use case is motivated by the fact that it will allow techno-economic comparisons of an implementation based on the UNIFY approach and the Universal Node with existing traditional implementations. We envision this use case to be initially implemented as a single function running on a virtualized infrastructure and to later evolve into a more modular implementation in which multiple reusable building blocks (routing, load-balancing, de- and en- capsulation….) will be orchestrated and dynamically placed on different nodes at different locations within the network.

3.1 Virtual BNG

The Broadband Network Gateway (also known as Broadband Remote Access Server – BRAS) routes traffic to and from broadband remote access devices such as digital subscriber line access multiplexers (DSLAM) or Customer Edges (CE).

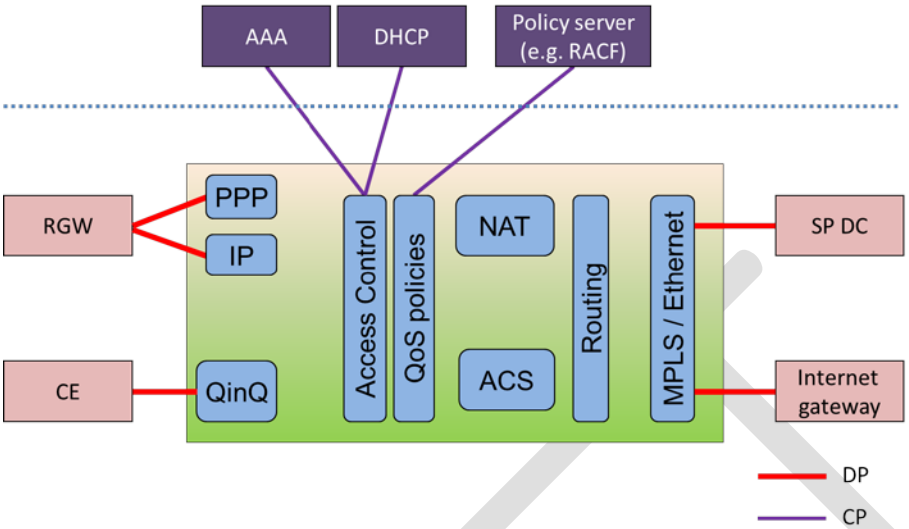

Figure 3.1: Virtual BNG use case

Its main components are the following:

- **Layer 2 connectivity on the access lines: tunnels (PPP, QinQ) or plain Ethernet.**
- **Access control, authentication, possibly personal firewall (AAA).**
- **QoS enforcement, possibly service based QoS with Deep Packet Inspection.**
- **Network Address Translation (NAT).**
- **IP address assignment, first IP hop from access, uplink IP routing.**
- **Possibly MPLS support.**

Further detail of the use case can be found in deliverable D2.1 "Use cases and initial architecture".

4 Requirements

Starting from the requirements from the Work Packages included in Appendix A, the impact on the data plane architecture/interfaces has been extracted and grouped according to the functional area involved. First groups cover two different areas of the data plane: network virtualization and resource sharing as novel aspects on the one hand, switching and traffic steering handled together on the other. Next groups cover considerations about other aspects, not novel as requirements but on the approach of the solution, specifically regarding performance.

Finally, in order to present a picture as complete as possible, the requirements extracted from the selected use case are also included as a specific group.

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY" and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [11].

4.1 Network virtualization and resource sharing

This section lists the requirements on the UN to support virtualization of the network and the sharing of resources by the different UN users, which is a responsibility of the UN owner. At this stage, the actual mechanism for implementing the network virtualization is not yet defined and its support could have impact on multiple layers. Definitely, the support for multi-tenancy imposes some requirements on the UN, mainly related to the traffic isolation and resource sharing enforced at the data plane level. In the current state-of-the-art, there are some proposals for implementing network virtualization out of the box (e.g. FlowVisor [9] or OpenVirteX [10]) with very little relation with the network node. The UN tries to take a step forward and provide the functionalities needed to support a carrier grade virtualization of network resources.

1. The UN MUST provide a mechanism to provide traffic isolation in the data plane.
2. The UN MUST support bandwidth and network resource sharing.
3. The UN MUST provide an interface to describe its available resources:
    a. Network resources (e.g.: ports, bandwidth).
    b. Compute resources (e.g. CPU, memory, storage).
4. The view of the available resources as seen by the Resource Controller MUST reflect all consumed resources e.g. those consumed by the UN low-level packet processing (compute, accelerators, etc.).
5. Requests for additional flow processing by the Network Controller MUST be validated against available (i.e. not committed) resources and MUST be rejected in case insufficient resources are available.

6. The UN SHOULD support traffic differentiation according to at least the following criteria:
    a. L1: physical incoming port.
    b. L2: VLAN.
    c. L3-L4: IP, port, DSCP/TOS.
7. The UN SHOULD also describe available accelerators that may be accessed directly by specific VNFs.
8. The UN MAY provide the option to configure direct access to a dedicated physical port or dedicated queue for a NF. Once a resource is dedicated, the control plane must be made aware that the resource is no longer available for other users.

