A Survey of Network Function Placement

Xin Li and Chen Qian
Department of Computer Science, University of Kentucky
Email: xin.li@uky.edu, qian@cs.uky.edu

***Abstract*—Ranging from web caches to firewalls, network functions play a critical role in modern networks. The emerging of Network Function Virtualization (NFV) has recently gained wide attention from both industry and academia, also making the study of their placement a popular research topic. This paper surveys recent network function orchestration frameworks, and particularly the network function placement strategies. We identify their design considerations as well as the companion advantages and disadvantages for different placing strategies in this paper.**

I. Introduction

Network functions (NFs), also known as "middleboxes", are playing an increasingly important role in modern networks, ranging from mobile networks, enterprise networks, to data-center networks. A recent survey [29] shows that the number of NFs is comparable to that of the forwarding devices, indicating their significance. NFs improve the network performance (e.g., WAN Optimizer, web proxy and video transcoder, load balancer), enhance the security (e.g., firewall, IDS/IPS) or monitor the traffic (e.g., lawful interception, passive network monitor).

Conventionally, NFs are built in dedicated hardware for performance concerns, which incur high capital investment and operating expense. Furthermore, they are hard to manage. Their replacement and upgrade involve non-trial human labor. In light of this situation, NFV [12] was proposed, aimed to address these issues by leveraging visualization technologies to consolidate NFs into general-purpose hardware platforms. NFV, along with Software-Defined Network (SDN), enables automated management of the whole life cycle of virtual network functions (VNFs), leading to resource efficiency and expense reduction. How NFs are placed directly determines some important metrics, such as operating cost and end-to-end latency. To achieve the benefits from NFV's ability to conveniently place VNFs anywhere anytime, a number of VNFs placement strategies are proposed for different NFV orchestration frameworks. Apart from centralized NFs, some other forms of NFs exist towards different objectives, and their placement is according quite different. We will discuss different NF frameworks and their placement issues in this paper.

The rest of this paper is organized as follows. We go deep into hardware NFs in Sec. II. We review some recent advances in NFV in Sec. III, followed by other forms of NFs described in Sec. IV. We discuss related placement-related issues in Sec. V and potential future research directions in Sec. VI. We finally conclude this paper in Sec. VII.

II. Hardware Network Function Placement

Until recently, NFs were mostly built in dedicated hardware. Though hardware NFs are high-capacity, they are power hungry and incur large capital investment. As a result, there are only a limited number of hardware NFs deployed in fixed points in the network. Different hardware NFs have their own placement considerations and thus different placement strategies, as we discuss in this section. TABLE. I summarizes the comparison.

A. Independent passive NFs

Some passive NFs can work independently to perform a complete task, without the interaction with other NFs. The placement strategies for this kind of NFs either tries to maximize the utility under the budget constraints or to minimize the total cost while a certain level of service is guaranteed.

The passive monitor placement falls in this category. The passive monitor is inside a router or a stand alone device that tap into a communication link. There is an intrinsic trade-off between the monitoring coverage and cost (which consists of the initial capital investment and operating expense). To this end, the proposed solution in [30] carefully places passive monitors and controls their sample rate, without chaning traffic routing pathes. Two integer linear programs (ILPs) are formulated respectively for two conflicting objectives: One is to maximize the fraction of IP traffic being sampled and the other is to minimize the total monitoring cost. The results of both optimization problems determine the number of monitors, their optimal place positions and their sampling rate. It can be further proven that both problems are NP-hard (maximizing monitoring coverage and minimizing cost inherent the hardness of SET COVER and MAX k-COVERAGE, respectively). Therefore, greedy heuristics are proposed to solve the optimization problems.

One problem of this static placement strategy is that varying traffic matrix renders the previously optimal solution sub-optimal. MeasuRouting [24] addresses this problem by strategically routing traffic to fix monitoring points while routing pathes are least disrupted. It is optional after the NF placement.

