Placement Strategies for Virtualized Network Functions in a NFaaS Cloud

Xin He* Tian Guo‡ Erich Nahum† Prashant Shenoy*
*University of Massachusetts Amherst
{xhe, shenoy}@cs.umass.edu
†IBM Thomas J. Watson Research Center
{nahum}@us.ibm.com
‡Worcester Polytechnic Institute
{tian}@cs.wpi.edu

***Abstract*—Enterprises that host services in the cloud need to protect their cloud resources using network services such as firewalls and deep packet inspection systems. While middleboxes have typically been used to implement such network functions in traditional enterprise networks, their use in cloud environments by cloud tenants is problematic due to the boundary between cloud providers and cloud tenants. Instead we argue that network function virtualization is a natural fit in cloud environments, where the cloud provider can implement Network Functions as a Service using virtualized network functions running on cloud servers, and enterprise cloud tenants can employ these services to implement security and performance optimizations for their cloud resources. In this paper, we focus on placement issues in the design of a NFaaS cloud and present two placement strategies—tenant-centric and service-centric—for deploying virtualized network services in multi-tenant settings. We discuss several trade-offs of these two strategies. We implement a prototype NFaaS testbed and conduct a series of experiments to quantify the benefits and drawbacks of our two strategies. Our results suggest that the tenant-centric placement provides lower latencies while service-centric approach is more flexible for reconfiguration and capacity scaling.**

I. Introduction

Traditionally enterprises have used middleboxes to implement various security and performance functions in their enterprise networks. These network functions include firewalls, deep packet inspection systems, and proxy caches among others.

As enterprise networks have become more dynamic in their needs, the use of specialized hardware middleboxes to implement network functions has become a drawback rather than a benefit. Network function virtualization (NFV) has emerged as a potential solution to enable enterprises to flexibly deploy and reconfigure network functions on-demand to handle the network's dynamic, scalability and security needs.

At the same time, enterprises have begun to move back-end applications from in-house data centers to the cloud. The cloud's pay-as-you-go model and on-demand resource allocation abilities are attractive to enterprises for hosting their application in a more cost-effective fashion while also handling workload dynamics. Indeed many new enterprises are entirely cloud based where their entire IT infrastructure—both internal and external facing applications—are cloud based.

In such scenarios, an enterprise needs to implement network security and performance functions in the cloud to guard their cloud-based servers—in order to implement the same network security and performance policies they would have implemented in their enterprise network. Since deploying or leasing middleboxes in a public cloud is not always possible, the use of NFV to implement these functions using commodity cloud servers is an attractive option.

In many cases, the cloud providers may themselves offer network functions as a service (NFaaS) to cloud-based enterprises so that they can lease storage and servers as well as appropriate network services to configure and guard their resources.

Motivated by such scenarios, in this paper, we study the design of a NFaaS cloud. Specifically we assume that a NFaaS cloud provides different network functions (e.g., firewall, intrusion detection system (IDS), caching, etc) that can be leased by a cloud-based enterprise for their cloud IT infrastructure.

We specifically examine how a cloud provider should design a multi-function multi-tenant NFaaS cloud from the placement perspective. We propose two different placement strategies for a multi-tenant NFaaS cloud and discuss the advantages and disadvantages of each approach. We conduct an experimental evaluation of these approaches using a small prototype NFaaS cloud and quantify their benefits and overheads. We believe that our insights can provide design guidelines on the placement of virtualized network functions in future NFaaS clouds.

II. Background and Related Work

In this section, we present background on cloud computing and network function virtualization.

A. Cloud Computing Background

We consider an enterprise that hosts its IT needs using cloud resources. The enterprise is assumed to acquire servers and storage resources from an Infrastructure-as-a-Service (IaaS) cloud to run its applications on this cloud-based infrastructure. Such a cloud-based IT infrastructure needs to incorporate network security policies and performance optimization—like an in-house enterprise network. That is, network traffic coming

978-1-5090-5409-1/16 $31.00 © 2016 IEEE
DOI 10.1109/HotWeb.2016.17

into the enterprise's cloud servers needs to undergo security checks (e.g., using firewalls, deep packet inspection systems, virus scanners, etc.) and may be subject to performance optimization (e.g., using in-network caches).

