Defining a generic OR-VNFM interface for configuring network functions

Defining a Generic OR-VNFM Interface for Configuring Network Functions

Davide Pezzolla, Ivano Cerrato, Fulvio Risso, Gabriele Castellano
Dept. of Computer and Control Engineering, Politecnico di Torino, Torino, Italy

***Abstract*—The ETSI model defines a generic architecture to deploy and configure virtual network functions. While many efforts from both academia and industry focus on the problem of deploying those virtual network functions, little attention has been given to the interface needed to configure such applications. This paper explores the problem of dynamically configuring virtual network functions and proposes an implementation for the ETSI MANO OR-VNFM interface that supports generic network functions by exploiting a message bus and YANG models.**

I. Introduction

The network function virtualization (NFV) paradigm is gaining momentum because it allows agile services provisioning as well as the reduction of the management and deployment cost of such services.

The deployment of virtual network functions (VNFs) on the infrastructure nodes requires the execution of two tasks. The first one consists in actually instantiating the VNF in some execution environment such as a virtual machine or a Docker container, while the second tasks includes the configuration of the deployed VNF, so that it can operate as required by the VNF owner (or tenant, in the following). Notably, a VNF must be configured at the boot, but its configuration can be further changed/updated during the VNF lifecycle.

The European Telecommunications Standards Institute (ETSI) has proposed a reference model for the NFV architecture that defines the main functional blocks, their associated reference points, and descriptors; particularly, such an architecture supports both the tasks introduced above and required to make a virtual network function working. While proposals such as OpenMANO [1] and Open Baton [2], which provides implementations of the ETSI infrastructure, mainly focus on the VNF deployment, it is not completely clear yet how to implement the dynamic service configuration, which should be achieved through the OR-VNFM interface [3].

This paper proposes an implementation of the ETSI OR-VNFM interface which is based on YANG models and a publisher/subscriber communication mechanism between the involved entities. YANG models, which allows each VNF to describe its configuration parameters, are increasingly used to configure physical network equipments, often coupled with the NetCONF protocol. Therefore, different from the several languages (and the associated tools such as Chef, Puppet or Ansible) widely deployed in virtualized environments to configure VNFs, a YANG-based configuration method facilitates the support for both VNFs running as virtual machines in data centers, and physical network functions, i.e., network functions implemented as dedicated appliances. Finally, the pub/sub model enables the VNF to send a request for configuration without knowing the entities to contact, whose advantages will be more detailed in the next section.

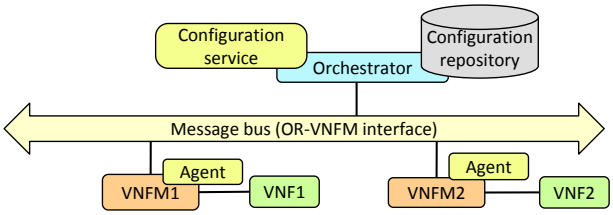

Fig. 1. Overall view of the configuration architecture.

II. Configuration architecture

In the ETSI MANO model, the task of configuring VNFs is assigned both to the Orchestrator and to the Virtual Network Functions Manager (VNFM), which communicate through the OR-VNFM interface.

In order to configure virtual and physical network functions, in our architecture we introduced a new **configuration service** module in the Orchestrator (Figure 1), whose main purpose is to handle the lifecycle of the VNF configurations, such as receiving configuration requests from the VNFM, retrieving the proper data from the configuration repository, and sending it to the proper VNFM; in addition, real-time configuration changes coming directly from the tenant itself should be handled as well. The configuration service receives configuration requests from the orchestrator, which is connected to a pub/sub message bus that carries on both control information (e.g., configuration requests) and data messages (i.e., the actual configuration).

According to our architecture in Figure 1, the OR-VNFM interface is actually implemented by a message bus[1], on which multiple entities are attached, namely the orchestrator (and, consequently, the configuration service), the **configuration repository**, and the VNFM **agents**. In fact, as shown in the picture, each VNFM includes an agent that is the component that actually communicates with the configuration service through the OR-VNFM interface in order to receive the configuration. Moreover, such an agent is also in charge of translating, through a proper backend, the received configuration from an high level formalism into a set of commands to be actually used to configure the VNFs (e.g., Openconfig messages, Chef commands). Finally, the agent pushes these commands to the VNFs under the control of the specific VNFM, which is compliant with the ETSI MANO model that assigns to the

[1]We use the Double Decker bus: https://github.com/Acreo/DoubleDecker.

VNFM the duty of pushing the configuration into the VNFs under its responsibility.

Our implementation of the OR-VNFM interface exploits a communication model based on the publisher/subscriber paradigm; in other words, messages are sent on the bus as belonging to a specific *topic*, and all the entities subscribed to that topic will receive the message. The bus provides flexibility to our architecture, as it allows the agent inside the VNFM to be generic and ask for configuration without knowing the real recipient of the message; in fact, the IP address of the configuration service can change from one infrastructure node to another. Even, the bus may allow the agent to receive configurations also directly from other VNFs, and not only from the configuration service. For example, the agent running along with a DHCP VNF may announce the IP addresses assigned to clients; the agent associated with a firewall may then use this information as a configuration input, and forbid Internet connection to all the address not assigned by the DHCP server.

