Verification and Configuration of Software-based Networks

Serena Spinoso
XXIX PhD cycle
serena.spinoso@polito.it
Advisor: Prof. *Riccardo Sisto*

29 June 2017

Scenario

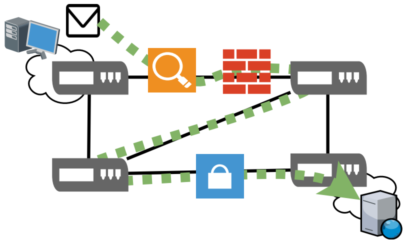
New opportunities in productive environments, like data centers, thanks to:

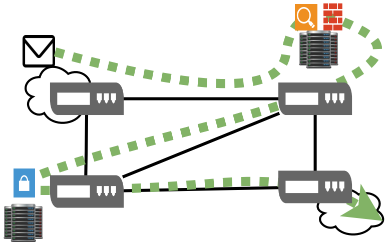
- ***Network Function Virtualization*** (NFV) decouples software implementation of the network functions from their physical counterparts
- ***Software Defined Networking*** (SDN) is in charge of chaining those functions to create network paths.

Scenario

Service Providers allow users to build ***Service Graphs*** by:

- selecting a set of ***Virtual Network Functions*** (VNFs)
  - DPI, NAT, Firewall ...
- specifying traffic forwarding through the selected VNFs
  - ***Service Function Chaining*** (SFC)





Thesis Goals

Many research topics in SDN/NFV-based networks:

- VNF placement, security enforcement, bandwidth optimization, ...

PhD goals:

1. *Formal verification of service graph requests*
2. *Model-based configuration of network functions*

Formal Verification of Service Graphs

Formal Network Verification

Formal verification checks the correctness of computer systems before putting them into use, by exploiting formal methods and mathematical reasoning

Formal verification can be applied to the networking field:

- **Network Verification** proves that a network model (e.g. network configurations) fulfils certain invariants

Formal Network Verification

Formal techniques for verification and property checking in *SDN/NFV-based networks* to avoid faults and errors at run-time

- flexibility of the offered network services
- very frequent network reconfiguration (e.g. user requests or management events)

Traditional model checking techniques run out of memory and time in case of complex network scenarios!

Formal Network Verification: Challenges

Service Providers need verification strategies:

- done *before* deploying the service graphs
- with *low* verification times
- with *fair* processing resources (e.g. CPU, memory)

Formal Verification of Service Graphs

- *Detecting Anomalies in Service Function Chains*
- *Checking Reachability in Service Graphs*

Detecting Anomalies in SFCs

Many SDN programming languages offer **forwarding policies**

- to specify traffic forwarding through chains of VNFs
- translated into flow entries in OpenFlow switches

Providers have to check the policy specification correctness

- faults in network configurations may arise at run-time

Detecting Anomalies in SFCs: contributions

State of the Art:

- verification of OpenFlow networks
  - *during or after* the switch configuration *deployment*
- just *conflict analysis* among OpenFlow entries

Contributions:

- language-independent checking mechanism
- early-detection of anomalies among forwarding policies
- customizable anomaly specification approach

Detecting Anomalies in SFCs: anomalies

Forwarding anomalies are any erroneous or unwanted policy specification done by the Service Provider

- possible faulty network conditions and states at run-time
- include conflicts, errors, sub-optimization, and more

Example:

- "`FW must process traffic before NAT`"
- "`User1 traffic must not pass through FW`"

Detecting Anomalies in SFCs: the approach

First Order Logic models for:

- Forwarding rules that compose a policy

$$r_i = (\mathcal{M}_i, \mathcal{C}_i) \; = \; (eth\_src_i = v_{eth\_src_i},\; eth\_dst_i = v_{eth\_dst_i},\\ eth\_type_i = v_{eth\_type_i},\; vlan\_id_i = v_{vlan\_id_i}, ip\_src_i = v_{ip\_src_i},\\ ip\_dst_i = v_{ip\_dst_i},\; ip\_proto_i = v_{ip\_proto_i},\; port\_src_i = v_{port\_src_i},\\ port\_dst_i = v_{port\_dst_i},\; c_i^1,\; \dots,\; c_i^n),\; r_i \in \mathbb{R}_{\mathbb{F}}$$

