Compound implementation of chained network functions and virtual resource management performance evaluation

Wolfgang Hahn, Borislava Gajic, Christian Mannweiler
Nokia Networks
St. Martin-Straße 76, 81541 Munich, Germany
{wolfgang.hahn; borislava.gajic, christian.mannweiler}@nokia.com

***Abstract*—The paper discusses different implementation options of service function chaining (SFC) in Cloud Computing. The motivation is to increase the efficiency of classical SFC of chaining multiple network functions implemented in dedicated virtual machines, especially due to its latency drawbacks for packet transfer. An alternative implementation of a compound SFC implementation in a single virtual machine is proposed. The focus is not primarily on the latency drawback but on an improved resource management. In order to evaluate the presented concepts a traffic model that resembles the traffic pattern of a real network is applied. Example calculations give an indication on the sensitivity of different parameters to efficiency of the SFC implementation options and proposed management solutions.**

Keywords— Service Function Chaining; Cloud computing; VNF Management; Resource pooling

I. INTRODUCTION

Due to the rapid increase of the number and versatility of mobile devices and applications the mobile network operators have been recently faced with the problem of increased TCO, CAPEX and OPEX. As one of the means to reduce operators' costs, the network functions virtualization (NFV) in common Data Center (DC) hardware for different network functions is increasingly utilized.

Virtualization is also applied for service function chaining (SFC). Today, mobile network operators are providing numerous value adding packet processing functions in the so called "Gi-LAN" (such as NAT, TCP optimizers, firewalls, video optimizers) that are implemented by chains of boxes the traffic needs to be routed through. Inefficiency and management complexity of those solutions have encouraged the work on optimizing the implementation.

A next level of network flexibility is expected to be introduced with the 5G architecture. The EU Horizon 2020 funded project 5G NORMA [1] aims to develop a network that is able to adapt its functions in RAN and Core to multiple services in a context-aware way. Enabling technologies used are adaptive (de)composition and allocation of virtualized mobile network functions and software-defined mobile network control. This includes the service-specific selection of both the flavor and sequence of the decomposed network functions. A consequence of this modular design is the increased need to efficiently manage, compose and connect network services from a high number of elementary functional modules.

(Note: In this paper, the term Network Function (NF) is used in the context of virtualization in general. The term Service Functions (SF) is used in the context of SF chaining.)

Most of the current approaches aim at virtualizing the former physical "box-functions" to become VNFs (Virtual Network Functions) and use the connectivity framework provided by the data center to create the required data path. Hereafter this implementation option is referred to as ***classical SFC design***, see Figure 1.

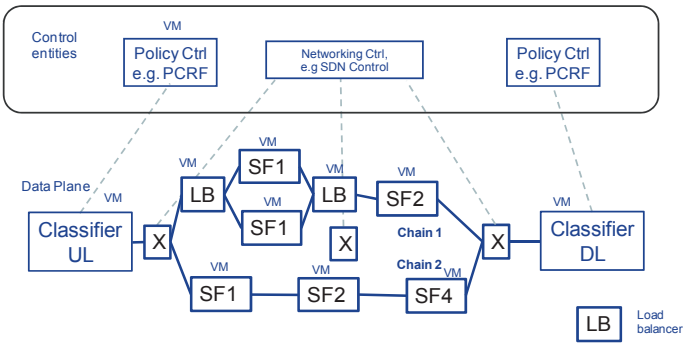


Fig. 1. A classical SFC design implementation option.

Different acceleration techniques (e.g. Intel DPDK, SR-IOV) can be used to improve the connectivity between VMs, however, a number of problems related to classical SFC design remain: (1) Additional latency and jitter are introduced by communication between virtual machines (even if accelerated) depending on the topology and different used acceleration mechanisms. (2) There is the need to scale and load-balance each individual SF. The problem arises from the fact that scaling in DC is usually implemented by increasing (scale out) or decreasing (scale in) the number of VMs used to run a specific network function. (3) Due to the high number of VMs and restrictions that need to be taken into account (delay

978-1-5090-0223-8/16/$31.00

budget, throughput...) by the virtualized infrastructure manager (VIM), resource allocation and scheduling of VM/VNF instantiation becomes more complex.
In order to address such constraints, another approach that relies on the implementation of a complete SFC (or at least parts of a chain) on a single virtual machine is proposed. Hereafter this SFC implementation option is referred to as ***compound SFC design***.

