Efficient Management Solutions for Software-Defined Infrastructures

Stuart Clayman, Lefteris Mamatas†, and Alex Galis
Dept. of Electronic Engineering, University College London, London, UK
† Dept. of Applied Informatics, University of Macedonia, Thessaloniki, Greece
Emails: s.clayman@ucl.ac.uk, emamatas@uom.gr, a.galis@ucl.ac.uk

***Abstract*—Novel evolutions in the networking world have been proposed under the umbrella of 5G initiatives. The NFV concept is related with the building blocks for virtual networks that are characterized as highly dynamic network environments. SDN performs logically-centralized network control that is decoupled from the data plane, enabling holistic network management. Furthermore, there is a recent trend towards lightweight virtualized network devices and servers bringing significant advantages in terms of adaptability and responsiveness to the network and service environment dynamics. The above paradigms can be combined together, resulting in unprecedented flexibility in Software Defined Infrastructures (SDI) operations. Such unified environments require new efficient and distributed management facilities that are characterized by scalability, reliability, and adaptability to the dynamic conditions in terms of resource availability and changing service and infrastructure requirements. To assist the evaluation of these components, we developed a distributed facility for testing, evaluating, and experimenting with the management of these SDI environments - the Very Lightweight Software-Driven Network and Services Platform (VLSP). It exhibits the following properties: (i) it is a complete integrated management platform for SDI environments; and (ii) it is distributed and scalable, making it suitable for a wide range of topologies and network service deployments.**

I. Introduction

Software-Defined Networks (SDNs) are realizing a major architectural shift in the Internet, such as the decoupling of network control from the data plane. Logically-centralized control is gradually replacing the distributed self-organized way the Internet behaves, mainly within the boundaries of a single organization domain, thus enabling decision-making aligned to an organization structure. As SDN deployment has been increasing, the focus of research has gradually moved from the phase of SDNs to Software-Driven Infrastructures, where novel management approaches can bring services and networks closer together [1]. The combination of server deployment within the network and devices that are gradually adopting the virtualization technologies, provides a network architecture concept with deeper integration of networks with the IT domains and their related operations through Network Functions Virtualization (NFV). This allows significant cost savings and more flexibility in service provisioning.

Such a shift in approach calls for a more unified and efficient management and control with software entities focusing beyond traffic engineering. Management in 5G should address at least the following 6 aspects: (i) efficient service function, (ii) optimized service function availability, (iii) service and VNF lifecycle automation, (iv) service placement automation, (v) efficient resource utilization, and (vi) dynamic resource up/down scaling (elasticity).

978-1-5090-0223-8/16/$31.00

Our previous experience [2] has highlighted the following critical issues with testing efficiency when managing virtualized systems: (i) the number of virtual machines that can run on a physical host is limited, the startup speed of a virtual machine is quite slow and its image size large; (ii) in terms of highly dynamic networks, and virtualized routers in particular, we observed that over 95% of the router functionality was never utilized in any experiments that were run; and (iii) there were some serious hurdles with the IP networking configuration of virtual machines and virtual routers. With these issues, it is difficult and complex to design and build 5G management systems that address these 6 highlighted aspects.

The VLSP integrates and unifies management of networks, compute, and services into a single platform. The whole system, including a lightweight virtual router, was designed and built from scratch to enable flexible experimentation of virtual infrastructures. VLSP has the following advantages:

- *Support of integrated SDN and NFV environments:* to uncover the potential of virtualized SDN solutions and NFV deployments, in a naturally integrated way, with common management and control facilities.
- *Lightweight virtual router implementation:* to allow evaluation of diverse scenarios, including the testing of various management lifecycle schemes and robustness to "unexpected" network/node conditions (such as sudden start up/shut down events exposing software deficiencies).
- *Distributed infrastructure implementing logically-centralized management and control:* to support optimizations, based on centralized decisions using a global system picture derived from an integrated monitoring infrastructure based on Lattice [3]. It also supports flexible and adaptable service provisioning, plus aspects beyond traffic engineering.
- *Experimentation on management:* to focus on the experimentation of distributed management and control components of SDI rather than data plane performance. VLSP brings the following benefits: (i) more routers can be deployed on each host, making it easier to test scalability and stability; (ii) enhanced distributed management, control and monitoring evaluations can be carried out, which is difficult to achieve in a running environment using a number of deployed data centers; (iii) suitable for both experimentation with virtual networks and lightweight virtual servers; and (iv) supports experiments with dynamic topologies (such

as migratable virtual routers).