There are some requirements which are related to the global architecture that is currently being discussed, so it is still not defined if they will impact the data plane or not but are included here for reference:

1. Traffic isolation implementation MUST be transparent to the UN users.
2. The UN exhibits a logical view to the controller. The definition of the correspondence between the exposed logical view and the actual physical topology and features of the UN is currently being defined.

4.2 Performance

A state-of-the-art UN should not perform orders of magnitude lower than the corresponding dedicated hardware solution in order to be a viable choice. This section intends to give ballpark guidelines on the lower bounds of the performance expected from the UN.

Clearly, the throughput is very dependent on the function implemented on the node. Hence, this section distinguishes between the different functionalities implemented on the UN, from base switching functionality to more complex functionality.

Base functions can be implemented without the need of additional code running in virtual machines. Most likely they will be implemented by programming flow rules in the virtual switch(es). Some examples are L2 and L3 (both IPv4 and IPv6) forwarding or NAT.

Extended functions start from anything that keeps state in the datapath, and, for the sake of a BRAS, this may be PPP in some GRE or PPPoE encapsulation. Note that this still does not mean that it has to be run as standalone code: it might be implemented by the virtual switch(es).

Complex functions (or black box functions) are functions that run in a separate environment because of their complexity or because they are coming from a non-trusted (e.g. 3rd party) application developer. Here the performance characteristics are unknown or at least not easy to predict, so the UN has to run these in a resource-controlled environment.

Unless stated otherwise, the performance numbers in this section are to be understood as "per CPU core" value and are referred to the system that is fully dedicated to the given function assuming high-end (e.g. Intel Xeon) processors. Note that with lower end CPU cores (e.g. Atom, ARM) these values should be scaled down linearly according to power consumption.

Scenario	Criteria	Value	Comments
Common Features	Minimum Number or slices	16	Inspired by the number of LSI in an NEC IP8800 switch.
	Time to handle any type of requests from controllers – at least send a "reject" in case of overload	100 ms	Depends on table size, control path bandwidth, CPU used on the datapath. Still, an upper bound would be critical.
Base Function L2 Forwarding	Throughput	8/2Mpps	For 64 / 1518 byte packets.
	Maximum delay processing packets	20 us	Between two PHY ports (without traffic management).
	Number of table entries	1M	Maximum table size.
Base Function L3 Forwarding	Throughput	3Mpps	For both 64 and 1518 byte packets.
	Maximum delay processing packets	30 us	Between two PHY ports.
	Number of table entries	1M	Table size.
Extended Function PPP termination	Throughput	3/1 Mpps	For 64 / 1518 byte packets.
	Maximum delay processing packets	50 us	Between two PHY ports.
	Number of table entries	100k	Table size.

Table 4-1: Performance requirements

1. **Overload control:**
    a. **The UN MUST remain stable and responsive during high load.**

b. The UN SHOULD support congestion reporting (towards the controllers).

c. The UN SHOULD support mechanisms for controlling the network load and distribute it towards other nodes (both control and data plane).

2. The UN SHOULD make use of the load distribution functionality (e.g. multiple queues and filters) present in the network interface hardware layer.

3. UN data plane SHOULD be able to scale down to hardware platforms targeted for CPE and up to more powerful nodes in the network or in data centers – given that the required resources are available.

4.3 Switching and traffic steering

This section focuses on the switching functionality that takes place inside the UN and which is the basis for implementing traffic steering. Traffic steering is a high level function that takes place in the upper layers outside the node since a network-wide view is needed to perform this functionality (e.g. routing). On the other hand, switching (i.e. forwarding) is a low level function that takes place inside the UN and only the node view is needed in this case. As a consequence, the following list of requirements is oriented to the switching functionality due to the scope of this deliverable.

As an example, figure 4.1 shows one of the use cases of UNIFY. The orange line itself is the logical path of the packet with the different modules and that would be traffic steering, while the individual decisions at each node to find the next one would be switching.