- *Pre-* and *provider-defined* anomalies

$$ip\_src_i = ip\_src_j \wedge ip\_dst_i = ip\_dst_j \wedge ip\_proto_i = ip\_proto_j \wedge\\ port\_src_i = port\_src_j \wedge port\_dst_i = port\_dst_j \wedge c_i^1 \neq c_j^1 \; \Rightarrow \text{Collision}(pi_i, pi_j)$$

netgroup

Detecting Anomalies in SFCs: classification

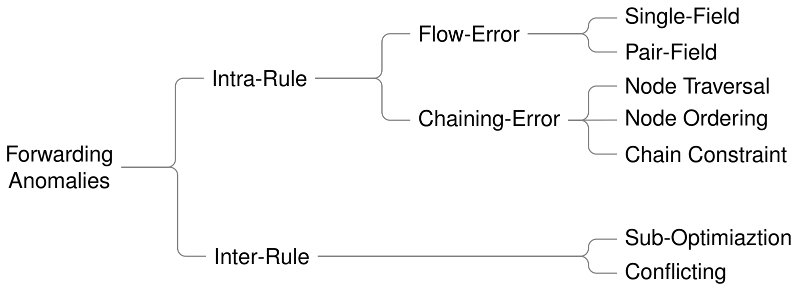
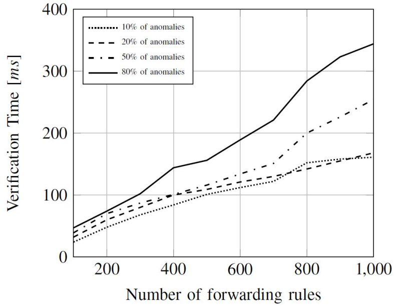
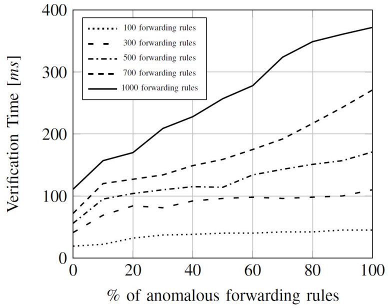


Figure: Hierarchy of anomaly classes.

Detecting Anomalies in SFCs: results

Verification time is influenced by both the *number* of forwarding rules and the *percentage* of these that trigger an anomaly





Reasonable verification times from both the NFV architecture and users perspectives!

Detecting Anomalies in SFCs: publications and next plans

Publications:

- *Valenza, F., Spinoso, S., Basile, C., Sisto, R., Lioy, A.*. **A formal model of network policy analysis**. In IEEE 1st International Forum on Research and Technologies for Society and Industry Leveraging a better tomorrow (2015).
- *Valenza, F., Su, T., Spinoso, S., Lioy, A., Sisto, R., Vallini, M.* **A formal approach for network security policy validation**. Journal of Wireless Mobile Networks, Ubiquitous Computing and Dependable Applications (2017).
- *Spinoso S., Sisto R.*, **Formally Specifying and Checking Policies and Anomalies in Service Function Chaining.** *Submitted major revision* to IEEE Transactions on Network and Service Management.

Next improvements:

- raise the abstraction-level of traffic flow modelling
- raise the expressiveness of the model
  - additional network operation (e.g. monitoring)

Formal Verification of Service Graphs

- *Detecting Anomalies in Service Function Chains*
- *Checking Reachability in Service Graphs*

Checking Reachability in Service Graphs

Forwarding errors may also be due to faulty VNF configurations at run-time

- e.g. faulty filtering rules in firewalls, wrong black list in anti-spams, DPIs etc...

Service Providers need more accurate modelling approaches to check the network correctness

Checking Reachability in SGs: contributions

State of the Art:

- most existing verification tools are OpenFlow-oriented
- they check network functions that take forwarding decisions based on packet headers only (stateless functions)

Contributions:

- Model networks and stateful VNFs
  - network functions that alter and forward packets based on internal states and algorithms

Checking Reachability in SGs: the approach

Boolean modelling and satisfiability checking techniques to verify **reachability properties** against **stateful VNFs**

VeriGraph:

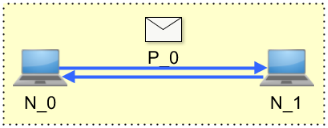
- Network and VNFs models are sets of FOL formulas
- Use Z3, an SMT solver, as verification engine
- Exploit Neo4J for service graph manipulation

Checking Reachability in SGs: network model

FOL formulas for modelling:

1. *network fundamentals*

$$(send(n_0,n_1,p_0,t_0)) \implies (n_0 \neq n_1 \land p_0.src \neq p_0.dst), \quad \forall n_0, p_0, t_0$$