B. Chained NFs

Some flows may be required, either by applications or users, to traverse through a given sequence of NFs. This is called service chaining (or policy chaining). For example, the network administrator may specify that all http traffic should follow the service chain: $firewall \rightarrow IDS \rightarrow proxy$, for security purposes. There are two requirements for service chaining:

NF type	Location	Traffic steering	Placement objective
Independent NFs	in-line	optional	max cov./min cost
Chained NFs	off-line	compulsory	min latency

TABLE I
COMPARISON BETWEEN INDEPENDENT PASSIVE NFS & CHAINED NFS.

- **Correctness** : The sequential order must hold.
- **Efficientcy** : Traffic should not traverse unnecessary NFs.

Service chaining is more complicated than using independent passive NFs. Trivially placing NFs in order on the routing path may require a prohibitively large number of NFs. Existing solutions leverage traffic steering to accomplish service chaining.

PLayer [15] is a policy-aware switching layer adhering to these two requirements. PLayer explicitly forwards different classes of traffic through dif sequences of NFs scattering around the network. PLayer does not use bump-in-the-wire fashion (in-line), because traffic is also required to not traverse unwanted NFs. To achieve this, off-line processing is utilized, which means traffic is explicitly forwarded to NFs plugged into switches for processing.

The second requirement of service chaining implies off-line NF placement.

PLayer can only use inflexible Layer-2 mechanisms (e.g, spanning tree) to deliver the traffic between NFs. The SDN-based solution StEERING [31] which allows fine-grained per-subscriber and per-application service chains can be applied to both L2/L3 networks. Like PLayer, StEERING also places the NFs in fixed off-line points and StEERING steers traffic to these NFs in order through centrally configuring the forwarding rules on the SDN switches. The new paradigm of SDN allows the network operators to easily implement the flexible traffic steering. Though the ability of flexible traffic steering enables the freedom to place NFs anywhere in the network without violating the requirements of service chaining, the placement strategy impacts the user performance. StEERING provides one placement strategy with the objective of minimizing the average time for the subscribers' traffic to traverse through all required NFs, in which the traffic delay is calculated as the sum of shortest-path delays between consecutive NFs. Since the search space is huge, StEERING adopts an approximation heuristic, rather than exhaustive search, to efficiently solve this placement problem.

III. NFV ORCHESTRATION AND PLACEMENT

The new paradigm of NFV enables the great flexibility as for the deployment, instantiation, configuration, and termination of virtual NFs on demand. The virtualization nature of NFV fosters a number of fabulous features which are not possible in conventional hardware NFs: cheap elastic virtual NFs, dynamic horizontal scaling and fast state migration, etc. Recently, a bunch of NFV frameworks have been proposed to efficiently manage the hardware resources. Different design spaces of these NFV frameworks lead to their own appropriate placement strategies. We compare different NFV frameworks and summarize how the design space affects the placement in TABLE .II[1]. Despite the variety, all existing NFV frameworks place VNFs at off-line servers.

A. Thread-based Framework

Consolidating software NFs into generic hardware platforms enables resource multiplexing as well as components reuse. CoMb [25] is a pioneer NFV framework, which can host modular VNFs. These NFs are contained in threads. In order to reuse software elements, the incoming packets are first processed by elements shared by all VNFs, such as protocol parser and session reconstructor. After that, these packets are classified and the policy shim layer then sends them to the corresponding thread-based NFs for further processing. CoMb assumes that the resource footprint of each thread is proportional to its workload, whereas alternative containers (e.g., virtual machine) introduce non-trial fixed overhead.

1) Monolithic Consolidating: Using the lightweight thread as the container stems great benefits with regards to service chaining and NF placement. The benefits are two-fold.

- CoMb [25] consolidates all the required NFs in a single thread for each traffic class, which drastically eases the management of the service chain. Moreover, such monolithic consolidating obviates the need for considering mangling NFs, which actively modify the traffic header or the payload, such as NATs, web proxies. Otherwise, a packet may be not recognized after mangling NFs, and thus we do not know which NF to process it next.
- One or more monolithic NFs are instantiated on the servers in the routing path. With centralized network-wide view, the processing workload are spatially distributed in order to balance the load.

