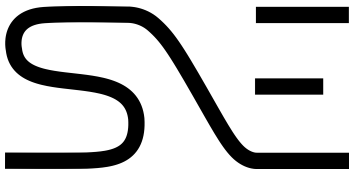

MSc System and Network Engineering
Research Project 1

Network Functions Virtualization

Bernardus A. Jansen, BSc
Bernardus.Jansen@os3.nl

February 11, 2018

Supervisors:
Dr. R. Malhotra
J. Kloots

Assessor:
Prof. Dr. C.T.A.M. de Laat

Universiteit van Amsterdam

Abstract

While virtualization of applications has been very popular for a number of years, virtualization of network functions is markedly less commonplace. With *aaS offerings increasingly popular and hardware increased in capacity and extended with support for virtualization, the network functions virtualization paradigm is interesting for both service providers and organizations looking to outsource their IT operations.

However, even with evolved technology, virtualizing network functions still has its challenges. In this paper we identify the challenges and opportunities for network functions virtualization. We find that NFV is suitable to be offered as a service, but full virtualization of network functions is still difficult to achieve.

Contents

1 Introduction

While server virtualization technologies have been mainstream for several years, virtualizing networking applications has not taken off as fast due to their demand for high throughput and low latency. With hardware capacities increased and tailored to support virtualization technologies, it may now be possible to meet these demands.

With the popular *aaS offerings commonly seen today, virtualizing network functions is a very interesting field for both service providers and organizations. Network Functions Virtualization (NFV) could allow external providers to offer functionality and performance that was previously only possible and offered by on-premises physical hardware. Service providers could utilize NFV to expand their offerings and enable organizations to outsource IT hardware and operations. This could in turn unburden these organizations from the administrative and operational overhead associated with physical hardware and pave the way for pay-per-use licensing models that may prove to be more cost-efficient than the current situation.

Though hardware and software technologies have been optimized allowing for virtualization of network functions, physical and technical limitations may still arise. For example, latency to offsite hardware will always be higher than to onsite hardware. In addition, dependencies among services may cause difficulties when transitioning an existing physical environment to a virtualized environment.

This paper will look into the current state of the network functions virtualization landscape and investigate the feasibility of replacing hardware based on-site network functions with virtualized network functions in an external provider network. This project will only focus on virtualizing network functions and not on application services such as e-mail and webservers.

1.1 Research question

The main research question of this project is as follows:

How can services in a campus network be aided by virtualization by an external service provider?

The main research question is divided into the following sub-questions:

- *Which network functions within campus networks are suitable to be virtualized?*
- *Which technical aspects need to be considered if an external service provider would decide to provide one or more of these virtualized functions?*
- *Does the distance of the virtualized platform from the campus affect the performance of the virtualized function? Is this performance dependent on the function itself?*
- *Is it feasible to just virtualize one function or are they so inter-dependent with other network functions in the campus domain that eventually a virtualized solution should be offered for all network functions within a campus network?*

1.2 Outline

In this paper we make a basic overview of a campus network and identify the network functions that are operated in such an environment. We then identify any challenges that may need to be taken into consideration should one wish to virtualize or outsource any or all network functions in the environment. We conclude with the discussion, our conclusion and pointers for future work into NFV.

2 Related Work

Virtualization of network functions has received significant attention from researchers and organizations. The white paper that coined the term *Network Functions Virtualization*[2] originated from a reaction to advances made in the areas of virtualization and software defined networking (SDN) and the observation that networks contained increasingly large numbers and varieties of hardware appliances, as also supported by a survey conducted by Sherry et al[20]. The paper was published by a team part of the European Telecommunications Standards Institude (ETSI). Since then, ETSI has spawned the NFV Industry Specification Group and actively continues researching NFV. ETSI also provides documents detailing various use cases where NFV may be useful[9] and a reference NFV framework architecture[10].

NFV has also received attention from the open-source community, with a number of open source projects facilitating NFV having spawned, such as OpenContrail[17] and the OPNFV project[18]. OpenContrail can be deployed as a platform to support virtual network functions and to manage the functions running on top of it. The OPNFV project was started by the Linux foundation and has as its goal to further the development of components for NFV platforms and to design a reference framework for NFV infrastructure.

Vendors that traditionally sold and supported hardware-based appliances have also identified NFV as a field offering opportunities. Cisco for example currently uses NFV both to provide a platform for customers to run virtual network functions with Cisco as a service provider[3] as well as to enable customers to virtualize a number of network functions on a single on-premises machine[4].

