Name Enhanced SDN Framework for Service Function Chaining of Elastic Network Functions

Sameer G Kulkarni*, Mayutan Arumaithurai*, Argyrious Tasiopoulos‡, Yiaonis Psaras‡,
K.K. Ramakrishnan†, Xiaoming Fu*, George Pavlou‡
*University of Göttingen, Germany, ‡University College London, †University of California, Riverside.

***Abstract*—Middleboxes have become an integral part of Internet infrastructure, providing additional flow processing for policy control, security, and performance optimization. Network Function Virtualisation (NFV) proposes the deployment of software-based middleboxes on top of commercial off-the-shelf (COTS), enabling the dynamic adjustment of Virtual Network Functions (VNFs), both in terms of instance numbers and computational power. The performance of Data center and Enterprise networks depend strongly on efficient scaling of VNFs and the traffic load balance across VNF instances. To this end, we present Name enhanced SDN framework for service function chaining of elastic Network functions (NSN) that extends the Function-Centric Service Chaining (FCSC) with load balancing functionalities to achieve efficient network utilization while reducing the switch flow rules by 2-4x compared to traditional SDN approaches.**

I. Introduction

Software Defined Networking (SDN) enables to realize the policy enforcement by providing greater flexibility and control in steering the packets through desired function chain. With logically centralized controller, it is easier to enforce heterogeneous policies for different flows and to steer the traffic across the network. In addition, with the global view of network topology, it is easier to monitor the resource utilization. Network Function Virtualization (NFV) has caused the paradigm shift towards deploying the soft middleboxes that provide flexible realization of network services with greater cost optimization [1]. SDN and NFV greatly augment to provide flexible and dynamic software-based network environment. On the other hand, Information-Centric Networks (ICN) and Named Data Networking (NDN) architectures introduce the naming layer to the network architecture that decouple the content/name from the location. This offers greater flexibility in routing the flows based on service types, without actually knowing the exact location in the network.

Traffic dynamics often trigger for reallocation and reconfiguration of network resources. In case of high demands, some resources end up being over-utilized, resulting into higher latency and SLA degradation, while on other occasions, end up being underutilized. In such circumstances, in order to meet the performance and energy objectives, the NF instances (NFIs) need to be dynamically instantiated or decommissioned or even relocated/migrated. However, to make it happen, several key decisions need to be made in terms of knowing when to instantiate, decommission or migrate the instance, which network instances need to be scaled, where in the network to place the instances and how to redistribute the load among the available instances. Several recent works [1], [2], [3] have tried to address these aspects within the hood of traditional or SDN framework.

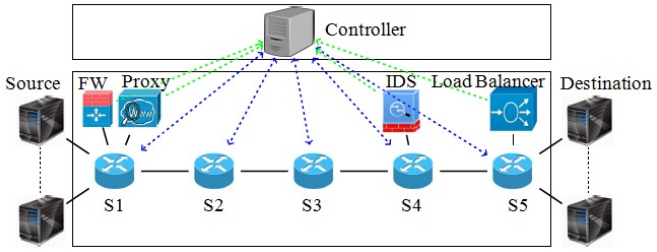


Fig. 1: NSN Architecture

FCSC [4] exploits the benefits of NDN in combination with SDN to provide a more flexible, scalable and reliable framework to realize service function chaining. However, it falls short of incorporating a reliable mechanism for applying load balancing over NFIs. Load balancing is fundamental to ensure efficient utilization of resources and to meet the SLA requirements. Herein, we present **N**ame enhanced **S**DN framework for service function chaining of elastic **N**etwork functions(*NSN*) that exploits named service instances and compliments the SDN framework by providing the capabilities of efficient load balancing and elastic scaling of VNF's via service instantiation, consolidation, while supporting flow redirections for achieving higher VNF utilization.

II. Related work

Slick [1] provides a programming model abstraction, where the SDN controller employs heuristic based approaches for estimating the dynamic placement, steering and consolidation of VNFs. However, load balancing is not explicitly addressed and the routing does not take into consideration the network load upon the load steering decisions. E2 [2] presents a NFV scheduling framework that supports affinity based NF placement while trying to minimize the traffic across switches as well as deploying dynamic scaling of NF instances. SIMPLE [3] primarily addresses the SDN based traffic steering approach that tries to optimize on the total rules. It relies on the ILP solver to provide online load balancing.

III. NSN Enhanced Architecture

We present the high level architecture and design of *NSN*, that incorporates name based network function instances and enhances SDN's capability to handle placement, routing and flow redirections.