Given proliferated service chains and moderate number of traffic flows, a large number of monolithic NFs are needed to scatter around in the network. If the thread was not lightweight, there should be not enough resources to host them.

2) Cross-border On-path Placement: A service chain may contain location dependent NFs which have preferred locations in the network. For example, web proxies would better to be closed to the clients in order to optimize user experience and reduce bandwidth consumption. On the other hand, some NFs that encrypt the traffic may inflate the bandwidth utilization and these NFs are suggested to be placed near the destination. It is location dependency that CoMb [25] falls short in.

To this end, MIDAS [1] is proposed as an extension to CoMb, allowing each traffic flow to be processed by NFs located at different machines along its routing path, as opposed to CoMb, where all required NFs are consolidated into a single machine. MIDAS is especially useful in the Internet, where location dependency impacts the performance significantly. Internet Service Providers (ISPs) have started deploying micro-datacenters, where CoMb servers can be hosted. NF location dependency may require the NFs to be provided by multiple NF providers (NFPs). MIDAS resolves the problems of CoMb

[1]In this table, CSamp does not have service chains, so it is not necessary to consider order preserving and mangling NFs.

NFV form	NFV framework	Placement strategy	On path?	Mangling NF?	Location dependency?	Order preserve?
Thread-based	CoMb [25]	Monolithic consolidating	✓	✓	x	✓
	MIDAS [1]	Cross-border on-path placement	✓	x	✓	✓
VM-based	E2 [22]	Path-loosely-controlled placement	x	x	x	✓
	Statos [10]	Path-loosely-controlled placement	x	✓	x	✓
	VNP-OP [4]	Path-tightly-controlled placement	x	✓	x	✓
	PACE [18]	Unordered placement	x	x	x	x
Other Forms	Slick [3]	Partial consolidating	✓	x	✓	✓
	CSamp [27]	On-path distributed placement	✓	N/A	✓	N/A
	ETTM [8]	Monolithic consolidating	✓	✓	x	✓

TABLE II
COMPARISON BETWEEN DIFFERENT NFV FRAMEWORKS

servers discovery and on-path processing establishment across multiple NFPs. Once the CoMb servers on the routing path are discovered, MIDAS assigns required NFs to CoMb servers hosted by multiple NFPs in two stages. The first stage is to compute each NFP's responsibility considering utilization balancing across NFPs and location dependency. The heuristic for the second stage is used to select CoMb servers within the scope of a single NFPs while respecting the order.

B. VM-based Framework

Isolation is an indispensable property for both performance and security, especially in the context of multi-tenant clouds, where the resources are shared among cloud tenants. Not for isolation, the busy VNFs may consume much more resources than other co-residing VNFs, leading to starvation and violation of the stringent SLO. Moreover, isolation is perceived as the key to cloud security. As a result, most NFV framworks contain NFs in virtual machines (VMs) for isolation. A running VM reserves a fixed amount of hardware resources (including CPU, memory, hard disk, etc.) regardless of its workload. Consequently, to the best of my knowledge, it is assumed in all VM-based frameworks that one instance of VNF consumes fixed resources and possesses fixed processing capacity. The heavyweight VM prohibits installing all the required VNFs on the routing path for every traffic flow for two main reasons:

- The VMs consumes substantial resources.
- The VM booting time is prohibitively long (tens of seconds). VM installation happens at a much coarser time-scale than traffic dynamics.

Instead, all these VM-based frameworks steer traffic flows between VNFs to fulfill service chaining. On the other hand, these VM-based frameworks are different from hardware NFs in that they can decide where to place VNFs in runtime, hence more freedom to optimize the performance&cost.

To achieve the best performance benefits from smart NF placements, they would better be augmented by other mechanisms to incorporate traffic dynamics, such as flow distribution (small time-scale load balancing), dynamic scaling in/out (resource efficiency), etc. We briefly discuss this topic in Sec. III-B4.

