A Service-Aware Virtualized Software-Defined Infrastructure

Lefteris Mamatas, *Member, IEEE,* Stuart Clayman, *Member, IEEE,* and Alex Galis, *Member, IEEE*

Abstract

The Internet infrastructure is gradually improving its flexibility and adaptability due to the incorporation of new promising technologies, such as the Software-Defined Networks and the Network Function Virtualization. The main goal is to meet the diverse communication needs of the users, while the global system operation satisfies the business and societal goals of the infrastructure and service providers. This calls for solutions that consider both local and global network viewpoints and provide sophisticated system control in a stable and predictable way, while being service-aware.

We propose a fully integrated solution along these lines, the Very Lightweight Software-Driven Network and Services Platform (VLSP) - a service-aware Software-Defined Infrastructure for Networks and Clouds. The VLSP consists of three main distributed systems: (i) a facility performing uniformly logically-centralized management and control of the infrastructure, called the Virtual Infrastructure Management, (ii) an information management infrastructure being able to maintain an accurate view of the infrastructure environment at both local and system levels, called the Virtual Infrastructure Information Service, and (iii) a lightweight virtualization hypervisor able to perform configuration changes in the infrastructure resources, called the Lightweight Network Hypervisor. We discuss representative use-case scenarios, while we demonstrate how VLSP tunes performance trade-offs for particular service demands.

I. Introduction and Motivation

The Internet is a global infrastructure that accommodates a wide range of applications with diverse Quality of Service requirements. A primary goal is to utilize the physical resources

L. Mamatas, S. Clayman and A. Galis are with the Department of Electronic and Electrical Engineering, University College London, London, UK. E-mails: {l.mamatas, s.clayman, a.galis}@ucl.ac.uk. L. Mamatas is currently a lecturer at the University of Macedonia, Greece.

according to the needs of the deployed network services, while at the same time Internet stakeholders, such as the network and service providers, target the realization of services in a manner consistent with their business plans. This diversity calls for flexibility resource control and management, with programmability and improved determinism in the system behaviour.

Along these lines, Programmable Networks [15], Software-Defined Networks (SDNs) and OpenFlow [1] were introduced. SDNs are characterized by: (i) decoupled network control from forwarding, with control embedded in a logically-centralized component, (ii) programmability via software functions interacting with the network, and (iii) appropriate abstractions that allow SDN applications and services to be network-aware. OpenFlow is a de jure standardized way to control flow tables in switches and routers. It allows a logically-centralized software application which has a global viewpoint of the network, called an SDN Controller, to interact with the network equipment and make changes in the flow tables of the network equipment. Higher-level SDN Applications interact with one or more SDN Controllers in order to manipulate their general behaviour, and to achieve significant performance improvements [2]. Two survey papers that cover the area of SDNs are [3] and [4].

The Network Function Virtualization (NFV) architectural concept [5] brings networks closer to IT domains and their related operations. NFV targets both flexibility in service provisioning and reduction of cost expenditure. SDNs and NFV, although independent, can be mutually beneficial and may co-exist in the same network environment. SDNs offer flexibility at the network control level, while virtualization is a good candidate technology for hiding network device heterogeneity.

Based on the above, we have envisaged the following *four research challenges* for the future evolution of SDN/NFV technologies with the aim of improving service-awareness in the infrastructure:

Ch 1. The enablement of new SDN applications, beyond centralized traffic engineering, without being constrained by existing hardware characteristics. Virtualization can hide the heterogeneity at the hardware level and can serve as a migration path towards adopting SDN-like technologies.

Ch 2. The introduction of new abstractions for realising sophisticated management features on top of flexible and programmable network control components. Such technologies will bridge the gap between the local viewpoints (i.e. solutions handling network control issues) and the global viewpoints (i.e. higher-level management features).

Ch 3. The fast and simple deployment of network resources that collectively support a large number of services, from the level of a single node up to a large-scale network.

Ch 4. The maintenance of a global picture at both network level and domain level, using logically-centralized intelligence, programmable techniques, and an abstracted design.

Consequently, a high-level unification and integration of the Virtualization, Network Function Virtualization, SDN, Programmability, and Management would need to be achieved. This calls for new infrastructure architectures, standardized interfaces, and management applications. As such, we foresee that the research focus would move towards the incorporation of SDN with NFV, whereby novel network management approaches can bring services and networks closer.

Targeting the above challenges, we present the Very Lightweight Software-Driven Network and Services Platform (VLSP) - a fully integrated open-source software-defined infrastructure and architecture for networks and clouds which we have designed and implemented from the scratch. VLSP is differentiated from SDN architectures [6] and Open Solution initiatives [7] by having the following architectural features:

(i) a deeper integration of NFVs and SDNs by introducing uniform virtualization of networks and applications,

(ii) unified management and control for both networks and clouds, reducing the management cost and complexity,

(iii) suitability for reliability and scalability evaluation through use of a lightweight virtualization hypervisor.

Additionally, VLSP supports the following novel features:

1) A software execution environment within the virtual routers, allowing the deployment, at run-time, of diverse network control and service components in order to enable programmability capabilities.
2) A focus on elastic, adaptable, and autonomic service provisioning and management. Supporting lightweight application components being realized as virtual resources deployed and managed uniformly by the same environment.
3) Hierarchical and distributed control components that offer scalable and logically-centralized network control and management, without overloading centralized software nodes. This enables the autonomic network and service management logic to be designed and operated

on a global network view.

4) An integrated and abstracted state information manipulation facility, building the global-picture for the system, while supporting local level and domain level views, yet adaptable to the diverse requirements of the involved entities.

