Prototype of an ICN Based Approach For Flexible Service Chaining in SDN

Mayutam Arumaithurai*, Jiachen Chen*, Eeran Maiti*, Xiaoming Fu* and K.K. Ramakrishnan†

* University of Goettingen, Germany, Email: arumaithurai.chen,maiti,fu@cs.uni-goettingen.de
† University of California, Riverside, CA, U.S.A. Email: kk@cs.ucr.edu

Abstract—With the widespread use of middleboxes in communication networks (e.g., for services such as Firewall, DPI, accounting, proxies, caching), additional processing beyond simple forwarding has become common. With the emergence of Network Function Virtualization (NFV), dynamically instantiating these functions becomes even more feasible. In addition, chaining such services for packet flows is key to efficiently delivering Internet services. In addition, Software Defined Networking (SDN) gives Internet Service Providers greater flexibility to provision such middlebox based services and route flows through them. However, with the dynamic instantiation of network functions, efficient and scalable use of these services requires separation of the services from their location. In this work, we implement and demonstrate an Information Centric Networking based solution that complements SDN for service chaining and provides benefits such as scalability, flexibility and reliability.

I. MOTIVATION

Supporting typical Internet services requires a large number of middleboxes to be deployed in the network, with packet flows going through a number of such functions. This capability of chaining services is commonly used (e.g., DPI, Firewall, proxies, policy and accounting services, caching etc.) for delivering information for commonly used applications (e.g., video (Youtube), email, social media (Facebook)). The advent of SDN and NFV promises to increase the flexibility and reliability of providing such services, since the middleboxes providing those services could be provided on demand at locations where it is most needed and efficient.

In our recent work [1], we argued that the existing approaches [2] [3] have a common problem of intricately coupling the management logic (“what”) with the node location (“where”) that limits flexibility and results in inefficiency. For example, when an SDN controller decides the functions a flow needs, it also decides the path the flow has to go through and sets up state on the intermediate switches. This results in limitations in scalability, reliability and flexibility, and have difficulty in adapting to the requirements of a large scale, dynamically changing middlebox set supported by NFV. We proposed Function-Centric Service Chaining (FCSC) [1] to enhance SDN with a service-name layer (built based on Information Centric Networking (ICN) [4]) to decouple the location of a particular network function instance from the identity of the function it provides. Such a decoupling facilitates the dynamic modification of the functions needed by a flow on the controller or the middleboxes (e.g., DPI, load balancer). This also enables switches to dynamically detect the load (popularity) of a certain function and accordingly instantiate/dispose of network function instances (co-resident with the switch or on some other node). The enroute function-based routing allows more dynamic use of the newly created

1http://www.projectfloodlight.org/floodlight/

instances and faster recovery from node/link failures.

The work in [1] described the concepts and demonstrated the benefits with the help of a simulation environment. This work extends [1] and reports on a prototype implementation of the service-name oriented FCSC approach. We use existing SDN utilities (albeit with minor modifications) in order to demonstrate that FCSC could indeed be realized. We describe the prototype environment, the choice of SDN utilities, the limitations of the prototype in achieving the advantages promised by FCSC and describe the example demonstration scenarios of service chaining that we intend to present. In the future, we plan to implement a full fledged service-name layer in order to realize the full benefits of the FCSC approach.

II. ARCHITECTURE

In order to realize the full potential of FCSC, a dedicated naming layer would be ideal. For reasons of expediency and convenience, we implemented FCSC using alternate layers and available utilities in SDN. The details of the implementation is presented below.

a) Controller: The availability of multiple instances of the same service (referred to by the same service-name) at different locations results in the presence of multiple-paths to the middleboxes hosting those services. E.g.,, in Figure 2, Service-name “A” is present in hosts connected to R3 as well as R1. Therefore the forwarding table in R2 could have two entries, one pointing to R3 and another to R1 for flows that require the service “A”. Another issue of service chaining is that the sequence of services required could result in the flow having to go in a loop before it reaches the destination. E.g., in Figure 2, a flow have to go to services: “DPI” followed by “A” and finally “B”. In this case, the path of the packets from the src to dst could look like: R6 -> R3 -> R2 -> R1 -> R3 -> R2 -> R5. We can observe that there is loop being created between R2, R1 and R3. Most of the OpenFlow controllers do not support the presence of loops and/or multiple paths. Amongst the ones that support the presence of loops and multipath (Floodlight and Ryu), the latter was used since
it supports OpenFlow versions till 1.4 while the former only supports OpenFlow version 1.0.

