NFV Research Group
Internet-Draft
Intended status: Informational
Expires: September 6, 2017

M-K. Shin, Ed.
ETRI
K. Nam
Friesty
S. Pack
KU
S. Lee
ETRI
R. Krishnan
Dell

March 6, 2017

Verification of NFV Services : Problem Statement and Challenges
draft-irtf-nfvrg-service-verification-03

Abstract

NFV relocates network functions from dedicated hardware appliances to generic servers, so they can run in software. However, incomplete or inconsistent configuration of virtualized network functions (VNFs) and forwarding graph (FG, aka service chain) could cause break-down of the supporting infrastructure. In this sense, verification is critical for network operators to check their requirements and network properties are correctly enforced in the supporting infrastructures. Recognizing these problems, we discuss key properties to be checked on NFV services. Also, we present challenging issues related to verification in NFV environments.

Table of Contents

1. Introduction

NFV is a network architecture concept that proposes using IT virtualization related technologies, to virtualize entire classes of network service functions into building blocks that may be connected, or chained, together to create network services. NFV service is defined as a composition of network functions and described by its functional and behavioral specification, where network functions (i.e., firewall, DPI, SSL, load balancer, NAT, AAA, etc.) are well-defined, hence both their functional behavior as well as their external interfaces are described in each specifications.

In NFV, a VNF is a software package that implements such network functions. A VNF can be decomposed into smaller functional modules or APIs for scalability, reusability, and/or faster response [ETSI-NFV-Arch],[ETSI-NFV-MANO]. This modular updates or composition for a network function may lead to many other verification or security issues. In addition, a set of ordered network functions which build FGs may be connected, or chained, together to create an end-to-end network service. Multiple of VNFs can be composed together to reduce management and VNF FGs. While autonomic networking techniques could be used to automate the configuration process including FG updates, it is important to take into account that incomplete and/or inconsistent configuration may lead to verification issues. Moreover, automation of NFV process with integration of SDN may lead the network services to be more error-prone. In this sense, we need to identify and verify key properties to be correct before VNFs and FGs are physically placed and realized in the supporting infrastructure.

1.1. Terminology

This document draws freely on the terminology defined in [ETSI-NFV-Arch].

2. Problem statement

The verification services should be able to check the following properties:

2.1. Dependencies of network service components in NFV framework

In NFV framework, there exist several network service components including NFVI, VNFs, MANO, etc. as well as network controller and switches to realize end-to-end network services. Unfortunately, these components have intricate dependencies that make operation incorrect. In this case, there is inconsistency between states stored and managed in VNF FGs and network tables (e.g., flow tables), due to

communication delays and/or configuration errors. For example, if a VNF is replicated into the other same one for the purpose of load balance and a new FG is established through the copied one, but all the state/DBs replication is not finished yet due to delays, this can be lead to unexpected behaviors or errors of the network service. Therefore, these dependencies make it difficult to correctly compose NFV-enabled end-to-end network services.

2.2. Invariant and error check in VNF FGs

In VNF FGs, an infinite loop construction should be avoided and verified. Let us consider the example. Two VNF A and VNF B locate in the same service node X whereas another VNF C resides in other service node Y [SIGCOMM-Gember]. Also, the flow direction is from X toY, and the given forwarding rule is A->C->B. In such a case, service node Y can receive two ambiguous flows from VNF A: 1) one flow processed by VNF A and 2) another flow processed by VNF A, B, and C. For the former case, the flow should be processed by VNF C whereas the latter flow should be further routed to next service nodes. If these two flows cannot be distinguished, service node Y can forward the flow to service node X even for the latter case and a loop can be formed. To avoid the infinite loop formation, the forwarding path over VNF FG should be checked in advance with the consideration of physical placement of VNF among service nodes. Also, reactive verification may be necessary, since infinite loop formation may not be preventable in cases where configuration change is happening with live traffic.

In addition, isolation between VNFs (e.g. confliction of properties or interference between VNFs) and consistent ordering of VNF FGs should be always checked and maintained.

2.3. Load balancing among VNF instances

In VNF FG, different number of VNF instances can be activated on several service nodes to carry out the given task. In such a situation, load balancing among the VNF instances is one of the most important considerations. In particular, the status in resource usage of each service node can be different and thus appropriate amount of jobs should be distributed to the VNF instances. To guarantee well-balanced load among VNF instances, the correctness of hash functions for load balancing needs to be verified. Moreover, when VNF instances locate in physically different service nodes, simple verification of load balancing in terms of resource usage is not sufficient because different service nodes experience diverse network conditions (e.g., different levels of network congestion)[ONS- Gember]. Therefore, it is needed to monitor global network condition as well as local resource condition to achieve the network-wide load balancing in VNF

FGs. Also, whether the monitoring function for network/compute/storage resources is correctly working should be checked.