In figure 1, we show the VLSP architecture, components and their relation [6] [7] to the ONF SDN ITU-T Y.3300 SDN, IRFT RFC 7426, ETSI NFV, OpenStack Neutron, and OpenDaylight architectures. ONF is working on OpenFlow standardization aspects and ETSI is studying the architecture of NFV from the network operators' perspective. Neutron augments OpenStack clouds with *networking as a service* capabilities, and is based on a loosely-coupled architecture, whereby service and virtual network device plugins realize the targeted behaviour and are hidden behind a common API. All available plugins present different performance trade-offs, scalability, manageability, and compatibility aspects, which cloud operators should weigh-up for themselves.

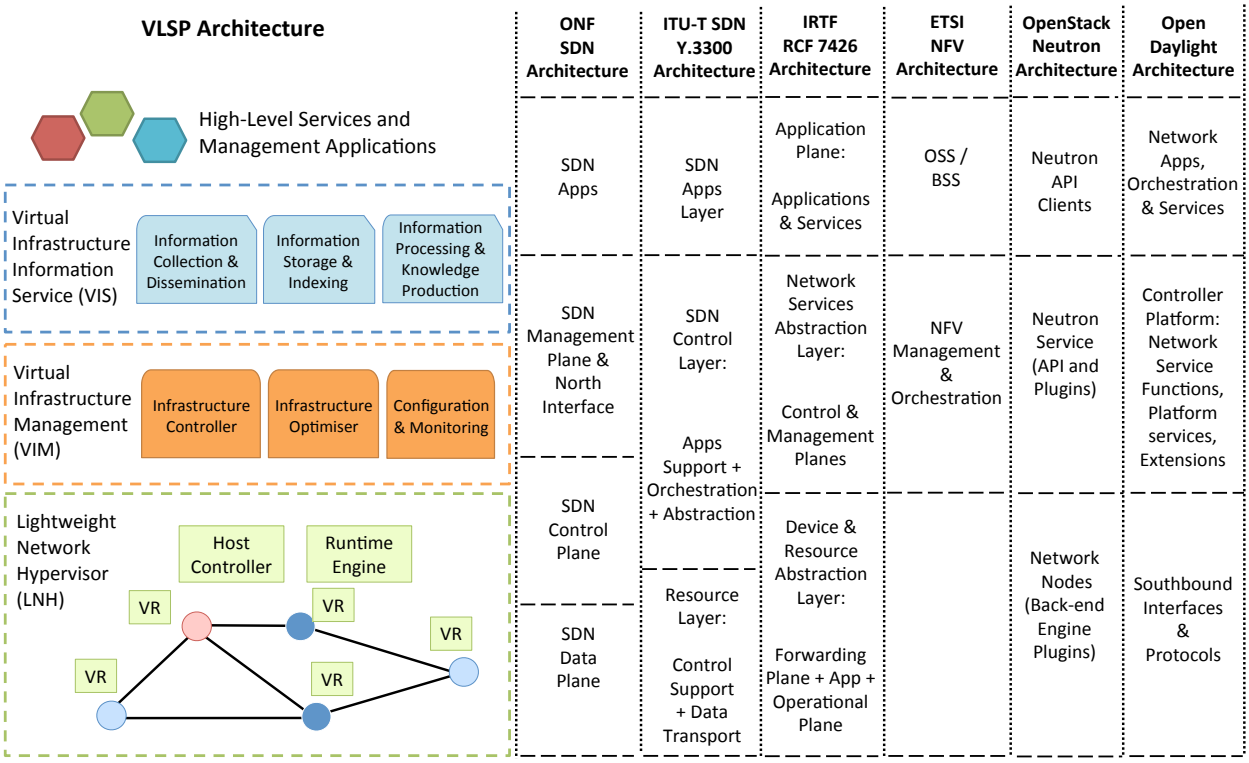


Fig. 1. VLSP architecture, components and relation to ONF SDN, ITU-T SDN, IRTF SDN, ETSI NFV, OpenStack Neutron and OpenDaylight architectures

Several SDN/NFV efforts have appeared within the IETF in the form of working groups or

working group initiatives. The data plane oriented approaches are enabling higher-level control (e.g. the Interface to the Routing System and the Abstraction and Control of Transport Networks groups), with some of them introducing application-awareness (e.g. the Application-Layer Traffic Optimization group). The Virtual Network Function Pools and the Network Function Virtualization Configuration groups propose design details for a more efficient usage of NFVs. The Service Function Chaining working group is studying the deployment of service functions in large-scale environments.

The ITU's Telecommunication Standardization Sector (ITU-T) hosts study groups on SDNs. SG13 focuses on future networks (e.g. clouds, mobile, virtual networks) and SG11 on relevant network protocols. In the IRTF, the Software-Defined Networking Research Group identifies future research challenges, including scalability, abstractions, security, and programming languages for SDN environments. Other SDN initiatives have a stronger focus on combining SDNs with NFV, including the IEEE SDN technical community.

Many researchers have pointed out the importance of both reliability and scalability in SDN environments. Shalimov et al [8] carried out an extensive study of seven SDN/OpenFlow controllers. They claim, based on their results, that current controllers do not scale well over the network cores and that they are not able to meet the increasing demands in communication. In our case, VLSP considers scalability as a basic design requirement.

In another case, Google documented an outage incident in its SDN WAN deployment [2], which could have been avoided if latency sensitive operations have received higher priority, compared to the throughput-intensive ones. The outage problem was detected at a very late stage, due to the lack of enough performance profiling and reporting. Levin at al. assess how inconsistency of SDN control state information significantly degrades performance of logically-centralized control applications [9]. In their work and many others, state information management is integrated within a corresponding SDN application or SDN controller.

