Per-user NFV services with mobility support

Per-User NFV Services with Mobility Support

Matteo D'Ambrosio, Mario Ullio, Vinicio Vercellone
Network Automation
TIM
Torino, Italy
Email: matteo.dambrosio@telecomitalia.it

Ivano Cerrato, Fulvio Risso
Department of Computer and Control Engineering
Politecnico di Torino
Torino, Italy
Email: ivano.cerrato@polito.it

***Abstract*—This paper presents an architecture to provide end-to-end per-user services with support to client mobility, designed according to the SDN and NFV paradigms. Our service platform dynamically configures and launches service requests when the client connects to the network, which are used by a multi-domain orchestration system to arrange the required network configuration and computational resources. Service configuration is dynamically updated when a movement of the client is detected, that is, when a client device changes its access point to the network. A prototype implementing the idea has been developed and validated over JOLNET, a real, geographical, OpenFlow-based experimental network connecting several sites in Italy and operated by Telecom Italia.**

I. Introduction

Network operators are introducing in carrier networks advanced techniques traditionally used in data centers, in order to execute network services (e.g., firewall and NAT) in virtualized environments running on general purpose hardware. This objective is achieved by exploiting the Software Defined Networking (SDN) and Network Function Virtualization (NFV) paradigms, which promise high flexibility in network operations, fast service creation, deployment and provisioning, optimal usage of resources in underlying infrastructures, adoption of cheap technology based on general purpose servers. This results in a great advantage for network operators, which can reduce infrastructure and operating costs while enhancing capabilities for efficient and fast/agile service deployability. In addition, the technique of Service Function Chaining (SFC) [1], enabled by the SDN paradigm, abstracts the service creation process by means of high level service chains/graphs and automates the service deployment of Virtualized Network Functions (VNFs) over the physical infrastructure.

In order to leverage the potential and the advantages brought by SDN, NFV and SFC in delivering network services, several approaches of resource orchestration and compute/network virtualization in carrier networks have been proposed and are under discussion in several standardization groups and research projects. Among the others, it is worth to mention the Service Function Chaining Working Group [2] in Internet Engineering Task Force (IETF), the NFV Industry Specification Group [3] in the European Telecommunications Standards Institute (ETSI), the ETSI Open Source NFV Management and Orchestration (MANO) project [4], and the EU-funded projects UNIFY [5] and 5GEx [6].

This paper presents what we call the VPN Plus Plus (VPNPP) service, which provides end-to-end per-user services to mobile clients. The proposed service is based on an open source, multi-domain architecture, which orchestrates compute and network resources both in data centers and carrier networks by means of a hierarchical orchestration framework. Particularly, in our proposal, a service platform dynamically configures and launches, at client/user access, a per-user service graph that is used by the multi-domain orchestration system to arrange the required network configuration and computational resources. The per-user service graph is dynamically updated and re-instantiated when the client moves from one access point to another.

A prototype of the VPNPP service has been implemented and validated, from a functional point of view, over JOLNET, a real, geographical, OpenFlow-based network, connecting several sites in Italy and operated by Telecom Italia.

II. The VPNPP service

The VPNPP service aims at provisioning end-to-end per-user services that are automatically reconfigured when the use mobile terminal (MT) moves from one network site to another.

At the service bootstrap, the VPNPP service configures the network to forward all the traffic generated on the infrastructure to an authentication service, in order to allow users to authenticate themselves. After a user authentication, a VPNPP session is started, which implies to launch dedicated VNFs that operate only on the traffic belonging to that specific MT, and which have been associated (out of band) to the specific user just authenticated.

Particularly, the service request for a user specifies a connection of the MT to the Internet and/or other intranets/private networks. The service is described through a service graph that indicates the service function chain that has to process the user's traffic (i.e., the traffic generated by the MT) before it reaches the Internet/intranet.

The VPNPP service deploys the VNFs on the closest computing resources to the MT attachment point; then, when the MT changes its attachment point to the network, VNFs are potentially restarted on those computing resources that are now closer to the new location.