2.4. Policy and state consistency on NFV services

In VNF FG, policy to specific users can be dynamically changed. For example, a DPI VNF can be applied only in the daytime in order to prohibit from watching adult contents while no DPI VNFs applied during the nighttime. When the policy is changed, the changed policy should be reconfigured in VNF service nodes as soon as possible. If the reconfiguration procedure is delayed, inconsistent policies may exist in service nodes. Consequently, policy inconsistency or confliction needs to be checked. Also in some situations, states for VNF instances may be conflicted or inconsistent. Especially when a new VNF instance is instantiated for scale-up and multiple VNF instances are running, these multiple VNF instances may have inconsistent states owing to inappropriate instantiation procedure [SIGCOMM-Gember]. In particular, since the internal states of VNF instances (e.g., the instantaneous state of CPU, register, and memory in virtual machine) are not easily-visible, a new way to check the VNF internal states should be devised.

2.5. Performance

In VNF FG, VNF instances can locate in different service nodes and these service nodes have different load status and network conditions. Consequently, the overall throughput of VNF FG is severely affected by the service nodes running VNF instances. For example, if a VNF instance locates in a heavily loaded service node, the service time at the service node will be increased. In addition, when a VNF FG includes a bottleneck link with network congestion, the end-to-end performance (e.g., latency and throughput) in the VNF FG can be degraded. Therefore, the identification of bottleneck link and node is the first step for performance verification or guarantee of the VNF FG [ONS-Gember]. After detecting the bottleneck link/node, the VNF requiring scale up or down can be identified and the relocation of VNF instance among service nodes can be determined

2.6. Security

How to verify security holes in VNF FG is another important consideration. In terms of security services, authentication, data integrity, confidentiality, and replay protection should be provided. On the other hand, several VNFs (e.g., NAT) can modify or update packet headers and payload. In these environments, it is difficult to protect the integrity of flows traversing such VNFs. Another security concern in the VNF FG is distributed denial of service (DDoS) to a

specific service node. If an attacker floods packets to a target service node, the target service node cannot perform its functions correctly. Therefore, such security attacks in the VNF FG should be detected and handled in an efficient manner. In the case of DDoS, adding a DDoS appliance as the first element in the service chain would help alleviate the problem. Moreover, unknown or unauthorized VNFs can run and thus how to identify those problems is another security challenge.

3. Examples - NS policy conflict with NFVI policy

Another target of NFV verification is conflict of NS policies against global network policy, called NFVI policy.

NFV allocates and manages NFVI resources for a network service according to an NS policy given in the network service descriptor (NSD), which describes how to govern NFVI resources for VNF instances and VL instances to support KPIs of the network service. Example factors of the NS policy are resource constraints (or deployment flavor), affinity/anti-affinity, scaling, fault and performance management, NS topology, etc.

For a network-wide (or NS-wide) management of NFVI, NFVI policy (or global network policy) can be provided to describe how to govern the NFVI resources for optimized use of the infrastructure resources (e.g., energy efficiency and load balancing) rather than optimized performance of a single network service. Example factors of the NFVI policy are NFVI resource access control, reservation and/or allocation policies, placement optimization based on affinity and/or anti-affinity rules, geography and/or regulatory rules, resource usage, etc.

While both of the policies define the requirements for resource allocation, scheduling, and management, the NS policy is about a single network service; and the NFVI policy is about the shared NFVI resources, which may affect all of the given network services globally. Thus, some of NS and NFVI policies may be inconsistency with each other when they have contradictive resource constraints on the shared NFVI resources. Examples of the policy conflicts are as follows:

<Example conflict case #1>

- NS policy of NS_A (composed of VNF_A and VNF_B)
  - Resource constraints: 3 CPU core for VNF_A and 2 CPU core for VNF_B
  - Affinity rule between VNF_A and VNF_B

o NFVI policy
- No more than 4 CPU cores per physical host

o Conflict case
- The NS policy cannot be met within the NFVI policy

<Example conflict case #2>

o NS policy of NS_B (composed of VNF_A and VNF_B)
- Affinity rule between VNF_A and VNF_B

o NFVI policy
- Place VM whose outbound traffic is larger than 100Mbps at POP_A
- Place VM whose outbound traffic is smaller than 100Mbps at POP_B

o Conflict case
- If VNF_A and VNF_B generate traffic in 150Mbps and 50Mbps, respectively,
- VNF_A and VNF_B need to be placed at POP_A and POP_B, respectively according to the NFVI policy
- But it will violate the affinity rule given in the NS policy