From our point of view, the state information, its manipulation, and the exchange capabilities can be abstracted away and realized through a separate component in the SDN architecture. The SDN applications and controllers should be able to communicate information based on their own diverse requirements and constraints. This requires not only supporting alternative methods to create the network-wide state, but also a flexible way to choose the most appropriate configuration each time.

In contrast to the related initiatives, we target our identified *four research challenges*, which we foresee as main research trends in the near future. We focus on aspects such as service-awareness, a better and uniform control, all of which are beyond performance issues, network protocol issues, and the existing constraints of deployed infrastructures. VLSP is based on a lightweight virtual router implementation, suitable for scalability and reliability evaluations. As we highlight in figure 1, the VLSP components could be integrated into existing deployed infrastructures at their equivalent architectural blocks, since they use similar design strategies (e.g. RESTful communication).

Section II discusses the VLSP design details and architecture. Section III provides representative use-case scenarios of our platform. Section IV highlights our experimental methodology and provides experimental results. Section V discusses the lessons we have learned during its design and implementation. Finally, section IV concludes this paper.

II. Platform Design and Architecture

The VLSP provides reusable management and control facilities that are utilized by additional software entities, called Management and Control Entities (MCEs) in this paper. These MCEs, when combined with the VLSP enable logically-centralized management and control of the system. A categorization of the different types of MCE, from a deployment related perspective, follows:

1) High-Level Services and Management Applications, which are responsible for the efficient operation of the whole system, at both the network and service levels. They take optimization decisions based on the global picture.
2) MCEs deployed at the physical hosts, controlling a part of the network, such as the SDN Controllers.
3) MCEs deployed at the virtual routers, which are responsible for resource-facing operations at the virtualization hypervisor level.

Figure 1 gives a high-level overview of the VLSP. An earlier version of the VIM and the LNH layers are presented in paper [10]. We have done further design and implemented an integrated working version of the VLSP and released it as open-source software[1]. In the following

[1]The VIS, the VIM, and the LNH components, the relevant documentation, and research papers can all be downloaded from: http://clayfour.ee.ucl.ac.uk

subsections, we elaborate all three of the VLSP layers and their main functions.

A. The Virtual Infrastructure Information Service

The VIS offers abstracted and logically-centralized information manipulation across all of the deployed software entities. In figure 2, we present a high-level view of the VIS architecture and its basic interactions. The VIS uses two separate interfaces for communication with the MCEs, namely: (1) the *Information Management Interface* which handles the configuration of information manipulation, including the MCE's registration to the VIS, the management of internal VIS information manipulation functions and the establishment, operation, and optimization of information flows; and (2) the *Information Exchange Interface* which offers the actual information transfer and exchange capability to the deployed MCEs.

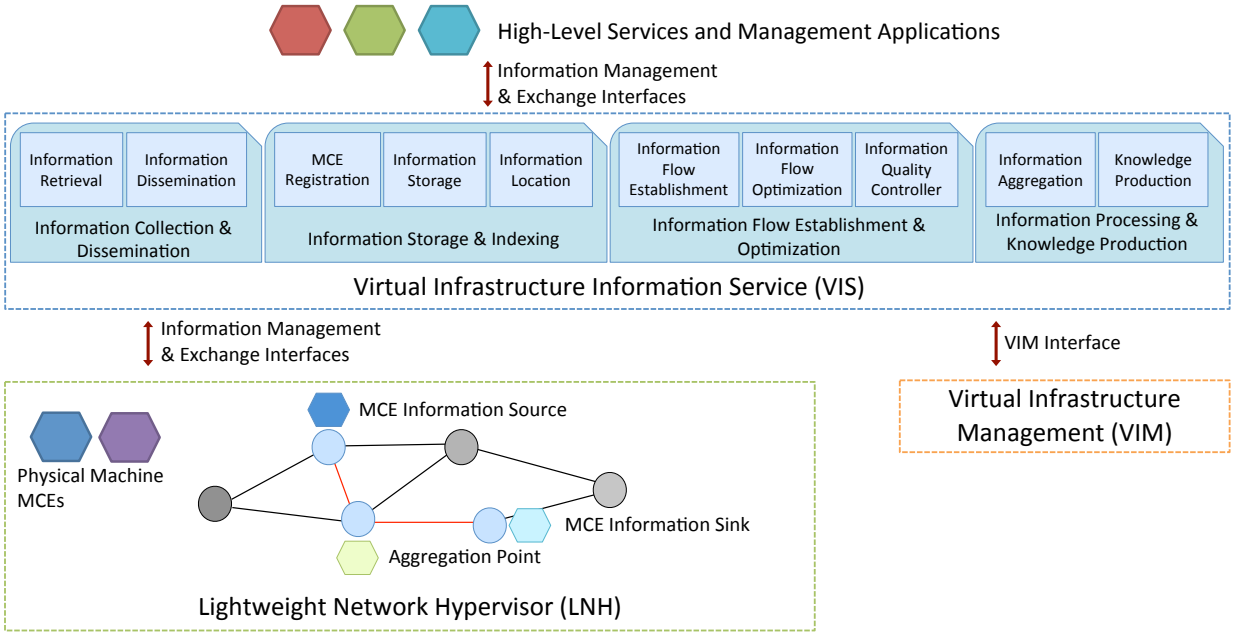


Fig. 2. The Virtual Infrastructure Information Service

The VIS has the following core functions:

- *Information Collection and Dissemination (ICD).* The ICD is responsible for organizing the communication of information, including the optimization of information flows. It offers facilities for *Information Retrieval* and *Information Dissemination*, plus the *Information Flow Controller*. Alternate selectable communication methods are supported, such as the

Push/Pull, Publish/Subscribe, and Direct Communication methods. The *Information Flow Controller* oversees such functions, including controlling the information flow establishment, operation, and relevant optimization aspects.