III. The VPNPP architecture

In order to realize the VPNPP service in a complex scenario encompassing a real geographical network, we designed the

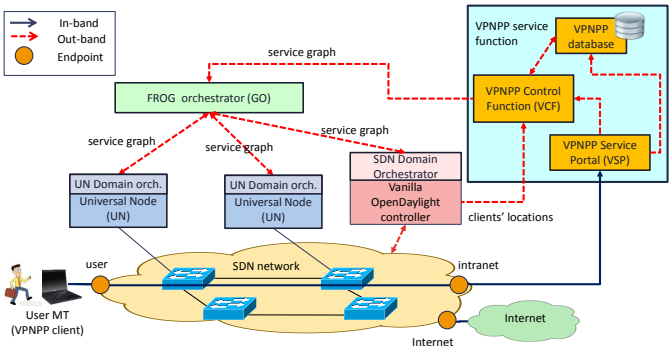

Fig. 1. VPNPP architecture.

hierarchical orchestration framework shown in Figure 1 and described in the remainder of this section.

A. VPNPP service platform

The VPNPP Service Portal (VSP), the VPNPP Control Function (VCF) and the VPNPP database constitute the VPNPP service platform.

The VPNPP database is exploited by the other modules of the VPNPP service platform to authenticate users and to keep track of clients currently connected. Moreover, it contains the user profiles that, among other information, specify the service graphs to be instantiated after the users authentication.

The VSP offers to mobile terminals an authentication server and the possibility to register to a list of available VNFs to be instantiated after the client has been authenticated. According to Figure 1, the VSP interacts with the VCF in the backend in order to notify the VCF itself about clients state changes (new client registrations, service authentications/disconnections).

The VCF is in fact the module that actually implements the service logic, and periodically queries an SDN controller (OpenDaylight (ODL), as shown in Figure 1) to get the list of connected clients and their locations. Each connected device is identified by its IP address, which is permanently associated to it. ODL records contain the following information for each connected device: IP address, MAC address, ingress node in the network, ingress port and VLAN. This information is then stored by the VCF in the VPNPP DB.

Based on the current position of a mobile terminal (retrieved through the SDN controller) and its registration state (kept by the VSP in the VPNPP DB), the VCF triggers service requests (in the form of service graph) to the Orchestration Layer in order to implement the intended network and compute configurations, as required by the specific user's profile.

B. The VPNPP service graph

The service graph sent by the VCF to the orchestration layer (the FROG global orchestrator in Figure 1) is actually a JSON message that includes the following information: the specification of requested VNFs; the endpoints of the service to be deployed in terms of physical ports of infrastructure domains (e.g., interface `p1p2` of domain `un_ve`) and VLAN IDs to be used to tag traffic coming from such an endpoint, as shown in Figure 4; the traffic steering rules, which indicate how endpoints and VNF ports should be connected to each other to realize the requested service.

C. The VPNPP orchestration system

The VPNPP orchestration system includes two levels: the first level consists of a global orchestrator that sits on top of many infrastructure domains, while the second level includes several domain orchestrators that take care of deploying services on the infrastructure domain under their responsibility.

As a global orchestrator, the VPNPP architecture exploits an enhanced version of FROG [7], an open source orchestrator whose northbound REST API can be used by control network functions (e.g., the VCF in case of VPNPP service) and service layer applications to send requests to deploy/update service graphs.

In order to support the VPNPP service, FROG has been extended to orchestrate a network composed of several infrastructure domains, each one with different capabilities (e.g., execute VNFs, create network paths, and more) and with its own domain orchestrator, as shown in Figure 1. Particularly, through its southbound API, the FROG orchestrator interacts: *(i)* with the SDN domain orchestrator (SDN-DO), which sits on top of an SDN network controlled by a vanilla ODL (ODL) controller, and *(ii)* with the Universal Node domain orchestrator (UN-DO), responsible of orchestrating a Universal Node infrastructure domain. More in detail, while the SDN domain consists of several OpenFlow switches that can be only exploited to create network paths, the Universal Node (UN) [7] is an infrastructure domain consisting of a single network node, based on COTS hardware, which can execute network functions (packet processing) as well as more traditional cloud workloads (e.g., server web), and which enables flexible traffic steering among the running functions or applications. Source code of the FROG orchestrator and the domain orchestrators described above is available at [8].

