Copyright Notice

Institute of Communication Networks and Computer Engineering
University of Stuttgart
Pfaffenwaldring 47, D-70569 Stuttgart, Germany
Phone: ++49-711-685-68026, Fax: ++49-711-685-67983
Email: mail@ikr.uni-stuttgart.de, http://www.ikr.uni-stuttgart.de

Data-Center Architecture Impacts on Virtualized Network Functions Service Chain Embedding with High Availability Requirements

Sandra Herker*, Xueli An†, Wolfgang Kiess‡, Sergio Beker‡, Andreas Kirstaedter*
*Institute of Communication Networks and Computer Engineering (IKR), University of Stuttgart, Germany
Email: Sandra.Herker@gmx.net, andreas.kirstaedter@ikr.uni-stuttgart.de
†Email: anxueli@gmail.com
‡DOCOMO Communications Laboratories Europe GmbH, Munich, Germany
Email: {kiess, beker}@docomolab-euro.com

***Abstract*—Network Functions Virtualization (NFV) is a recent networking trend gaining a lot of attention from telecom operators and vendors. It promises to virtualize entire classes of network node functions within a data-center and to deliver network services in the form of Virtualized Network Function (VNF) service chains using commercial off-the-shelf hardware and IT virtualization technologies. However, availability gets an important issue when purpose-built telecom hardware designed for the "fives nines" standard via built-in failure protection and recovery mechanisms is replaced by the off-the shelf hardware. With commercial off-the-shelf data-center hardware, failure probabilities could be higher than in traditional physical network infrastructure. Thus with NFV, infrastructure availability has to be considered all the way from the physical right up to the hypervisor layer and resilience mechanisms need to be built into the software and service provisioning design. In this work, we model different backup strategies for VNF service chains and provide algorithms for their resilient embedding in the data-center. Further, we answer the question which data-center topologies offer the best cost-per-throughput relation for a given resilience/availability for VNF service chains.**

I. Introduction

In current telecommunication networks, network functions (NFs) are implemented as a combination of vendor specific hardware and software. In contrast to that, ETSI proposed a concept called Network Functions Virtualization (NFV) [1] to improve cost efficiency compared to dedicated hardware and software implementation. NFV focuses on the use of commercial off-the-shelf (COTS) hardware, i.e., general-purpose servers, to provide NFs via software virtualization techniques. With NFV, the NFs get implemented as software building blocks in the form of Virtual Machines (VMs) on industry standard high volume servers, switches and storage using IT virtualization technologies. These blocks are then connected together in the form of Virtualized Network Function (VNF) service chains to support the desired network services. A VNF service chain defines an ordered or partially ordered set of abstract functions and ordering constraints that must be applied to packets and/or frames[1]. The composition of the VNFs is different from a typical application component composition [2]. One challenge is to formalize a request for chaining several VNFs together, while considering the possible dependencies among them and to fit the requirements of the tenant applications. For placing the functions in the operator's network, the requirements of individual requests as well as the overall requirements need to be met.

Using the NFV concept, the perception of availability will shift from a per-network-element viewpoint to the consideration of end-to-end service availability. One important availability requirement in NFV is the service continuity, i.e., the end-to-end availability of telecommunication services. The VNF needs to ensure the availability of its part of the end-to-end service, just as in the case of a non-virtualized NF. VNF failures shall never impact other applications, hardware failures shall only affect those VMs assigned to that specific hardware, connectivity failures shall only affect connected NFs, etc. As well as designing availability into a VNF we can also design availability into service chains. Considered availability levels are in the range of the classical "five nines" (i.e. high availability of services, when the downtime is less than 5.26 minutes per year). However, for the purpose of the Internet of Things, telecommunication networks may well have to support higher service availability values - as required e.g. by machine control and other safety-critical applications.

Our purpose is to create VNF service chains with a requested availability. As the VNF service chains are deployed in a data-center (DC), we develop an embedding algorithm for resilient deployment of VNF service chains in the DC. Another important question is which DC topologies are principally best suited for the resilient VNF service chains. Therefore, we investigate in this work which DC topology offers the best cost-per-throughput performance for given VNF service chain availability levels.

*The work in this paper was done while at DOCOMO Communication Laboratories Europe GmbH.

†Now with Huawei Technologies Duesseldorf GmbH, European Research Center. Work in this paper was done while at DOCOMO Communication Laboratories Europe GmbH.