b) Service Naming: In order to perform service-name based forwarding and service chaining, we need the ability to use service-names (or labels) and also the ability to compose multiple functions together. For reasons of expediency and convenience, we used the approach of using MPLS-like labels, which also provides label stacking that enables us to compose multiple functions. A datagram can contain multiple MPLS labels that are stacked, and the labels can be removed, swapped or more labels can be added as needed. The limitations of this approach arises from the fact that the labels are stacked and current implementations follow the strict rule of not looking at more than one label in a packet. Therefore, the intermediate nodes are unable to make a forwarding decision based on all the services required for that flow and it is complicated to change the order or services provided. Moreover, since the switches do not have complete knowledge of all the services required by the flow, they are unable to make full use of their ability to instantiate services on demand.

c) Switch: The switches that were considered for the implementation were Open vSwitch\(^3\) and OFSoftswitch\(^4\). Open vSwitch provides MPLS functionality only in the userspace and does not support multiple MPLS label stacks. OFSoftswitch on the other hand supports multiple MPLS label stacks and is highly customizable. The CPqD version of OFSoftswitch 13 was used for this implementation, but with modifications in the source code to react to the availability (or absence) of new middlebox services attached to it by informing the controller and as well as neighboring nodes (via the controller or special UDP messages).

III. DEMONSTRATION

With the help of live demonstrations, we will present how we emulated a service-name layer using current tools (with modifications) available in the current SDN architecture and compare it to a standard SDN based service-chaining solution. We will use a simple custom topology that contains multiple middleboxes (see Figure 2) on Mininet\(^5\) (Version 2.2.0). R1-R6 are FCSC capable switches. N1-N4 and DPI provide functions A, B and DPI as noted in Figure 2. Src and Dst are the source and destination of a flow of interest. Ctrl is the central controller in an SDN solution. The link latency between the switches is 2ms and the latency between switches and the end-systems (middlebox, src, dst, control) is 10ms. The bandwidth on the link is 100 Mbps (large enough to support the flow). For standard SDN solution, a learning switch application was run on the controller for making routing decisions. The design and utilities used for evaluation are completely compatible with existing SDN solutions, hence the experiments can be replicated and or modified, or deployed readily based on requirements. The evaluation was carried out in OpenFlow protocol 1.3. Below we list the detailed demonstration scenarios:

Scenario 1: Forwarding rules for services offered by the network will be pro-actively set up. I.e., flow tables would be populated with paths to reach services offered in the network. Therefore, unlike in current SDN, new flows do not have to wait at the ingress till a packet is forwarded to the controller and a new forwarding path is set up by the controller (see Figure 1(a)).

Scenario 2: A middlebox providing DPI service will dynamically change the list of services a flow requires to demonstrate the changes in policies and we will demonstrate how FCSC is able to react quicker than existing SDN solutions (see Figure 1(b)).

Scenario 3: A middlebox providing a certain service will be switched off while packets of a flow are being processed by it to emulate failure and we will demonstrate how FCSC is able to quickly route to a new middlebox providing the same service (see Figure 1(c)).

Scenario 4: A new instance of a middlebox providing a service will be instantiated in a more convenient switch and we will demonstrate how an FCSC flow is able to quickly make use of that middlebox (see Figure 1(d)).

IV. CONCLUSION

The main aim of this work is to demonstrate how a service-name based forwarding engine was implemented for service chaining with the use of existing SDN utilities and solutions. With the help of simple examples, this work will demonstrate the benefits of such a service-name based forwarding engine at the SDN data plane for service chaining and compare it to existing solutions. In the future, we intend to implement a fully fledged naming layer based solution for service chaining in an SDN environment.

REFERENCES


\(^3\)http://openvswitch.org/\n\(^4\)https://www.sdncentral.com/projects/cpqdofsoftswitch/\n\(^5\)http://mininet.org/api/annotated.html