- *Information Storage and Indexing (ISI).* The ISI provides storage and indexing functionalities to the VIS. The *MCE Registration* module allows the MCEs to express their information handling requirements and capabilities. The ISI function maintains an MCE registry, storing specifications for the available information to be retrieved or disseminated. The *Information Storage* module offers alternative storage options, according to requirements and characteristics, specified during an MCE's registration phase.
- *Information Processing and Knowledge Production (IPKP).* The IPKP augments VIS with information processing, aggregation, and global-picture information production capabilities. The *Information Aggregation* module applies aggregation functions (e.g. MAX, MIN, AVERAGE) to the collected data before they are stored or disseminated. The data may be filtered at the aggregation level for optimization purposes. The *Knowledge Production* module generates global-picture information through processing and/or aggregating information.

B. The Virtual Infrastructure Management

The Virtual Infrastructure Management (VIM) provides high-level control and management of the virtual infrastructure. In figure 3, we show the VIM with its basic functions and interfaces. It is responsible for the manipulation and lifecycle of virtual topologies and the service and management software running on top of them, to ensure continued operation and consistency. The VIM interacts with the other two VLSP layers through the *Virtual Infrastructure Management Interface*. The core VIM architectural components are as follows:

- *Infrastructure Controller.* The *Infrastructure Controller* acts as a control point for managing the virtual elements and the applications they support. It accepts all of its input via the *VIM Interface* from High-Level Management Applications (i.e. for service orchestration aspects) and from the VIS (i.e. for information management related activities). The *Service Orchestrator* performs the automatic allocation and full lifecycle management of distributed application nodes which run on the virtual routers and realize particular network services. The *Infrastructure Controller* is also responsible for the allocation and efficient operation

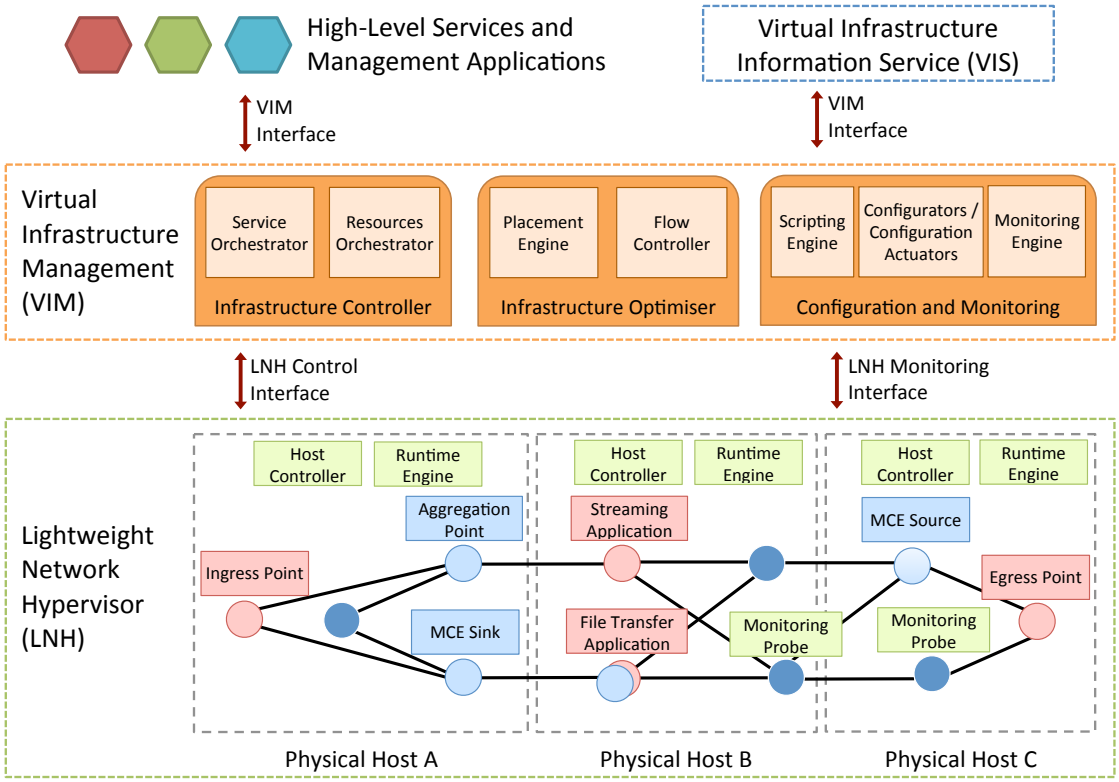


Fig. 3. The Virtual Infrastructure Management and the Lightweight Network Hypervisor

of the virtual resources, through the *Resources Orchestrator*. This module handles the optimal placement of the virtual routers, the decisions to add or remove new resources, the manipulation of the corresponding virtual links etc. It manages the full lifecycle of the virtual resources, aligned to the availability of physical resources and the service requirements.

- *Infrastructure Optimizer.* This function handles VLSP optimization aspects for efficient allocation of virtual resources and the corresponding data paths. It is responsive to the changes in network conditions or requirements, triggering appropriate optimization processes or mitigating stability problems (such as the relocation of service nodes or virtual routers). The *Placement Engine* performs the actual placement of all VLSP entities according to the current topology and the virtual network elements load. The *Flow Controller* is responsible for performing logically-centralized data flow allocation. It takes decisions on the establishment of particular data flows. The same component realizes the information flows being handled from the VIS.