[1]https://datatracker.ietf.org/doc/draft-ietf-sfc-problem-statement/, September 2015

978-1-4673-9526-7/15/$31.00 ©2015 IEEE

TABLE I
MTBF VALUES FOR DIFFERENT DC COMPONENTS (NUMBERS TAKEN FROM [4], [3], CISCO SWITCHES AND INTEL SERVERS)

DC component	MTBF (hours)	MTTR (hours)
Server	$0.6667 \times 10^4 - 10.95 \times 10^4$	7-8
ToR switch	$14.5 \times 10^4 - 17.52 \times 10^4$	2.9
Aggregation switch	$8.76 \times 10^4 - 20 \times 10^4$	2.1
Core switch	60×10^4	2.1

In the following sections, we first explain related work in Section II and then introduce the involved system components in Section III. The resilient VNF embedding algorithm for achieving high availability is explained in Section IV. Section V analyzes the performance of different DC architectures for resilient VNF service chain placement. Finally, we conclude our work in Section VI.

II. BACKGROUND AND RELATED WORK

A. Availability of DC components

VNF service chain failures can be caused by server failures and/or DC network failures. A server may fail due to errors of the involved hardware (processor, memory, storage discs, power supply, network interfaces, etc.). Further, there can be software failures caused by the hypervisor or the VM instances. Network failures can be caused by the switches, the cabling, the load balancers (LBs) or other components. In this work we concentrate on hardware failures in the DC servers and switches.

Different studies (e.g. [3], [4], [5] and [6]) provide statistical data on element failures within DCs. The study [3] about network-related failures in DCs found out that usually node (e.g. switches and servers) failures are due to maintenance. Top-of-rack (ToR) switches are reported to be most reliable, however, as a lower priority component they show high downtimes. LBs are least reliable and experience many short lived faults. The Mean Time To Repair (MTTR) and Mean Time Between Failures (MTBF) values of a three years' collection of DC failure event logs [4] are shown in Table I.

The study in [5] details server failure characteristics and shows that they mainly (about 78%) can be attributed to hard disk events. In [6] the major failure reasons of servers and VM failures are examined. Reasons for failure are wear-and-tear of server, over-aggressive consolidation/repeated on-off cycles and temperature rise.

For our study, we use the MTBF and MTTR values from Table I and determine from them the availability values for each component. For the servers we use the lower MTBF values as we consider low cost servers in the study.

B. Resilient VM placement in DCs

VM chain placement plays a crucial role in the layout of VNF service chains (which are built of VMs) in a DC, and this has been investigated in many works. Here we concentrate on resilient VM placement in DCs.

The authors of [7] and [8] focus on availability-aware Virtual Data-Centers (VDC) embedding. The technique to compute the availability of a VDC in [7] considers both the heterogeneity of DC networking and computing equipment in terms of failure rates and availability, and the number of redundant virtual nodes and links provisioned as backups. The authors of [9] designed an availability-aware scaling approach to improve overall system availability while maintaining the communication costs. They used algorithms to resize the VMs to meet the requirement about availability. Machida et al. [10] and Xu et al. [11] both aim to minimize the backup resources, i.e. number of redundant VMs on a minimal set of active servers in a DC while guaranteeing a certain protection level. The work in [12] considers VNF in cloud and presents a solution for the resilient deployment of VNFs, using OpenStack for the design and implementation of the proposed service orchestrator mechanism.

In difference to the work mentioned above, this paper focuses on placement of NFV type applications with high availability constraints and the suitability of different DC architectures for a resilient VNF service chain embedding. Further we evaluate influences of different parameters on different DC architectures with the use of the algorithm.

III. SYSTEM MODEL

A. Data-center topology

Our goal is to find a data-center (DC) topology that minimizes the cost to deploy NFV type services and is efficient for the resilient embedding.

