POLITECNICO DI TORINO

Dipartimento di Automatica e Informatica

PhD in Computer and Control Engineering

XXIX cycle

Supervisor

prof. Riccardo Sisto

Verification and Configuration of Software-based Networks

PhD Candidate: *Serena Spinoso*

Introduction

The innovative trends of Network Function Virtualization (NFV) and Software Defined Networking (SDN) have posed never experienced opportunities in productive environments, like data centers. While NFV decouples software implementation of the network functions (e.g., DPI, NAT, etc) from their physical counterparts, SDN is in charge of chaining those functions to create network paths.

One of the new opportunities is to make the service-provisioning models offered by Data Center Providers (DCPs) more flexible, by enabling users to build their own networks (named *service graphs*): users can select the Virtual Network Functions (VNFs) to use and can specify how packets have to be processed and forwarded in their networks.

In the context of Virtualized and Software-Defined Networks, this PhD thesis spans mostly topics related to the verification of SDN/NFV networks and their configuration. In particular, we studied novel approaches for verifying formally the validity of network properties in the service graph requests, comparing them with state of the art proposals, and we propose also an effective solution for installing configuration parameters into VNFs (e.g., blacklists into DNS filters) with a model-based approach.

Challenges

In the context of this thesis, we faced several challenges related to the goals we aim to achieve.

For what concerns network verification, we must explore verification strategies that: *(i)* are performed before deploying the service graphs, in order to avoid unexpected network behaviours; (ii) take a reasonable amount of time from a VNF Orchestrator point of view, with fair processing resources (e.g. CPU, memory and so on). This is because we are in the context of flexible services, where the reconfiguration of network functions can be frequently triggered, both in case of user request and in case of management events.

From a VNF configuration perspective, instead, DCPs should also take care of implementing a strategy to deploy VNF configurations in order to complete the service request and/or eventually to fix bugs in case of verification failures. Here, we can envision several challenges to address: *(i)* network functions may require different ways (REST API, CLI, SMTP, etc...) for being configured and *(ii)* the semantic of a configuration depends on the network function itself (e.g., router parameters are clearly different from firewall ones).

Forwarding Policy Verification

The literature has proposed several strategies for building service graphs and one of these is based on forwarding policies. By *forwarding policy* we refer to a high-level specification of packet forwarding, by indicating which packet flows are forwarded in each network path.

With respect to SDN-related verification, the majority of the proposed tools operate on network configuration rules (commonly OpenFlow), which means to perform network verification after having translated the high-level policies into OpenFlow entries.

Our contribution is the definition of a formal model, based in First Order Logic (FOL), that allows the automatic detection of a set of anomalies in forwarding policies. In particular, an *anomaly* is an erroneous specification of forwarding policies, which may cause misleading network conditions and states.

One key factor that distinguishes our work from existing approaches is early detection of policies anomalies (i.e., before translating such policies into OpenFlow entries), in order to speed up the fixing phase, without even starting service deployment.

In order to prove the usefulness of the approach in a real network scenario, a Java-based prototype that exploits Drools as inference engine has been implemented. The achieved results of the approach are verification times in the order of milliseconds (i.e., about 400ms) for networks of reasonable size (i.e., 300 end-hosts with 1000 forwarding policy rules).

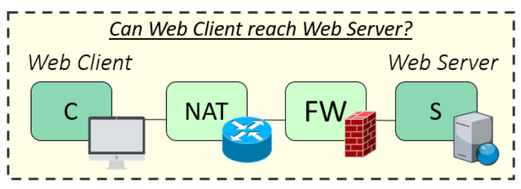
Network Property Verification

A recent key advancement in network verification has been the development of solutions that consider the presence of *stateful* VNFs, i.e. functions that may dynamically change the forwarding path of a traffic flow according to their local algorithms and states (e.g., IDSs).

A scalable verification approach has been proposed in [3], based on the off-the-shelf SMT (Satisfiability Modulo Theory) solver Z3. According to this solution, the network and the involved VNFs are modelled as a set of FOL formulas. Those formulas are passed to Z3 in order to verify some reachability-based properties (e.g., "*can A reach B traversing C?*").