Advantages of a compound SFC design are:

- Predictable performance, especially for the E2E latency of a service chain. There is no need for complex topology or transport mechanisms considerations for achieving the required latency.
- Easier load balancing as not required for individual SF but per service function chain.
- Easier and very dynamic scaling in and out – resource reuse (removing not used SF chains and deploying new SF chains on already existing VMs)

A disadvantage of the compound SFC is the missing isolation of individual SFs. But this matter seems less important for functions that are provided by one vendor. Isolation can be achieved for the complete SFC only. Furthermore, requirements of the individual SFs in terms of VM resource restrictions need to be taken into account when composing the SFC.

The remainder of the paper is organized as follows. Section 2 gives a brief overview of the related work. Section 3 describes the new concept of inner management loop applied with the compound SFC design. Section 4 gives an explanation on the traffic model used to evaluate the different SFC implementation options along with performance calculation of different SFC implementations and management solutions. The obtained results are presented and the main findings discussed. Finally, Section 5 draws conclusions.

II. Related work –

The key enablers for constantly delivering cutting edge network functions at lower costs are NFV [2] and Software Defined Networking (SDN) [3]. Whereas NFV enables deploying the NF as a software on standardized hardware, SDN decouples the control plane and the data plane, thus allowing the programmability on the flow level. Often, NFV and SDN are seen as enablers for SFC implementation. However, deploying the SFC as a composition of VMs running the individual SF has a latency issue due to the communication between VMs. This has led to numerous research activities to ensure fast packet processing in DC with general-purpose hardware. An overview is given in [4]. In particular virtual switches are accelerated especially for chaining network functions [5] or programmable NICs are used for this purpose [6].

Another option is to utilize the DirectPath I/O [7] acceleration technique. Passthrough and SR-IOV [8] are further approaches for acceleration of VM to VM communication. Some recent works [9] have analyzed the benefits of using the Data Plane Development Kits (DPDKs) for running VNFs. The results show very good (i.e., near-native) performance. However, as mentioned above, although networking performance between VMs that host the SF can be improved by such acceleration means, the benefits are subject to availability and might not be sufficient to overcome the latency introduced by the classical SFC deployment.

III. Inner and outer management loop for compound SFC

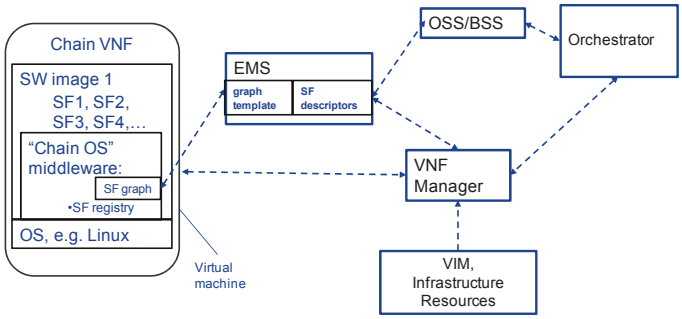
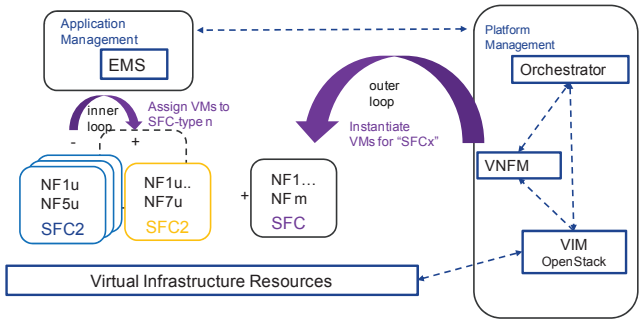
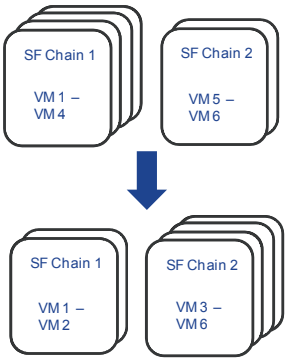