In the VPNPP architecture, the FROG orchestrator is in charge of the following tasks: *(i)* receive, from domain orchestrators, information about the capabilities of the available domains (i.e., what domains can do), *(ii)* receive and process service requests from the VCF, *(iii)* calculate physical and virtual resources required to implement the requested service, *(iv)* determine routes and policies to be enforced, and *(v)* split the input service graph in subgraphs to be sent to the proper domain orchestrators for service instantiation. Notably, the possibility to receive domain capabilities from domain orchestrators, as well as the ability to split a service graph based on such capabilities, and on information associated with the endpoints in the graphs in order to instantiate VNFs as close as possible to the users' clients, have been added to FROG in order to realize the proposed VPNPP service.

The SDN-DO receives and processes service requests coming from the upper layer orchestrator and implements requested service policies in the SDN network via the vanilla ODL controller (Hydrogen release) northbound API. Particularly, the SDN-DO asks to the ODL controller to configure

the underlying switches in order to create the paths described in the service graph. It is worth mentioning that the ODL controller is equipped with a "host tracker" module, which keeps track of the connected devices by monitoring ARP packets entering in the SDN network. This way, as described in Section III-A, it can provide the proper information to the VCF when such a module queries the controller itself.

The UN-DO receives and processes service requests coming from the upper layer orchestrator and implements requested services in the UN domain. In order to satisfy a service request, the UN-DO starts the proper VNFs either as Docker containers or virtual machines executed through the QEMU/KVM hypervisor. Then, it creates the proper paths between VNFs ports and with the physical network by configuring the forwarding tables of a virtual switch using OpenFlow 1.3 (the current implementation of the prototype supports both the standard and the DPDK-based Open vSwitch).

IV. JOLNET

JOLNET [9] is a distributed infrastructure deployed by Telecom Italia, which consists of a geographical SDN network connecting several Universal Nodes and OpenStack sites at the edge. Resources are partitioned among several tenants through the mediation of a virtualization system based on a special purpose Openflow controller with network virtualization capabilities (FlowVisor) and a cloud controller (OpenStack) for the computing resources. A network slice (i.e., an overlay network with dedicated resources) and a number of virtual machines can be assigned to each tenant.

A. The JOLNET VPNPP slice

A small slice on JOLNET (Figure 2) has been allocated to the VPNPP service. It corresponds to the SDN domain shown in Figure 1, which is controlled by a vanilla ODL controller, Hydrogen release, equipped with the Host Tracker module mentioned above. The VPNPP slice is associated with a set of VLAN tags; all traffic entering the JOLNET with one of these tags is recognized as part of the VPNPP slice. JOLNET FlowVisor automatically redirects all OpenFlow messages related to this traffic to the VPNPP ODL Controller.

As shown in Figure 2, the VPNPP slice integrates two physical Access Nodes (AN) and two Universal Nodes (UNs), which provide computing resources to the per-user service chains instantiated after the users authentication. Universal Nodes are deployed in the Turin (TO) and in the Venice (VE) sites. Mobile terminals are supposed to access the VPNPP slice by attaching to the ANs, located in two different Points of Presence (POPs), namely Trento (TN) and TO. It is worth to note that, in our validation, MTs are emulated as virtual machines on the ANs. Each VM has a virtual NIC connected either to the local network infrastructure or to a remote location by means of a bridge and tunnel arrangement, and switch between the two connections. The resulting behavior is the same of a device moving and attaching between TN and TO ANs.

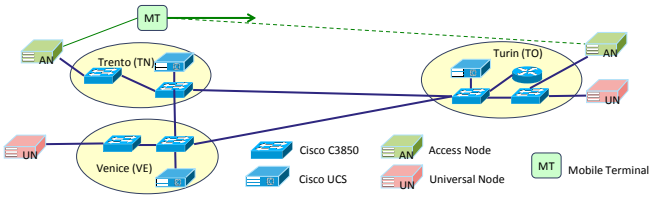

Fig. 2. Experimental setup on JOLNET for the VPNPP service.

