Ready-to-Deploy Service Function Chaining for Mobile Networks

Roberto Bifulco, Anton Matsiuk, Alessio Silvestro
NEC Laboratories Europe, Germany – E-mail: <firstname.lastname>@neclab.eu

***Abstract*—The dynamic composition of virtualized network functions promises to ease the introduction of new services in mobile networks, while reducing provisioning costs. This approach, usually called Service Function Chaining, has been the focus of several research efforts. However, current systems rely on new tunneling protocols and on network infrastructure changes. In this paper, we introduce CATENAE[1], an efficient Service Function Chaining system for mobile networks. Our system deals with today's network functions, including the ones that perform changes to the packet's header, without using any tunneling protocol and without requiring network modifications. Our approach is ready to be deployed in mobile networks, integrates seamlessly with legacy network management systems and introduces no overhead in the (virtual) network functions.**

I. Introduction

Network operators deploy network functions to enforce their policies and to provide additional services on top of plain connectivity [11]. Content caching, NAT, TCP optimization, video transcoding, HTTP header enrichment, are examples of such services. Despite their ubiquitous usage [32], network functions deployment is still performed by modifying the network topology. That is, network functions are hard-wired on the network traffic's path. The inflexibility and complexity of this approach is not acceptable when network functions are implemented by means of software running in virtual machines, as envisioned in the case of Network Function Virtualization (NFV) [6]. In fact, hard-wiring would hinder the benefits brought by the possibility of dynamically deploying virtual network functions (VNFs) on general purpose servers. Therefore, there is a growing interest on *Service Function Chaining* (SFC) systems [13], which enable the flexible deployment of network functions while guaranteeing their configurable and dynamic chaining.

In general, a SFC system assigns a network flow entering the managed network to a chain of functions, and steers the flow through the functions of such chain, according to the chain's functions ordering [22]. A number of challenges arise when addressing the design of a SFC system. First, assigning a network flow to its chain requires network traffic classification, an operation that is critical for the system scalability since it should be performed for all the handled traffic. Second, traffic forwarding should be performed according to the chain the traffic belongs to, instead of following the typical forwarding approach, e.g., based on IP routing. Third, network flows are usually bi-directional, that is, there is an upstream and a downstream direction and a network function, e.g., a firewall, may need to handle both of them. This requires to perform a coordinated classification of upstream and downstream flows, and the enforcement of symmetric paths for the two directions. Finally, network functions may have dynamic and opaque behaviors that modify the network traffic in unknown ways, which may introduce a need for traffic reclassification or even make the traffic unclassifiable [25].

To address these challenges, a number of SFC systems have been already proposed ([25], [7], [2], [34], [26]). However, they usually target green field or long term deployments. In fact, they require a number of changes either in the network hardware [25] or in the network functions [26], or in both [7]. In other cases, they require modifications to the network architecture [2]. Ready to deploy solutions, which don't require such changes, may instead not handle all the aforementioned challenges. For example, some SFC systems are unable to deal with opaque network functions actions [25], [34]. Regardless of the adopted solutions, the proposed systems address SFC in a general way, supporting a broad range of deployment scenarios without considering their specific properties and constraints. That is, they usually adopt a "one-size-fits-all" approach. While we recognize the intrinsic value of such a general solution, we also notice that not all the deployment scenarios share the same set of requirements, with the final result of SFC systems that provide unnecessary features for the specific scenarios in which they are deployed. At the same time, such systems usually fail to satisfy a critical requirement of many today's production deployments, i.e., the SFC solution should introduce minimum impact on the legacy infrastructures [3], [18].

In this paper, we argue that it is possible to simplify the implementation of a SFC system, by carefully tailoring the SFC solution to its specific deployment scenario. Our main contribution is to demonstrate that this statement holds true for the practical case of implementing SFC in mobile networks. To this aim, we present the design and implementation of CATENAE, a systems that supports SFC in today's mobile networks without introducing new protocols, without changing the legacy infrastructure and without changing network functions behavior. CATENAE leverages the unique properties of a mobile network's scenario to provide the desired functions chaining features, including the handling

[1]Catenae is a Latin word that means "chains".