In this paper, we present the advantages of VLSP with experimental results and detailed implementations with an operational evaluation of VLSP's non-functional characteristics. Two experimental scenarios are presented showing (1) the management of diverse topologies, which demonstrates the flexibility and scalability of VLSP, and (2) the management of data streams, which shows how VLSP handles monitoring data streams. The main VLSP architectural artefacts, with a discussion of representative use-cases, are presented in [4]. The VLSP open source solution is available at [5].

A summary of the remaining paper follows: Section II gives background information and contrasts the proposed platform with the related work. Section III discusses the design and implementation details of the VLSP. Section IV highlights our experimental methodology and shows our evaluation results. Finally, section V concludes the paper.

II. Background and Related Work

A number of open-source initiatives have appeared, bringing together NFV with SDNs, such as Open Daylight [6]. It is common in deployed networks, including cloud and data center networks, to use system-level virtual machines (e.g. Xen, VMware) plus embedded software routers (e.g. Click [7] or Quagga). However, these VMs are too heavy for large-scale experimentation of their management features. Furthermore, NFVs and SDNs have a parallel evolution that have decoupled aspects, particularly for the management functions.

Other researchers have also observed the same serious issues with managing virtualized systems and they have proposed a number of lightweight virtual servers and routers, such as Mirage OS [8], OSv [9] and ClickOS [10]. These solutions are based on unikernels (or Cloud Operating Systems), [11], where the application can be defined with a high-level programming language. Its compilation output is a very lightweight and single-purpose virtual machine, keeping the essential part of the OS system only. For this reason they are called a Library OS. These virtual machines may be a few megabytes in size and boot up quickly enough that they can startup with the establishment of a TCP connection or the reception of a DNS request packet. So far, such proposals are being deployed within existing cloud or SDN environments that are not designed especially for these lightweight virtualized network devices.

A number of management solutions have been proposed for the studied flexible infrastructures. The CONTENT architecture proposes an orchestrater (at the cloud service layer) federating the IT resources from distributed sites with user-to-data-center and inter-data-center multi-layer connectivity services, managed by an SDN-based network layer. The UNIFY consortium [12] devises means to orchestrate, verify and observe end-to-end service delivery from premises / enterprise networks, using aggregation over core networks, into data centers. The T-NOVA project [13] plans to exploit the concept of NFV allowing operators not only to deploy Virtualized Network Functions (VNFs) for their own needs, but also to offer them to their customers as value-added services. OpenMANO is an open source project that provides a practical implementation of the reference architecture for Management and Orchestration under standardization at ETSI's NFV ISG [14].

Many research institutes have constructed national-level SDN testbeds, such as "OF@TEIN" and "OFELIA" [15]. However, these are large-scale deployments which may not be suitable for frequent experimentations. In some cases such infrastructure was not available and the well-known SDN emulator Mininet was used instead (e.g. [16]). Virtualization is used for emulating heterogeneous network technologies as well. Most of these deployments offer virtualized environments being supported by OpenFlow switches. In our case, VLSP proposes a virtual network that consists of SDN-enabled virtual routers. The Mininet [17] is suitable for small networks (i.e. runs on a single host) and does not support network dynamicity and performance assessment of virtualized hosts [18].

Many promising open-source initiatives are present in the area of SDN/NFV, such as OpenStack [19], Open Daylight [6] and OPNFV [20]. Open Daylight 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. Neutron augments OpenStack clouds [21] with *networking as a service* capabilities. ONF is working on OpenFlow standardization aspects [22] and ETSI [23] is studying the architecture of NFV from the network operators' perspective. Several SDN/NFV efforts have appeared within the IETF (e.g. [24]) in the form of working group initiatives. The ITU's Telecommunication Standardization Sector (ITU-T) hosts study groups on SDNs (e.g. SG13) which focus on future and 5G networks. In the IRTF [25], the Software-Defined Networking Research Group identifies future research challenges for SDN environments. Other SDN initiatives have a stronger focus on combining SDNs with NFV.