Fig. 2. Compound SFC design

The implementation as shown in Figure 2, containing all software of different SFs in a single VM, allows to activate a particular SFC as compound SFC during runtime. This is done by configuration management (e.g., under the control of the element management system, EMS). The configuration data e.g. in the format of a SFC graph representation is processed by a "Chain OS middleware" in the Chain VNF VM (details of that middleware are not in the scope of this paper).



(a) Inner and outer loop concepts



(b) Example of reprogramming of VMs in inner loop

Fig. 3. Compound SFC implementation

By shifting resource management responsibilities to a local SFC management level, the management system is able to react to changed traffic conditions much faster compared to the level of scaling and orchestrating VMs. Such localized management system can be used to adapt the usage of the VM in a way that the available resources are distributed among the SFC in an optimal way: during runtime, VMs can be reconfigured in a way that in those VMs hosting underutilized chains another chain requiring a higher network throughput can be activated.

This mechanism introduces an "inner loop" of control for optimal use of allocated resources/VMs for service function chaining (Figure 3). As a consequence, the alternative operation of increasing or decreasing the overall number of VMs for SF Chaining (referred to as "outer loop") by involving entities such as the Orchestrator needs to be done less frequently.

Advantages of the inner loop management include easier and very dynamic scaling in and out per chain by resource reuse (removing not used SFCs and deploying new SFCs on already existing VMs). The process can be performed much more dynamically than VM scaling.

IV. PERFORMANCE EVALUATION

To evaluate the concepts of compound SFCs and inner vs. outer loop management, a set of calculations based on different use cases has been carried out. Performance indicators used are the number of VMs needed for a given traffic load and the number of scaling operations. The next section presents the utilized traffic model and basic assumptions.

A. Traffic model and assumptions

As a baseline for calculating the performance of the compound SFC and inner/outer loop a traffic model over one week that resembles real network traffic has been used. For simplification, traffic of one day is considered only.

The calculations assume the existence of 1 million users in the considered network that generate the traffic to be carried in two different SFCs (SFC1 and SFC2). SFC1 is composed of SF1 and SF2, whereas SFC2 is composed of SF1 and SF3. It is further assumed that traffic from different SFCs is handled in dedicated VMs that deploy the software images for required SFCs. Such dedicated VMs are part of the common VM pool that is initially created by the Orchestrator. The objective of such a pool of VM instances is to have a (reasonable) number of VMs reserved for SFs or SFCs. One of the challenges therefore includes pool dimensioning with pool size possibly varying over time.

The investigations included the evaluation of performance for different numbers of VMs in a pool (ranging from 30 VMs up to 250 VMs) as well as different portions of VMs that deployed the software images for particular SFCs (e.g. 75%-25% or 50%-50% between SFC1 and SFC2, respectively). The inner loop, i.e. reprogramming of underutilized VMs to serve the more demanding SFC, is done once the load on VMs drops below a threshold (total traffic load is equally spread over VMs hosting a particular type of SFC, e.g., SFC1). The threshold parameters can be adapted to optimize the algorithm, here 22% was used. On the other hand, there is a need to trigger such reprogramming of VMs only if the load of VMs handling one chain goes above a threshold, 80% in the example. In order to support service continuity, of always at least one underutilized VM is not re-programmed, such that it can continue serving even underutilized chains.

For the VM capability it is assumed that one VM can handle 1 Gbps of SF1 traffic, 2 Gbps of SF2 traffic and 3 Gbps of SF3 traffic, which leads to the conclusion that one VM can handle 0.66 Gbps of SFC1 traffic and 0.75 Gbps of SFC2 traffic.