Traditionally such features have been implemented using middleboxes—dedicated hardware devices that are deployed in the network—to perform these functions.

In a cloud-based setting, we assume that the infrastructure cloud provider provides these functions as cloud services—allowing the enterprise to lease firewalls, caches, etc. similar to leasing servers and storage. We assume that the cloud provider supports a rich mix of network services that may be needed by an enterprise. The same benefit as infrastructure clouds hold in this case such as the pay as you go model, the ability to scale up service capacity, and on-demand resource allocation.

B. *Network Function Virtualization*

While the cloud provider can provide cloud-based network services by deploying middleboxes on behalf of cloud custom, it is more effective to use network function virtualization to implement these services using commodity servers.

In this case, a service such as a firewall, IDS or a cache is implemented as software that runs inside a virtual machine and the VM runs on commodity servers. Virtualizing network functions has become popular since it offers a number of benefits over the middlebox approach – such as reducing capital cost, shortening deployment cycles and the ability to handle the needs of a dynamic network.

In our scenario, NFV is a natural fit since the cloud provider is already leasing servers and can use these commodity servers to deploy virtualized functions and offer the network functions as a service (NFaaS) to customers.

A customer can lease various network functions as cloud services and chain them together to implement the desired network security policies and performance optimization. For instance, the customer (i.e., the enterprise) could lease a firewall service, an IDS service, a DNS service, a cache service and configure them so that network traffic flow through them transparently.

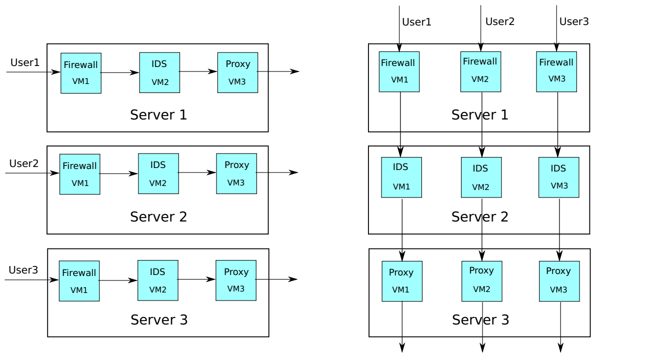
III. Placement Issues in a NFaaS Cloud

Many design issues arise in deploying a NFaaS cloud. In this paper, we specifically focus on placement issues in a multi-tenant NFaaS cloud, which we discuss next.

A. *Placement Strategies*

A cloud provider can employ one of two placement strategies in a multi-function multi-tenant NFaaS cloud: tenant-centric and service-centric.

In a tenant-centric approach, VMs comprising all network services leased by a tenant are mapped onto a single server or a group of co-located servers (e.g., servers on the same network rack). Fig 1(a) shows tenant-centric placement for three different tenants, each of whom is using three different network services. While the figure shows all services reside on



(a) Tenant-centric Placement. (b) Service-centric Placement.

Fig. 1. Two placement strategies of deploying network functions in a multi-tenant NFaaS cloud

a single server, higher capacity services may require multiple servers that are co-located.

In contrast, a service-centric placement approach maps VMs running the same service for different tenants on the same server or on a co-located group of servers. Fig 1(b) which depicts this approach shows that each server (or rack(s)) hosting the same service.

B. *Tradeoffs*

The two placement strategies offer a number of tradeoffs – both from a cloud provider's and a cloud tenant's perspective.

Network latency: The primary advantage of customer-centric service placement is that it optimizes the network latency of packets as they traverse from one network function to the next (e.g., from firewall to IDS). Since the services belonging to a tenant reside on the same server or the same rack, network latency is minimized. In the service-centric placement approach, services belonging to a tenant may reside on different racks, requiring packets to traverse to multiple switches when going from one service to the next.

Flexibility and Scaling: In many scenarios, existing network functions might need to be updated. For example, a tenant may want to implement new network functions in their network, or may choose to replace one IDS with another. In scenarios where a tenant's network traffic is increasing, the resource allocated to a NF may have to be scaled up ("elastic scaling"). In a tenant-centric approach, such reconfiguration require free resources on the server or rack hosting the tenant's current services. If such idle capacity is unavailable, either other tenant have to be moved to other servers/racks to free up resources or the new service has to be placed on a more "distant" server, diminishing the latency advantage of the approach. A service-centric approach does not suffer from such a drawback, since new services (or resizing of existing ones) can be achieved by choosing any server with sufficient capacity without regard to network proximity.