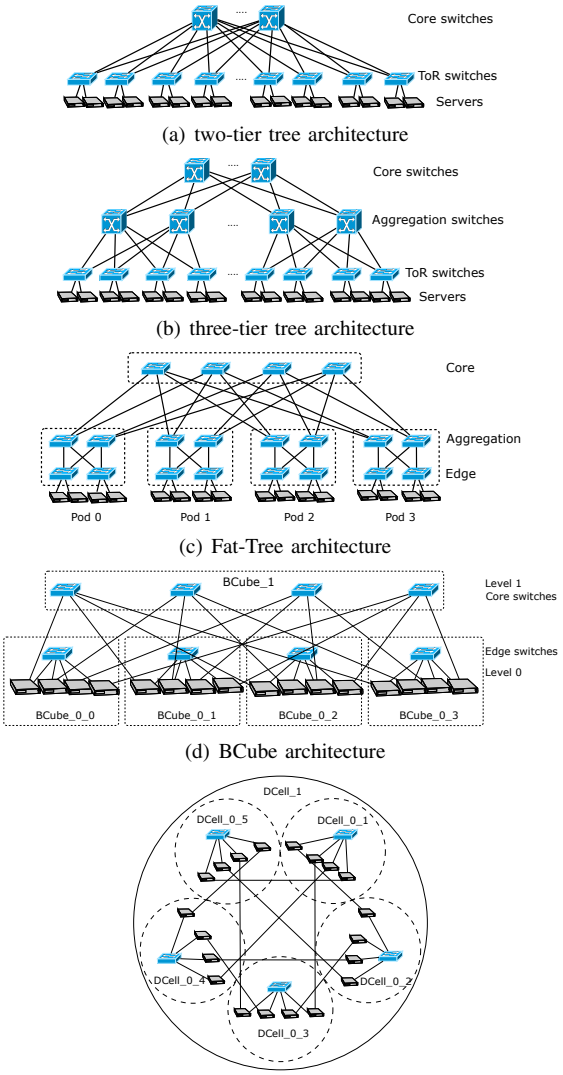
The entire DC can be modeled as a graph which consists of vertices and edges. The vertices represent switching nodes, i.e. core switch, aggregation switch and top-of-rack switches (ToR), and servers. The edges represent physical links between servers and switching nodes and between switching nodes. Each server can be used to host one or multiple VMs. We assume that all servers have the same configuration in terms of CPUs, RAM and storage.

The following DC topologies are considered in our analysis.

- Switch centric
  - Two-tier tree architecture as shown in Figure 1(a).
  - Three-tier tree architecture as shown in Figure 1(b).
  - Fat-Tree topology as shown in Figure 1(c) is a three-tier architecture that uses Clos topology [13].
- Server centric
  - BCube as shown in Figure 1(d) is a recursively defined structure and uses servers and switches for packet forwarding [14].
  - DCell as shown in Figure 1(e) is also defined recursively and uses servers and switches for packet forwarding [15].

B. NFV Service Chain modeling

VNF service chains provide typical network functions like DPI, firewall, encryption and tunneling to the customers of the operator offering the network service. Such service is



(a) two-tier tree architecture

(b) three-tier tree architecture

(c) Fat-Tree architecture

(d) BCube architecture

(e) DCell architecture

Fig. 1. Data-center architectures

expected to process a large number of "parallel flows". Here, in this paper we define a "parallel flow" as all the packets exchanged between two end systems that are located outside the DC. I.e., this definition of a traffic flow differs from the classical Internet 5-tupel "flow" definition. We assume that the individual traffic flows do not have inter-dependencies between each other as they e.g. result from different customers. Further, we neglect in the following the traffic needed for configuration/management/control of the VMs and only consider the data traffic passing along the chain of VMs. If a VNF application needs to be run in a chain of VNFs, these VNFs are deployed independently on VMs, which could be located on the same server or different servers. The packets of any traffic flow then need to traverse the VNFs of a VNF service chain in a specific order thus determining the internal sequence that a traffic flow passes through the topology of the DC. The VNFs themselves can be provided by the same VNF vendor or different vendors.

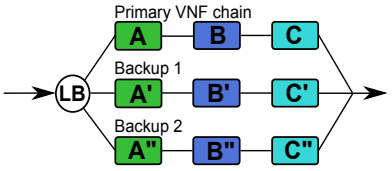


Fig. 2. Backup deployment strategy 1

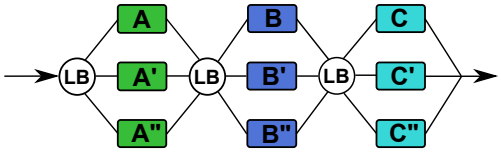


Fig. 3. Backup deployment strategy 2

C. Different backup deployment strategies for reliable VNF chains

The question is how to deploy VNF chains with predefined levels of availability in the DC network. As already a single failure in one part of the VNF service chain breaks the whole chain, we have developed two different backup deployment strategies: Strategy 1 is a simple backup of the complete VNF service chain via a load balancer (LB) connecting all chains at their beginning (c.f. Figure 2 with a primary chain of three VNF functions A-B-C and two backup chains A'-B'-C' and A"-B"-C"). In strategy 2 (c.f. Figure 3 resource pooling) the individual nodes (VNF functions) of the chains are connected to LBs that can redistribute the traffic to one of the corresponding VNFs of the backup chains if one VNF of the primary chain is broken.