On the technical side of NFV, a number of solutions exist that may assist in achieving high networking performance in a virtualized environment. Hardware extensions such as Single-root Input/Output Virtualization (SR-IOV)[12] and AMD-Vi/Intel VT-d[5][1] have been introduced that allow to pass hardware through to virtual machines, enabling them to directly address the hardware without going through the hypervisor, reducing overhead in virtualized environments.

To reduce the overhead within the operating system itself, frameworks such as the Data Plane Development Kit (DPDK)[16] and New API (NAPI)[19] have been developed. A main advantage of DPDK is that it enables to handle all packet processing in userland, meaning it is no longer necessary to switch from kernel- to usermode when processing packets. It also allows one to write their own libraries for processing packets, bypassing the default networking stack. The main advantages of NAPI are that it reduces system load while processing packets by reducing the number of interrupts and drops packets in advance when the system is overloaded, before any processing is done on them[6].

Another novel approach to assist software packet processing is PacketShader[7]. PacketShader utilizes graphics processing units (GPU) to process networking packets. As GPUs have large numbers of processing cores, these cores can be utilized to massively parallelize processing of networking packets. This paralellism may be especially interesting for an NFV platform where a single physical server hosts multiple virtual network functions.

3 Network Function Infrastructure

Educational institutions are a typical example of organizations that have extensive networks with large numbers of services and network functions as well has high demands for security, while not having IT as their core business. Due to their educational

nature, these institutions generally also have a large amount of users compared to their number of employees. For these organizations, maintaining a large number of network functions as well as staff to manage and support it may be seen as an undesirable side effect of their core business. According to SURFnet internal research, smaller educational instances are already commonly seen to outsource their IT management, but remain confronted with the administrative overhead of procuring and operating physical hardware. These institutions might therefore profit from outsourcing not only the management and operational side of their network, but also the actual infrastructure and network functions themselves to an external service provider.

3.1 Common Network Functions

Network functions that are currently commonly found in organizations as hardware appliances are firewalls, routers, proxies, VPN, Load Balancers, Intrusion Detection Systems (IDS) and WAN optimizers[20]. A generic term for these devices is *middleboxes*.

Different types of middleboxes can have very different requirements for networking performance depending on their function. Routers, firewalls and intrusion detection systems typically handle large amounts of data as practically all traffic to and from the organization network has to traverse these devices, while load balancers typically only handle incoming traffic to one or more specific applications. VPN appliances are commonly used to enable users to access hosts and services within the organization remotely, and may have lower requirements with regards to throughput and latency. The different requirements can impact the feasibility of virtualizing these functions.

3.2 Network Function Topology

As hardware middleboxes are inherently physical devices, they generally have a fixed position in a network. This in particular holds for appliances such as firewalls and routers. While the position of a network function in a network generally does not change over time, different organizations may structure their networks differently. Some organizations may for example handle firewalling at the outer edges of their network with a limited number of high-capacity firewalls, while others may choose to place a larger number of 'smaller' firewalls more internally in the network. An example of the former with a single generic appliance is shown in Figure 1.

The topology of the network functions can have implications for the feasibility of outsourcing network functions to an external service provider. For example, should an institution wanting to utilize NFV only have a limited-bandwidth uplink, it may not be feasible to move core routers and firewalls that handle internal routing off-site. This would mean all internal traffic will have to traverse the uplink that has only a limited capacity. Furthermore, moving only a subset of internal network functions off-site may also have unwanted consequences. Should an organization maintain the edge of their network all traffic passes through on-site but only move smaller-scale network functions that may be more suitable for virtualization off-site to a service provider, traffic to and from these functions will have to traverse their connection to the internet multiple times. An example of such a situation is shown in Figure 2. The increase in latency and bandwidth consumption may also be a concern when an organization gradually transitions from a physical on-site setup to a virtualized off-site setup by first outsourcing smaller core appliances.