Packing and Resource Utilization: A tenant-centric approach enables hosting of heterogeneous services on the same server. This enable a CPU-intensive (but less I/O intensive) service to be co-located with an I/O-intensive (but less CPU-intensive) one, yielding better utilization of various resources of potentially a denser packing—a strategy that has been successfully employed in general-purpose VM placement.

A service-centric approach hosts homogeneous services (belonging to different tenants) on a server of packing density is determined by the most bottlenecked resources of each service. However, homogeneity of services is not without advantage. Since all service on a server run the same code, it provides opportunities for better memory and cache utilization (e.g., page cache sharing across services). It may also be possible to employ containers (lightweight VMs) rather than VMs to further exploit this homogeneity.

Performance Interference: Whenever multiple services reside on a physical server, there is the potential for performance interference. In the service-centric case, such interference will be cross-tenant, while in the tenant-centric case, the interference will be cross-service interference. Fortunately, resource partitioning features of VMs (or containers) can isolate co-resident services and minimize the impact of such interference.

In summary, whether to user tenant-based or service-based placement depends on the cloud provider's objective – tenant-based deployment has better latency, potentially better packing but less flexibility for reconfigurations, capacity scaling than a service-based approach and vice versa.

IV. Experimental Evaluation

In this section, we quantify the benefits and overheads of the two placement strategies using an experimental evaluation. Since the space is limited, here we only provide some of our experimental results. More evaluation can be found in our technical report [10]

A. Prototype NFaaS Cloud

We have built a prototype NFaaS cloud using several open-source components. Our prototype provides three network services: (i) a network firewall that is implemented using Linux *IPtables*, (ii) an intrusion detection system that is implemented using *Snort* 2.9.8.2 [1], and (iii) an in-network web cache that is implemented using *Squid* 3.3.8 [2]. All three components are popular and widely-used system that we deploy as virtualized services inside virtual machines. The use of virtualization enables benefits such as rapid deployment, flexible placement and flexible resizing when needed. Further, we implement both tenant-centric and service-centric placement in our NFaaS prototype. In both cases, we use Linux bridging to enable a tenant to "chain" network functions as per their needs (see Figure 2). We do so by appropriately configuring routing tables on each VM. In tenant-centric placement, services belonging to a cloud tenant are placed on the same server when possible (or on nearby servers when VMs are too large to pack onto a single server). In service-centric placement, VMs belonging to a service are packed together for co-location.

B. Experimental Setup

We deployed our NFaaS prototype on a testbed of five physical server as shown in Figure 2. Each server has an Intel Xeon X3430 2.4GHz Quad-Core CPU, two gigabit physical NICs, 8 GB of RAM and 1TB 7200 RPM disk. All machines run Ubuntu 12.04 and use KVM as a virtual machine (VM) hypervisor. We use one server to house clients and one to house web servers belonging to tenants. The other three servers run virtualized network functions inside VMs. Each VM is pinned to a physical CPU core with 2048 MB RAM allocated. We also create two virtual NICs that allow network traffics to traverse from one NIC to the other. Client HTTP traffic is generated using *httperf* [3] on one server, and this traffic traverses through a tenant's three network services before reaching the web server. We monitor and measure resource utilization of various VMs using the *dstat* [4] tool.

We configure the *iptables* firewall with 1000 rules and configure *Snort 2.9.8.2* with their default rule set. Finally, we configure *Squid 3.3.8* to cache frequently accessed web pages. In tenant-based deployment, each physical server contains three VMs belonging to a tenant running different network functions connected as a chain through Linux bridges as illustrated in Figure 1(a). In service-based deployment, each physical server contains three VMs running the same network function belonging to different tenants as illustrated in Figure 1(b).