- *Configuration and Monitoring.* The *Configuration and Monitoring* function supports the VIM interactions with the High-level Applications and the LNH. The *Configurators and Configuration Actuators* handle operations related to the translation of certain management and control commands to configuration settings at the level of LNH. It provides alternative configuration options to the management entities for a targeted network behaviour. The *Monitoring Engine* is responsible for low-level monitoring activities, such as the manipulation of monitoring probes in the virtual routers. It collects information to be further processed from the VIS (either aggregated or global-picture information) and used from the VIM as an input for optimization decisions.

C. The Lightweight Network Hypervisor

The Lightweight Network Hypervisor (LNH), as shown in figure 3, includes a fully operational lightweight Virtual Router (VR) combined with Virtual Network connectivity and its associated low-level control components. The LNH communicates with the other two VLSP layers for network control activities through the *LNH Control Interface*. This communication involves enforcement of the VIM decisions regarding the deployment and configuration of virtual networks and network services. Furthermore, it handles the establishment and efficient parameterization of the information flows being overseen from the VIS. The VIM *Monitoring Engine* communicates with a number of deployed *Monitoring Probes* in the VRs, in order to collect real-time state information via the *LNH Monitoring Interface*. The LNH consists of:

- *The Virtual Router.* The virtual router is a software element that offers basic network protocol functionality (e.g. routing and transport), an isolated execution environment, and a virtual sockets API. Our focus here was to provide a lightweight foundation for a scalable infrastructure (i.e. the VRs can bootstrap and shutdown in less than a second). The routers support core IP functionality, such as the distance vector routing protocol, the time-to-live option etc. The router offers an application layer interface enabling the deployment of Java software applications at run-time. These act as the service elements within the platform by using a virtual sockets API which can send and receive datagrams or packets.
- *The Host Controller.* The VRs are controlled by distributed components residing at each physical host, the *Host Controllers (HCs)*. The HCs directly control the VRs within a host

by passing commands from the VIM, such as to manipulate virtual routers, links, application nodes and their configuration.

- *The Runtime Engine.* The *Runtime Engine* complements the *Host Controller* with functionalities related to the runtime operation of the deployed virtual networks and network services. An example being to enforce periodical network activities, such as network maintenance processes and timely detection of failures.
- *The Monitoring Probes.* Each virtual router is instrumented with the VLSP monitoring system, based on Lattice [11], in order to gather data on the virtual network or the hosted applications via modules called *Monitoring Probes*. The monitoring system collects the raw data and passes in for further processing onto the VIS.

III. Use-Case Scenarios

Here we present an example workflow involving all three of the VLSP layers. Assume a High-Level Application expressing particular service requirements: a request to deploy a web-based application to be used by a given number of users. The application also requests global-picture information on the web application performance, such as the average response time. The *service deployment request* is directed to the VIM through the *VIM interface* and the *information flows establishment request* to the VIS through the *Information Management Interface*.

The VIM invokes the *Infrastructure Controller*, which decides to deploy a minimum number of required web-servers hosting the web application and a number of web-proxies that may be used to improve communication performance. Furthermore, the same component, after a request from the VIS, decides to deploy: (i) a number of monitoring probes sampling web application performance, and (ii) an aggregation point to calculate the average values. All of the above are handled by the *Service Orchestrator*.

The *Resource Orchestrator* organizes the deployment of virtual machines on a number of physical servers. The *Infrastructure Optimizer* uses the *Placement Engine* to decide the most efficient locations of the involved application nodes (i.e. the web-servers, web-proxies and the aggregation point), in order to utilize physical resources in an optimal way, as in [10]. The *Flow Controller* deploys: (i) the data flows between the web-servers, web-proxies and the clients, and (ii) the information flows that communicate and process information regarding the web application performance. All flows and node placements are optimized according to an active

global performance goal (for the minimum energy consumption, as an example).

The monitoring probes are handled by the *Monitoring Engine*. The *Scripting Engine* and the *Configurators and Configuration Actuators* modules handle the appropriate representations and configuration settings. Those settings are communicated to the LNH, for it to enforce all the above network and service decisions using the appropriate *Host Controllers* and *Runtime Engines* which are handling the required virtual routers. After this, the deployed network service is ready to offer the web-based application to users.

In practice, all of the above configuration decisions may be revoked (i) by the High-Level Application, in the case that the active performance goal changes, or (ii) by the VIM, if it foresees issues that may cause faults or performance problems. In the experimental results section, we study a similar scenario in which the VIM requests, after some time, a change in the global performance goal — it enforces a general guideline to stabilize the average response time.

Some of the use-case examples that have benefited from the VLSP platform are presented here:

- The VLSP was used as a soft core network connected to hardware based wireless and mobile network elements in order to realize network services on top of heterogeneous network paths. The VLSP operated efficient data flows and enabled allocation of resources near the service consumers, based on the unified local and global network views and the high-level service requirements. We demonstrated a fully working and integrated system in the context of UniverSELF project[2].
- The VLSP has been used to implement an Information-Centric Networking (ICN) testbed. Specifically, it has been utilized in the implementation of the CURLING ICN algorithm [12], which employs a hop-by-hop hierarchical content-based publish-subscribe paradigm to content distribution.
- Within the Dolfin project[3], the aim is to optimize the energy consumption within the limits of a single Data Center and in a group of Data Centers, based on system virtualization techniques and the optimal distribution and placement of virtual machines. To realize such a system, we augmented the VLSP with energy-aware monitoring, modeling, and resource

[2]UniverSELF project website: http://www.univerself-project.eu

[3]Dolfin project website: http://www.dolfin-fp7.eu

allocation.