The availabilities A of the different strategies are calculated as follows if we assume that the failures of the different components are independent (i.e. the different components are deployed on different physical devices):

Strategy 1

$$A = p_{LB}(1 - (1 - p_{VNF}^{n})^{b+1}) \tag{1}$$

Strategy 2 resource pooling

$$A = (p_{LB}(1 - (1 - p_{VNF})^{b+1}))^{n} \tag{2}$$

p_{LB} is the availability of the LB and p_{VNF} is the availability of a VNF component. n is the number of VNFs in a VNF service chain. b is the number of backups per VNF component.

IV. High Availability Service Chain Embedding

A. Heuristic Algorithm for High Availability

To achieve the requested service availability and save resources, the smallest possible number of backup elements has to be added to the primary VNF chain. The idea is to first embed the primary VNF chain and recursively add one backup chain while calculating the service availability. The algorithm stops if the requested availability is met or the maximum

number of backup chains (e.g. 10 chains) is reached to avoid excessive numbers of intermediate switches in the backup chains.

Step 1:
For each virtual node in the primary VNF chain calculate and select a server node candidate that fulfills the virtual node requirements (capacity, ...).

Step 2:
For each virtual link in the primary VNF chain embed it using the Constrained-based Shortest Path (CSPF) algorithm on bandwidth that satisfies the bandwidth requirements.

Step 3:
Construct the backup graph using one of the different backup deployment strategies.

Step 4:
Embed the backup chain while considering the constraints on CPU and bandwidth.

Step 5:
Calculate the service availability of the primary plus backup chain(s).

Step 6:
Check if the availability requirement is fulfilled
- Yes, stop the procedure and report its success.
- No, calculate another backup chain (repeat Step 4 and 5).

Step 7:
If after a certain number of backups the requested service availability is not fulfilled, stop the procedure and report its failing.

B. VNF Service Chain Placement Strategies

The VNF Service Chain Placement (VSCP) strategy determines how the VNF service chain is mapped to the VM level in the DC. It plays an important role in terms of consumed computing/storage resource and internal bandwidth of a DC. The goal is to map as many VNF service chains as possible in one DC to maximize operator's revenue. As the strategies how and where the operator will place the VNF functions in the DC are still unknown today, we developed the following two different VSCP strategies:

1) Local VSCP: The idea of the local placement is to keep all the VMs that run VNF application sub-functions as close as possible to minimize the DC internal consumed bandwidth and number of hops for interconnecting the VMs. In the case of reliability this means that all VMs belonging to the primary VNF service chain are embedded as close as possibly (e.g. same server). However, the primary VMs and their corresponding backups are not embedded using local VSCP and are not allowed to be on the server.

2) VNF Vendor Based VSCP: A VNF service chain may contain VNFs provided by different vendors, for instance, DPI from company A and tunneling from company B. To ensure maximum isolation of VNFs from different vendors (e.g. for security reasons) and avoid the potential influence from the hypervisor and also security concerns, the servers can be pooled or clustered [16]. To cover this use case, we introduce a vendor based VSCP strategy: An individual server must only contain VMs from a single vendor, while these VMs may still be mixed in the same rack. Again, the VMs on the same chain should be placed as close as possible to the others in the VNF chain.

C. Detailed Description of the Embedding Algorithm

For the identification and embedding of the backup chains, we first make a few assumptions for simplicity: All servers have the same availability. The links between the switches (ToR, agg, core) have 100% availability as we only consider server and switch failures in this work. LB can be embedded on switches. Therefore the LB have the availability of the component (e.g. switch) it is embedded. Generally, the components (VMs) of any individual backup chain must be placed on different servers than their counterparts in the primary and other backup chains.

As the starting point, we calculate for each DC topology a so called "availability matrix" indicating the shared risk between any two servers. Each matrix element is calculated considering the probability of a failure happening in any of their common parent switches of the two servers (while entering the DC from the core switch) and the probability of a failure happening at the same time in their own private path below the common parent switches [9]. The availability value is calculated from the shortest distance from one server to the other. If the shortest distance is known the number of intermediate switches (ToR, aggregation, core) can be determined. The availability A between two VMs hosted on the servers u and v is calculated as in Equation 3 where $C(u,v)$ are the set of the common parent switches of u and v. The sets $N(u)$ and $N(v)$ contain the parent nodes belonging to its own path (i.e. switches facing out of the DC excluding the common parent switches) of server u and v respectively and the server itself. A_n is the availability of a node n in the DC (which could be a switch or a server).

$$A = 1 - \big((1 - \prod_{n \in C(u,v)} A_n) + \prod_{n \in C(u,v)} A_n \times (1 - \prod_{x \in N(u), x \notin C(u,v)} A_x) \times (1 - \prod_{y \in N(v), y \notin C(u,v)} A_y)\big) \quad (3)$$