2. *VNF behaviour* (e.g. NAT)

$$(send(nat, n_1, p_1, t_1) \land \neg isPrivateAddress(p_1.dst)) \implies p_1.src = ip\_nat$$
$$\land \exists(n_0, p_0, t_0) \mid (t_0 < t_1 \land recv(n_0, nat, p_0, t_0) \land isPrivateAddress(p_0.src)$$
$$\land p_0.dst = p_1.dst), \ \forall(n_1, p_1, t_1)$$

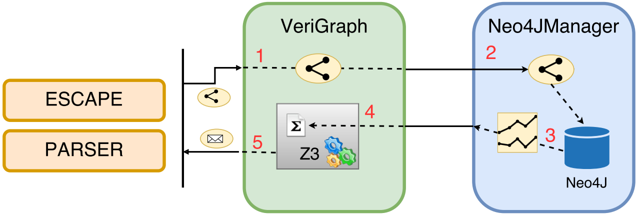
Checking Reachability in SGs: VeriGraph

Part of the SP-DevOps toolkit (UNIFY) and integrated into a VNF Orchestrator (*ESCAPE*)

- `https://github.com/netgroup-polito/verigraph`

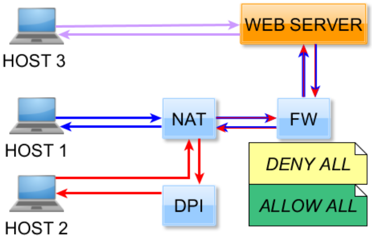
Part of the D-release of *Parser* (OPNFV)

- `https://github.com/opnfv/parser`



Checking Reachability in SGs: VeriGraph

Reach. Property	Result	VeriGraph	Z3	Verification Time
Host1 ->Server	X	263ms	30ms	293ms
Host2 ->Server	X	326ms	28ms	354ms
Host3 ->Server	V	256ms	54ms	310ms
Host1 ->Server	V	250ms	107ms	357ms
Host2 ->Server	V	295ms	65ms	360ms
Host3 ->Server	V	282ms	61ms	343ms



Checking Reachability in SGs: publications and next plans

Publications:

- *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 4th European Conference on Service-Oriented and Cloud Computing, 2015.

Next improvements:

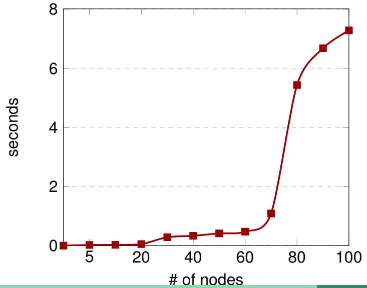
- Make the verification approach scalable
- Reduce the complexity of the modelling technique

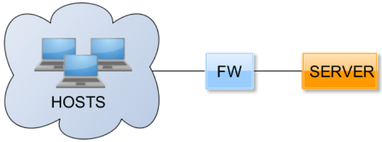
Checking Reachability in SGs: scalability issues

Boolean modelling-based approach is more promising then model checking techniques

- *FOL is not a decidable logic*

Make network and VNF models in ***Skolemized form*** (without existential quantifiers)





Network Function Configuration

Network Function Configuration

Service Providers have to *configure* VNFs to complete the service graph deployment

- filtering rules in firewalls, IP addresses for NATs...

Cloud Managers (CMs) rely on external configuration services

- e.g. Puppet, Chef, Ansible,...

Network Function Configuration: Challenges

Flexible configuration approaches have to consider:

- many configuration strategies per function
  - REST API, CLI, SMTP, etc...
- configuration semantics depend on the function types
  - router and firewall parameters are clearly different

Network Function Configuration

- *Seamless Configuration of Virtual Network Functions*

Seamless Configuration of VNFs: contributions

State of the art:

- existing configuration services are targeted to expert users
- they use one configuration strategy
- they rely on VNF-specific plug-ins

Contributions:

- Enable a **model-based VNF configuration** in CMs to:
  - hide the low-level details to the users
  - support many configuration strategies
  - use vendor-agnostic and function-independent modules

Seamless Configuration of VNFs: overview

Service Providers need configuration modules that are *vendor-agnostic* and *function-independent*:

- ***Translator***
  - translates the configuration parameters into a particular format required by a VNF
- ***Gateway***
  - delivers the produced configuration into the VNF

Seamless Configuration of VNFs: inputs

VNF Object Model (VNF OM)