III. The VLSP System

VLSP provides a complete environment from the *protocol stack* up to the *service management* level, including a tailor-made monitoring facility. Consequently, we are able to experiment with new complete 5G network management and control facilities over virtual networks based on our lightweight virtual entity resembling both servers and routers.

A. VLSP Architecture

The architecture consists of three main layers to provide the management of the full virtual infrastructure (as in figure 1):

1) the *Application Layer* executes *Management Applications* that define the software components and functions of a network service, together with their configuration parameters and operations. The *Application Layer* allows high-level services and management applications to operate while having a holistic view of the network.
2) the *Orchestration Layer* consists of software elements which perform most of the management and orchestration activities and is in charge of managing the full lifecycle operations of the virtual resources in the network and the allocation of the applications running on the virtual nodes. It provides an *Infrastructure Controller* which orchestrates the elements of a service and its associated resources, an *Infrastructure Optimizer* which optimizes

the service nodes and resources in terms of optimal placement, configuration and data flow operation, and the *Configuration and Monitoring* entity providing configuration related processes (such as configuration representation) plus monitoring of the network and service resources using our own monitoring facility.

3) the *Infrastructure Layer* contains both the *Virtual Infrastructure* which represents the virtual resources that make up the virtual networks, and the *Data Center Infrastructure* which are the physical machines hosting the virtual entities. It includes a *Host Controller* which acts as a Network Hypervisor, presenting abstractions for starting, stopping, and configuring virtual elements. It is also responsible for the runtime operation of the virtualized resources and the applications they host.

VLSP is a distributed management infrastructure that has centralized functionality and is responsible for the setup, configuration, optimization, and shutdown of network entities. As depicted in Figure 1, it takes input from various *Management Applications* regarding various high-level requirements (e.g. a performance goal for the global system operation) and translates that input into network and service resource configuration enforced through a set of *Host Controllers*.

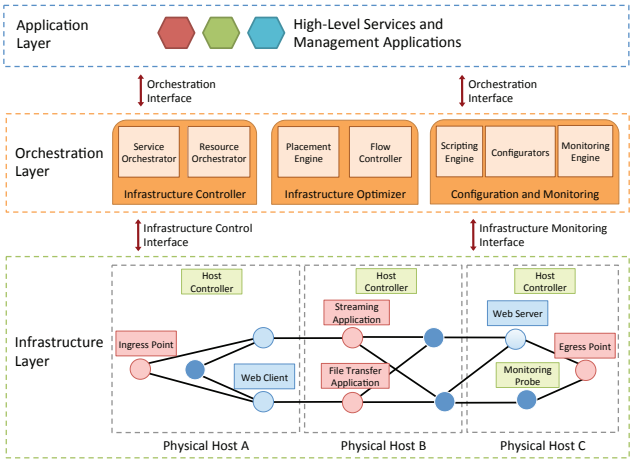


Fig. 1. Overall System Architecture and Components

In order to manage the challenging and dynamic infrastructures of virtual networks there needs to be a monitoring system which collects data and reports on the behavior of both the physical resources (e.g. CPU usage, memory usage) and the virtual resources (e.g. utilization level of the virtual links). These monitoring data items are sent to the *Infrastructure Optimizer* and *Controller* components. The former uses the monitoring information in order to take decisions regarding network strategies and the latter enforces these decisions.

To allow fully dynamic management and control of the system elements we have a *Scripting Engine* which allows the definition of scenarios, resources, or other software entity parameters. We use the Clojure language for the dynamic configurations. This allows us to perform fully *Software-Defined Operations* using a functional language that has an expressive representation of the configuration settings, while being very brief. Calling a function like `(topo-line a-name 10)` defines a network structure of 10 nodes and 9 links. This structure is only a *software representation* of the virtual network. W Another function *activates* the network onto the infrastructure, and hence deploys the virtual routers and virtual links. We use scripting for networks, services, and static configuration files. All communicated information is transmitted over REST channels and using JSON descriptions.