978-1-4673-9486-4/16/$31.00

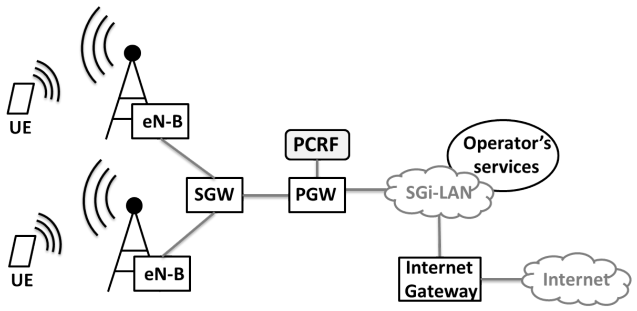

Fig. 1. LTE network architecture.

of opaque network functions' actions. Traffic forwarding is performed by rewriting network packets' header to steer network flows from one function to the next one in the chain. Rewriting rules are configured using SDN software switches, which are anyway deployed at the servers hosting VNFs [20]. Flow re-classification after a VNFs is done by creating per-VNF VLAN topologies, using an approach conceptually similar to [31]. By implementing a proof of concept prototype, we demonstrate that CATENAE does not add per-packet processing overheads, it integrates nicely with legacy network management systems and it is fully compatible with legacy network infrastructures and functions while supporting millions of network flows.

The remainder of the paper is organized as follows. Sec. II introduces background information and related work. In Sec. III we present the CATENAE's design while in Sec. IV we evaluate a proof of concept implementation. Sec. V discusses our design choices and their implications. In Sec. VI we conclude the paper.

II. Background and Related Work

This section presents relevant background information about the mobile networks in which CATENAE can be deployed, introduces the current work on SFC performed by standard organizations like IETF, and provides an overview of the SFC solutions proposed by the research community.

A. Mobile networks

Our work is focused on implementing SFC in Long-Term Evolution (LTE) cellular networks (cf. Fig. 1). A LTE network gives connectivity to a user equipment (UE) using a radio network provided by a set of eNode-Bs (eNBs), which are deployed by the operator over a geographic area. The eNBs encapsulate UE's network flows in a tunnel that, traversing the Serving Gateway (SGW), brings user's IP packets to the Packet Data Network Gateway (PGW). The PGW is the UE's gateway towards IP networks, i.e., all the IP traffic coming and going to the operator's IP network (and to the Internet) goes through the PGW. Also, the PGW is the point where the UE's IP address actually exists in the network. The Policy and Charging Rules Function (PCRF) provides the PGW with the policies to handle users' traffic, e.g., it provides the QoS configuration. After the PGW, user's packets are sent to the SGi-LAN, which is the place where the operator provides additional services [33]. The SGi-LAN is usually an Ethernet network, where network functions are deployed and wired together either physically or logically (e.g., defining VLANs). Network functions can be either *transparent*, i.e., they don't modify packets' header, or *opaque*, i.e., they modify packets' header. After the packets have been processed by the various functions, they are finally delivered to an Internet Gateway (IGW), that forwards them to the Internet.

We highlight a few points about LTE networks, which will help in understanding the design decisions presented in Sec. III. First, operators plan to replace legacy network functions with virtualized ones, by deploying, in the SGi-LAN, a relatively small number of servers (e.g., less than a hundred) that will host VNFs. Thus, we expect a SFC system will deal with VNFs in the number of thousands and that these VNFs are connected to each other by a L2 network, since the SGi-LAN is usually a traditional Ethernet network. Second, the network traffic exposes properties which are typical of LTE deployments. That is, the upstream flows (i.e., those generated at the users) are usually much smaller in size than the downstream flows [12]. Also, the connections are (almost) always initiated in the upstream direction. Finally, since private IP addresses are usually in use on the UE-side, network operators always deploy a NAT-like function [28].

B. SFC in standards

The IETF is the main standard organization that is dealing with SFC, stating the SFC problem in RFC7498 [27], and defining the architecture of an SFC system in RFC7665 [9]. In the IETF architecture, a network function is relabeled *Service Function* (SF). Thus, a *Service Function Chain* is an abstract definition of an ordered set of SFs.

The incoming traffic, e.g., in the upstream direction, at the edge of an SFC-enabled Domain, is classified by a Service Classification Function, in order to perform traffic steering through the correspondent chain. The Service Classification Function adds an SFC-Encapsulation to the classified packets. Notice that the architecture defines the encapsulation format as independent from the network encapsulation protocol used to interconnect the elements. This way, the SFC system does not necessarily need an homogeneous network between the chain's functions, and can instead support more complex scenarios that enable Service Providers to use different technologies. The SFC-Encapsulation is used by another component of the architecture: the Service Function Forwarder (SFF). SFFs read the SFC-encapsulation to send network packets to directly attached SFs, or to forward them to the SFF to which the next function in the chain is attached. For instance, a network switch may host a SFFs function if extended to read the SFC-encapsulation format. Since the RFC7665's architecture assumes that SFs can deal with the SFC-encapsulation format, SFC-unaware functions (e.g., legacy network functions) are supported by the usage of an SFC-Proxy. An SFC-Proxy removes the SFC-encapsulation at the ingress of an SFC-unaware area and add it again on

the egress of that area. An end-of-chain classifier has the responsibility to remove the SFC-encapsulation when packets exit the SFC-enabled Domain, and to classify the packets belonging to the downstream traffic.