A. Name based Routing

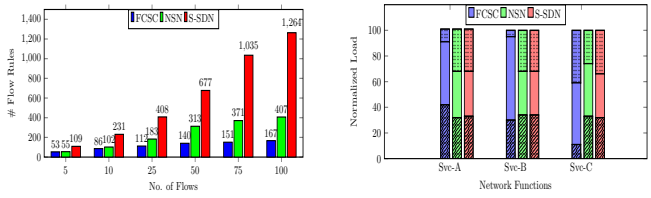
NSN enhances FCSC's name based routing mechanism to perform NFI based routing, wherein all the NFIs are uniquely identified by a name. Policy enforcement on the flows is performed by the controller by encoding the names of the sequence of network functions that are required for the flow to pass through. This aspect is essentially the concept of Information-Centric Networking, wherein the function that a flow requests is decoupled from the location where this function/service is going to be executed. That is, packets indicate through their headers the service function they require and the network is responsible for routing those packets towards the right location. This notion of location-independence can support real-time flow steering and redirection to dynamically instantiated/re-located services out of the box. Once a packet goes through a NF and the corresponding service is executed, the header is modified to remove this service from the chain of required services.

A key difference compared to current IP based SDN solutions is that the intermediate switches do not need to maintain per flow forwarding information or similar fine-grained forwarding rules, but only need to store forwarding information to reach the named instances. The switches that only route packets at specific service instances, have to keep a single forwarding rule for each service instance. Thus, the state maintained at intermediate routers is proportional to the number of instances and not to the number of flows. Moreover, these rules can be set in a proactive manner as soon as an NFI is instantiated, removed or re-located. Only the ingress switches and the edge switches connected to NFI that are servicing the flow, keep a per flow state forwarding table to ensure that the right labels (i.e., the next hops service instances) are placed on the flow's header.

Another advantage compared to existing IP based solutions is that when an NFI is removed or re-located, in the case of NSN, only the forwarding entries to these instances need to be changed, whereas in the case of current solutions, all entries pertaining to flows that are being serviced by this NFI needs to be modified. Similarly, in case of flow redirection, the proposed scheme provides a notion of atomic rule update as it needs one rule update. NFI node just needs to change the NFI tag to another instance.

B. VNF Placement and Elastic Scaling

NSN supports placement, instantiation, removal and relocation of instances to better support the dynamic requirements of flows. The NSN architecture can facilitate for different heuristic based placement mechanisms. NSN enables SDN framework to make quicker heuristic based placement decisions, and allows for finer and quicker course corrections to redistribute the load by either redirecting flows via other instances and/or by instantiating, removing or relocating the NFIs, and thereby enables to overcome the disadvantages of making such heuristic based decisions instead of time consuming and complex ILP [3] decisions. SDN controller can periodically monitor the utilization of the NFIs and network



(a) Total Switch rules for flows. (b) Load across different NFs.

Fig. 2: Evaluation Results indicating Flow rule optimization and load balancing characteristics.

link utilization, that can assist in identifying the optimal placement of NFI's that need to be dynamically instantiated. One such approach is to compute the optimal location by accounting available resources with greater affinity for the flows. On the same lines, the under-utilized NF instances can be decommissioned. NSN estimates the load on each instance based on the gathered link statistics, flows in the system, and the explicit notifications from instances. Based on the load thresholds, it can then dynamically instantiate and decommission specific network instances to ensure the instance utilization rates are kept within the optimal levels.

IV. Implementation and Evaluation

We implemented NSN, as set of modules on top of POX SDN contoller in Python (around 2500 lines of code). We use the Open vSwitch (2.4.0) for SDN switches and implemented custom network modules in linux using python scapy utility.

We evaluate NSN using the Mininet network emulator and use the data center tree topology. As in [3], each flow has a policy chain of 3 distinct NFs that originate and terminate at random nodes. Key focus of our evaluation is to measure and quantify the benefits in terms of overall number of flow rules required for steering and load across active network instances. We compare NSN with Standard SDN (S-SDN) that employs IP 5-tuple based rule setting and with FCSC that employs rule setting purely based on the named network functions.

Figure 2(a) depicts the total number of flow rules installed across all switches in the network for different number of flows. Figure 2(b) indicates the normalized average load observed across all the active instances of services A, B and C. We can see that NSN provides significant reduction in the total number of rules stored at switches compared to S-SDN. Moreover, NSN ensures load balancing capability identical to that of the S-SDN solution.

Acknowledgement

This work was supported by EU FP7 Marie Curie Actions CleanSky ITN project Grant No. 607584, and NICT EU-JAPAN GreenICN project Grant No. 608518.

References

[1] B. Anwer, T. Benson, N. Feamster *et al.*, "Programming slick network functions," in *ACM SOSR*. ACM, 2015.
[2] S. Palkar, C. Lan, S. Han *et al.*, "E2: A framework for nfv applications," in *ACM SOSP*, 2015.
[3] Z. A. Qazi, C.-C. Tu, L. Chiang *et al.*, "Simple-fying middlebox policy enforcement using sdn," in *ACM SIGCOMM*, 2013.
[4] M. Arumaithurai, J. Chen, E. Monticelli *et al.*, "Exploiting icn for flexible management of software-defined networks," in *ACM ICN*, 2014.