The *Infrastructure Layer* consists of a number of virtual entities, which represent the Virtual Infrastructure, instantiated across a number of physical machines, that are part of a Data Center infrastructure. The virtual entities are logically independent software elements which communicate with each other via network interfaces. The network traffic is made up of datagrams that are communicated between the virtual entities. Each virtual entity can be dynamically created or destroyed within the virtual network, and has a management control connection to a *Host Controller*. This sends the instructions to start up or shutdown entities on the local machine as well as to setup or tear-down connections with other virtual routers or machines. Each virtual entity has a built-in monitoring probe for determining the usage of the network resources and another probe to monitor CPU, thread, and memory usage. The data provided by the probes is collected and can be used by the *Orchestration Layer* to manipulate the virtual infrastructure.

B. VLSP Implementation and Deployment

We have created a working implementation of the architecture described in the section III-A called the Very Lightweight Software-Driven Network and Services Platform (VLSP). It has been implemented for the purpose of testing and evaluating various aspects of managing SDN and highly dynamic virtual environments, in particular the 6 aspects of: (i) efficient service function, (ii) optimized service function availability, (iii) service and VNF lifecycle automation, (iv) service placement automation, (v) efficient resource utilization, and (vi) dynamic resource up/down scaling (elasticity). VLSP is a testbed that consists of a large number of virtualized entities which execute on a number of physical machines. The testbed has been validated for some of these aspects in previous work we have undertaken on virtualized and highly dynamic networks [26].

The testbed set up has all the components described in section III-A, including: (i) a supervisor and experimental controller realizing the basic functionalities of the *Orchestration Layer*; (ii) per-host *Host Controllers*; and (iii) the *virtualized entity*, which runs inside a JVM. The testbed is configured by the *Infrastructure Controller* running on one physical server, and the *Host Controllers* running on physical machines that host virtual entities. Under control from the *Orchestration Layer*, the individual *Host Controllers* start or interact with virtual entities when needed. The choice of *Host Controller* is decided by the *Placement Engine*, which uses an algorithm to determine the *best* physical machine to deploy the new virtual entity. The *Infrastructure Controller* also sends requests, via a *Host Controller*, to connect virtual routers together via virtual network links.

In the VLSP testbed, the virtual servers and routers are autonomous entities, with the virtual network executing independently from the control elements. The control parts interact with the virtual layer, as depicted in Figure 2. We also allow static configurations and dynamic configurations for the Configuration and Monitoring component and the Infrastructure

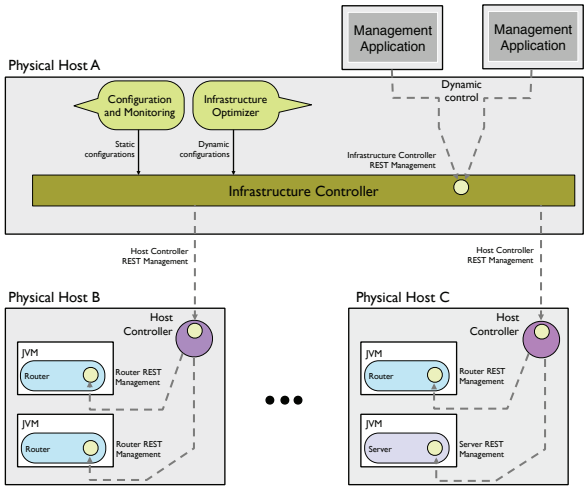


Fig. 2. Component Setup plus Control Path to Virtual Entities

Optimizer. Both of these components enable the dynamicness in the *Infrastructure Controller*. The start up and shutdown of virtual entities and links is managed by the *Infrastructure Controller* and is performed by the *Host Controller* that resides on each host. The *Host Controller* behaves in the same way that a hypervisor does in other virtualised environments, by starting virtual entities on the same physical host, but can also pass on *Infrastructure Controller* commands to the entities.