Network Service Header. While there are no standards defined for the SFC-Encapsulation format, a currently discussed proposal is the Network Service Header (NSH). The NSH is composed by a Base Header (32 bits), a Service Path Header (32 bits) and zero or more Context Headers. The Base Header provides information about the Service Path Header and the payload protocol. The Service Path Header is composed by a Service Path ID to identify the chains and a Service Index to provide location within the chain. Context Headers carry opaque metadata and variable length encoded information. The NSH header is located between the original packet/frame and the overlay network encapsulation protocol, if any. In fact, current NSH-based prototypes usually assume that an overlay network, e.g., based on VxLAN, connects SFFs. The original data unit, e.g., a L2 frame or a L3 packet, thus, is encapsulated within different transport protocols such as VLAN, VxLAN, GRE, Ethernet, etc. When a SF receives a packet coming from a Service Chain, it will decrement the Service Index header in order to update the location of the packet within the chain. At the end of the chain, an end-of-chain classifier will remove the NSH header and forward the packet normally. NSH is transport independent because it can be used with different encapsulation protocols. It provides information about the chain each packet belongs to, through the Service Path ID header, and the location within the chain, through the Service Index Header. Context Headers make possible to share network and service metadata (L2-L7) that enable to re-classify the packets after a SF.

C. SFC in research

A number of proposals have been presented by the research community, in order to address the challenges of SFC.

SIMPLE [25] provides SFC using an SDN network. It implements inter-switch tunnels to aggregate the traffic with common destinations, in order to reduce the total number of forwarding rules in the SDN switches' forwarding tables. When such optimization is not required, hop-by-hop fine granular forwarding rules are used instead. Traffic reclassification, after an opaque network function, is performed using a dynamic module which analyzes the similarities between packets entering and exiting the network function. However, such solution shows limited accuracy and it introduces significant delays in the network flows.

To overcome such limitations, FlowTags [7] suggests the modification of the network functions in order to provide contextual information, in the form of a tag, which is added to the processed network packets, in order to perform traffic classification. The tags are defined by a centralized controller and cached at the network functions, using an approach similar to the handling of network packets at the controller in OpenFlow networks [21]. Like in SIMPLE, packets forwarding is performed writing appropriate forwarding rules in the SDN switches along the path.

Using an SDN network to perform traffic steering is the solution adopted also by StEERING [34]. In this case, the authors leverage a smart encoding of the forwarding rules in a multi-table switch's pipeline, in order to scale the total number of supported chains and network flows, still providing fine-grained traffic steering. However, StEERING is not able to reclassify the traffic in presence of opaque network functions.

Finally, SoftCell [14] presents a solution that takes into account the deployment scenario's properties to simplify the implementation of SFC in mobile networks. To the best of our knowledge, and putting aside CATENAE, it is the only proposal that explores such an approach. To be deployed, SoftCell requires a network of SDN switches and a modification of the mobile network's architecture. For instance, SoftCell removes SGW and PGW functions, and therefore removes LTE's mobility management introducing a custom solution instead. Traffic classification is performed at switches co-located with the eNBs for the upstream direction, while classification for downstream traffic is performed leveraging information encoded in the source IP address/transport port of outgoing packets. In fact, traffic is assumed to be always initiated in the upstream direction, thus, any downstream packet will carry in the destination IP address/transport port the original upstream flow's encoded value.

III. Design

This section presents our design choices, the CATENAE's architecture, the traffic steering method and gives an overview on possible deployment options in LTE infrastructures.

The main objective of CATENAE's design is to provide SFC while minimizing the impact on current infrastructures. To this aim, our design decisions are taken in the light of the properties characterizing the deployment scenario, i.e., the LTE network. We make a number of observations that motivate our design decisions. First, the main and most important observation is that network functions are connected using an Ethernet network, while user traffic is composed of IP packets, since the tunnel that brings the traffic from eNBs to the PGW only transports IP packets. Thus, the user traffic is agnostic to the L2 packets header and therefore we can manipulate the L2 header to perform traffic forwarding according to our needs. Second, the upstream flow is always started before the downstream flow, and upstream traffic's throughput is usually orders of magnitude smaller than downstream one. Because of these two observations we can perform traffic classification in the upstream direction using a software classifier. In fact, while the classifier is traversed by all the user traffic, it could be still able to scale to handle millions of flows, if these flows contribute a relatively small aggregated throughput. A third observation is that there is always a NAT function deployed in a chain [32]. This allows us to perform traffic classification in the downstream direction without adding a dedicated end-of-chain classifier. In fact, we mandate the deployment of NAT functions as the last chain's