C. Network Function Micro Benchmark

We first configure our network service based on Figure 2 in which each network runs inside its own VM. Client VMs are instructed to contact the web server, at a specified workload intensity, to get a 50 KB web page for 1 minute. All HTTP traffic traverses through these three NFs, and we measure the average resource utilization of NFs on their corresponding VMs. As we increase network traffic, from 100 to 500 connections per second, the CPU utilization of all network functions increase linearly as shown in Figure 3. Specifically, the web proxy is more sensitive to increasing network traffic than the firewall and IDS, with up to 60% more CPU consumption. The network utilization also grows directly with increasing network traffic. In addition, we find the disk and memory utilization vary only slightly with increasing network traffic. This is because the memory state of the firewall and IDs depend on factors such as the size of the rules set and not on the workload. Also, while the cache size depends directly on the number of frequently accessed pages, it is not strongly related to the request rate. We thus omit these measurement results but use them in our simulation (see Table I).

*R*esult: The CPU and network utilization of our services increase nearly linearly with the workload while memory and disk utilization are less sensitive to the request rate.

D. Virtualization Overhead

As shown in IV-C, NFs are heavily CPU and network consuming. In this experiment, we measure underlying hypervisor's CPU and network utilization for both placement approaches. We set up bare-bone VMs for tenant-based and service-based respectively as shown in Figure 1. The network

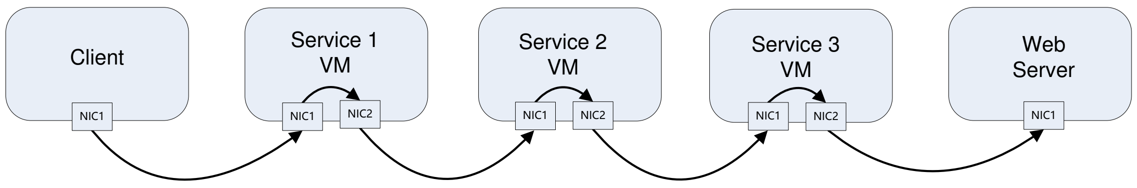


Fig. 2. Experimental Setup

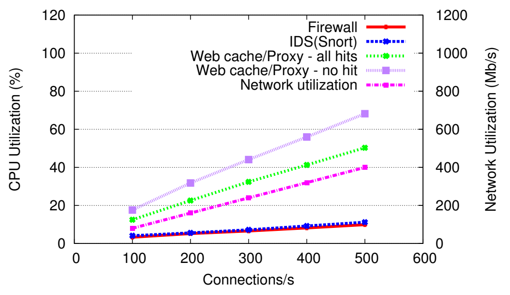
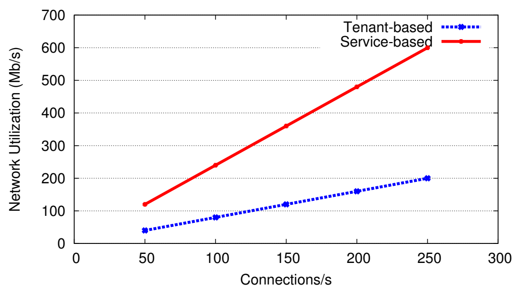
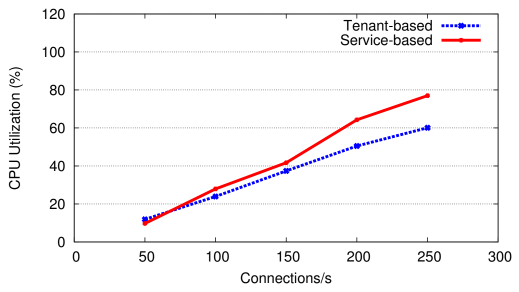


Fig. 3. CPU and network utilization of network functions



(a) Networkutilization of tenant-based approach and service-based approach



(b) CPU utilization of tenant-based approach and service-based approach

Fig. 4. Hypervisor CPU and Network utilization of the two deployments.

traffic of each tenant is configured to be the same for both approaches. We instrument the hypervisors in both tenant-based and service-based deployments to measure the utilization.

In Figure 4(a), we show that the NIC bandwidth consumed by tenant-based deployment is only 1/3 of that consumed by service-based deployment. This is because in tenant-based approach, traffic between network functions goes through internal bridges within a physical server's boundary while in service-based approach, the traffic goes between physical servers through physical NICs. This makes tenant-based approach much more bandwidth efficient than service-based approach. However, this bandwidth saving is not acquired for free. From Figure 4(a), we can see that although NIC throughput of service-based approach is three time than that of tenant-based, the CPU utilization of service-based approach is only slightly higher. This is because forwarding packets through internal bridges in tenant-based approach is CPU consuming and brings overhead for the hypervisor.