The monitoring system *Lattice* [3] has been used for monitoring virtualised services in federated cloud environments, monitoring virtual networks [27], and as the monitoring system for an information management platform that aggregates, filters, and collects data in a scalable manner within virtual networks [2]. Lattice has been proven to be ideal for the task of collecting monitoring data for various types of dynamic network environments. Each virtual entity has at least one probe that can generate data. Monitoring data is also collected from each *Host Controller*. This data is send to the *Monitoring Engine* of the *Orchestration Layer* that processes it, and is the data that is used by the *Placement Engine* for determining where a new virtual entity is placed.

IV. Results

In this section we present our evaluation and validation of VLSP for managing virtual entities. First, we detail our experimental setup, the relevant methodological issues, and our experimental scenarios. Then we present and discuss the experimental results from each scenario, starting with the scenario details and the performance metrics used each time. The goal of the following experiments is to: (i) highlight the scalability, flexibility, and adaptability of the VLSP for a diverse set of topologies and network service deployments, and (ii) show that VLSP can operate with a wide range of network and service environments.

The following experimental scenarios we developed:

- **Scenario 1 - Management of Diverse Topologies:** The first scenario demonstrates the flexibility and scalability of VLSP. We deployed various topologies across a set of servers in our testbed, showing the rapid creation and removal of topologies that can enable dynamic services. Our six 5G management aspects i highlight: efficient resource utilization, dynamic resource elasticity, VNF lifecycle automation, and service placement automation. We see the speed of creating virtual routers and virtual links, as well as their deletion times.
- **Scenario 2 - Management of Data Streams:** The second scenario shows how the complete VLSP environment manages monitoring data streams. We create applications handling management data of network operations that are deployed across a virtual topology. We demonstrate the adaptability behavior of VLSP and also its scalability advantages from another view point, compared to the first scenario. This highlights the aspects of efficient service function and optimized service function availability. In our runs, the topology adapts dynamically under the VLSP control as nodes are created and deleted.

In all of our experiments we used the following hardware: (i) 2 servers with 2 Intel 2.5GHz CPUs (4 cores) and 8GB of memory, (ii) 4 servers with 8 AMD Opteron 2.347GHz CPUs (4 cores) and 32GB of memory, and (iii) 5 servers with 16 Intel Xeon 2.27GhZ CPUs (4 cores) and 32GB of memory.

A. Scenario 1 - Management of Diverse Topologies

Here we validate the management capabilities of VLSP through measuring the router / link startup and deletion times for creating a number of representative virtual topologies. The studied topologies are commonly seen in Data Centers - namely the grid and tree topologies. We created: (i) a 10 X 10 grid; (ii) a tree with a spanout of 4 and depth of 4; and (iii) a tree with a spanout of 16 and depth of 2. We programmatically created the topology descriptions using the Clojure language. For example, the method `(topo-grid 10 10)` creates a 10 X 10 Grid, the method `(topo-tree 4 4)` creates a Tree (4, 4), and the method `(topo-tree 16 2)` creates a Tree (16, 2). Any size of grid, tree, or other topology can be devised and created by VLSP.

In order to demonstrate the distributed nature of VLSP over a number of physical servers, we deployed each of the 3 topologies across all of the machines in the test-bed. After ten iterations each time, we decreased the number of servers by one, under the control of the *Orchestration Layer*. The experiment stops when the number of physical servers is too small to accept the required number of virtual routers. In table I, we show the number of virtual routers and virtual links in each of the three topologies. Each experimental run starts with the creation of a new network topology and, during the run, we gather the following metrics for analysis. After the topology creation, we shut the topology down and gather deletion times.

Topology	No of routers	Mean time (ms)	Std Dev (ms)	No of links	Mean time (ms)	Std Dev (ms)
Grid (10 X 10)	100	839.9	10.5	188	194.3	1.9
Tree (4, 4)	85	822.2	9.7	84	181.5	2.6
Tree (16, 2)	17	804.9	9.7	16	159.5	2.5

TABLE I. Topology sizes / times w.r.t no. of routers & links

Router Startup Time - the time taken to start a virtual router includes the JVM creation time, the loading of the relevant classes, and the time to initiate the required objects to ensure the router is in a *ready state*.