<Example conflict case #3>

o NS policy of NS_C (composed of VNF_A and VNF_B)
- Resource constraints: VNF_A and VNF_B exist in the same POP
- Auto-scaling policy: if VNF_A has more than 300K CPS, scale-out

o NFVI policy
- No more than 10 VMs per physical host in POP_A

o Conflict case
- If CPS of VNF_A in POP_A gets more than 300K CPS,
- and if there is no such physical host in the POP_A whose VMs are smaller than 10,
- VNF_A need to be scaled-out to other POP than POP_A according to the NFVI policy
- But it will violate the NS policy

4. Requirements of verification framework

The verification framework addressed in this document follows [ETSI-NFV-Testing]. [ETSI-NFV-Testing] covers the following aspects of pre-deployment testing: 1) assessing the performance of the NFVI and its ability to fulfil the performance and reliability requirements of the VNFs executing on the NFVI, 2) data and control plane testing of VNFs and their interactions with the NFV Infrastructure and the NFV MANO,

and 3) validating the performance, reliability and scaling capabilities of network services.

A verification framework for NFV-based services also needs to satisfy the following requirements:.

- R1 : It should be able to check global and local properties and invariants. Global properties and invariants relate to the entire VNFs, and local properties and invariants relates to the specific domain or resources that some of the VNFs are using. For example, Loop-freeness and isolation between VNFs can be regarded as global. The policies that are related only to the specific network controllers or devices are local.

- R2 : It should be able to access to the entire network states whenever verification tasks are started. It can directly manage the states of network and NFV-based services through databases or any solution that specializes in dealing with the network topology and configurations, or can utilize the functions provided by NFV M&O and VNFI solutions to get or set the states at any time.

- R3 : It should be independent from specific solutions and frameworks, and provide standard APIs.

- R4 : It should process standard protocols such as NetConf, YANG, OpenFlow, and northbound and southbound interfaces that are related network configurations, and used by OSS.

5. Challenging issues

There are emerging challenges that the verification services face with.

5.1. Consistency check in distributed state

Basically, NFV states as well as SDN controllers are distributed. writing code that works correctly in a distributed setting is very hard. Therefore, distributed state management and consistency check has challenging issues. Some open source project such as ONOS offers a core set of primitives to manage this complexity. RAFT algorithm [RAFT] is used for distribution and replication. Similarly, Open daylight project has a clustering concept to management distributed state. There is no "one-size-fits-all" solution for control plane data consistency.

5.2. Intent-based service composition

Recently, Intent-based high-level language is newly proposed and discussed in open source project. The Intent allows for a descriptive way to get what is desired from the infrastructure, unlike the current NFV description and SDN interfaces which are based on describing how to provide different services. This Intent will accommodate orchestration services and network and business oriented SDN/NFV applications, including OpenStack Neutron, Service Function Chaining, and Group Based Policy. A Intent compiler that translates and compiles it into low level instructions (e.g., SDN controller/OpenStack primitives) for network service components. In this sense, error checking and debugging are critical for reliable Intent-based service composition.

5.3. Finding infinite loops

General solutions for the infinite loop can lead to intractable problem (e.g. the halting problem). To make the verification practical and minimize the complexity, some of the restrictions are required. Finding cycle can be processed in polynomial time but the restriction could be too much for some cases that service functions or network flows requires finite loops.

5.4. Live traffic verification

It is known fact that the complexity of verification tasks for the real and big problem is high. A few invariants can be checked in real-time but it would be impossible if the size of VNFs increases or properties to be checked are complex.

5.5. Languages and their semantics

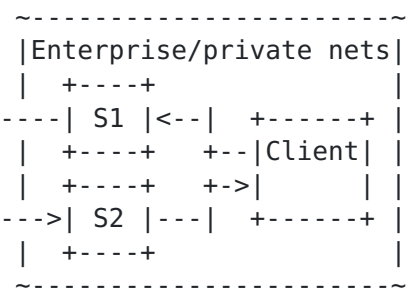
For the verification, configurations and states of VNFs need to be precisely expressed using formal semantics. There are many languages and models, and it is impractical for the verification frameworks to support all of the existing languages and models. Languages and semantic models optimized to the verification framework need to selected or newly developed.

5.6. Stateful VNFs with multiple physical views

The correctness of VNFs whose behaviors depend on the previous states (packets, actions, etc) and whose physical entities are multiple should be checked differently than the stateless ones. Such VNFs include firewall, load balancer, NAT, flow rules with counter or soft timeout.