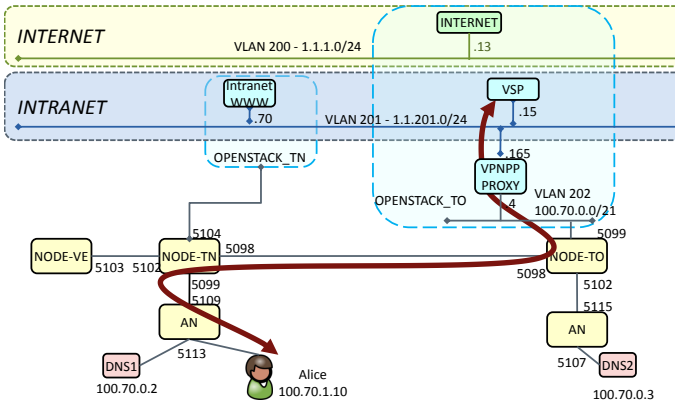

Fig. 3. Default network paths in the VPNPP slice over JOLNET.

V. Validation

In order to validate the VPNPP architecture, we initially arranged the VPNPP slice as shown in Figure 3, which enforces all traffic from unregistered clients attached to the network at the TN and TO sites, to be forwarded to the VSP through a VPNPP proxy (both located in the JOLNET OpenStack cloud system). Hence, when a new MT performs an HTTP GET request, the VSP captures the message and provides a login/password challenge to the user.

During the evaluation, the VPNPP database includes the user Alice, which is associated with a user profile that guarantees only access to the Internet, but limited access to the Intranet (she can reach the authentication service, but not the Intranet WWW server). When Alice attaches to the TN site and authenticates herself, her authentication state is notified to the VCF, which is also aware of the Alice's location thanks to the ODL controller. Therefore, the VCF is now aware that Alice is at a certain ingress port on the AN in Trento, and then it instantiates a new service graph to start the VPNPP session for Alice (Figure 4(a)). In particular, the VCF provides such a graph to the FROG orchestrator, which looks for computing (VM firewall for Alice) and networking (e.g., VLAN tags) resources, calculates paths, splits the request among the compute (UNs) and network (SDN) domains, and finally sends the subgraphs to the proper domain orchestrators.

Since the UN in Venice is the closest one to the TN site, which does not have any UN, the FROG orchestrator asks the UN-DO associated with such a domain to set up the ALICE_FW virtual machine, while the ODL-DO configures the network paths through the JOLNET VPNPP slice as

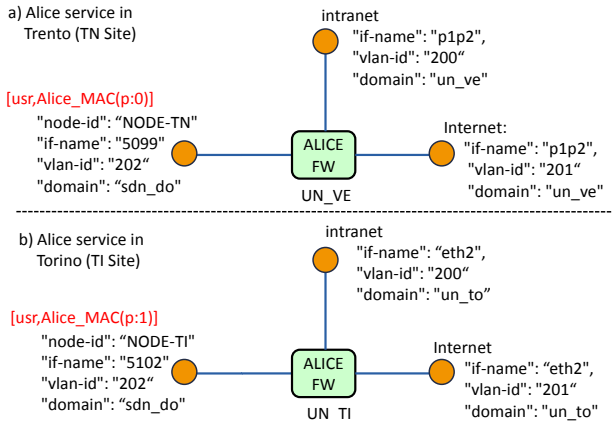

Fig. 4. Service graphs for Alice when connected to TN (a) or TO (b) sites.

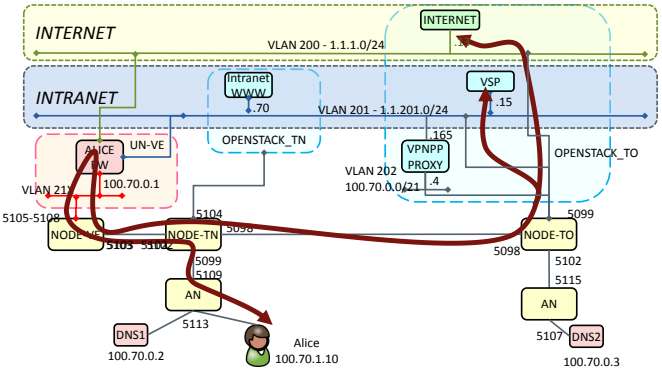

Fig. 5. Available paths for Alice after her authentication in Trento.

required. The firewall is configured according to the Alice's profile; available paths for Alice after the deployment of the service graph are shown in Fig 5.