The proposed solution has been implemented in a tool released under the AGPLv3 license, named VeriGraph [4]. VeriGraph takes also the functional configurations of all deployed VNFs (e.g., filtering rules on firewalls) into account.

Let us consider the service graph shown in Figure 1, where we have verified the reachability between client and server, with different firewall configurations (i.e., the firewall drops packets in the first verification-run, while it allows the traffic in the second run). In particular the adopted approach achieves verification times in the order of milliseconds (see table in Figure 1), which is compliant with the timing limitations needed by a VNF Orchestrator.



Firewall rule	Verification Result	Verification Time
DROP	**UNSAT**	354ms
ALLOW	**SAT**	360ms

Figure 1: *Service graph and verification times.*

Model-based Configuration

One of the weaknesses of Cloud Managers is that they still miss a way to configure the VNFs in a seamless and flexible way.

The proposed solution relies on a model-based configuration approach, which means defining a representation (i.e., a model) of the main configuration parameters for each VNF implementation in the requested service graph. The configuration data model, named *VNF Object Model*, is then automatically processed for generating the actual configuration parameters and for pushing them into the VNF.

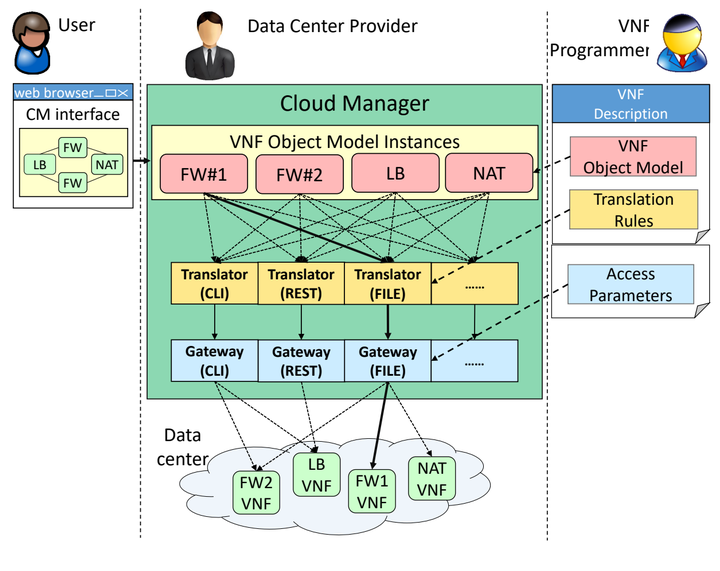


Figure 2: *Seamless configuration of network functions.*

To achieve this goal, we need the installation of further software modules (Figure 2): *(i)* the *translator* modules take care of translating the configuration parameters into a particular format required by a VNF, by using an internal instance of the VNF Object Model where the parameters are stored, and a set of *Translation rules*, which are directives to translate the OM instance's content into the specific structure/format required by the VNF; *(ii)* the *gateway* modules take care of delivering the produced configuration into the VNF, by using a set of *Access Parameters*, indicating how to contact the VNF and to update its configuration following one of the configuration strategies (e.g., REST, configuration file, etc.) already supported by the function.

The achieved results of this work, *w.r.t.* the current state of the art, are: *(i)* the exploitation of a model-driven approach that achieves a higher flexibility in CMs and *(ii)* the insertion of non-VNF-specific software modules in CMs to avoid changes in the VNF implementation, while the state-of-the art proposals are based on VNF-specific modules.

Bibliography

[1] Spinoso S., Leogrande M., Risso F., Singh S., Sisto R., "Seamless Configuration of Virtual Network Functions in Data Center Provider networks". Submitted to: Journal of Network and Systems Management (Springer).

[2]Spinoso S., Sisto R., "Formal Verification of Forwarding Policies". Submitted to: Transactions on Network and Service Management (IEEE).

[3]Spinoso S., Virgilio M., John W., Manzalini A., Marchetto G., Sisto R., "Formal Verification of Virtual Network Function Graphs in an SP-DevOps Context. In: Fourth European Conference on Service-Oriented and Cloud Computing (ESOCC2015), Taormina, Italy, 15-17 September, 2015.

[4] https://github.com/netgroup-polito/verigraph.