Attachment A: Virtual Function Placement and Chaining in Network Function Virtualization

Traditionally, network functions are purpose-built in proprietary appliances, known as middle-boxes [1], which are used to enhance network security (e.g., Intrusion Detection System (IDS)), and/or network performance (e.g., Load Balancers that balance the traffic load). Network traffic flows are directed through middle-boxes of respective functions in a given order based on the predetermined policies. As an example in Fig.1, the traffic from working-from-home employees to the company server needs to flow through three middle-boxes: an NAT (Network Address Translator), a Firewall, and an IDS for enhanced security. If the access is originated from office employees, however, the traffic only goes through a Firewall due to less security concerns. In a typical enterprise network, middle-boxes can be in the same order of magnitude as rudimentary network components (i.e., switches and routers) [2].

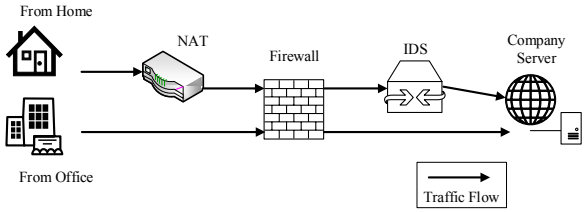

Fig. 1. Example of Chained Services through Middleboxes

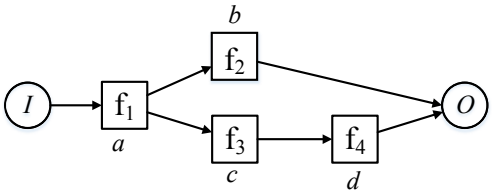

Fig. 2. Forward Graph

This type of purpose-build middle-boxes leads to huge challenges to the network management. One one hand, given the proprietary feature of middle-boxes, the configuration/repairment of middle-boxes requires ad-hoc expertise which varies among different middle-boxes. One the other hand, given the wide deployment, middle-boxes become a significant source of network failures in both the dimensions of scale and frequency. For instance, in 2012, a bug in a load balancer update at *Google* caused the service degradation for 8% to 40% of *Gmail* customers [3]. According to a recent field study, middle-boxes contribute to 43% of high-severity incidents in data centers [4].

Recently, ETSI (European Telecommunications Standards Institute) advocates a proposal, namely network function virtualization (NFV), to address above issues [5]. A virtual function is a *software* instance of a middle-box's functionality. With NFV, the function of a middle-box is separated from proprietary appliances, and deploying a new middle-box (say a Firewall) is transformed to running a virtual function (i.e., a Firewall software) on any commodity hardware. As an analogy, one can image that *iPhone*'s functions are virtualized and hence can be run on a laptop. NFV bears two major advantages comparing to middle-boxes: first, generic hardware can be adopted to run a variety of functions, which is more cost-efficient and requires no ad-hoc expertise in management; second, given the software nature, middle-boxes can be created/terminated in a flexible manner. Repairing a virtual function can be easily done by robooting the software. With NFV, network traffic flows need to pass through a group of virtual instances of customized functions (instead of middle-boxes) in a given order. Technically, the set of virtual instances are referred to as a *service function chain* [6]. Different from middle-boxes, the virtual function instances are not fixed and can be decided dynamically. Given a service function chain, it is critical to decide where to place the required virtual functions, i.e, the *placement* problem, and how to steer the traffic to pass through the placed functions, i.e. the *routing* problem. Given the space limitation, in the following, we use a simple example to briefly explain these two problems and discuss our plan of study.

Placement of Service Functions: In NFV, network traffic flow needs to be steered through a group of virtual functions (e.g., NAT, Firewall, and IDS in Fig. 1) in a given order. This group is generally represented as a directed graph, known as the *Forward Graph* [6]. Figure 2 shows an example of a forward graph, where a service chain request consists of four virtual function nodes (i.e., a to d with function f_1 to f_4, respectively), and the entry point I and exit point O (the entry point and exit point are generally fixed at particular locations). The directed link (i.e., the arrow) indicates the order of the function chains. For instance, the link from Node a to Node b indicates that function f_1 has to be applied to the flow before f_2. Also, each function requires a number of computing resources, and each link requires a number of bandwidth resources. For simplicity, we assume that each function in Fig. 2 requires 100 units of computing resources, and each link requires 5 Gb/s of bandwidth resources. The required virtual functions are instantiated at respective commodity servers of a physical network. An example of physical network is shown in

Fig. 3, where there are six commodity servers that are connected by eight links. We assume that all the nodes have 100 units of computing resources except Node B, and all links have 10 Gb/s of bandwidth except Link A-B.