Link Startup Time - the time taken to start a virtual link between two virtual routers includes the negotiation between the routers to set-up both ends of the link to ensure the link is in a *ready state*.

Router Deletion Time - the time taken to delete a virtual router includes ending all of the executing virtual applications and shutting down all the virtual links attached to that particular router. Deleting a link also requires negotiating with the other end of the link to ensure a bilateral disconnection and consistent state.

Across all of the three experiments the router creation time is around 820ms, the link creation time is around 180ms, and the router (plus link) deletion times is around 30ms. The test runs have been executed 10 times to ensure replicability of our observations. We deemed 10 replications appropriate for safe analysis (which produced a very low standard deviation of the values, as shown in Table I). According to these results, we can confirm the scalable behavior of the VLSP in terms of rapidness in the manipulation of virtual routers / links. The topology type, size and number of physical servers have an insignificant impact on the virtual entities creation and deletion times, making VLSP suitable to provide the virtual network infrastructure supporting a wide-range of deployed services.

B. Scenario 2 - Management of Data Streams

To highlight how the VLSP operates as a complete system, we devised a scenario where *management data* is processed and manipulated. Dealing with *management data* flows is extremely important for virtual infrastructure management as so many entities can be created dynamically. A number of *management data* clients communicate with data servers, potentially using data proxies. The VLSP monitors the behavior of the *management data* flows and requests a change in a global performance objective when it detects abnormal behavior. In our case, it trades a detected jitter in the average response time of the requested data for a slight increase in the average value of this particular performance metric. We tested this scenario with large topologies in order to investigate the behavior of VLSP when considering efficient functionality.

We have created data management applications for Data Clients, Data Servers, and Data Proxies. The Data Clients periodically transmit performance measurements to the management component of VLSP by communicating with the most appropriate VLSP nodes. We deployed the Data Clients and Data Servers randomly. The VLSP determines the optimal position of Data Proxies on VLSP nodes according to the topology size and using the PressureTime placement algorithm [26], whereby the number of Data Proxies increases with the topology size. This aspect involves two components of the *Orchestration Layer*, the *Infrastructure Optimizer* which enforces optimization decisions by communicating with the *Infrastructure Controller*, and the *Placement Engine* of the *Orchestration Layer* which points each of the Data Clients and Servers to the most appropriate Data Proxy, where the current strategy is to choose the Data Proxy node closest to them. Then, after a warm-up period, the communication begins.

Each experimental run starts with the creation of a new network topology. This involves interactions between the *Infrastructure Controller* and the corresponding *Host Controllers* deployed on all 11 physical servers. The topology consists of a number of virtual routers and a number of virtual links created randomly. The link details are picked from a distribution (i.e. a discrete distribution with a minimum of one, to maintain connectivity). Each new virtual router is dynamically assigned to the physical machine with the least processing load.

All of the experiments have a stochastic nature, with random network topologies and random placements of Data Clients and Servers etc. Again, the test runs have been executed 10 times to ensure replicability of our observations. The average values of all above metrics are calculated every 10 seconds in a separate metric collection aggregator. For each run, data is sampled in order to gather the following metrics:

Average Response Time - the average time taken from the request of a data set from a Data Client to the point that it is received.

Average VLSP CPU load - the average CPU load associated with the VLSP. This allows us to monitor the VLSP behaviour in terms of processing requirements.

Total Memory Storage Used - the total memory storage used in the VLSP for the data flow configuration and the potentially cached data in the proxies.

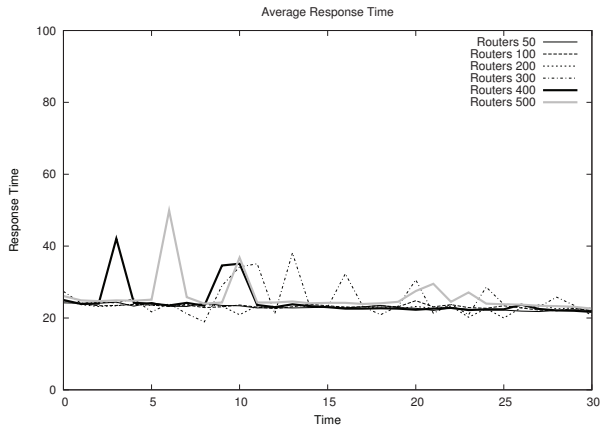
The main goal of the experiment is to investigate the VLSP behaviour in terms of scalability and stability. As shown in figures 3(a) and 3(b), large scales can be reached (up to 500 virtual routers) and VLSP manages to mitigate the jitter issue.