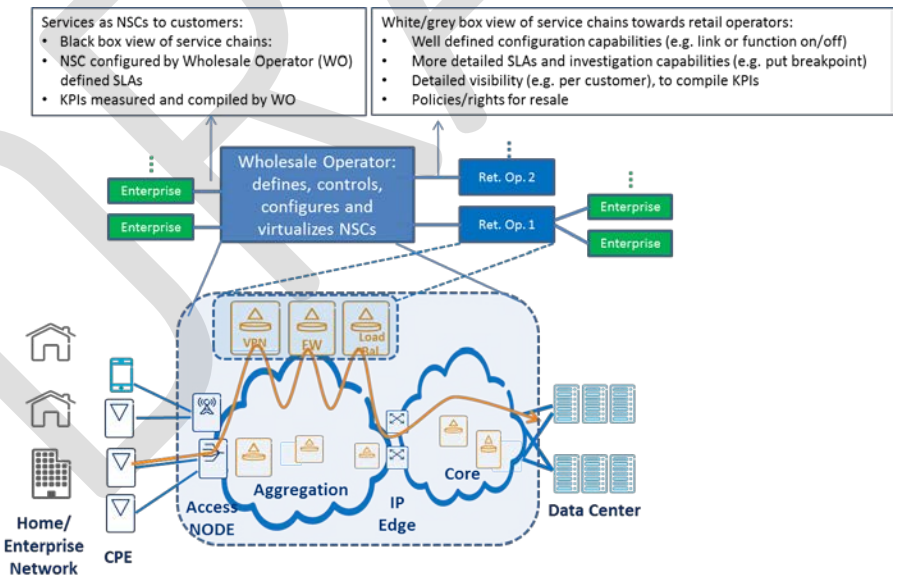

Figure 4.1: Switching and traffic steering in UNIFY use case

1. The UN MUST contain virtual switching functionality that allows switching traffic between the physical ports and the virtual ports of the VNFs present on the node.
2. The UN MUST allow the network controller to dynamically program the virtual switch(es) to enable and control the traffic steering.
3. The UN MUST support switching between physical and virtual ports according to at least the following criteria:
    a. L1: physical port.
    b. L2: VLAN, MAC address, MPLS tag.
    c. L3-L4: IP, port, protocol, DSCP/TOS.
4. The UN MUST NOT introduce any reordering of the packets of IP flows (i.e. the distribution of the traffic between multiple processing cores must be done in such a way that the order of the packets of each IP flow is preserved).
5. The UN MUST provide the mechanisms for the orchestration layer to establish connectivity to the NFs deployed in it (both control and management interfaces).
6. The UN MAY allow the control plane to deploy additional virtual switches (independent switching domains).

4.4 Data plane configuration

The requirements in this section are oriented to configuration management and apply to any configuration related to the data plane. Functional requirements related to the configuration of the capabilities of the UN are included in other sections according to the area they have an impact on.

1. The UN SHOULD support a mechanism to automatically configure the data plane when the node is brought into service.
2. Configuration of network and services SHOULD be performed in a stateful manner: steps are finally completed when acknowledged. If no or negative replies are received from the to-be-configured device, the configuration should be restored.
3. UN configuration data SHOULD be kept separate from operational and statistical data.
4. The UN SHOULD support the configuration of redundant controllers (both for Network and Resource control).

4.5 Monitoring

An important aspect of network operations is the capability to monitor and troubleshoot the infrastructure, which requires dedicated functions to be present in network nodes. This section aims at presenting the requirements that need to be satisfied by the UN in order to guarantee network observability and monitoring, which represent a set of fundamental components to guarantee the proper network operation.

In particular, the rest of this Section will provide a list of the requirements that have to be satisfied by the data plane of the UN to properly support measurement and monitoring functionalities. Notice how the set of mechanisms presented are derived from the ones used in main routers, e.g., the ones covered in standards such as IETF RFC 2863 [12], RFC 2819 [13] and RFC 3273 [14], and, in the context of the Metro Ethernet Forum, in the MEF 31 document [15].

Note that these methods require extra resources from the UN meaning that finer grain observation means higher resource usage. We recognize that this may have a non-negligible impact on the UN and prevent the system to reach the performance targets specified in this document. Because of this all of the monitoring functions are activated on a per request basis. There is always a possibility to further extend the observation functionality by developing additional observation VNFs and deploy them to the UN. Note that part of the functionality described below will most likely also be deployed as VNFs.

The specific requirements for the data plane of the UN are:

1. It MUST support passive measurement methods in order to provide proper estimations of given network metrics (throughput, loss, delay, jitter) on different levels of granularity (packet, flow, links, etc.).
2. It MUST allow Observation Points to instantiate, maintain and update sets of counters. Optionally, it should support also more complex structures (e.g., arrays) instead of single values for counters.
3. It MUST allow packet manipulation to add timestamps, mark a certain packet e.g. for monitoring, tracing, trend analysis, troubleshooting.
4. It MUST support sampling strategies on different levels of granularity (packet, flow, etc.).
5. It MUST support the retrieval of device state information (flow table states and counters, CPU and memory usage, buffers, etc.) on different levels of granularity (per device, per EE, per service function, but also per flow table, per table entry, etc.) to ensure proper SLA and troubleshooting management. In particular, it must allow metering of traffic at least on the following levels:
    a. At the node level (including VM)
    b. At the VNF level

c. At the flow level

Additional monitoring levels (e.g., at the user / customer level) are possible as well and can be achieved by defining the proper filter at the flow level.

6. It MUST support dynamic activation of the above monitoring features, so that these features only run when requested.
7. The same functionality defined for passive measurements SHOULD be supported also with respect to active measurement methods. Examples of management tools adopting these measurements are described in IEEE 802.1ag [16] for LANs and MANs, in ITU-T Y.1731 [17] for the Ethernet technology, or in IETF RFC 6371 [18] for MPLS.
8. It SHOULD provide atomic reads across different counters, or at least provide proper timestamps to ensure coherence among the read values.
9. It SHOULD allow instantiation of software-defined counters, i.e. small pieces of dynamically loaded software that is executed directly in the data plane (and not in dedicated VNF), and enable arithmetic operations on them (e.g., counting of total VoIP traffic by summing values retrieved by counters related to different VoIP protocols).
10. It SHOULD allow Observation Points to perform packet lookups up to L7, at least on a per-packet base

In addition to the above requirements, we can envision additional requirements that should be implemented in dedicated Virtual Network Functions but that may have an impact on the data plane as well. Particularly:

1. Observation Points MUST be able to exchange messages between the nodes where they are running on (should be possible between the basic existing NICs).
2. Observation Points SHOULD be able to perform packet lookups up to L7, for at least the first given bytes of a flow (e.g. 2048-bytes of HTTP packet as in today's security gateways).