Conclusion: Tenant-based deployment is less NIC bandwidth consuming than tenant-based deployment because packets go through internal bridges between network functions. Both tenant-based and service-based deployment incur non negligible hypervisor CPU overhead that increases linearly with network traffics. Specifically, to achieve the same throughput for each tenant, service-based deployment requires much more network bandwidth and slightly more CPU resources than service-based deployment.

E. Packing Efficiency

Next, we compare the packing efficiency of the tenant-based and service-based approaches. The packing efficiency is measured as the number of servers required to handle the same workload level—that is, we simulate different scenarios in which NFaaS customers request different resources for their network functions. Specifically, we divide NFaaS customers into three groups, i.e., small, medium and large, based on their resource requirements. In Table I, we list resource requirements of different customers in the form of CPU and Memory. These requirement values are taken directly from our micro benchmarks and correspond to NFs handling 300, 600 and 1200 connections per second respectively. In addition, the NFaaS cloud is simulated with four different types of physical servers and we show the corresponding configurations in Table II. Finally, we choose to use *best fit* algorithm that places all network functions of next customer to eligible servers with smallest amount of available CPU cores, for both deployment modes. Here a server is eligible only if it has enough available CPU and memory capacity.

We first look at the total number of small physical servers required to place large-sized customers. As shown in Figure 5(a), deploying NFs service-basedly can save up to 30% server resources compared to tenant-based deployment. In

NFaaS Customer Type	Firewall	IDS	Web Proxy
Small	(0.1, 71)	(0.1, 462)	(0.5,147)
Medium	(0.2, 71)	(0.2, 470)	(1, 151)
Large	(0.4, 71)	(0.4, 485)	(2, 160)

TABLE I
NFaaS customer configuration for simulations. Resource requirements are specified in the form of number of CPU cores and memory (MB).

Server Type	Cores	Memory (GB)
Small	4	8
Medium	6	12
Large	8	16
Xlarge	12	24

TABLE II
Server configurations for our simulations.

essence, service-based deployment outperforms tenant-based deployment because it has more eligible servers to choose from during each placement decision. For example, any physical server with less than 1.4 CPU cores is not eligible for tenant-based deployment.

Next we compare the number of large servers required to host a thousand customers with different resource requirements. We observe that the benefits of service-based deployment over tenant-based deployment increase from 3% to 30% when we need to handle customers with more resource demands. Similarly, this is because a larger customer leads to more potential waste of spare server resources.

Finally, we total the number of servers required to run NFs for one thousands customers of varying resource requirements.

In Figure 5(c), we show that tenant-based placement performs worse when only has access to small servers with limited resources. As we increase the server size, the difference between tenant-based and service-based placements converges. This is because having access to larger servers offsets the resource constraints exerted by tenant-based placement.

Result: The service-centric approach has higher packing efficiency than a tenant-centric deployment, with up to 30% fewer physical servers. In particular, tenant-based approach has poor packing efficiency when placing customers with higher resource demands or on to smaller servers.

F. End-to-end Performance

In this experiment, we evaluate the end-to-end response time and bandwidth of deploying a network service using either tenant-centric or service-centric placement. We create a logical topology where user generated requests have to traverse firewall, IDS and then web proxy in sequence. Similar to Figure 1, for the tenant-centric approach, we place all three services on the same quad-core server, while for the service-centric approach, we place these services on three different servers.

For each physical topology, either tenant-based or service-based, we begin the experiment by sending HTTP requests to fetch web pages from the web server. Specifically, we limit the request rate to 100 connections per second to avoid bandwidth congestion and make sure that all user requests go through all three middleboxes. We run each experiment for one minute and measure the average response time over all requests. As shown in Figure 6, tenant-centric placement incurs up to 20% lower latency, for both web page sizes, under similar server load. The response time difference is because packets traversing through multiple physical servers in the service-centric placement. We expect to observe an even higher performance gap if middleboxes were running in different racks for service-based deployment.