Flowstream [11] is the first VM-based framework, which was published as early as 2009. The authors of Flowstream noticed the narrowed performance gap between commodity switches/servers and customized high-end switches/hardware NFs, and successfully predicted the rise of commodity hardware to process traffic. NFs are consolidated to servers in the form of VMs. The traffic flow routes depend on the fined-grained control of SDN. Nevertheless, Flowstream is just a high-level design without concrete implementation.

1) Path Loosely Controlled Placement: There are existing placement algorithms, resolving the problem of mapping VMs onto physical machines in the cloud. Some VM-based frameworks directly rely on these placement algorithms where VNFs are viewed as ordinary VMs.

The baseline VM placement algorithms only consider server hardware resources, such as Least Used Host placement algorithm and Least Busy Host placement algorithm [7]. The rapid growth of distributed analytics (e.g., MapReduce) and time-sensitive applications raise a big concern about network performances. Hence, network-aware placement algorithms are proposed. E2 [22], a VM-based NFV framework, relies on them to minimize intra-server traffic when mapping each VNF instance to a particular server, because inter-server communication incurs less latency and bandwidth between servers. E2 [22] targets Central Offices (COs), where VNFs are clustered and interconnected by a Layer-2 network. COs are very common in the backhaul of mobile networks to provide a wide variety of services. The initial NF placement involves four steps. The first step is to merge individual service chains into a single policy graph (pGraph), in order to identify the relationship between all NFs. The second step is sizing. E2 determines the number of instances for each NF in the pGraph given the estimate of the load on a NF and the per-instance capacity. The workload of a NF will be evenly distributed among the its VNF instances. The third step is converting the pGraph to an iGraph, or the "instance graph". In the iGraph, each node represents an instance of NF and the edge weight captures the traffic demand between two VNF instances. The final step is the actual instance placement with the objective to minimize inter-server traffic. The optimization problem is modeled as graph partition, which E2 solves by an iterative local search algorithm based Kernighan-Lin heuristic.

Statos [10] is another network-aware NFV platform being able to correctly forward traffic in face of mangling NFs (we have discussed them in Sec. III-A1). Statos exhaustively

enumerates the possible downstream pathes for each mangling NF in the policy graph. The mangling NF are cloned as many times as the number of downstream pathes, with each clone being responsible for one downstream path. By simply cloning the mangling NFs when there is ambiguity, Statos ensures the correctness of service chaining even though this transformation potentially increases the number of needed VNF instances. Finally, Statos employs network-aware VM placement algorithms (such as TMVPP [20]) to avoid congestion.

2) Path Tightly Controlled Placement: In existing VM placement algorithms, it is assumed that the routing path between each VM-pair is determined merely by the physical locations of VMs. Even though sometimes traffic engineering is allowed, the network operators still cannot explicitly control the routing path. With the tremendous convenience and flexibility bought by SDN, some NFV frameworks control the routing pathes between NF pairs, enabling joint placement and routing optimization for better performance.

VNP-OP [4], is an optimization problem referred as Virtualized Network Function Orchestration Problem, trying to minimize the cost (which can be further broken down into deploy cost, energy cost and cost of forwarding traffic) , penalty for SLO violation and resource fragmentation by carefully placing VNFs as well as forwarding traffic through best available pathes. This optimization is formulated as an Integer Linear Program (ILP) taking into account server/link capacity constraints and service chaining. VNP-OP can be reduced to trans-shipment problem, and therefore is NP-hard. No doubt, a heuristic is proposed to solve this problem.

Charikar et al. first theoretically analyzed this joint optimization problem by recognizing it as a new extension to multi-commodity flow problem [5].

3) Unordered Placement: The aforementioned VM-based NFV frameworks all strictly preserve the sequential order specified by the service chain. However, sometimes, the service chain can be partially ordered or even completely unordered. For example, from a security stand point, there is little difference to put a passive monitor before or after a DPI. PACE [18] addresses the VNF placement for unordered service chains in cloud, with the objective to satisfy as many tenants' requests as possible.