4.6 Security requirements

Regarding the security requirements for the Universal Node, it's considered that most of the general requirements in the area of security deal with layers considered in other Work Packages that will deploy some services over the Universal Node. Nevertheless, and even if the architecture from WP2 is not available, it's thought that access to the low level configuration and management of the UN by an infrastructure owner will be needed and that, in this scenario, it should take into account some specific security requirements.

1. Access to the UN SHOULD be protected using appropriate standard security mechanisms, wherever applicable and convenient, like authentication, authorization, data encryption, data confidentiality and data integrity.
2. Access to the UN configuration and management functionalities SHOULD be divided in multiple subsets according to support different levels of privilege of the authenticated entity.
3. A register of security related events SHOULD be maintained locally or sent by a trusted method to a central location.
4. Access to the UN SHOULD include mechanisms that will deter brute force attempts or denial of service attacks.
5. The UN interfaces SHOULD support encryption where applicable in order to support secure scenarios.

4.7 Selected use case requirements

BNG provides an access point for the subscribers to the broadband services provided by the Service Provider (SP). The BNG aggregates traffic from a large set of subscribers connecting over an access network, and routes it to the network of the service provider. As the entry point into the SP network, the BNG also manages subscriber sessions, possibly including the following aspects:

- Control plane aspects:
  - Authentication (including the possibility to enter credentials in web-interfaces), Authorization (including per service policing) and Accounting.
  - Address assignment incl. first-ip-hop (DHCP or IPv6 address autoconfiguration).
  - Support of DDNS.
- Data plane aspects:
  - Quality of Service, e.g. priority handling, bandwidth limitation.
  - Multicast support.
  - Forwarding / tunneling the traffic of given customers to other operators/providers.
  - Ability to support VPN pass-through.

Today, the typical NSP+ISP scenario (where an operator requests the access to a customer of another one) is implemented in the BNG with tunnels (VPN, L2TP....) between the BNG of the providing operator and a gateway in the network of the requesting provider.

In the future other models are imaginable as well as detailed in use case "wholesale operator":

- BNG could provide virtualized physical resources, where other operators could install their own functions (like the IaaS model).
- BNG could provide software resources, e.g. different AAA blocks, data path protocols, which are used by other operators in a "drag & drop app store" manner (PaaS model).
- Combination of both.
- BNG should support various interfaces (including mobile) for connecting users or other operators.

This leads to new requirements / functional aspects like:

- Network isolation.
- Resource provisioning, monitoring and enforcement.

4.7.1 Use case performance requirements

In this section the UN refers to a single x86 blade which contains 2 x86 CPUs with 6-8 cores each. In other hardware scenarios (e.g. x86 with accelerators), the UN refers to a node with approximately the same power budget, which is around 200W. If there are alternatives with significantly different power characteristics – either above or below – it should be noted.

In order to assess the performance of the UN in the BNG use cases, the following parameters will need to be defined:

- Number of active subscribers handled by a single BNG = 100k.
- Total BNG bandwidth: Mpps with 64B and 1500B long packets.
  - Basic functionality (tunneling, routing) = 50 Mpps.
  - With all features enabled = 30 Mpps.
- Maximum bandwidth per subscriber (up and down):
  - Without rate limiting = 10 Gbps.
  - With rate limiting = 5 Gbps.

- **Maximum bandwidth of an IP flow (5 tuple), two direction = 10 Gbps.**
- **Number of entries in the routing table (uplink) = 1M.**

5 Functional specification

To make sure the vision is complete and coherent, the approach is based on a series of processes related to the data-plane that are supported by the different functional blocks detailed. At the same time, all the requirements identified in the previous section are covered by those functional blocks.