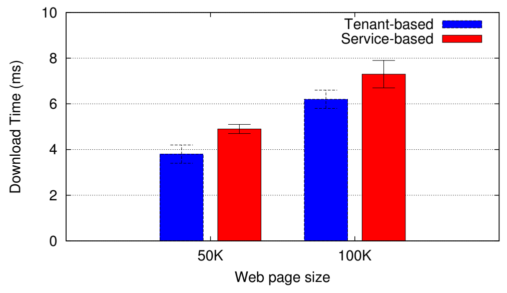


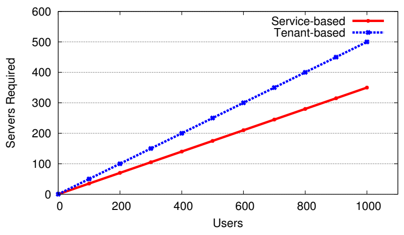
Fig. 6. Comparison of response time. Tenant-based deployment can achieve up to 20% improvement comparing to service-based deployment when all network functions run in the same rack.

Next, we use the same physical topologies and measure the maximum throughput for topologies. We use `iperf` to generate TCP traffic and measure the end-to-end throughput during steady state. We observe that tenant-based deployment can support up to 942 Mbps when using a dedicated physical server with 1Gbps NICs. This is because tenant-based deployment has full access to physical NICs. However, we only observe around 310 Mbps throughput in service-based deployment. The low throughput in mainly due to the traffic having to traverse through multiple physical NICs that are shared with multiple tenants. In this case, the 1Gbps network interfaces on each server are shared among three cloud tenants.

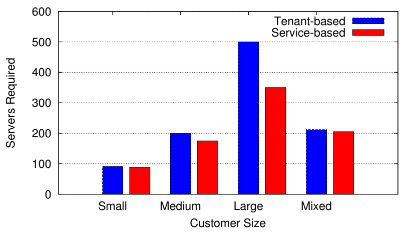
Result: Deploying network functions using a tenant-centric approach can lead to better end-to-end response time and higher throughput than the service-centric approach.

V. Related Work

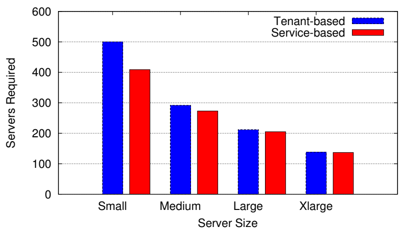
The promise of implementing software-based network functions and running them on commodity high-volume servers has attracted significant attention from the research community [8]. In NFV, individual network functions are implemented in software and use virtualization to replace its hardware counterpart. Efforts to improve packet processing performance using commodity NICs [5], [16] and packet transfers in the virtualized environments [17], [11] have significantly improve the performance of running virtualized NFs in the commodity settings. Almost always, NFs are used in combination to form network services and these end-to-end services introduce new problems in managing the end-to-end performance [13], [18], [15], [9], especially in the multi-tenant cloud environments [19], [6].



(a) Packing for Large-sized Customers.



(b) Impact of Customer Types. Mixed means equal number of each type of customers.



(c) Impact of Server Sizes.

Fig. 5. Comparison of packing efficiency for tenant-based and service-based deployment under different scenarios.

Unlike traditional VM placement [7], [12], [14] where the VM resource needs may not be known in advance, in our case, the characteristics of each network function is known in advance, enabling more optimal placement.

VI. Conclusions

In this paper, we studied how network functions as a service can be deployed in cloud environments. We studied two different placement strategies, tenant-centric and service-centric, for deploying network functions in the cloud. Our experimental evaluation using a NFaaS prototype and simulations show that tenant-centric placement can achieve better network performance because it avoids cross-service traffic from traversing network switches, which saves physical bandwidth and reduces network delay. In contrast, the service-centric approach is easier to manage and deploy; simulations using real measurements show that this approach yields better resource utilization in the cloud. Our results demonstrate the tradeoffs of the two approaches and provide guidance on which approach to choose based on the overall design goals.

VII. Acknowledgments

We thank the anonymous reviewers for their helpful comments. This work was supported in part by NSF grants 1422245, 1339839 and an internship at IBM Research. Any opinions, findings, and conclusions or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of the funding agencies.