For each simple VNF service chain the (primary) nodes of the VNF chain are embedded according to the selected VSCP strategy (local or vendor based). After successfully embedding the nodes, the links in between are embedded using the Dijkstra shortest path algorithm. After the simple VNF service chain is successfully embedded, its availability is enhanced by adding the backup nodes according to one of the backup deployment strategies. Then the backup nodes of the VNF chain are embedded according to the selected VSCP strategy of the VNF chain. For each node in the backup chain, suitable candidate backup servers according to the VSCP strategy and the capacity constraints (i.e. CPU

capacities) are identified. For each of these candidates the risk shared availability with the primary server is checked using the availability matrix explained above. The candidate node with the lowest shared risk (highest entry in the availability matrix) and also the shortest distance to the primary node is selected and embedded. This continues with all backup nodes in the backup chain.

If all backup nodes are successfully embedded the backup links need to be mapped. For the local VSCP strategy, the algorithm tries to embed using a shortest path that is maximum switch-node disjoint to the primary links (i.e., this mean that any joint switch node contained in the primary chain should be avoided to insure less shared risked nodes and links to and increase the availability). For all other backup chains the algorithm tries to avoid as much as possible joint intermediate switches between backup groups. For the vendor based VSCP strategy and backup deployment strategy 2, the links are mapped using shortest path and links disjoint paths between primary backup and backup-backup links.

After embedding the first backup chain the service availability is calculated and if - necessary and still possible - additional backup chain(s) are determined and embedded (as described above).

This algorithm is then used in the evaluations below to compare the performance of the different DC topologies in terms of the cost per throughput relation at the required availability level of the service chain.

V. Performance Analysis of different DC topologies for the resilient deployment strategies

Our simulation framework for analyzing the cost and availability performance of the different DC topologies for the deployment of resilient VNF chains is custom-built and was written in Java.

A. Simulation setup

The following DC parameters and VNF chain parameters were used in the simulations:

1) Data center parameters: The DC size is determined by the amount of servers, which is within the range $[400, 4000]$. Each server has 10 cores and can host up to 10 VMs. Each VM can occupy one or multiple cores within a server. The availability values for the server and switches are shown in Table II. The bandwidth allocation within a DC is shown in Table III. For the 2-/3-tier architectures (with 24 servers per rack), we use four core switches. The other topologies (Fat-Tree, BCube and DCell) are built with low cost switches with a low number if switch ports. To achieve 20 Gbps bandwidth, two 10 GbE links are used together, for instance by applying Ethernet link bundling.

2) NFV parameters: We assume that there are four VNFs per VNF service chain. All the VMs are connected one after the other to form a service chain. The incoming packets enter the VNF chain in the first VM and traverse all the other VMs and leave the chain at the last VM. If there are more than one core switch in the DC, the incoming traffic will be routed into and also out of the DC using the same core switch. We assume that the required packet processing capability for each VNF is a random number between $0.65 - 2$ Gbps/core[2]. Therefore, if one VNF requires 1 Gbps/core processing capability, it needs a VM with 5 cores in order to process 5 Gbps incoming traffic. The requested VNF service chain availabilities are chosen from 0.999, 0.9999, 0.99999, 0.999999.

TABLE II
Data-center component availability parameters

DC component	availability
Server	0.999
ToR switch	0.9999
Aggregation switch	0.9999
Core switch	0.99999

TABLE III
Data-center bandwidth parameters

DC bandwidth	server-ToR	aggregation	core
2-tier	10 Gbps	-	100 Gbps
3-tier	10 Gbps	100 Gbps	100 Gbps
Fat-Tree	10 Gbps	20 Gbps	20 Gbps
BCube	20 Gbps	-	20 Gbps
DCell	20 Gbps	-	20 Gbps

B. Influence of different parameters

For each parameter the simulation is run 100 times and the average number of successful embedded VNF chains is calculated.

1) Different DC sizes: We vary the DC size between 400 and about 4000 servers and use the local VSCP strategy with the backup strategy 1. We see the performance of the different DC topologies for a requested service availability of "fives nines" in Figure 4: The 2-tier topology performs best. The second best in this case is the 3-tier topology. One reason for this performance is the fact that these two topologies use modular core switches with high availabilities. The other topologies were mostly built with low cost switches with lower availabilities. Further, we see that the Fat-Tree topology has a higher embedding rate compared to BCube/DCell topologies. This can be attributed to the fact that Fat-Tree contains more switches in its DC topology whereas BCube and DCell are server-centric DC topologies. They partly use servers to work as switching nodes which results in lower successful VNF chain embeddings due to the lower availability of servers compared to switches. Figure 5 shows the corresponding cost in relation to the average embedded VNF service chains. For comparing the cost we use our previous work [19] where we have modeled the server and switch cost. The 2-tier architecture has lowest cost in all cases.