Finally, the problem of service function chain placement answers such a question: *given a group of forward graphs (e.g., Fig. 1), where should we instantiate the required functions in the physical network (e.g., Fig. 3)?* An example of placement is shown in Fig. 4 where functions f_1 to f_4 are created in Nodes A, D, C, and E, respectively (assuming that the entry and exit points are fixed at Nodes A and F, respectively). Node B with only 50 units of computing resources are not used as each function needs 100 units of computing resources. Although this example appears to be simple, the placement decision can be very challenging in reality for reasons that will be discussed below. In fact, the placement problem is an NP-Complete problem, which is a type of problem that is extremely hard to solve for large instances.

Routing of Service Functions: The traffic flow also has to be steered to pass through the placed functions in the desired order (i.e., the chaining). One needs to further answer the *routing* problem: *how is the network flow routed from one function to the next function (starting/ending at the entry/exit point, respectively) to follow the order specified by the forward graph?* An example of the routing decision for the example above is shown in Fig. 5. For instance, as function f_1 is followed by f_2 in the forward graph in Fig. 2, the flow travels the physical nodes in the sequence of A-C-B-D to pass functions f_1 and f_2 on Node A and D in order. Note that as Link A-B lacks bandwidth, the sequence A-B-D is not used. Although the routing problem appears to be dependent on the placement, in practice, these two problems need to be addressed jointly. For instance, if the placement is done without the consideration of routing, one may lead to an impasse that no bandwidths are available to connect the placed virtual functions.

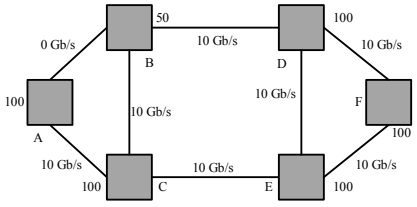

Fig. 3. An Example of Physical Network

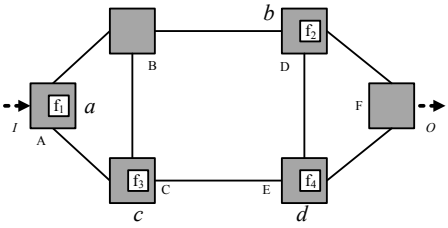

Fig. 4. An Example of Function Placement

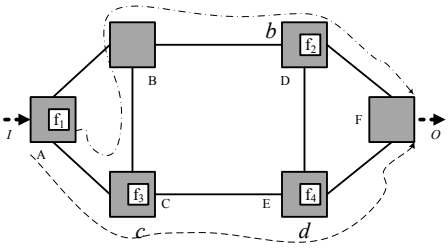

Fig. 5. An Example of Function Routing

Plan of Study: The difficulty of the virtual function placement and routing lies on a few facts: first, one has to cope with a group of forward graphs (only one above) that competes for the resources; second, multiple functions can co-exist at the same server, and functions may be incompatible; third, to achieve the optimal decision, the placement and routing have to be jointly considered. Given the NP-Completeness of this problem, in this project, we will present an *optimal* solution based on the Integer Linear Programming (ILP) transformation that jointly addresses the placement and routing problem. Logically, the ILP model is equivalent to the latter, but a state-of-the-art ILP solver software - *IBM CPLEX*, can obtain an optimal solution for the former. We will also seek time-efficient near-optimal solutions based on approximation algorithms, which have a guaranteed near-optimal performance. Overall, we expect the solutions found in this project will serve a basis that enables a virtualized enterprised network that embraces network function virtualization.

REFERENCES

[1] D. Joseph and I. Stoica, "Modeling middleboxes," *IEEE Network*, vol. 22, no. 5, pp. 20–25, 2008.
[2] J. Sherry and S. Ratnasamy, "A survey of enterprise middlebox," *Tech. Rep. UCB/EECS-2012-24, University of California, Berkeley,*, 2012.
[3] "Why gmail went down: Google misconfigure load balancing servers(updated)." [Online]. Available: http://arstechnica.com/
[4] R. Potharaju and N. Jain, "Demystifying the dark side of the middle: A field study of middlebox failures in datacenters," in *Proceedings of the ACM IMC*, 2013, pp. 9–22.
[5] NFV, "http://www.etsi.org/technologies-clusters/technologies/nfv," 2017.
[6] RFC7498, "Problem statement for service function chaining." [Online]. Available: https://datatracker.ietf.org/wg/sfc/documents/