When Alice disconnects from Trento site and moves and reconnects to Turin site, this behavior is recognized by the VCF, which updates the Alice's service graph (see Figure 4(b)) and sends it to the FROG orchestrator. At this point, the ALICE_FW instance in the UN at the TN site is terminated, a new instance is started at the UN in TO, and network paths are recalculated, resulting in the scenario of Figure 6.

Alice exits from the VPNPP service either by logging out through the authentication server, or by cutting the connection with the Access Network. The VCF is then notified (by the authentication server in the first case, or by polling the ODL controller in the latter) and sends to the FROG orchestration a request to delete the Alice's service graph. Computing and network resources used to implement such service session are then released by the proper domain orchestrator(s).

As a qualitative evaluation, the current version of the prototype launches a VPNPP service graph in 45s in average. Most of the delay is due to the boot time of the firewall VM (10-40 seconds). The remaining time is spent along the control path (VSP, VCF, FROG Orchestrator, Domain Orchestrators).

To conclude, a demonstration video showing the proposed

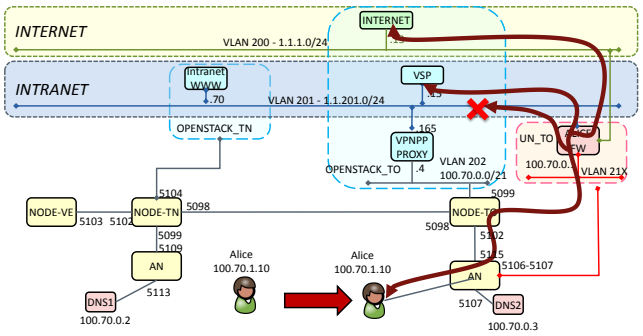

Fig. 6. Available paths for Alice when she moves to Torino.

VPNPP service at work is available at [10].

VI. Conclusion

The VPNPP service and architecture presented in the paper provides end-to-end per user services through a two-layers orchestration framework. The proposed idea has been validated through an open source prototype running over JOLNET, an experimental, geographical, OpenFlow-based network. The VPNPP experiment has shown hierarchical orchestration of network services over multiple technological domains and dynamic reconfiguration of per-client service function chains on a geographical setup.

As a future work, we will address performance and scalability issues related to the mobility support in the proposed framework. We will also implement the modules that compose the VPNPP service function (Figure 1)) as VNFs that are part of a service graph; this would provide more flexibility since it would allow, for instance, to reuse the same slice for multiple purposes by just deploying/undeploying such a service graph.

References

[1] P. Quinn and T. Nadeau, "Rfc7498 - problem statement for service function chaining." [Online]. Available: https://tools.ietf.org/html/rfc7498

[2] "Service function chaining (sfc)." [Online]. Available: https://datatracker.ietf.org/wg/sfc/charter/

[3] "Network functions virtualisation." [Online]. Available: http://www.etsi.org/technologies-clusters/technologies/nfv

[4] "Open source mano." [Online]. Available: https://osm.etsi.org/

[5] "Unify: unifying cloud and carrier network," 2013. [Online]. Available: http://www.fp7-unify.eu/

[6] "5g exchange," 2015. [Online]. Available: http://www.5gex.eu

[7] I. Cerrato, A. Palesandro, F. Risso, M. Su, V. Vercellone, and H. Woesner, "Toward dynamic virtualized network services in telecom operator networks," *Computer Networks*, vol. 92, Part 2, pp. 380 – 395, 2015, software Defined Networks and Virtualization.

[8] Netgroup @polito. The FROG (v.4). https://github.com/netgroup-polito/frog4.

[9] "Jolnet: a geographical sdn network testbed." [Online]. Available: https://www.softfire.eu/jolnet/

[10] M. D'Ambrosio, V. Vercellone, M. Ullio, F. Risso, and I. Cerrato. Demo prototype vpnpp with dynamic service reconfiguration. [Online]. Available: https://www.dropbox.com/s/bkm82fbw7o0myq7/VPNPP_Presentation%26Demo%20v2.avi?dl=0