2) Different requested service availability: Next we examined the DC topologies for requested service availability levels

[2]One CPU core can forward 10 Gbps traffic in general [17]. For a typical middlebox application (e.g., a Firewall) the throughput per CPU core is 2.8 Gbps for a packet size of 64 byte and 10 Gbps for a packet size of 1024 byte [17]. Other functions like carrier grade NAT, Software BRAS and Intrusion Detection System have lower throughput, only about 1 to 1.7 Gbps for a packet size of 64 byte. Packet forwarding via the servers like in BCube and DCell is assumed with a rate of 10 Gbps per core [18].

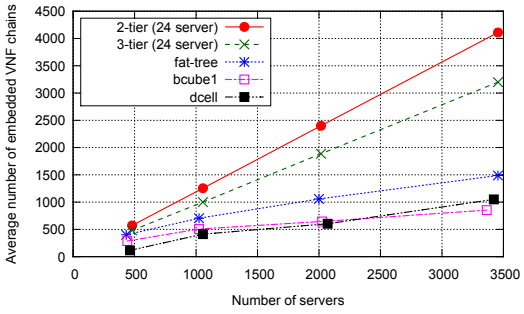


Fig. 4. Successful embedded VNF chains for 0.99999 and local VSCP

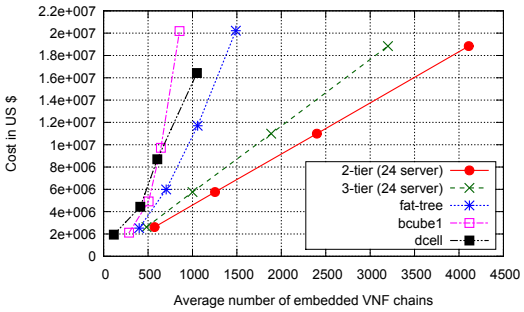


Fig. 5. Cost to successful embedded VNF chains for 0.99999 and local VSCP (lower is better)

from "three nines" to "six nines". The number of successfully embedded VNF chains decreases with rising requested service availability: Especially Fat-Tree shows more successful embeddings for "three nines" requested service availability (Figure 7) than for "six nines" (Figure 6). For "six nines" each VNF service chain needs on average two backup chains to achieve that high service availability. When the service availability decreases, one single backup chain is sufficient. While the absolute number of chains that can be embedded decreases for all topologies (which is to be expected), their relative ranking does not change.

In Figure 8 different requested availabilities and the successful embedded service chains are shown for the 2-tier topology.

3) Different VSCP: We further compared the local and vendor-based VNF service chain placement algorithms for the backup deployment strategy 1. For the VNF vendor based VSCP strategy we assure each function of the service chain (being from a different vendor) has to be placed on a different server. In this case, the routing paths for primary and backup chains within a DC tend to be longer than for the local VSCP at all DC topologies. Further, often an additional backup chain is required for achieving the same availability if the vendor based VSCP is used. The result is higher bandwidth and VM consumption and therefore less VNF chains can be embedded. For example, with requested availability of "five nines" the 2-tier topology with the local VSCP can embed about double the number of VNF chains as in the case of using the vendor based VSCP that needs an additional backup chain. The same effect can be experienced with higher availability values. If the requested service availability decreases, the influence of the VSCP strategies is decreased in the embedding and the