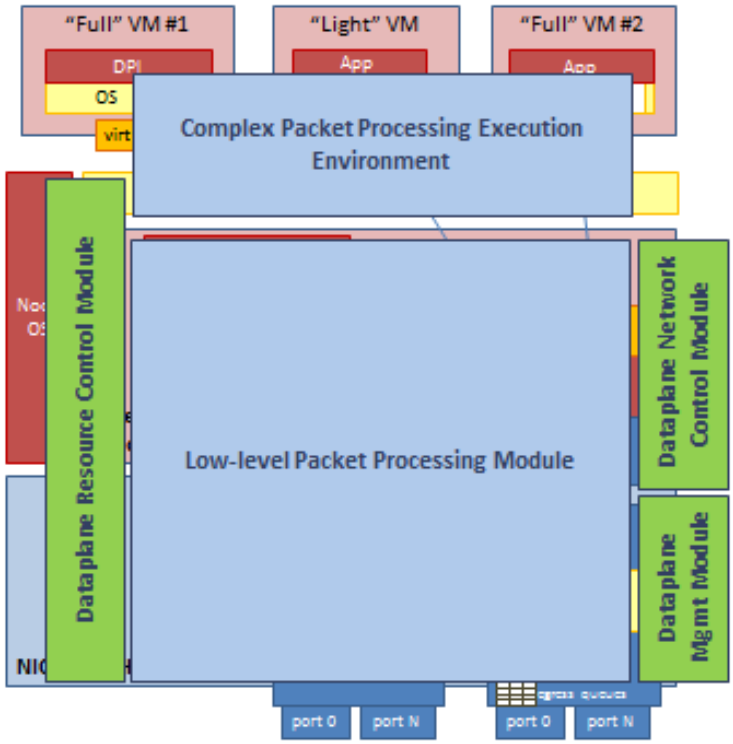

Figure 5.1: Current UN architecture – modular view

The blocks can be logically separated to packet processing blocks (blue) and control blocks (green)

5.1 Packet processing blocks

There are two logically separated packet processing blocks: the low-level packet processing module and the complex packet processing execution environment.

Note that hardware accelerators can be handled two ways: network chips that extend the NIC functionality can be seen as an extended I/O layer, while application accelerators (such as crypto engines) can be seen as light VMs, which in this case implement the application code in hardware.

5.1.1 Low-level packet processing module

This module is the engine executing the high-performance packet processing tasks, such as:

1. Traffic switching between physical and virtual ports according to configuration from control module.
2. Load balancing, link aggregation, ECMP.
3. Basic packet processing functionalities, such as:
    a. Ethernet / MPLS switching
    b. IP forwarding (IPv4, IPv6)
    c. Tunneling (encapsulation/decapsulation) except security tunnels.
4. QoS related functions, such as:
    a. Rate limiting.
    b. Marking.
    c. Priority based queuing, color based scheduler, hierarchical queuing, probably also WFQ.
5. Other low-level packet processing:
    a. Firewall with simple L2-L4 rules.
    b. Port based NAT.

The module's capabilities and control are exposed to the UN users through the DP network control module.

5.1.2 Complex packet processing execution environment

This block is responsible for supplying an execution environment for the more complex or 3rd party packet processing (or server) applications. The EE's main features that would be used by the applications:

1. Virtual port handling.
2. Resource separation, resource control.
3. Intra node load balancing, load distribution.

5.2 Control blocks

They are controlled by 3 logically different control modules: the DP resource control module, the DP network control module, and the DP management module.

5.2.1 Data plane management (DP-M) module

This module includes bootstrapping of the node, some general functions (e.g. security) and dynamic monitoring of required UN counters or features.

1. Bootstrapping the UN with regards to the data plane.
    a. The UN owner defines the default configuration for when the UN starts.
    b. On start the UN applies the defined configuration that would include at least.
        i. Setting up the interfaces to connect to the control plane (both resource and network controllers).
        ii. Deploying the default virtual switch.
        iii. Deploying the DP-RC and DP-NC modules.
2. Supports security features related to control plane communication (e.g. manage encryption of control messages between the UN control modules and their respective controllers).
3. Metrics configuration and collection.
    a. The UN user selects through the control plane the metrics to be collected.
    b. The DP-M validates/parses the request, configures the required counters/observation points, if required, start observation processes.
    c. After a given time or periodically, the DP-M recovers the information from these sources and sends the answer to the UN user.

5.2.2 Data plane resource control (DP-RC) module

This module includes all functional aspects that the UN owner requires in order to configure and manage the resources in the data plane and make them available to the UN users.

The DP-RC module has the following responsibilities:

1. Initiates and keeps contact with the Resource Controller.
2. Hides hardware-specific details: abstracts resources before exposing them to the resource controller, does the abstract resources ←→ physical resources mapping.
3. Manages compute, storage and network resource sharing.
4. Manages UN extra capabilities (accelerators, lightweight apps).
    a. Exposes the available capabilities through the resource controller.
    b. Through the resource controller the UN user requests the instantiation of one or more of the available capabilities or the removal of such resources.

c. The DP-RC creates an instance of the capability for the requesting UN user or removes it when it is not required anymore.

5. Manages network slicing to provide traffic isolation.

6. Maps queues to virtual ports.

5.2.3 Data plane network control (DP-NC) module

There are two possible approaches for controlling the networking part of the UN. One approach is that the UN users control their respective virtual network resources directly, so there would be direct control connection between the UN and the UN users network controller. The other approach is that the UN users control the UN node via the infrastructure provider's network controller. This would allow the infrastructure provider to hide the details of the UN node.

Whichever solution will be used we need to handle the network control interface in the UN node with the DP-NC module. This module implements the interface through which the UN user(s) control the UN network resources that have been assigned to them.

1. Interfaces with the Network Controller(s) allowing the UN user(s) to control switching between the physical and virtual ports in the low-level packet processing module.

2. Manages virtual switch configuration.

   a. The DP-NC module exposes switching capabilities through the NC(s).

   b. Through the NC(s) the UN User requests the configuration of a packet processing rule (add/modify/delete).

   c. The DP-NC parses and validates (from the isolation point of view) the request and applies the processing rule if it is valid by inserting it into the low-level packet processing module.