IV. Experimental Results

Here we present our experiments with the main use-case scenario discussed in the previous section. We particularly focus on how a change in the general performance goal by the VIM is realized, because of scalability and stability problems. We next detail our experimental methodology and then we present our experimental results.

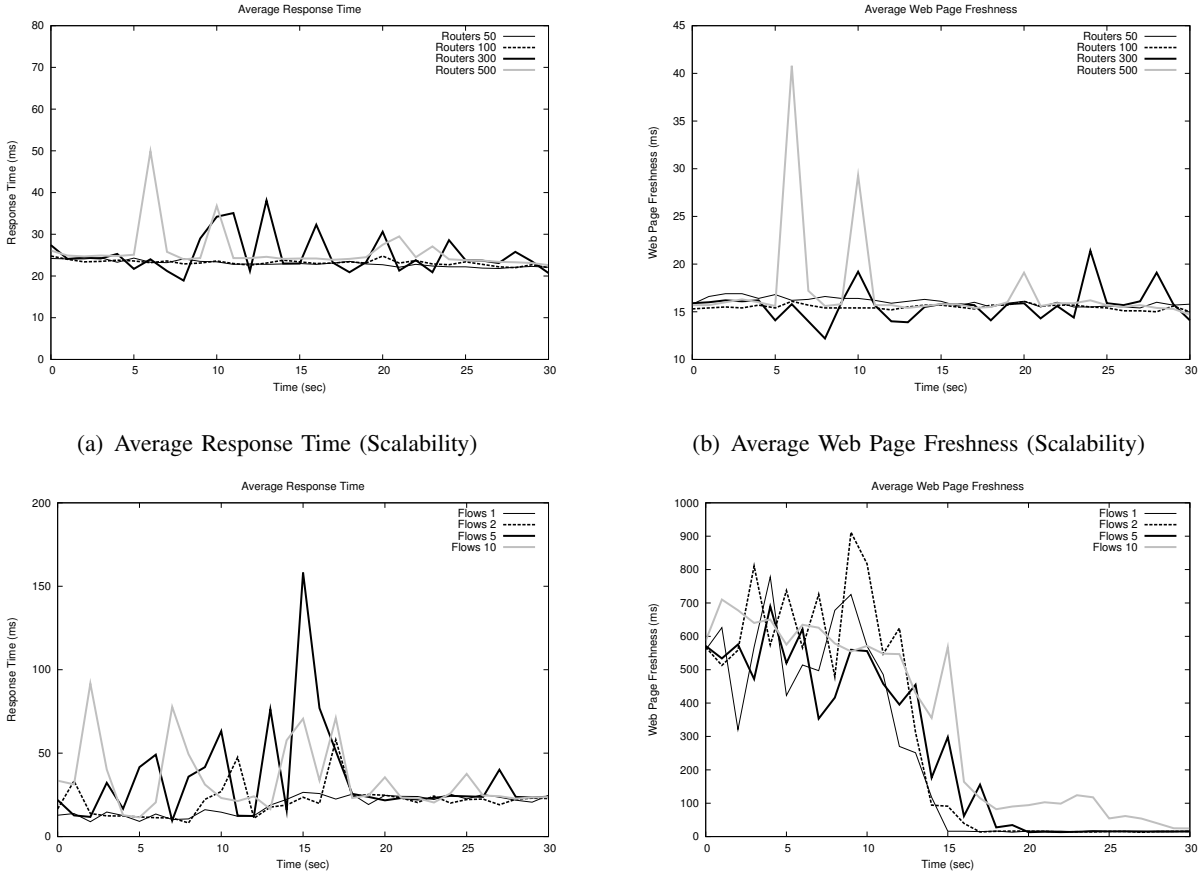
In our test-bed, we used 11 servers each with 4 CPU cores and 32GB of physical memory. The VLSP platform software itself consists of over 600 Java classes and more than 100,000 lines of code. On a desktop machine it is possible to deploy topologies of around 70 virtual routers. We have tested a deployment of 700 communicating virtual routers in complex multipath topologies on 11 dedicated physical servers. The underlying monitoring framework used by the testbed on the routers, known as Lattice, is described in [11] and is available as open-source software[4].

Every experimental run includes the creation of a new network topology. Such a process involves interactions between the VIM and the corresponding *Host Controllers* deployed at all physical servers. We have created small and simple data applications resembling Web Clients, Web Servers, and Web Proxies. The VLSP determines the most appropriate data paths for the data flows by having the global network view as an input and using the dynamic node selection algorithms presented in paper [13]. This activity involves a number of distributed nodes being deployed over the virtual network (i.e. the Web Servers, the Web Proxies and the aggregation point).

As the next step, the VIM *Placement Engine* assigns all of the Web Clients and Web Servers to the most appropriate Web Proxies (namely the Web Proxy that is closest). Then, after a warm-up period, the communication begins. The Web Clients periodically transmit performance measurements to the VLSP over the negotiated data flows using the following metrics:

- *Average Response Time.* The average time taken from the request of a webpage from a Web Client, to the point that it is received.
- *Web Page Freshness* The time taken from a webpage update to the point it reaches the requesting Web Client.