PACE [18] envisages the feasibility of enforcing policies in cloud using traditional L2 spanning tree, L2 shortest path, Openflow-based routing, source routing and pswitch-based routing [15]. In PACE, both offline and online optimization problems are in the form of ILP, which can be efficiently solved by leveraging LP relaxation.

4) Incorporating Dynamics: Statos [10] uses the end-to-end application performance as an indicator of VNFs being the performance bottleneck, in the face of which Statos employs a combination of three mechanisms to resolve the bottleneck at increasingly coarser time-scale: flow distribution, horizontal scaling and instance migration.

E2 [22] provides hooks for VNFs to report their instantaneous load and it also measures the processing latency of each VNF instance to detect overload. In case of overloading, dynamic scaling out occurs.

Sometimes on-going flows may be required to re-distributed to other VNFs to balance the workload. However it is problematic if the VNF is stateful. The expensive live migration of the whole virtual machines [6] may happen at a much coarser time-scale than the elastic provision. OpenNF [14], rather than copying the whole virtual machine, only manages associated VNF internal states and network forwarding states during flow re-distribution. According to OpenNF, VNF states are classified into three categories based on the their scope: per-flow, multi-flow and all-flow. NFs are required minor modification such that they can expose APIs to export/import states.

IV. Other Forms of Network Functions

Even though the NF frameworks described in this section still belong to NFV, they are so special that we use a separate section for them.

A. Element-based Framework

The NFs we discuss above, either hardware, thread-based or VM-based, which vertically integrate basic modules (e.g., protocol parse, encryption/decryption), can independently complete a particular packet processing task. On the other hand, Slick [3] takes another way which allows the operators to implement NFs as a chain of lightweight functions (e.g, checksums) which are placed across the network for reuse. These lightweight functions, referred as elements in Slick, are able to be reconfigured at runtime.

The placement consists of two steps. The first step is to consolidate elements if necessary. The Slick controller decide whether to consolidate contiguous elements onto a single machine or distribute them across multiple machines based on each element's inflation factor, which is defined as $log(f_{out}/f_{in})$ where f_{out} and f_{in} denote its output and input traffic volumes respectively. To reduce the link bandwidth consumption, it is intuitive to place elements with negative inflation factors near the sources and others near the destination. Therefore, the controller breaks the element list into sub-lists, each of which is placed in a single machine, such that the bandwidth consumption is minimized. The second step is to place consolidated elements. The placement strategy is that consolidated elements with negative (positive) inflation factors are placed onto the node on the longest common path of all traffic closed to sources (destinations). The rest consolidated elements are placed in a way such that the average path length for all traffic is minimized. Unlike CoMb [25] using monolithic consolidating, the placement strategy of Slick is partial consolidating.

B. Distributed NFs

Nowadays, most NFs are resided at certain locations in the network, each responsible for a static portion of the flow space. With ever growing network scale and traffic volume, these centralized NFs become the performance bottleneck. Moreover, steering traffic to them incurs non-trivial overhead.

CSamp [27] is a brave endeavour towards scalability by employing a system-wide approach to coordinate distributed NFs for fined-grained flow level monitoring. In CSamp, the monitor NFs are implemented as applications inside routers. To avoid duplicated sampling due to multiple monitor NFs on the routing path, CSamp uses a hash-based packet selection to achieve the coordination while obviating the overhead of explicit communication. When these distributed NFs are placed, a network-wide goal, such as maximizing the coverage, is achieved by distributing the workload to monitor NFs across the network while respecting the resource constraints.

Apart from distributed monitoring, distributed redundant elimination [2] and distributed IDS/IPS [26] have already been implemented.