3. Exposes networking metrics from the data plane.

4. Implements required security measures to restrict access (e.g. authentication and encryption of the communication with the UN users to prevent access to resources allocated to another user).

Security is considered through all the processes. This does not imply that the security related functions must be executed for each step, for example: a secure channel could be used for all the interactions of the corresponding actor with the UN, it would be established once and used transparently by all the other processes. The approach to be used in the Universal Node will be defined later on.

5.3 Mapping of selected use case requirements to functional specification

This section aims at highlighting which portions of the functional specification of the UN will be involved to satisfy the requirements of the selected use cases.

Most of the control plane requirements (e.g., user authentication) are not under the responsibility of the data plane of the UN and must be handled by the component in charge of the control plane of the network. However, some aspects may have an impact on the UN operations. For instance, the arrival of a new client *may* trigger some reconfiguration actions in the data plane, possibly involving different modules. Some examples are the following.

- **Creation of a new network partition (e.g., new VLAN) and/or update of an existing network partition in order to handle the traffic of the new user.**
- **Reconfiguration of the virtual switches of the UN in order to redirect some portion of traffic (e.g., the traffic to/from the new user) to an existing network service.**
- **Reconfiguration of some existing modules implementing selected data plane functions such as QoS, traffic shaping, tunneling, etc.) in order to configure the appropriate parameters for the new user.**
- **Creation of a new virtual switch in the UN that has to handle the traffic of a new user that is not a customer of the network operator (i.e., it belongs to a different ISP)[1]. This *may* require also the allocation of a new VM that contains the network services associated to that ISP and the consequent reconfiguration of the switching path in the UN in order to redirect the traffic to the proper functions.**

Apart for the case in which users belong to different ISP, users with similar requirements are expected to be handled together in an aggregated way, so there would no need for specific virtual machines for the different users and thus we do not foresee the necessity to start new network services, when a new user connects. In fact, we assume that the required network services are already active and hence only a network reconfiguration step is required.

[1] **We assume that the network operator will assign a different virtual switch to each ISP in order to guarantee better traffic isolation.**

List of abbreviations and acronyms

Abbreviation	Meaning
AAA	Authentication, Authorization, Accounting
API	Application Programming Interface
BNG	Broadband Network Gateway
BRAS	Broadband Remote Access Service
CE	Customer Edge
CPE	Customer Premise Equipment
CPU	Central Processing Unit
DDNS	Dynamic Domain Name System
DHCP	Dynamic Host Configuration Protocol
DP	Data plane
DPDK	Data Plane Development Kit
DP-M	Data plane management
DP-NC	Data plane network control
DP-RC	Data plane resource control
DSLAM	Digital Subscriber Line Access Multiplexer
ECMP	Equal Cost Multipath routing
EE	Execution Environment
ETSI	European Telecommunications Standards Institute
GRE	Generic Routing Encapsulation
IETF	Internet Engineering Task Force
HTTP	Hypertext Transfer Protocol
IEEE	Institute of Electrical and Electronic Engineers
IO	Input Output
IP	Internet Protocol
ISP	Internet Service Provider
ITU-T	International Telecommunication Union – Telecommunication Standardization Section
L2TP	Layer 2 Tunneling Protocol
LAN	Local Area Network
MAN	Metropolitan Area Network
MEF	Metro Ethernet Forum
MPLS	Multiprotocol Label Switching

NAT	Network Address Translation
NC	Network controller
NF	Network Function
NIC	Network Interface Controller
NSP	Network Service Provider
OVS	Open vSwitch
OS	Operating System
PaaS	Platform as a Service
PPP	Point-to-Point Protocol
PPPoE	Point-to-Point Protocol over Ethernet
QoS	Quality of Service
QoE	Quality of Experience
RFC	Request For Comments
SP	Service Provider
UN	Universal Node
VLAN	Virtual LAN
VM	Virtual Machine
VNF	Virtual Network Function
VoIP	Voice over IP
VPN	Virtual Private Network
WP	Work Package

Appendix: Requirements extracted from MS2.1

This appendix contains an excerpt of milestone MS2.1, an early draft of deliverable D2.1 available at the time of writing, with those requirements from Work Packages WP2, WP3 and WP5 that could have an impact on the data plane. Requirements from WP5 have been directly considered instead of derived from the milestone document.

4.1 Requirements from WP2 (Use Cases and Architecture)

Generic requirements

4.1.1 Business requirements from the operator side

The proposed main requirements that an operator would like are:

- **The framework shall allow multiple operator / providers to access the control, management and data planes**
- **(…)**

4.1.2 Operational requirements

These are the main operational requirements from an operator's point of view:

- **Configuration data should be separated from operational and statistical data at the various devices/systems used in the process,**
- **(…)**
- **Configuration of network and services should be performed in a stateful manner so that steps are finally completed when acknowledged or if no or negative replies are received from the to be configured device, the original configuration must be restored**
- **(…)**

4.2 Requirements from WP3 (Service Programming, Orchestration and Optimization)

(…)

4.2.2.2 Orchestration

The abstract description of the service graphs has to be translated into different types of resources and the components have to be configured and controlled appropriately. This task is done by the orchestration

component. Orchestration and its subtasks are mainly performed automatically, however, providing controllability/programmability of some algorithms would be desirable.