Configuration messages exchanged trough the OR-VNFM interface are defined according to a specific OpenConfig-derived YANG model associated with the VNF; this approach provides the possibility to configure any type of network function, as well as it allows to exploit models already defined by the OpenConfig industry group. Particularly, the YANG model describes the VNF in a way that is independent from the specific implementation of the function itself; this independence allows, e.g., the tenant, to configure the VNF without knowing its implementation details, which are only know by the agent on the VNFM.

We based our OR-VNFM interface on OpenConfig-like messages and a pub/sub communication model, instead of using tools such as Ansible, Chef or Puppet, for the following reasons. First, those tools require that the agent knows the configuration service, while our approach is more generic and allows the agent to be agnostic with respect to the identity of the entity that configures the VNFs (the configuration service or even other VNFs). Second, in case we have to configure a function already supporting OpenConfig models, the same model can be used both for the OR-VNFM interface and on the VNFM-VNF interface. However, our architecture can exploit the above mentioned tools to implement the VNFM-VNF interface, e.g., to configure network functions running in virtual machines (they cannot be used for physical functions, which usually do not include, e.g., a Puppet agent).

III. Validation

To validate our proposal we deployed, on the Universal Node[2], a service whose VNFs must be configured in order to allow the user to connect to the Internet, as shown in Figure 2 (since this work focuses on the OR-VNFM interface, we deploy both the VNF and the VNFM on the same virtual machine). To this purpose, either the system administrator or the user itself (through some out-of-band configuration method) can access to a web page, which is dynamically created by the configuration service based on the YANG models associated with the deployed VNFs. The web page allows to modify the configuration of each VNF, as shown by the screenshot in

[2]https://github.com/netgroup-polito/un-orchestrator

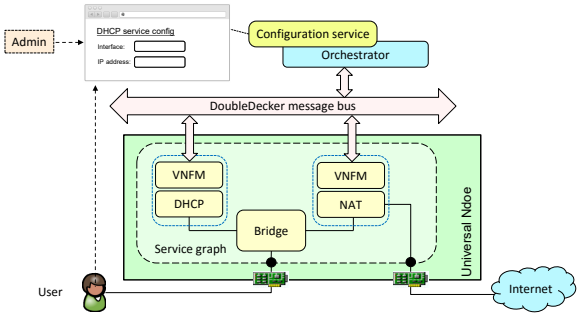

Fig. 2. Example of a real deployment on the Universal Node.

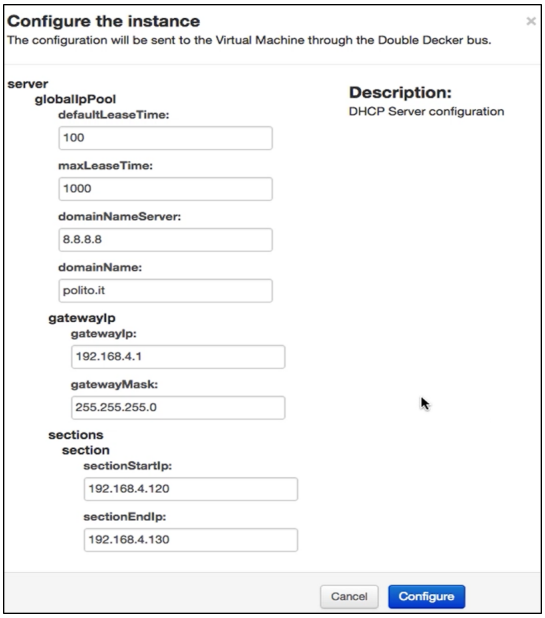

Fig. 3. Screenshot of the DHCP configuration panel, auto-generated from the YANG model.

Figure 3 that refers to the DHCP configuration panel. When the form is completed, the web page is translated into a JSON message coherent with the VNF YANG model and pushed to the VNFM by means of the OR-VNFM interface described in Section II.

In this scenario, the bootstrapping of both VNFs, which terminates when the VNFM retrieves, from the configuration service, a configuration previously stored in the configuration repository and applies it the the VNF, is approximatively 2s.

Acknowledgements - This work was conducted as part of the CC4BA project, which is partially funded by EIT Digital (Call 2016). The authors wish also to thank Fabio Mignini, Pontus Sköldström, Bertrand Pechenot and Antonio Manzalini for their help in the early stages of the work.

References

[1] D. Lopez, "Openmano: The dataplane ready open source nfv mano stack," in *IETF Meeting Proceedings, Dallas, Texas, USA*, 2015.

[2] "Open baton:," http://openbaton.org.

[3] "Network functions virtualisation (nfv); management and orchestration," http://www.etsi.org/deliver/etsi_gs/NFV-MAN/001_099/001/01.01.01_60/gs_nfv-man001v010101p.pdf.