C. Host-based Framework

The NFs we have talked about are all residing inside the network, and that is why we also call them "middleboxes" too. In order to exploit the increasing computing power shipped with advanced technologies (such as multi-core architecture) in commodity servers, ETTM [8] are proposed to radically move the NFs onto participating endpoints to provide a reliable and trustworthy NFV framework. The NFs running on the endpoints are contained in VMs in order to isolate NFs and end-user applications, and they are logically controlled by a central controller. Though endpoints are notoriously famous for being insecure, ETTM benefits from the trusted computing module (TPM) available in many current computers to provide NFs with the Attested Execution Environment (AEE). These distributed NFs residing in endpoints use Paxos [17] distributed consensus algorithm as a lever to provide consistent decisions, also as a way to provide fault-tolerance and reliability. Since the traffic is only processed at endpoints, similar to CoMb [25], the required NFs in the service chain are piled at the endpoints. Even though the NFs are contained in VMs, the abundant resources in each endpoint are enough to support its own service chain.

V. Discussions

Can different forms of NFs be mixed together?

The hybrid environment has already been discussed in literature. VNF-P [21] focuses on a hybrid scenario where the services are provided by both fast dedicated hardware NFs (for baseline workload) and flexible on-demand VNFs (for burst workload). VNF-P uses a similar method to what presented in Sec. III-B2 to place VNFs and to steer traffic. E2 [22] introduces a high-performance and flexible data plane where Click [16] modules (e.g., classifier and TCP reconstructor, etc.) are embedded to accelerate the packet processing. The VNFs in E2 need to explicitly call the API exported by the dataplane to achieve the benefit from such data path. In this sense, E2 is actually a hybrid of VM-based and element-based framework.

Can VM-based framework provide on-path placement?

On-path placement is desired for it introduces minimal inference to traffic engineering and obviates the need for complicated forwarding rules in routers. However, the overhead of VMs hinders on-path placement in existing VM-based NFV frameworks. ClickOS [19] opens this possibility, even though it is not explicitly stated in the original paper.

ClickOS [19] is meant to be built as a platform specially optimized for NFV to make fast deployment possible. ClickOS is a Xen-based virtual machine running Click [16] configurations in tailored guest OS environment. Since ClickOS adopts Click [16] as the programming abstraction, ClickOS only preserve the minimal code to support the running of Click. The ClickOS image size is further reduced by implementing various back-end drivers in the Xen host OS and only keeping a uniform front-end driver in the ClickOS virtual machine. As a result, the ClickOS image size is as small as $5MB$ and the ClickOS virtual machine can boot in 30 milliseconds. Due to the small memory footprint, a server can easily support up to hundreds of ClickOS VMs. Enhanced with high-performance inter-VM communication (e.g, NETVM [13]), it is possible to pile ClickOS VMs in a single server on the routing path to provide services.

Can we outsource NFs?

Sherry et al. implemented APLOMB [28], a service to oursource NFs in cloud without significant degrading of performance. By leveraging DNS-based redirection, the incoming traffic of an APLOMB customer is routed directly to the cloud first (for outsourced NF processing), then to the customer. The latency due to the indirection is negligible, because of the violation of triangle inequality in inter-domain routing and that cloud providers usually have a good connection to the Internet.

Where do service chains come from?

Service chains are one of the most important inputs when we run placement algorithms. The aforementioned work focuses on different methods to implement service chaining, whereas PGA [23] tries to answer an orthogonal question: how does the policy chain be specified. In the enterprise network, multiple entities independently declare their own network policies. PGA is a fast automation tool to compose independent network policies into a global-scope one for each traffic class. By leveraging a graph structure, PGA allows the detection and resolving of the conflict, if there is any.

VI. Challenges and Future Work

VM-based NFV frameworks are studied most recently due to the mature technologies it adopts from tens years of virtual machine development. However, sometimes directly applying them to NFV would result in sub-optimal performance.