Orchestration must be able to

- (...)
- establish connectivity to the network function
- (...)
- steer traffic to these network functions according to desired network service chain.

(...)

Orchestration should support dynamic/on-line operations and must be able to

- on-line modify SG connectivity and connectivity requirements as well as NF requirements
- (...)

Orchestrator has to optimize network behavior, resource usage, etc. in a flexible way. The optimization task should be done

- (...)
- supporting dynamic/on-line reconfiguration/re-optimization during a session.
- (...)

Beside automatic (default) optimization, the flexible configuration of the mechanism should also be supported where

- (...)
- manual pinning of NFs to certain resources is also supported
- (...)

4.3 Requirements from WP4 (Service Provider DevOps)

This section presents the requirements derived from the initial ideas of the SP-DevOps concept, as discussed in the DoW [ref] as well as in recent research papers published by the UNIFY consortium [DCC paper] and [SDN4FNS].

(...)

4.3.1 Monitoring and Observability

(...)

Nr.	WP4 Requirement Description	Arch. layer	Scope	UCG
2	Observability points should be able to observe basic network metrics (throughput, loss, delay, jitter) on different levels of granularity (packet, flow, links, etc.) through active and passive methods (retrieval of counters, active measurements through OAM tools, etc.)	DP	Infra	1-3
3	Observability points should be able to observe physical and logical device states (flow table states and counters, EE CPU and memory usage, buffers, etc.) on different levels of granularities (per device, per EE, per service function, or per flow table, per table entry etc.) to provide KPIs in order to ensure SLAs and troubleshooting purposes	DP	Infra	1-3
5	The DevOps framework requires computational and storage resources in physical and virtual nodes (super set of requirements 5a to 5e)	DP	Infra	1-3
5a	The DevOps framework should be able to instantiate additional counters in physical/virtual devices	DP/ RC	Infra	1-3
5b	Universal nodes should allow for performing arithmetic operations on the counters data on the physical/virtual device	DP	Infra	1-3
5c	Universal nodes should allow for performing packet manipulation (for adding timestamps, mark a certain packet for monitoring, tracing, trend analysis, troubleshooting) on the counters data on the physical/virtual devices	DP	Infra	1-3
5d	Universal nodes should allow for implementation of simple decision function on the physical/virtual device (thresholds for notifications, aggregation of events, etc.)	DP	Infra	1-3
5e	The DevOps framework should allow for implementation of configurable sampling strategies on the physical/virtual device on different levels of granularity (packet, flow, etc.)	DP	Infra	1-3
6	Observability points should be able to asynchronously report exceptions wrt. to pre-configured or adaptively set thresholds and limits (changes, performance degradations, SLO breaches, etc.)	DP	Infra	1-3
7	Observability points should be able to regularly report observability data in user specified time intervals, with a time-granularity of at least XX ms.	DP	Infra	1-3
8	Observability points should have the capability to perform packet lookups up to L7, for at least the first XXX bytes of a flow (e.g. 2048-bytes of HTTP packet in todays security gateways)	DP	Infra	1-3
9	The reaction time of observability points should (optionally) be specifiable in terms of an upper limit	DP/ RC	Infra	1-3

4.3.2 Validation and Verification

Validation and verification tools are part of both service orchestration and resource control layers of the architecture and offer possibilities to debug service components during design time to ensure intended functionality, as well as ensuring resource availability and verification of parameters settings during deployment time based on the specifications of service-chains and components. During deployment time, the validation and verification tools may need to operate within specified bounds (e.g. time limits) to ensure rapid service-chain (re-)deployment. The majority of the currently identified requirements apply to the domain level.

Nr.	WP4 Requirement Description	Arch. layer	Scope	UCG
16	Verification of service chain deployment should be possible wrt resource dimensioning and availability of the resource assigned to the particular service chain	RC/SO	SC	1-3
17	Verification of service chains should be possible during (re-)deployment time and operations wrt to customers specified policies, e.g resource availability and specified resource dimensions	RC/SO	SC	1-3
21	Universal nodes may need to support sandboxed EE for untrusted service functions and validation purposes	DP	Infra	1-3

4.3.3 Troubleshooting

Troubleshooting mechanisms will need to operate on several levels of the architecture and should be able to initiate observability points as part of localizing the cause of a fault or performance degradation. Current troubleshooting requirements apply mainly to the domain level. The outcome of a troubleshooting session may be used to notify the service owner, and/or as input to the resource control/service orchestration parts for resource management to ensure continuous service delivery (e.g. trigger re-optimization of resources). To properly perform troubleshooting, the DevOps framework should in general have possibility to retrieve current and historical information from physical and virtual entities. Troubleshooting may include e.g. identifying the cause of a performance degradation to rule inconsistencies or physical problems in the infrastructure.