Both SFC1 and SFC2 are handling the traffic that resembles smartphone users pattern, however the throughput requirements of SFCs are shifted in time (peak of traffic is occurring with the difference of 6 hours, the peak throughput requirements of SFC1 is 20% higher than that of SFC2).

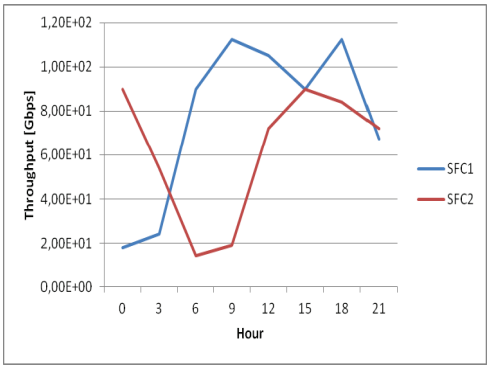


Fig. 4. Traffic model 2

B. Results

The first performance evaluation test focused on calculating the overall number of required VMs. The VM pool size and the share of the VMs handling SFC1 and SFC2 in a pool is varied. The pool of VMs represent the set of dedicated virtualized resources which serves to accommodate the traffic load of the traffic model. The portion of available resources in the pool are defined that are dedicated to each of the SFCs (pool share). In order to evaluate the compound SFC solution, different shares between SFCs (ranging from 90% of the pool which is dedicated to SFC1 and 10% dedicated to SFC2, towards 10% being dedicated to SFC1 and 90% to SFC2) are used. For classical SFC design, three sharing schemes are assumed, one assuming completely flexible (optimal) allocation of VMs from the pool to SFs that are building blocks of SFCs, i.e. best share, whereas the other two approaches assume some percentage of VMs in a pool dedicated to specific SFs. It is noticeable from Figure 5 that regardless of the initial SFC pool share among VMs in a pool the number of required VMs needed for handling the traffic model stays relatively stable for compound SFC implementation. This effect is a result of inner loop triggering. When initial pool share is not proportional to the traffic pattern the inner loop will be more active and will balance the initial mismatch.

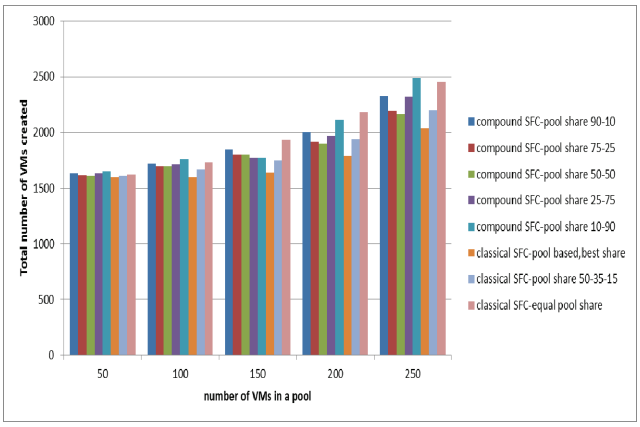


Fig. 5. The number of VMs needed for handling traffic model 2 when different VM pool sizes and SFC shares within the pool are applied

As shown in Figure 5, the compound and classical design show similar results. Due to the full flexibility of classical design, i.e. best share approach between VMs in the pool, the classical design shows slightly better results.In order to get more concise results, further focus was put on only one pool size that guarantees the normal operation of the network. This pool size was calculated as 80% of the number of VMs needed in a busy hour. Thus the pool size is 225 VMs. Furthermore, from Figure 5 it is noticeable that some of the SFC pool shares are more adequate for a given traffic mix, e.g. 75%-25% and 50%-50%. Therefore, in further analysis those pool shares have been selected.

Figure 6. shows obtained results for the considered pool shares and the compound approach compared to classical SFC design using the optimal pool share. Hereby the "total mismatch" in terms of the number of VMs that are additionally required on top of available VM pool for handling the traffic mix are shown.