To be more specific in the area of VNF placement, the traffic pattern between production VMs are different from that of VNF. For example, several VM placement algorithms, like TMVPP [20], assume that the traffic matrix is known before VM placement and the traffic bandwidth consumption between each VM pair is stable. These assumptions do not hold in the context of NFV anymore, where the traffic become unpredictable. For instance, people now begin to build security infrastructure based on NFV [9] and the intra-VNF matrix varies vigorously if DDoS attacks are mounted, which are

unpredictable in general. Moreover, the dynamic scaling in/out of VNFs compels efficient online placement algorithms. Fortunately, NFV also offers new opportunities for performance optimization. Even though we cannot get the traffic matrix in NFV context, there are correlations between intra-VNF traffic. For example, suppose we have an IDS with steady rate of incoming traffic. If a flow is suspicious to be malicious, the traffic of that flow would be sent to another network function (say a DPI) for further analysis; otherwise, the traffic would be sent to the firewall. Clearly, the traffic rate to the firewall and that to the DPI are negatively correlated. Given such information, the NFV framework could strategically place these two kinds of VNFs nearby. Since they are less likely to encounter the peak workload simultaneously, the hardware resources are safely and efficiently multiplexed.

The mangling NFs imposes a big challenge which does not exist in traditional VM placement. Current solutions mainly rely on the global ID encoded in each packet to ensure correct forwarding in face of mangling NFs. These global IDs are difficult to aggregate, so that the forwarding rules in switches are inflated. Scalability is a big issue for such solutions. Stratos [10] is a good start to exploit the power of VNF placement to address mangling NFs, even though in this solution more VNFs are initiated than necessary. If using global ID, we could try to place mangling NFs near the destinations so that upstream switches on the routing path avoid rules related to the global IDs. We envision that the combination of the two mechanisms, global ID and VNF placement to name, is worthy further investigation to incorporate mangling NFs in a more scalable and efficient manner.

VII. Conclusion

In this paper, we provide a comprehensive survey regarding the placement issues of both conventional hardware network functions and currently popular virtualized network functions. We survey different network function frameworks and present how their design considerations affect the network function placement strategies. We conclude with a discussion of other topics closely related to network function placement and possible future research challenges and opportunities associated with it.

References

[1] A. Abujoda and P. Papadimitriou. Midas: Middlebox discovery and selection for on-path flow processing. 2015.

[2] A. Anand, V. Sekar, and A. Akella. Smartre: an architecture for coordinated network-wide redundancy elimination. In *Proc. of ACM SIGCOMM*, 2009.

[3] B. Anwer, T. Benson, N. Feamster, and D. Levin. Programming slick network functions. In *Proc. of ACM SOSR*, 2015.

[4] M. Bari, S. R. Chowdhury, R. Ahmed, and R. Boutaba. Orchestrating virtualized network functions. *arXiv preprint arXiv:1503.06377*, 2015.

[5] M. Charikar, Y. Naamad, J. Rexford, and K. Zou. Multi-commodity flow with in-network processing. *Manuscript, www. cs. princeton. edu/˜ jrex/papers/mopt14. pdf*.

[6] C. Clark, K. Fraser, S. Hand, J. G. Hansen, E. Jul, C. Limpach, I. Pratt, and A. Warfield. Live migration of virtual machines. In *Proc. of the USENIX NSDI*, 2005.

[7] E. Clayman, S.and Maini, A. Galis, A. Manzalini, and N. Mazzocca. The dynamic placement of virtual network functions. In *Proc. of IEEE NOMS*, 2014.

[8] C. Dixon, H. Uppal, V. Brajkovic, D. Brandon, T. Anderson, and A. Krishnamurthy. Ettm: a scalable fault tolerant network manager. In *Proc. of USENIX NSDI*, 2011.

[9] S. K. Fayaz, Y. Tobioka, V. Sekar, and M. Bailey. Flexible and elastic ddos defense using bohatei. In *USENIX Security*, 2015.

[10] A. Gember, A. Krishnamurthy, S. S. John, R. Grandl, X. Gao, A. Anand, T. Benson, V. Sekar, and A. Akella. Stratos: A network-aware orchestration layer for virtual middleboxes in clouds. *arXiv preprint arXiv:1305.0209*, 2013.