[4]Lattice home page: http://clayfour.ee.ucl.ac.uk/lattice/



(a) Average Response Time (Scalability)

(b) Average Web Page Freshness (Scalability)

(c) Average Response Time (Handling Jitter Issue)

(d) Web Page Freshness (Handling Jitter Issue)

Fig. 4. VLSP Validation Results

In our experiments, we have stress tested our infrastructure with large topologies. The main goal here is to investigate its behaviour in terms of scalability and stability and how resource exhaustion can be tackled by changing the global performance goal. As shown in figures 4(a) and 4(b), large scales can be reached (in this case we go up to 500 virtual routers).

After some time during the runs, the VLSP detects stability problems (in this case, increased jitter in the average response time). When this occurs the *Service Orchestrator* enforces a change in the global performance goal that bypasses the usage of web proxies so that the web clients communicate with the web servers directly. As a consequence of this change the jitter decreases, the average response time increases, and we observe in figure 4(d) a significant improvement in the web page freshness. We can see in figures 4(c) and 4(d) that the VLSP can trade the increasing response time jitter for a slight increase in the average response time. Such strategies

can be associated with control loops that detect and tackle systematic stability problems.

V. Lessons Learned

Since we have been continuously designing and implementing VLSP features for a number of years, we have faced a number of issues that we document in this section. In a nutshell, we observed the following:

- It is difficult to experiment with medium to large-scale topologies without having scalability and stability as corner-stone requirements. In our case, we started building on top of XEN virtual machines which hosted a very small Linux distribution and routing software. However, as this was still too heavy for our needs, we ended up building our own virtual router implementation supporting the essential features only. Lightweight communication protocols are important as well, so we implemented basic transport and routing protocols. All of our management-level protocols run on top of REST.
- Efficient resource consumption calls for a clear identification and handling of the involved performance trade-offs. For example, a physical server may run out of memory while the other resources can be minimally utilized. The logically-centralized view and control give unique capabilities along these lines. Furthermore, many OS related configuration parameters need to be carefully adjusted, such as the heap memory allocation.
- The architectural aspects and relevant design abstractions need to avoid replicating functionalities, such as overlapping communication protocol features.
- There is a need to focus on design and integration aspects beyond simple performance improvements. The lesson from operating system design is that flexibility and adaptability may nominally reduce performance but can enable a greater number of services that were not possible before.

The reader may benefit from our experience for their own software design, development, and implementations of large scale distributed systems.

VI. Conclusions

We have presented VLSP, a complete design and implementation of a novel Service-Aware Virtualized Software-Defined Infrastructure that exhibits the main aspects of autonomic service provisioning and network / service management. We highlighted its relation to well-known

SDN, NFV architectures and relevant open-source initiatives. VLSP includes: (i) the VIS, which provides an accurate, timely, and complete view of the network, (ii) the VIM, which brings sophisticated high-level network management for virtual network topologies, and (iii) the LNH, which abstracts resources and enforces decisions taken. The above three distributed components inter-operate in order to provide a number of control loops that realize the management and operations that actually utilize the physical resources according to the needs of the deployed network services.

In summary, the novel qualities of our proposal are as follows: (i) we designed and implemented a uniform software-defined infrastructure from the scratch, bringing together the NFV with SDN technologies, (ii) we introduced a lightweight virtualization hypervisor, suitable for quick and scalable deployment of network infrastructures, having both computation and connectivity nodes - as a suitable facility to experiment with our ideas, (iii) we proposed a new infrastructure bringing context-awareness in such environments, associating global-picture information with local views, requirements and resource constraints. Last but not least, we demonstrated how these ideas work together to tune performance trade-offs towards planned network behavior.

We have shown how VLSP has been used for various network evaluations, and in future we plan to explore using the VLSP for flexible service creation and deployment focusing on distributed networked cloud infrastructures and network service chaining. This involves extensions of the VLSP with methods and processes elaborating and using [14] for continuous dynamic operation of services, service composition, aggregation and management of service blocks, including service delivery based on orchestration, programmability [15] and automatic (re)deployment.

ACKNOWLEDGMENT

This work is partially supported by the European Union DOLFIN (http://www.dolfin-fp7.eu) and UniverSELF (http://www.univerself-project.eu) projects.

REFERENCES

[1] N. McKeown, T. Anderson, H. Balakrishnan, Parulkar *et al.*, "Openflow: enabling innovation in campus networks," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 2, pp. 69–74, 2008.

[2] S. Jain, A. Kumar, S. Mandal, J. Ong, L. Poutievski *et al.*, "B4: Experience with a globally-deployed software defined wan," in *Proceedings of the ACM SIGCOMM 2013 conference on SIGCOMM*. ACM, 2013, pp. 3–14.

[3] D. Kreutz, F. M. V. Ramos, P. Veríssimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: A comprehensive survey," *CoRR*, vol. abs/1406.0440, 2014.

[4] F. Hu, Q. Hao, and K. Bao, "A survey on software defined networking (sdn) and openflow: From concept to implementation," *Communications Surveys Tutorials, IEEE*, vol. PP, no. 99, pp. 1–1, 2014.

[5] M. Chiosi, D. Clarke, P. Willis, A. Reid *et al.*, "Network Functions Virtualisation," White paper at the SDN and OpenFlow World Congress, ETSI, Tech. Rep., 2012.