A better strategy for migrating an existing on-site network function infrastructure to an external service provider is to first migrate the network functions located at the

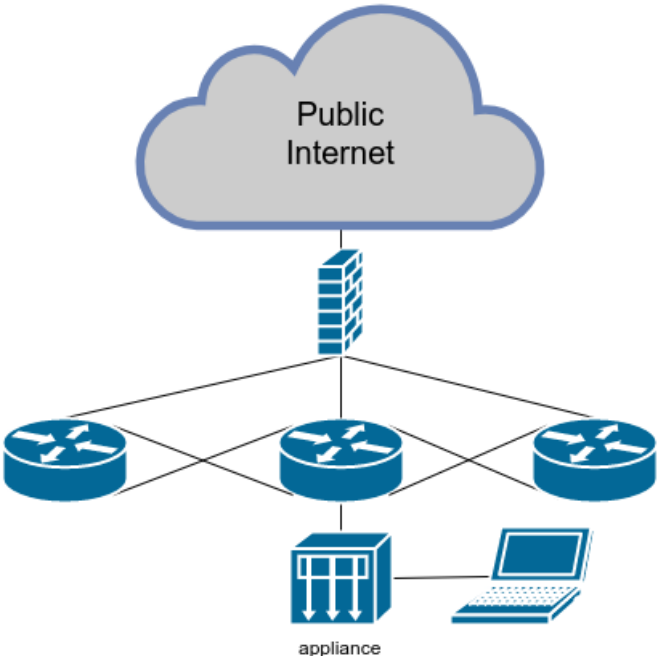

Figure 1: Simple example of traditional campus network function infrastructure.

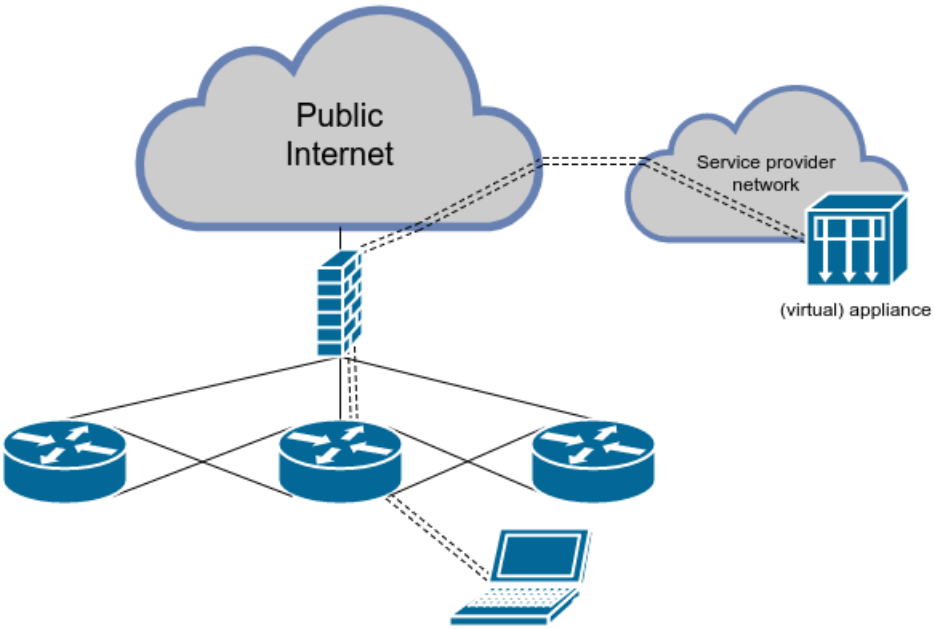

Figure 2: Example setup with off-site core appliance. The dashed lines signify a tunneled connection.