- Case 1:
  If a firewall service is implemented over two physical Openflow switches, there could be two paths that the client-server packets go through. If the packets between client and server go through the same switch, firewall functions correctly. However if packets from client to server go through S1 but packets from server to client come back through S2, those flows could be blocked and lead to false-negative result.

  To mitigate the situation, states of all instances for one logical VNF must be considered to verify the correctness.



- Case 2:
  If there are VNFs whose behavior depend on the previous VNF, those dependency must be considered as well.

  For example, if firewall and load balancer gets packets go through NAT service, they need to know the header mapping information that the NAT have set to correctly process their functions. If the FG consists of IPS followed by DPI and those functions are connected different switches, the switch connecting DPI must know if the incomming packets should be forwarded to DPI or not. Port knocking is also well-known example of stateful function.

  To mitigate the situation, the states of all VNFs having behavioral dependency must be considered when they are verified.

6. Gap analysis - open source projects

Recently, the Open Platform for NFV (OPNFV) community is collaborating on a carrier-grade, integrated, open source platform to accelerate the introduction of new NFV products and services [OPNFV]. Open Daylight (ODL) is also being tightly coupled with this OPNFV platform to integrate SDN controller into NFV framework [ODL].

This clause analyzes the existing open source projects including

OPNFV and ODL related to verification of NFV services.

6.1. OPNFV

6.1.1. Doctor

The Doctor project provides a NFVI fault management and maintenance framework on top of the virtualized infrastructure. The key feature is to notify unavailability of virtualized resources and to recover unavailable VNFs.

While the Doctor project focuses only on faults in NFVI including compute, network, and storage resources, the document discusses broader fault management issues such as break-down of the supporting infrastructure due to incomplete or inconsistent configuration of NFV services.

6.1.2. Prediction

The Prediction project provides a data collection for failure prediction framework. The failure prediction framework diagnoses or verifies which entity is suspected to be progressing towards a failure and which VNFs might be affected due to the predicted anomaly.

While the Prediction project focuses only on fault prediction in NFVI compute, network, and storage resources, the document includes broader fault management and prediction issues such as faults in the NFV service deployment and operation.

6.1.3. Resource Scheduler

The Resource Scheduler project provides an enhanced scheduler for optimizing the performance of the VNFs. In particular, this project supports resource isolation. For example, when a VNF strictly requires low latency, strongly isolated compute resources can be allocated to the VNF.

The Resource Scheduler project only focuses on optimizing the performance of individual VNFs without considering the end-to-end performance (e.g., latency and throughput) in NFV services.

6.1.4. Moon

The Moon project implements a security management system for the cloud computing infrastructure. The project also enforces the security managers through various mechanisms, e.g., authorization for access control, firewall for networking, isolation for storage, and

logging for tractability.

Note that the main interest of the Moon project is the DDoS attack to a service node and the IDS management for VNFs. A wider range of security issues in the NFV service verification need to be discussed.

6.1.5 Bottlenecks

The Bottlenecks project aims to find system bottlenecks by testing and verifying OPNFV infrastructure in a staging environment before committing it to a production environment. Instead of debugging the deployment in production environment, an automatic method for executing benchmarks to validate the deployment during staging is adopted. For example, the system measures the performance of each VNF by generating workload on VNFs.

The Bottlenecks project does not consider incomplete or inconsistent configurations on NFV services that might cause the system bottlenecks. Furthermore, the Bottlenecks project aims to find system bottlenecks before committing it to a production environment. Meanwhile, the draft also considers how to find bottlenecks in real time.

6.2. ODL

6.2.1. Network Intent Composition

The Network Intent Composition project enables the controller to manage and direct network services and network resources based on intent for network behaviors and network policies. Intents are described to the controller through a new northbound interface, which provides generalized and abstracted policy semantics. Also, the Network Intent Composition project aims to provide advanced composition logic for identifying and resolving intent conflicts across the network applications.

When the reconfiguration upon the policy (i.e, intent) is delayed, policy inconsistency in service nodes may occur after the policy is applied to service nodes. While the Network Intent Composition project resolves such intent conflicts only before they are translated into service nodes, this document covers intent conflicts and inconsistency issues in a broader sense.

6.2.2. Controller Shield

The Controller Shield project proposes to create a repository called

unified-security plugin (USecPlugin). The unified-security plugin is a general purpose plugin to provide the controller security information to northbound applications. The security information could be for various purposes such as collating source of different attacks reported in southbound plugins and suspected controller intrusions. Information collected at this plugin can also be used to configure firewalls and create IP blacklists for the network.