[6] SDN De-facto / Dejure Standards, "IRTF, "Software-Defined Networking (SDN): Layers and architecture terminology," Tech. Rep., January 2015. [Online]. Available: https://tools.ietf.org/html/rfc7426; ONF, "Software-Defined Networking: The New Norm for Networks," Open Network Foundation, Tech. Rep., 2012. ; ITU-T, "Recommendation y.3300 (2014) - Framework of Software-Defined Networking, https://www.itu.int/rec/t-rec-y.3300201406-i/en; Recommendation y.3001 (2012) - Future Networks: Objectives and Design Goals, http://www.itu.int/rec/trec-y.3001-201105-i; Recommendation y.3011 (2012) - Framework of Network Virtualization for Future Networks, https://www.itu.int/rec/t-rec-y.3011-201201-i/en," Tech. Rep. ; ETSI NFV : M. Chiosi, D. Clarke, P. Willis, A. Reid et al., "Network Functions Virtualisation," White paper at the SDN and OpenFlow World Congress, ETSI, Tech. Rep., 2012." Tech. Rep.

[7] Open Source initiatives, "OPNFV, "Open Platform for NFV, https://www.opnfv.org/.; OpenDaylight, "SDN and NFV platform that enables network control and programmability," Tech. Rep., http://www.opendaylight.org/.; OpenStack, "OpenStack Networking (Neutron)," Tech. Rep., https://wiki.openstack.org/wiki/Neutron." Tech. Rep.

[8] A. Shalimov, D. Zuikov, D. Zimarina, V. Pashkov, and R. Smeliansky, "Advanced study of sdn/openflow controllers," in *Proceedings of the 9th Central & Eastern European Software Engineering Conference in Russia*. ACM, 2013, p. 1.

[9] D. Levin, A. Wundsam, B. Heller, N. Handigol, and A. Feldmann, "Logically centralized?: state distribution trade-offs in software defined networks," in *Proceedings of the first workshop on Hot topics in software defined networks*. ACM, 2012, pp. 1–6.

[10] S. Clayman, E. Maini, A. Galis, A. Manzalini, and N. Mazzocca, "The dynamic placement of virtual network functions," in *1st IEEE / IFIP International Workshop on SDN Management and Orchestration*, 2014.

[11] S. Clayman, A. Galis, and L. Mamatas, "Monitoring virtual networks with lattice," in *Network Operations and Management Symposium Workshops (NOMS Wksps), 2010 IEEE/IFIP*. IEEE, 2010, pp. 239–246.

[12] W. K. Chai, N. Wang, I. Psaras, G. Pavlou, C. Wang, G. G. de Blas, F. J. Ramon-Salguero, L. Liang, S. Spirou, A. Beben *et al.*, "Curling: Content-ubiquitous resolution and delivery infrastructure for next-generation services," *Communications Magazine, IEEE*, vol. 49, no. 3, pp. 112–120, 2011.

[13] R. G. Clegg, S. Clayman, G. Pavlou, L. Mamatas, and A. Galis, "On the selection of management/monitoring nodes in highly dynamic networks," *Computers, IEEE Transactions on*, vol. 62, no. 6, pp. 1207–1220, 2013.

[14] B. Rochwerger, D. Breitgand, E. Levy, A. Galis *et al.*, "The reservoir model and architecture for open federated cloud computing," *IBM Journal of Research and Development*, vol. 53, no. 4, pp. 4–1, 2009.

[15] A. Galis, S. Denazis, C. Brou, and C. Klein, *Programmable networks for IP service deployment*. Artech House, 2004.

Lefteris Mamatas Lefteris Mamatas (https://sites.google.com/site/emamatas/) is a lecturer at the Department of Applied Informatics, University of Macedonia, Greece. Before that, he was a senior researcher at the University College London, Space Internetworking Center /Democritus University of Thrace and DoCoMo Eurolabs in Munich. He received his Ph.D. from the Department of Electrical and Computer Engineering, Democritus University of Thrace in Greece. His research interests lie in the areas of software-defined networks, network management, opportunistic networks and energy efficient communication. He participated in several international research projects, such as Dolfin (FP7), Autonomic Internet (FP7), UniverSELF (FP7), Ambient Networks (FP6) and others. He published more than 40 papers in international journals and conferences. He served as a TPC chair for the WWIC 2012 and E-DTN 2009 conferences and as a guest editor for the Elsevier Ad Hoc Networks Journal.

Stuart Clayman Stuart Clayman received his PhD in Computer Science from University College London in 1994. He has worked as a Research Lecturer at Kingston University and at UCL. He is currently a Senior Research Fellow at UCL EEE department. He co-authored over 30 conference and journal papers. His research interests and expertise lie in the areas of software engineering and programming paradigms; distributed systems; virtualised compute and network systems, network and systems management; networked media; and knowledge-based systems. He has been involved in several European research projects since 1994. He also has extensive experience in the commercial arena undertaking architecture and development for software engineering, distributed systems and networking systems. He has run his own technology start-up in the area of NoSQL databases, sensors and digital media.

Alex Galis Alex Galis (a.galis@ucl.ac.uk, www.ee.ucl.ac.uk/ agalis) is a Professorial Research Fellow In Networked and Service Systems at University College London. He has co-authored 10 research books and more that 200 publications in the Future Internet areas: system management, networks and services, networking clouds, virtualisation and programmability. He co-authored in 2004 Programmable Networks for IP Service Deployment one of the first text book on management and programmbility of virtualised network and computing resources (http://en.wikipedia.org/wiki/Active_Networking). He was a Vice Chair of the ITU-T SG13 / Group on Future Networking (www.itu.int/ITU-T/focusgroups/fn/index.html), where 5 ITU-T SDN related recommendations were developed. He is currently the co-chair of the IEEE SDN publication committee (http://sdn.ieee.org) and TPC co-chair of IEEE Network Softwarization 2015 (NetSoft 2015 http://sites.ieee.org/netsoft/).