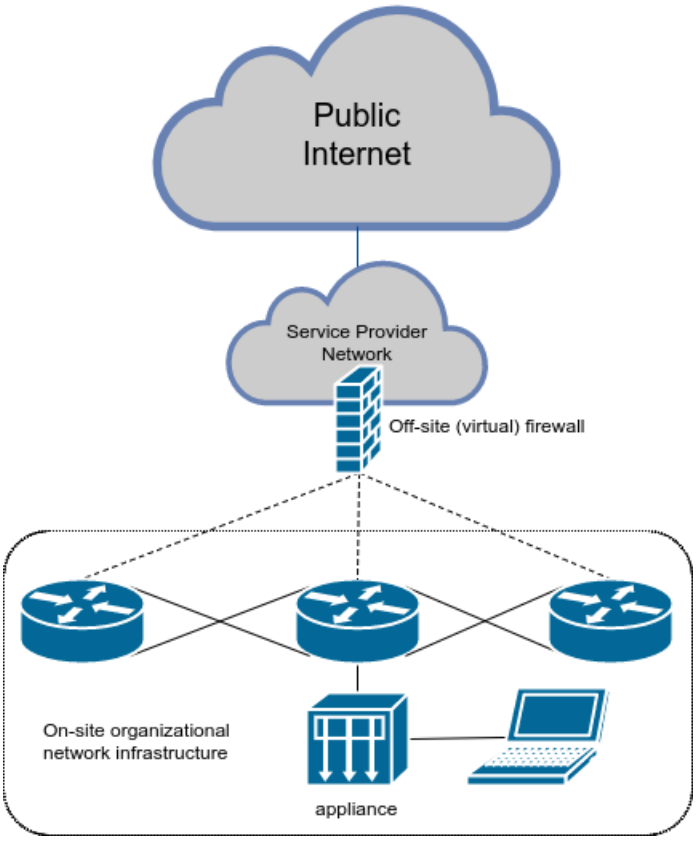

Figure 3: Example setup with off-site edge firewall.

edge of the network, as depicted in Figure 3. In this scenario, no additional bandwidth and latency are added as only the physical topology of the network changes, with the firewall now located in the service provider network, whereas the logical topology of the network remains the same as it is in Figure 1. A disadvantage of this setup is that it requires for all traffic to and from the campus network to flow through the service provider network.

4 Technical Considerations

Existing physical appliances typically handle packet processing in hardware in application specific integrated circuits. When network functions are moved to commodity hardware, processing of packets has to be done in software. In this scenario, existing software and network stacks can induce an unwanted and potentially significant overhead, hampering networking performance. The overhead is increased further when packet processing is done in a virtualized environment due to the added overhead of a hypervisor.

To give an indication of the speeds at which packets have to be processed in high-throughput scenarios, when receiving 64-byte (84 bytes on the wire including the 20-byte inter-frame gap and preamble) packets at 10Gbit/s, $\frac{10 * 10^9}{84 * 8} = 14.88 * 10^6$ packets have to be processed in a single second. This means processing a single packet can take at most $\frac{1}{14.88 * 10^6} = 67$ ns to be able to achieve wirespeed performance. To put this into perspective, a single context switch takes over 1000 ns or even much longer, depending on the CPU[21][13]. These numbers suggest that with off-the-shelf software and operating systems, multi-million packets per second performance may be very hard to achieve. In a performance test conducted by the developers of ClickOS, a Linux virtual machine running on the Xen hypervisor could only be seen to process up to 625,000 packets per second[15]. This is number over 20 times less than what is required to sustain 10Gb/s of throughput with small packets. Even with large packet sizes, the virtual machine in the test only managed to achieve throughput of up to 6.46Gb/s at about 530,000 packets per second[15].

While processing larger numbers of packets per second is possible[14], these levels of performance can only be achieved when the hardware is utilized as effectively as possible. This, for example, means that virtual machines should be pinned to specific CPU cores in order to optimally profit from branch prediction and to reduce the likelihood of cache misses, which are very costly timewise and can lead to packet loss in high-throughput scenarios. In addition, as most servers hosting virtual machines have multiple CPUs, these CPUs generally operate in their own Non-Uniform Memory Access (NUMA) domain. This means one has to take note of the NUMA domain of the CPU cores a virtual machine is running on. When these cores are in a different NUMA domain than the networking hardware, throughput and latency are negatively affected, hampering networking performance[22].

To enable high-speed packet processing in virtual machines, hardware extensions have been developed that allow passing through of (networking) hardware to virtual machines[12][5][1]. With direct hardware access, virtual machines can utilize frameworks such as DPDK to directly process packets in user mode with custom network stacks. Software overhead can be reduced even further by developing special Unikernel network functions. Unikernels are highly specialized applications where the software is specifically tailored for its intended purpose and integrated in a small kernel that is intended to be run as a virtual machine. A downside for Unikernels in NFV is however

that they rely on emulated hardware as they lack specific hardware drivers. While it would be possible to develop Unikernels with support for specific (passed-through) hardware, this in turn reduces flexibility when compared to general purpose operating systems.

While there are a number of methods that allow to achieve the demands for network functions in a virtualized environment on commodity hardware, the need for direct hardware access and specially tailored applications means the virtual network functions are not truly platform agnostic, reducing flexibility. This was already identified as a technical challenge and an area where further development is needed in the original NFV whitepaper in 2012[2]. When the virtualized environment relies heavily on passed-through devices and extensive knowledge of the underlying hardware, managing and operating such an environment could end up being quite comparable to operating an environment based on hardware appliances.