V. CONCLUSIONS

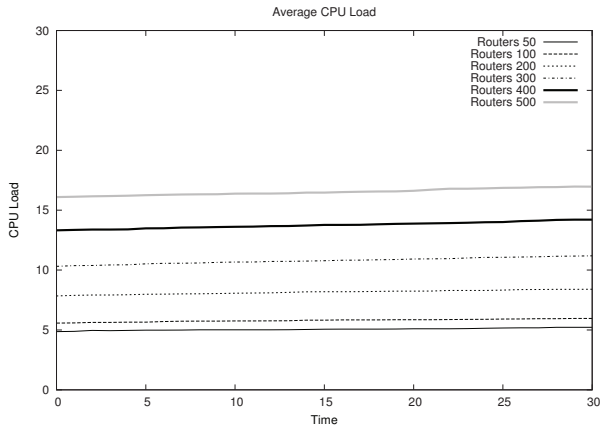
Within the 5G initiatives, flexible and dynamic virtualised infrastructures bring programmability and adaptability into the network environment by abstracting away direct management and control. We introduced a management platform, the Very Lightweight Software-Driven Network and Service Platform (VLSP), which is specially designed to meet the characteristics of such dynamic environments in terms of: (i) user and service requirements, (ii) constraints of underneath physical resources, and (iii) user and virtual machine mobility. VLSP has integrated operations for all the software defined elements through one logically-centralized element. We have evaluated this software defined infrastructure and shown that it is lightweight and suitable for scalability and stability tests.

VLSP is a lightweight implementation which allows the evaluation of diverse scenarios. The goals for using the VLSP architecture over using a hypervisor, running a standard virtual machine and standard OS, were to have better simplicity and non-functional characteristics (i.e. better scalability; lower resource utilization; quicker startup speed; reduced heaviness; more networking flexibility and eliminating the issue where 98% of the router functionality is not needed). From the results presented we have achieved these goals. Finally, we aimed for VLSP to be fully distributed. This has been achieved, as the management components as well as the host controllers and the virtual routers themselves can be deployed across any number of physical hosts. Using a management feature, the physical hosts can be turned offline and online, as required.

Our immediate plan is to introduce new management services and applications to exploit the potential of our infrastructure.



(a) Average Response Time



(b) Average VLSP CPU Load

Fig. 3. Results Without Data Proxy

ACKNOWLEDGEMENT

This work was partially supported by the EU projects: DOLFIN [28], 5GEX – "5G Multi-Domain Exchange" [29] and SONATA – "Service Programming and Orchestration for Virtualized Software Networks" [30].

REFERENCES

[1] A. Galis, J. Rubio-Loyola, S. Clayman, L. Mamatas, S. Kuklinski, J. Serrat, and T. Zahariadis, "Software enabled future internet - challenges in orchestrating the future internet," in *Mobile Networks and Management*. Springer, 2013, vol. 125, pp. 228–244.

[2] L. Mamatas, S. Clayman, M. Charalambides, A. Galis, and G. Pavlou, "Towards an information management overlay for emerging networks," in *Network Operations and Management Symposium (NOMS)*. IEEE, 2010, pp. 527–534.

[3] S. Clayman, "The lattice monitoring framework open-source software," http://clayfour.ee.ucl.ac.uk/lattice/.

[4] L. Mamatas, S. Clayman, and A. Galis, "A service-aware virtualized software-defined infrastructure," *Communications Magazine, IEEE*, vol. 53, no. 4, pp. 166–174, 2015.

[5] S. Clayman, L. Mamatas, and A. Galis, "The very lightweight software-driven network and services platform (vlsp) open source software," University College London, http://clayfour.ee.ucl.ac.uk/usr/.