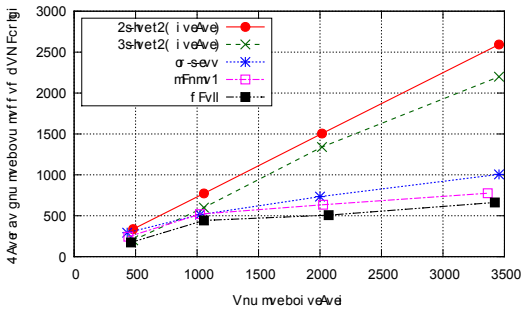


Fig. 6. Successful embedded VNF chains for 0.999999 and local VSCP

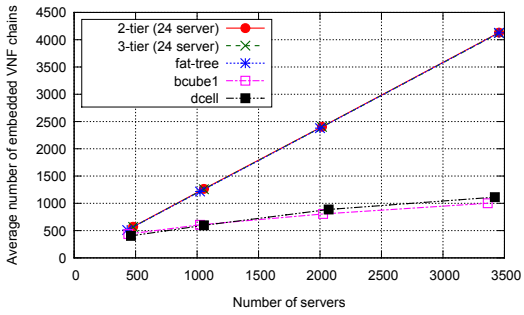


Fig. 7. Successful embedded VNF chains for 0.999 and local VSCP

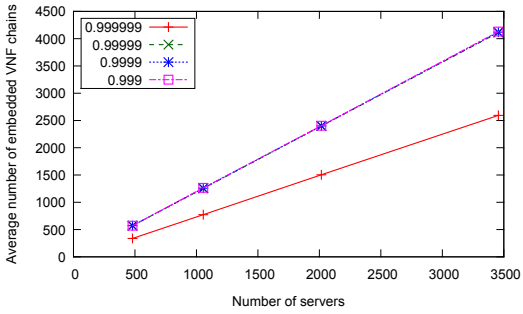


Fig. 8. Impact of the different requested service availabilities for 2-tier

resulted embedded VNF chains. An example result for the different DC topologies with a server size of about 3500 is shown in Figure 9 for different VSCPs and availabilities.

4) Different backup deployment strategies: We also compare the different backup deployment strategies. For the simulation we examine the deployment strategy 2 with the vendor based VSCP. If the strategy 2 is combined with the

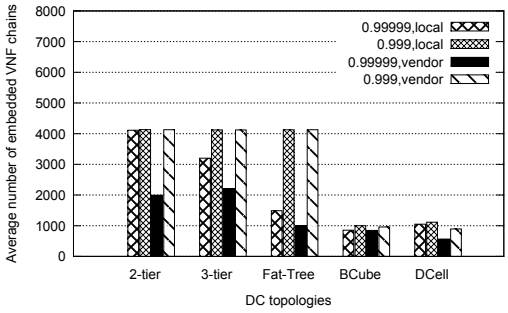


Fig. 9. Impact of VSCP strategies for requested service availabilities 0.99999 and 0.999

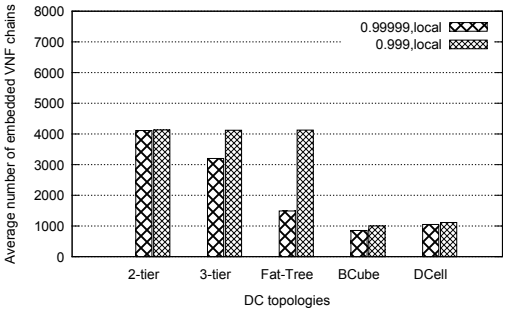


Fig. 10. Impact of the backup deployment strategy for availability 0.999

local VSCP it would be equal to strategy 1 with the local VSCP. In Figure 10 the strategy 1 with vendor based VSCP is compared to strategy 2 for the different DC architectures with about 3500 servers. From the simulation, strategy 2 (resource pooling) can embed only successfully the VNF with the requested availability for "three nines" and "four nines". This can be attributed to the common LBs in the backup graph which become critical and prohibit high availability values. For a requested availability of 0.9999, Fat-Tree, BCube, DCell cannot successfully embed the chain. This is due to the facts that the LB is embedded in one of the switches and that the switches in Fat-Tree, BCube and DCell have lower availability than those switches used for the 2-/3-tier topologies (which can embed the LB in one of the highly available core switches).

C. Discussion: Recommendations for network operators derived from the results

The switch centric topologies are better suited for achieving high availability values. The 2-tier architecture shows best performance followed by the 3-tier topology. The fully meshed of the 2-tier architectures has advantages in embedding against the 3-tier architecture because there are more paths to route backup chains. Generally, high-cost and reliable switches are needed in the DC topology to achieve high availability service chains. The server centric topologies have the disadvantage that the servers are less reliable than switches and therefore the performance of these topologies for reliable VNF is lower compared to the switch centric ones.

Further, inefficiencies arise when employing a vendor-based VNF service chain placement (VSCP): Due to the fact that an individual server must only contain VMs from a single vendor, the primary and backup chains generally get longer compared to a local VSCP and additional backup chains are required such that in the end less chains can be embedded.

VI. Conclusion

In this work we developed a VNF service chain embedding algorithm that considers service availability constraints and examined using our algorithm the ability of different DC architectures to deploy resilient NFV type applications. Further we compared the cost of the different architectures and their performance for embedding VNF service chains with requested service availability. Different backup deployment strategies and VSCP strategies are compared for each DC topology. Further, we gave some recommendations for future NFV type applications deployment for resiliency in DCs. From the results the "best" DC topology for achieving high availability for VNF service chain is a 2-tier tree topology.