5 Opportunities for NFV

To effectively offer network functions virtualization as a service, traffic through the virtualized network functions should be able to reach its destination directly, without having to be tunneled back through the network the traffic originated from, as already shown in Figure 3. This means NFV has its main applications in scenarios where the service provider is also the backbone provider for its customers. In this case the client network segmentation can be done from within the provider network, and allows for network segments with reduced throughput requirements to be virtualized separately from the main network, without its traffic first having to go through routers in the client network.

The main disadvantage of outsourcing edge network functions to an external service provider is however that these functions generally require the highest throughput and lowest latency of all network functions. As organizations may still wish to outsource these functions, service providers may benefit from setting up a hybrid NFV platform where network functions are in part virtualized on commodity hardware and partially hosted on traditional hardware appliances. These hardware appliances allow to move the high-throughput edge network functions off-site and to virtualize the core network functions with lower performance requirements behind it, all in the provider network. This setup can also assist in migrating an existing on-premises infrastructure to the service provider. With the edge network migrated to its network, the service provider can then choose to continue running the high-throughput network functions on hardware appliances, or to divide them up into a number of 'smaller' network functions that handle less traffic each, and may allow them to be virtualized. While in a hybrid setup hardware appliances are still part of the equation, them being operated by the service provider still frees clients from having to operate and manage the specialist hardware. In addition, as technology supporting NFV matures, the service provider may be able to transition the hosted network functions to a fully virtualized platform.

NFV may also offer advantages for organizations that wish to maintain and operate their network functions themselves. NFV may for example allow them to consolidate multiple network functions onto a single platform, freeing them from maintaining and acquiring specific hardware, and allows for added flexibility with regards to introducing new functions into the network.

6 Discussion

In this project, physical distance between network functions was not considered as a factor in determining whether NFV may be considered useful for an organization. While placing network functions off-site adds latency compared to on-premises functions, most organizations already centralize their network functions in certain locations, which means the physical distance data from an end user to its destination has to traverse is not necessarily increased significantly. This however only holds when the service provider is located relatively close to its client base, and not for example situated in another country, at a scale where the physical distance may lead to a more significantly increased latency.

Security implications of virtualizing network functions were also not specifically considered in this project. Some organizations may find it undesirable to have 'internal' traffic traversing an external connection. While encryption can be applied to traffic to and from the service provider, this may be very costly for high bandwidth connections. In addition, the virtual nature of the network functions platform may also open the network up to vulnerabilities in the platform itself.

7 Conclusion and Future Work

Virtualizing network functions offers numerous advantages over hardware appliances. From the perspective of the operator however, the advantages are much less pronounced when it comes to virtualizing high-performance network functions. This means hardware appliances are likely to remain commonplace in networks in the near future. Service providers looking to add NFV to their service portfolio might therefore consider setting up hybrid platforms where certain services may be hosted on commodity hardware and others on hardware appliances. This setup already provides most of the advantages a full NFV setup has from the perspective of the customer, with the administrative load of managing the remaining hardware appliances ending up at the service provider, for which the cost of managing these devices is relatively small compared to their existing IT operations.

Further development of both hardware and software may be necessary to extend what can be achieved with network functions virtualization. On the hardware side there may be a market for new devices that can assist in packet processing for a virtualized environment. Such hardware could take the form of PCI cards to be inserted in commodity servers, or as a new type of hardware appliance that can be utilized to support many types of virtualized network functions on a single piece of equipment. On the software side, specialized network functions Unikernels may prove to be very interesting for achieving high performance.

Existing research into software packet processing may also be interesting for NFV if it can be extended to include or apply to virtualized environments. Existing technologies such as using GPUs to assist in packet processing[7][11] may for example be very interesting in virtualized environments hosting multiple network functions due to the parallel nature of their processing. New research may be necessary to make GPU cores available to multiple virtual machines simultaneously, as opposed to entire graphics cards passed through as is already possible using VT-d and AMD-Vi. Technologies such as NVIDIA GRID[8] may already be useful for such applications.

In this project we have looked into migrating existing hardware-based network functions infrastructure to a hosted (virtualized) setup. This however is not necessarily complete, and possible network functions migration strategies may be an interesting area for further research in itself.