References

[1] Snort. https://www.snort.org/.
[2] Squid Web Cache. http://www.squid-cache.org/.
[3] Httperf. https://sourceforge.net/projects/httperf/.
[4] Dstat. http://dag.wiee.rs/home-made/dstat/.
[5] Intel Data Plane Development Kit. http://dpdk.org.
[6] A. Bremler-Barr, Y. Harchol, D. Hay, and Y. Koral. Deep packet inspection as a service. In *Proceedings of the 10th ACM International on Conference on Emerging Networking Experiments and Technologies*, CoNEXT '14, pages 271–282, New York, NY, USA, 2014. ACM.
[7] N. M. Calcavecchia, O. Biran, E. Hadad, and Y. Moatti. Vm placement strategies for cloud scenarios. In *Cloud Computing (CLOUD), 2012 IEEE 5th International Conference on*, pages 852–859. IEEE, 2012.
[8] ETSI. Network functions virtualization: Architectural framework. http://www.etsi.org/deliver/etsi_gs/nfv/001_099/002/01.01.01_60/gs_nfv002v010101p.pdf, 2013.
[9] A. Gember, A. Krishnamurthy, S. S. John, R. Grandl, X. Gao, A. Anand, T. Benson, A. Akella, and V. Sekar. Stratos: A network-aware orchestration layer for middleboxes in the cloud. *CoRR*, abs/1305.0209, 2013.
[10] X. He, T. Guo, E. M. Nahum, and P. Shenoy. Placement strategies for virtualized network functions in a nfaas cloud. Technical Report UM-CS-2016-009, College of Information and Computer Sciences, University of Massachusetts Amherst, Amherst, Massachusetts, September 2006.
[11] J. Hwang, K. K. Ramakrishnan, and T. Wood. NetVM: High performance and flexible networking using virtualization on commodity platforms. In *11th USENIX Symposium on Networked Systems Design and Implementation (NSDI 14)*, pages 445–458, Seattle, WA, Apr. 2014. USENIX Association.
[12] D. Jayasinghe, C. Pu, T. Eilam, M. Steinder, I. Whally, and E. Snible. Improving performance and availability of services hosted on iaas clouds with structural constraint-aware virtual machine placement. In *Services Computing (SCC), 2011 IEEE International Conference on*, pages 72–79. IEEE, 2011.
[13] J. Martins, M. Ahmed, C. Raiciu, V. Olteanu, M. Honda, R. Bifulco, and F. Huici. ClickOS and the art of network function virtualization. In *11th USENIX Symposium on Networked Systems Design and Implementation (NSDI 14)*, pages 459–473, Seattle, WA, Apr. 2014. USENIX Association.
[14] K. Mills, J. Filliben, and C. Dabrowski. Comparing vm-placement algorithms for on-demand clouds. In *Cloud Computing Technology and Science (CloudCom), 2011 IEEE Third International Conference on*, pages 91–98. IEEE, 2011.
[15] S. Palkar, C. Lan, S. Han, K. Jang, A. Panda, S. Ratnasamy, L. Rizzo, and S. Shenker. E2: A framework for NFV applications. In *Proceedings of the 25th Symposium on Operating Systems Principles*, SOSP '15, pages 121–136, New York, NY, USA, 2015. ACM.
[16] L. Rizzo. Netmap: A novel framework for fast packet i/o. In *Proceedings of the 2012 USENIX Conference on Annual Technical Conference*, USENIX ATC'12, pages 9–9, Berkeley, CA, USA, 2012. USENIX Association.
[17] L. Rizzo and G. Lettieri. VALE, a switched ethernet for virtual machines. In *Proceedings of the 8th International Conference on Emerging Networking Experiments and Technologies*, CoNEXT '12, pages 61–72, New York, NY, USA, 2012. ACM.
[18] V. Sekar, N. Egi, S. Ratnasamy, M. K. Reiter, and G. Shi. Design and implementation of a consolidated middlebox architecture. In *Presented as part of the 9th USENIX Symposium on Networked Systems Design and Implementation (NSDI 12)*, pages 323–336, San Jose, CA, 2012. USENIX.
[19] J. Sherry, S. Hasan, C. Scott, A. Krishnamurthy, S. Ratnasamy, and V. Sekar. Making middleboxes someone else's problem: Network processing as a cloud service. In *Proceedings of the ACM SIGCOMM 2012 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, SIGCOMM '12, pages 13–24, New York, NY, USA, 2012. ACM.