[11] A. Greenhalgh, F. Huici, M. Hoerdt, P. Papadimitriou, M. Handley, and L. Mathy. Flow processing and the rise of commodity network hardware. *ACM SIGCOMM CCR*, 39(2), 2009.

[12] R. Guerzoni et al. Network functions virtualisation: an introduction, benefits, enablers, challenges and call for action, introductory white paper. In *SDN and OpenFlow World Congress*, 2012.

[13] J. Hwang, K. K. Ramakrishnan, and T. Wood. Netvm: high performance and flexible networking using virtualization on commodity platforms. In *Proc. of USENIX NSDI*, 2014.

[14] A. Jacobson, R. Viswanathan, C. Prakash, R. Grandl, J. Khalid, S. Das, and A. Akella. Opennf: enabling innovation in network function control. In *Proc. of ACM SIGCOMM*, 2014.

[15] D. A. Joseph, A. Tavakoli, and I. Stoica. A policy-aware switching layer for data centers. In *Proc. of ACM SIGCOMM*, 2008.

[16] E. Kohler. *The Click Modular Router*. PhD thesis, Massachusetts Institute of Technology, 2000.

[17] L. Lamport. Paxos made simple. 2001.

[18] L. E. Li, V. Liaghat, H. Zhao, M. Hajiaghay, D. Li, G. Wilfong, Y. R. Yang, and C. Guo. Pace: policy-aware application cloud embedding. In *Proc. of IEEE INFOCOM*, 2013.

[19] J. Martins et al. Clickos and the art of network function virtualization. In *Proc. of USENIX NSDI*, 2014.

[20] X. Meng, V. Pappas, and L. Zhang. Improving the scalability of data center networks with traffic-aware virtual machine placement. In *Proc. of IEEE INFOCOM*, 2010.

[21] H. Moens and F. De Turck. Vnf-p: A model for efficient placement of virtualized network functions. In *Proc. of IEEE CNSM*, 2014.

[22] S. Palkar, C. Lan, S. Han, K. Jang, A. Panda, S. Ratnasamy, L. Rizzo, and S. Shenker. E2: A framework for nfv applications. In *Proc. of ACM SOSP*, 2015.

[23] C. Prakash, J. Lee, Y. Turner, J. M. Kang, A. Akella, S. Banerjee, C. Clark, Y. Ma, P. Sharma, and Y. Zhang. Pga: Using graphs to express and automatically reconcile network policies. In *Proc. of ACM SIGCOMM*, 2015.

[24] S. Raza, G. Huang, C. Chuah, S. Seetharaman, and J. P. Singh. Measurouting: a framework for routing assisted traffic monitoring. *IEEE/ACM Transactions on Networking*, 20(1):45–56, 2012.

[25] V. Sekar, N. Egi, S. Ratnasamy, M. K. Reiter, and G. Shi. Design and implementation of a consolidated middlebox architecture. In *Proc. of USENIX NSDI*, 2012.

[26] V. Sekar, R. Krishnaswamy, A. Gupta, and M. K. Reiter. Network-wide deployment of intrusion detection and prevention systems. 2010.

[27] V. Sekar, M. K. Reiter, W. Willinger, H. Zhang, R. R. Kompella, and D. G. Andersen. csamp: A system for network-wide flow monitoring. In *Proc. of USENIX NSDI*, 2008.

[28] 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. 2012.

[29] J. Sherry and S. Ratnasamy. A Survey of Enterprise Middlebox Deployments. Technical report, EECS, UC Berkeley, 2012.

[30] K. Suh, Y. Guo, J. Kurose, and D. Towsley. Locating network monitors: complexity, heuristics, and coverage. *Computer Communications*, 29(10), 2006.

[31] Y. Zhang, N. Beheshti, L. Beliveau, G. Lefebvre, R. Manghirmalani, R. Mishra, R. Patneyt, M. Shirazipour, R. Subrahmaniam, C. Truchan, et al. Steering: A software-defined networking for inline service chaining. In *Proc. of IEEE ICNP*, 2013.