References

[1] Darren Abramson, Jeff Jackson, Sridhar Muthrasanallur, Gil Neiger, Greg Regnier, Rajesh Sankaran, Ioannis Schoinas, Rich Uhlig, Balaji Vembu, and John Wiegert. Intel virtualization technology for directed i/o. *Intel technology journal*, 10(3), 2006.

[2] Margaret Chiosi, Don Clarke, Peter Willis, Andy Reid, James Feger, Michael Bugenhagen, Waqar Khan, Michael Fargano, Chunfeng Cui, Hui Deng, et al. Network functions virtualisation: An introduction, benefits, enablers, challenges and call for action. In *SDN and OpenFlow World Congress*, pages 22–24, 2012.

[3] Cisco. Cisco NFV Infrastructure. `https://www.cisco.com/c/en/us/solutions/service-provider/network-functions-virtualization-nfv-infrastructure/index.html`. (Accessed on 2018-01-19).

[4] Cisco. Enterprise Network Functions Virtualization (NFV). `https://www.cisco.com/c/en/us/solutions/enterprise-networks/enterprise-network-functions-virtualization-nfv/index.html`. (Accessed on 2018-01-19).

[5] Advanced Micro Devices. AMD I/O Virtualization Technology (IOMMU) Specification, 2007.

[6] Linux Foundation. networking:napi [linux foundation wiki]. `https://wiki.linuxfoundation.org/networking/napi#advantages`, November 2016. (Accessed on 2018-01-24).

[7] Sangjin Han, Keon Jang, KyoungSoo Park, and Sue Moon. Packetshader: a gpu-accelerated software router. In *ACM SIGCOMM Computer Communication Review*, volume 40, pages 195–206. ACM, 2010.

[8] Alex Herrera. Nvidia grid: Graphics accelerated vdi with the visual performance of a workstation. *Nvidia Corp*, 2014.

[9] European Telecommunications Standards Institute. Network Functions Virtualization (NFV) Use Cases. ETSI GS NFV 001, European Telecommunications Standards Institute, October 2013.

[10] European Telecommunications Standards Institute. Network functions virtualization: Architectural framework. ETSI GS NFV 002, European Telecommunications Standards Institute, December 2014.

[11] Anuj Kalia, Dong Zhou, Michael Kaminsky, and David G Andersen. Raising the bar for using gpus in software packet processing. In *NSDI*, pages 409–423, 2015.

[12] Patrick Kutch. PCI-SIG SR-IOV Primer. Technical report, Intel, 2011.

[13] Chuanpeng Li, Chen Ding, and Kai Shen. Quantifying the cost of context switch. In *Proceedings of the 2007 workshop on Experimental computer science*, page 2. ACM, 2007.

[14] Marek Majkowski. How to receive a million packets per second. `https://blog.cloudflare.com/how-to-receive-a-million-packets/`, 06 2015. (Accessed on 2018-01-18).

[15] Joao Martins, Mohamed Ahmed, Costin Raiciu, Vladimir Olteanu, Michio Honda, Roberto Bifulco, and Felipe Huici. Clickos and the art of network function virtualization. In *Proceedings of the 11th USENIX Conference on Networked Systems Design and Implementation*, pages 459–473. USENIX Association, 2014.

[16] DPDK Project. DPDK project charter. `http://dpdk.org/about/charter`. (Accessed on 2018-01-09).

[17] OpenContrail Project. About OpenContrail. `http://www.opencontrail.org/about/`. (Accessed on 2018-01-15).

[18] OPNFV Project. OPNFV website. `https://www.opnfv.org/`. (Accessed on 2018-01-08).

[19] Jamal Hadi Salim, Robert Olsson, and Alexey Kuznetsov. Beyond softnet. In *Annual Linux Showcase & Conference*, volume 5, pages 18–18, 2001.

[20] Justine Sherry, Shaddi Hasan, Colin Scott, Arvind Krishnamurthy, Sylvia Ratnasamy, and Vyas Sekar. Making middleboxes someone else's problem: network processing as a cloud service. *ACM SIGCOMM Computer Communication Review*, 42(4):13–24, 2012.

[21] Benoit Sigoure. How long does it take to make a context switch? `http://blog.tsunanet.net/2010/11/how-long-does-it-take-to-make-context.html`, November 2010. (Accessed on 2018-01-24).

[22] Chengwei Wang, Oliver Spatscheck, Vijay Gopalakrishnan, Yang Xu, and David Applegate. Toward high-performance and scalable network functions virtualization. *IEEE Internet Computing*, 20(6):10–20, 2016.