[6] OpenDaylight, "SDN and NFV platform that enables network control and programmability," Tech. Rep., http://www.opendaylight.org.

[7] R. Morris, E. Kohler, J. Jannotti, and M. F. Kaashoek, "The click modular router," in *ACM Transactions on Computer Systems*. Citeseer, 2000.

[8] A. Madhavapeddy, R. Mortier, C. Rotsos, D. Scott, B. Singh, T. Gazagnaire, S. Smith, S. Hand, and J. Crowcroft, "Unikernels: Library operating systems for the cloud," *SIGPLAN Not.*, vol. 48, no. 4, pp. 461–472, Mar. 2013.

[9] A. Kivity, D. Laor, G. Costa, P. Enberg, N. HarÉl, D. Marti, and V. Zolotarov, "Osv – optimizing the operating system for virtual machines," in *2014 usenix annual technical conference (usenix atc 14)*, vol. 1. USENIX Association, 2014, pp. 61–72.

[10] J. Martins, M. Ahmed, C. Raiciu, V. Olteanu, M. Honda, R. Bifulco, and F. Huici, "Clickos and the art of network function virtualization," in *Proceedings of the 11th USENIX Conference on Networked Systems Design and Implementation*, ser. NSDI'14. Berkeley, CA, USA: USENIX Association, 2014, pp. 459–473.

[11] A. Madhavapeddy and D. J. Scott, "Unikernels: Rise of the virtual library operating system," *Queue*, vol. 11, no. 11, pp. 30:30–30:44, Dec. 2013.

[12] UNIFY, "Unify project," https://www.fp7-unify.eu/.

[13] T-Nova, "T-nova project," http://www.t-nova.eu/.

[14] ETSI ISG, "Network functions virtualisation (nfv); architectural framework."

[15] A. Köpsel and H. Woesner, "Ofelia–pan-european test facility for openflow experimentation," in *Towards a Service-Based Internet*. Springer, 2011, pp. 311–312.

[16] J. Batalle, J. Ferrer Riera, E. Escalona, and J. A. Garcia-Espin, "On the implementation of nfv over an openflow infrastructure: Routing function virtualization," in *Future Networks and Services (SDN4FNS), 2013 IEEE SDN for*. IEEE, 2013, pp. 1–6.

[17] B. Lantz, B. Heller, and N. McKeown, "A network in a laptop: rapid prototyping for software-defined networks," in *Proc. of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks*. ACM, 2010, p. 19.

[18] H. Kim, J. Kim, and Y.-B. Ko, "Developing a cost-effective openflow testbed for small-scale software defined networking," in *Advanced Communication Technology (ICACT), 2014 16th International Conference on*. IEEE, 2014, pp. 758–761.

[19] OpenStack, "OpenStack Open Source Cloud Computing Software," Tech. Rep., http://www.openstack.org.

[20] OPNFV, "Open Platform for NFV," Tech. Rep., http://www.opnfv.org.

[21] Neutron, "OpenStack Networking - Neutron," Tech. Rep., https://wiki.openstack.org/wiki/Neutron.

[22] ONF, "Software-Defined Networking: The New Norm for Networks," ONF White Paper, Tech. Rep.

[23] ETSI, "ETSI Network Functions Virtualization," Tech. Rep., http://www.etsi.org/technologies-clusters/technologies/nfv.

[24] IETF, "IETF Software Driven Networks," Tech. Rep., http://www.ietf.org/proceedings/82/sdn.html.

[25] IRTF, "Software-Defined Networking Research Group (SDNRG)," Tech. Rep., https://irtf.org/sdnrg.

[26] 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.

[27] 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.

[28] DOLFIN, "Data centres optimization for energy-efficient and environmentally friendly internet (dolfin fp7 project)," http://www.dolfin-fp7.eu.

[29] 5GEX, "EU H2020 - 5G Multi-Domain Exchange (5GEX) project," https://5g-ppp.eu/5gex/.

[30] SONATA, "EU H2020 - 5G Service Programing and Orchestration for Virtualized Software Networks (SONATA) project," https://5g-ppp.eu/sonata/.