References

[1] M. Chiosi, D. Clarke, P. Willis, A. Reid, J. Feger, M. Bugenhagen, W. Khan, M. Fargano, C. Cui, H. Denf *et al.*, "Network functions virtualisation: An introduction, benefits, enablers, challenges and call for action," White paper ETSI technical report, Tech. Rep., 2012.

[2] S. Mehraghdam, M. Keller, and H. Karl, "Specifying and placing chains of virtual network functions," in *IEEE 3rd International Conference on Cloud Networking (CloudNet '14)*. IEEE, 2014.

[3] P. Gill, N. Jain, and N. Nagappan, "Understanding network failures in data centers: measurement, analysis, and implications," in *Proceedings of the ACM SIGCOMM*. ACM, 2011, pp. 350–361.

[4] R. Potharaju and N. Jain, "When the Network Crumbles: An Empirical Study of Cloud Network Failures and Their Impact on Services," in *Proceedings of the 4th ACM Symposium on Cloud Computing (SOCC '13)*, 2013.

[5] K. V. Vishwanath and N. Nagappan, "Characterizing Cloud Computing Hardware Reliability," in *Proceedings of the 1st ACM Symposium on Cloud Computing (SOCC '10)*, 2010.

[6] W. Deng, H. Jin, X. Liao, F. Liu, L. Chen, and H. Liu, "Lifetime or Energy: Consolidating Servers with Reliability Control in Virtualized Cloud Datacenters," in *IEEE 4th International Conference on Cloud Computing Technology and Science (CLOUDCOM '12)*, 2012.

[7] M. G. RABBANI, M. F. ZHANI, and R. BOUTABA, "On Achieving High Survivability in Virtualized Data Centers," *IEICE TRANSACTIONS on Communications*, vol. 97, no. 1, pp. 10–18, 2014.

[8] Q. Zhang, M. Zhani, M. Jabri, and R. Boutaba, "Venice: Reliable Virtual Data Center Embedding in Clouds," in *Proceedings IEEE INFOCOM*, April 2014, pp. 289–297.

[9] W. Wang, H. Chen, and X. Chen, "An availability-aware virtual machine placement approach for dynamic scaling of cloud applications," in *9th International Conference on Ubiquitous Intelligence & Computing and 9th International Conference on Autonomic & Trusted Computing (UIC/ATC)*. IEEE, 2012, pp. 509–516.

[10] F. Machida, M. Kawato, and Y. Maeno, "Redundant virtual machine placement for fault-tolerant consolidated server clusters," in *IEEE Network Operations and Management Symposium (NOMS '10)*. IEEE, 2010.

[11] J. Xu, J. Tang, K. Kwiat, W. Zhang, and G. Xue, "Survivable virtual infrastructure mapping in virtualized data centers," in *IEEE 5th International Conference on Cloud Computing (CLOUD '12)*, 2012.

[12] M. Scholler, M. Stiemerling, A. Ripke, and R. Bless, "Resilient deployment of virtual network functions," in *5th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT '13)*. IEEE, 2013.

[13] M. Al-Fares, A. Loukissas, and A. Vahdat, "A scalable, commodity data center network architecture," in *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 4. ACM, 2008, pp. 63–74.

[14] C. Guo, G. Lu, D. Li, H. Wu, X. Zhang, Y. Shi, C. Tian, Y. Zhang, and S. Lu, "BCube: a high performance, server-centric network architecture for modular data centers," *ACM SIGCOMM Computer Communication Review*, vol. 39, no. 4, pp. 63–74, 2009.

[15] C. Guo, H. Wu, K. Tan, L. Shi, Y. Zhang, and S. Lu, "DCell: a scalable and fault-tolerant network structure for data centers," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 4, pp. 75–86, 2008.

[16] A. Gulati, A. Holler, M. Ji, G. Shanmuganathan, C. Waldspurger, and X. Zhu, "VMware Distributed Resource Management: Design, Implementation, and Lessons Learned," in *VMware Technical Journal*, Spring 2012.

[17] "D5.3: Application Development and Deployment," FP7 CHANGE Project, Tech. Rep., 2013.

[18] L. Popa, N. Egi, S. Ratnasamy, and I. Stoica, "Building Extensible Networks with Rule-based Forwarding," in *Proceedings of the 9th USENIX Conference on Operating Systems Design and Implementation*, ser. OSDI'10, 2010.

[19] S. Herker, X. An, W. Kiess, and A. Kirstaedter, "Evaluation of Data-Center Architectures for Virtualized Network Functions," in *IEEE ICC 2015 - Third International Workshop on Cloud Computing Systems, Networks, and Applications (CCSNA) (ICC '15)*, 2015.