In terms of security services, the document covers authentication, data integrity, confidentiality, and replay protection. However, the Controller Shield project only covers authentication, data integrity, and replay protection services where the confidentiality service is not considered.

6.2.3. Defense4All

The Defense4All project proposes a SDN application for detecting and mitigating DDoS attacks. The application communicates with ODL controller via the northbound interface and performs the two main tasks; 1) Monitoring behavior of protected traffic and 2) Diverting attacked traffic to selected attack mitigation systems (AMSs).

While the Defense4All project only focuses on defense system at the controller, this document includes broader defense issues at the service node as well as the controller.

6.3. Summary

The verification functions should spreads over the platforms to accomplish the requirements mentioned in clause 3. The correctness of NFV- based services and their network configurations can be checked in the NFV MANO layer which has the entire states of the VNFs. Each NFVI needs to provide verification layer which composed of policy manager, network database and interfaces (e.g. REST APIs). Local properties and invariants can be verified inside the specific NFVI, and the global properties and invariants can be checked by merging local verification results from the related NFVIs.

The verification service provides verification functions to NFV MANO, NFVI, and any other low-level modules such as SDN controllers. For the platform independency, it provides standard APIs to process the verification tasks. It also uses standard APIs provided by OSS such as OpenStack (Neutron) and Open Daylight. The compiler and interpreter translate standard description languages and protocols into the internal model which optimized to the verification tasks. It can process user-defined properties to be checked as well. The properties to be checked whether they are user-defined or pre-defined invariants are managed by property library. The verifier maintains a

set of verification algorithms to check the properties. The network database inside the verification service manages the global network states directly or indirectly.

A PoC can be implemented using OpenStack (Neutron) and Open Daylight. The modules related to verification framework can reside in between network virtualization framework (e.g. OpenStack Neutron) and SDN controller (e.g. Open Daylight). Neutron and Open Daylight uses standard APIs provided by verification service to accomplish verification tasks. The initial use case for the PoC could be, in particular, any of security, performance, etc as mentioned in clause 2.

7. Security Considerations

As already described in clause 2.6, how to verify security holes in VNF FG is very important consideration. In terms of security services, authentication, data integrity, confidentiality, and replay protection should be provided. On the other hand, potential security concern should be also carefully checked since several VNFs (e.g., NAT) can modify or update packet headers and payload.

8. Acknowledgements

The authors would like to thank formal methods lab members in Korea University for their verification theory support.

9. References

9.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

9.2. Informative References

[ETSI-NFV-Arch] ETSI, "Network Function Virtualisation (NFV); Architectural Framework," 2014.

[ETSI-NFV-MANO] ETSI, "Network Function Virtualiztion (NFV) Management and Orchestration," 2014.

[ETSI-NFV-Testing] ETSI, "Network Function Virtualiztion (NFV) Pre-deployment Testing; Report on Validation of NFV

Environments and Services," 2016.

[SIGCOMM-Qazi] Z. Qazi, C. Tu, L. Chiang, R. Miao, V. Sekar, and M. Yu, "SIMPLE-fying Middlebox Policy Enforcement Using SDN," in Proc. ACM SIGCOMM 2013, August 2013.

[ONS-Gember] A. Gember, R. Grandl, A. Anand, T. Benson, and A. Akella, "Stratos: Virtual Middleboxes as First-Class Entities," ONS 2013 and TR.

[SIGCOMM-Gember] A. Gember, R. Viswanathan, C. Prakash, R. Grandl, J. Khalid, S. Das, and A. Akella, "OpenNF: Enabling Innovation in Network Function Control," in Proc. ACM SIGCOMM 2014, August 2014.

[HOTSDN-Ghorbani] S. Ghorbani, B. Godfrey, "Towards Correct Network Virtualization", HOTSDN 2014.

[RAFT] https://raftconsensus.github.io/.

[ODL] "OpenDaylight SDN Controller, "http://www.opendaylight.org/

[OPNFV] "Open Platform for NFV, "https://www.opnfv.org/

Authors' Addresses

Myung-Ki Shin
ETRI
161 Gajeong-dong Yuseng-gu
Daejeon, 305-700
Korea

Phone: +82 42 860 4847
Email: mkshin@etri.re.kr

Ki-Hyuk Nam
Friesty

Email: nam@friesty.com

Sangheon Pack
Korea University

Email: shpack@korea.ac.kr

Seungik Lee
ETRI
161 Gajeong-dong Yuseng-gu
Daejeon, 305-700
Korea

Phone: +82 42 860 1483
Email: seungiklee@etri.re.kr

Ramki Krishnan
Dell

Email: Ramki_Krishnan@dell.com