Nr.	WP4 Requirement Description	Arch. layer	Scope	UCG
22	The DevOps framework should be able to trigger new observability points, reconfigure existing observability point or read data from already existing observability points for temporary troubleshooting purposes	RC	SC	1-3
23	The DevOps framework should have access to current states of physical/virtual devices, as well as access to timestamped historical data, logs, and events stored in the physical/virtual devices in (close to) real time	RC/SO	NF	1-3
24	The DevOps framework should have access to information about network topology, device types,	RC/SO	Misc	1-3

	device capabilities, etc. In (close to) real time.			

(…)

4.3.5 General WP4 SP-DevOps requirements

Availability of timestamps and possibility to aggregate observability information from different physical/virtual entities throughout domains are important features to perform higher-level analytics from both monitoring and troubleshooting perspectives, offering possibilities to monitor different aspects within and across services as well as the infrastructure for the purpose of e.g. detecting and localize performance degradations and managing resources efficiently. As these requirements span over several of the categories above we here list them separately.

Nr.	WP4 Requirement Description	Arch. layer	Scope	UCG
29	Observability points should have access to timestamping capabilities inside physical and virtual network elements and functions (e.g. EE and universal node)	DP	NF	1-3

References

[1] L. M. Z. L. K. Z. T. Gergely Pongracz, "Cheap Silicon: a Myth or Reality?Picking the right dataplane hardware for Software Defined Networking," 16 August 2013. [Online]. Available: http://conferences.sigcomm.org/sigcomm/2013/hotsdn.php. [Accessed 4 March 2014].

[2] "ClickOS and the Art of Network Function Virtualization," [Online]. [Accessed 13 February 2014].

[3] BISDN, "The eXtensible DataPath Daemon (xDPd)," [Online]. Available: https://www.codebasin.net/redmine/projects/xdpd/wiki. [Accessed 13 february 2014].

[4] Open Networking Foundation, "OpenFlow Switch Specification v1.3.3," 13 September 2013. [Online]. Available: https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-spec-v1.3.3.pdf. [Accessed 10 February 2014].

[5] M. C. G. C. Luigi Rizzo, "Transparent acceleration of software packet forwarding using netmap," 25-30 March 2012. [Online]. Available: http://info.iet.unipi.it/~luigi/netmap/20110729-rizzo-infocom.pdf. [Accessed 5 March 2014].

[6] Intel, "Intel® DPDK vSwitch - OVS," [Online]. Available: https://01.org/packet-processing/intel%C2%AE-ovdk. [Accessed 13 February 2014].

[7] "OpenStack," [Online]. Available: https://www.openstack.org/. [Accessed 13 February 2014].

[8] "Open Daylight," [Online]. Available: http://www.opendaylight.org/. [Accessed 13 February 2014].

[9] ON.LAB, "FlowVisor," [Online]. Available: http://onlab.us/flowvisor.html. [Accessed 18 February 2014].

[10] ON.LAB, "OpenVirteX," [Online]. Available: http://tools.onlab.us/ovx.html. [Accessed 13 February 2014].

[11] Internet Engineering Task Force - Networking Group, "RFC2119 - Key words for use in RFCs to Indicate Requirement Levels," March 1997. [Online]. Available: http://tools.ietf.org/search/rfc2119. [Accessed 13 February 2014].

[12] Internet Engineering Task Force - Network Working Group, "RFC2863 - The Interfaces Group MIB," June 2000. [Online]. Available: http://tools.ietf.org/search/rfc2863. [Accessed 13 February 2014].

[13] Internet Engineering Task Force - Network Working Group, "RFC2819 - Remote Network Monitoring Management Information Base," May 2000. [Online]. Available: http://tools.ietf.org/search/rfc2819. [Accessed 13 February 2014].

[14] Internet Engineering Task Force - Network Working Group, "RFC3273 - Remote Network Monitoring Management Information Base for High Capacity Networks," July 2002. [Online]. Available: http://tools.ietf.org/search/rfc3273. [Accessed 13 February 2014].

[15] The Metro Ethernet Forum, "Service OAM Fault Management Definition of Managed Objects," January 2011. [Online]. Available: http://metroethernetforum.org/Assets/Technical_Specifications/PDF/MEF_31.pdf. [Accessed 13 February 2014].

[16] Institute of Electrical and Electronic Engineers, Inc., "802.1ag Connectivity Fault Management," December 2007. [Online]. Available: http://www.ieee802.org/1/pages/802.1ag.html. [Accessed 13 February 2014].

[17] International Telecommunication Union – Telecommunication Standardization Section, "Y.1731 : OAM functions and mechanisms for Ethernet based networks," November 2013. [Online]. Available: http://www.itu.int/rec/T-REC-Y.1731/en. [Accessed 13 February 2014].

[18] Internet Engineering Task Force (IETF), "RFC6371 - Operations, Administration, and Maintenance Framework for MPLS-Based Transport Networks," September 2011. [Online]. Available: https://tools.ietf.org/html/rfc6371. [Accessed 13 February 2014].

[19] "Open vSwitch," [Online]. Available: http://openvswitch.org/. [Accessed 13 February 2014].