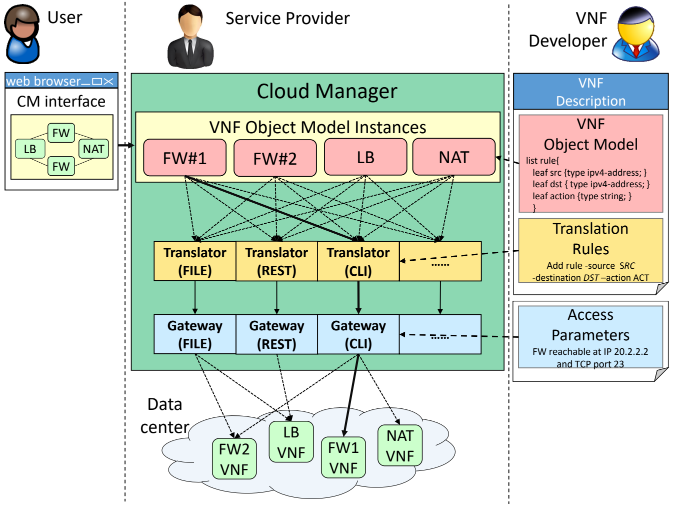
- representation of the main VNF configuration parameters
- one VNF Object Model Instance is created for each VNF

Translation Rules

- directives to translate the VNF OM instance into the structure/format required by the VNF

Access Parameters

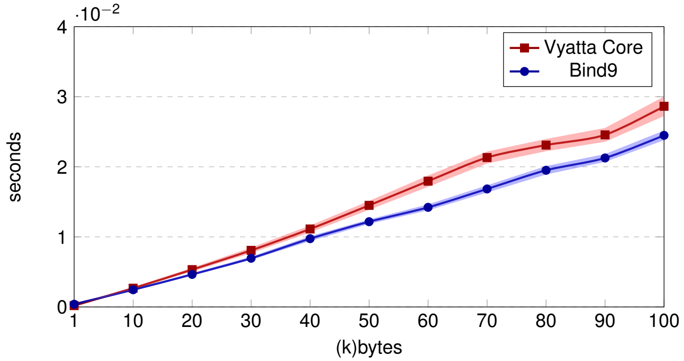
- directives to push down the VNF configuration



```
list rule{
  leaf src {type ipv4-address; }
  leaf dst { type ipv4-address; }
  leaf action {type string; }
}
```

Seamless Configuration of VNFs: results

Two types of VNF were successfully configured in software cloud architecture provided by **PLUMgrid, Inc**:



Seamless Configuration of VNFs: publications and next plans

Publications:

- *Spinoso, S., Leogrande, M., Risso, F., Singh, S., Sisto, R.*, **Automatic Configuration of Opaque Network Functions in CMS.** In IEEE/ACM 7th International Conference on Utility and Cloud Computing (2014).
- *Spinoso, S., Leogrande, M., Risso, F., Singh, S., Sisto, R.*, **Seamless Configuration of Virtual Network Functions in Data Center Provider Networks.** In Journal of Network and Systems Management (2017).

Next improvements:

- integration with the verification service to
  - check the correctness of the VNF configuration generated
  - enable an automatic fixing in case of errors

Conclusions

Improvements to the state of the art in many aspects of a network service life-cycle:

- *Verification of anomalies in a forwarding policy*
- *Reachability analysis in service graphs*
- *Model-based functional configuration of network functions*

Publication List

2014:

- *Spinoso S., Leogrande M., Risso F., Singh S., Sisto R.*, **Automatic Configuration of Opaque Network Functions in CMS.** In: 1st International Workshop on Network Virtualization and Software-Defined Networks for Cloud Data Centres (NVSDN 2014).

2015:

- *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.
- *Valenza F., Spinoso S., Basile C., Sisto R., Lioy A.*, **A Formal Model of Network Policy Analysis** In: First International Forum on Research and Technologies for Society and Industry (RTSI2015), Turin, Italy, 16-18 September, 2015.

Publication List

2016:

- *Valenza F., Su T., Spinoso S., Lioy A., Sisto R., Vallini M.*, **A formal approach for network security policy validation**. Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications.

2017:

- *Spinoso S., Leogrande M., Risso F., Singh S., Sisto R.*, **Seamless Configuration of Virtual Network Functions in Data Center Provider networks.** Journal of Network and Systems Management, Springer.
- *Spinoso S., Sisto R.*, **Formal Verification of Forwarding Policies.** Submitted major revision to: Transactions on Network and Service Management, IEEE.

Planned:

- *Spinoso S., Virgilio M., Marchetto G., Sisto R.*, **VeriGraph: Verifying complex service Graphs.** To submit in 2017.

Questions