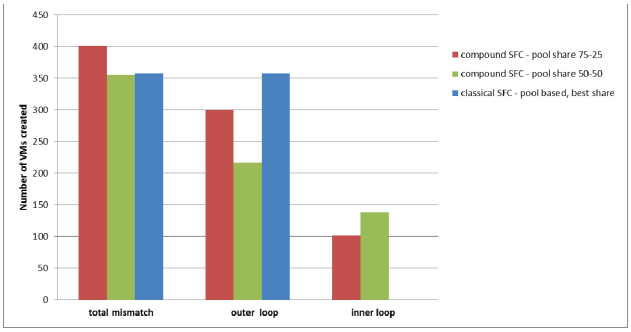


Fig. 6. The number of VMs that are needed on top of the available pools. The mismatch in number of needed VMs that is handled in outer loop and inner loop.

In other words, this is the number of VMs on top of initial VM pool that need to be created additionally or need to be made available through other means. This can be handled either by the Orchestrator by instantiating completely new VMs in outer loop or by reprogramming of already existing VMs in inner loop. Reprogramming the existing VMs achieves resource savings by reusing existing VM machines, and faster reaction to changed traffic demands. Reprogramming of VMs through local management entities takes less time than creating the new VMs through an Orchestrator. As illustrated in Figure 6, the classical SFC approach does not have the inner loop possibility and all mismatches in the number of VMs are handled over the more "expensive" outer loop. Triggering of inner loop by the compound approach significantly reduces the need of new VM instantiations, thus saves resources and time.

V. CONCLUSIONS

In this work, different implementation options for SF Chaining have been evaluated, where all SFs are implemented in separate VMs and the compound approach where SFs are implemented in a single VM. Further management options for achieving better resource reuse and faster response time were analyzed, i.e. the inner loop concept. The results show a comparable resource requirements for compound and classical SFC design. However, with the use of the inner loop, further resource savings and faster response time to changing traffic demands can be achieved.
As a next step, a proof of concept implementation is planned. Future work might allow more detailed comparison of the approaches both w.r.t. latency in the traffic path and dynamics in virtual network function management and configuration.

VI. ACKNOWLEDGMENT

This work has been performed in the framework of the H2020-ICT-2014-2 project 5G NORMA. This information reflects the consortium's view, but the consortium is not liable for any use that may be made of any of the information contained therein.

VII. REFERENCES

[1] EU H2020-ICT-2014-2, 5G NORMA: https://5gnorma.5g-ppp.eu, 2015

[2] M. Chiosi, et al. "Network Functions Virtualization", SDN and OpenFlow World Congress, Darmstadt, Germany, October 22-24, 2012.

[3] "Software Defined Networking: The New Form for Networks", White paper, ONF Open Networking Foundation, April 2013.

[4] K. Tsiamoura, „A Survey of Trends in Fast Packet Processing", Proceedings of the Seminars "Future Internet" (FI) and "Innovative Internet Technologies and Mobile Communications" (IITM), TU Munich 2014

[5] I. Ivano Cerrato, Guido Marchetto, Fulvio Risso, Riccardo Sisto, Matteo Virgilio, "An Efficient Data Exchange Algorithm for Chained Network Functions" IEEE 15th International Conference on High Performance Switching and Routing (HPSR), 2014

[6] Y. Luo, E. Murray, T. L. Ficarra, "Accelerated Virtual Switching with Programmable NICs for Scalable Data Center Networking " Proceedings of the second ACM SIGCOMM workshop on Virtualized infrastructure systems and architectures, 2010

[7] "Network I/O Latency on VMware vSphere 5 – Performance Study" VMWare Technical White Paper, 2012.

[8] "PCI-SIG SR-IOV Primer: An Introduction to SR-IOV Technology", Intel White Paper, January 2011.

[9] J. DiGiglio and D. Ricci, "High Performance, Open StandardVirtualization with NFV and SDN", joint Technical White Paper by Intel Corporation and Wind River, http://www.windriver.com/whitepapers/ovp/ovp_whitepaper.pdf, 2015