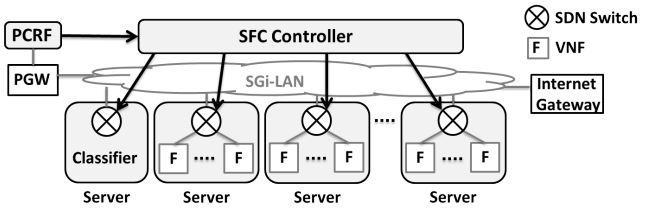

Fig. 2. CATENAE's architecture.

functions, which is anyway already a common practice. A NAT function performs a mapping between upstream traffic and corresponding downstream traffic, in order to apply address translation. When an upstream's packet traverses the NAT, its source IP address is rewritten with a NAT's routable IP address. Thus, any corresponding downstream traffic's packet will be delivered to the NAT, having the destination IP address set to the NAT's routable IP address. Since the NAT function is first hit by the already classified upstream traffic, the NAT will associate any downstream traffic to its upstream flow [14]. The remainder of this section presents the way we capture these observations in the designed architecture and in the corresponding traffic steering method.

A. Architecture

CATENAE's architecture (cf. Fig. 2) is composed of 4 elements: the classifier, which performs traffic classification on the packets entering the SGi-LAN, in the upstream direction; the VNFs' switches, which are deployed at the servers and connect VNFs with the SGi-LAN; the SGi-LAN itself, i.e., an Ethernet network, that connects classifier and VNFs' switches with each other; the SFC Controller, that configures classifier and VNFs' switches in a coordinated way to enforce function chains. Both the classifier and the VNF's switches are SDN *software* switches (e.g., they implement OpenFlow), while the SGi-LAN implements a typical MAC learning algorithm. Thus, the SFC Controller does not change the SGi-LAN network's operations, but uses it as a mere transport network between VNFs located on the servers.

The classifier handles all the uplink traffic coming from the PGW. This design decision may rise scalability concerns, since the classifier is a software switch. However, the classifier handles only uplink flows, which are contributing just a fraction of the overall load (cf. Sec. IV). Thus, CATENAE enforces symmetric paths for upstream and downstream flows, in respect to the network functions, but only upstream flows are processed by the classifier. The classification for the downstream traffic is instead performed by the NAT functions, which are always a chain's last function.

The SFC Controller offers a function chains configuration interface, which could be connected to e.g., the PCRF of the LTE architecture. Upon reception of a chain installation request, the SFC controller implements the chain by installing forwarding entries at all the involved switches. Function chains are described by a list of flow identifiers (FIDs) and a list of functions. Each FID includes one or more of the following fields: IP addresses, transport ports and the IP header's DSCP field. Also, while the FID always defines the upstream direction of a flow, it also identifies the downstream direction as well. In fact, the downstream direction of a flow can be identified by switching source IP addresses and transport ports values with destination ones. The functions' list contains the chain of network functions for the flow identified by the FIDs, specified in the order in which the upstream flow should traverse them. The last function in the list is the chain's exit point, i.e., a NAT. Each network function is further described by a network location. Network locations can be both provided with a static configuration or the SFC system can perform a lookup for the location using a different interface, e.g., connected to a VMs management system.

B. Traffic steering

Traffic steering is the process of defining the network paths for network flows, according to an explicit policy. CATENAE performs traffic steering configuring each of the managed switches (including the classifier) to classify an incoming packet, retrieve the chain it belongs to and forward it to the chain's next function. Since Ethernet networks perform packet switching based on Ethernet destination addresses, CATENAE performs packets delivery to a given function, over the SGi-LAN, configuring the switches to rewrite Ethernet addresses. In the remainder of this paragraph we describe the operations for upstream and downstream cases.

Upstream. Upstream flows are first handled at the classifier, which uses the FIDs to classify packets and send them to the respective first chain's function. If the function is *transparent*, the function's switch delivers the packet directly to the function and re-classifies it using the FIDs, after the function's processing. When a function is *opaque*, packets' header values change, making the system unable to reclassify flows using the FIDs. Also, all the functions coming after an opaque one are handled as opaque functions by the system. In fact, once a packet's header has been changed, the original FIDs don't match the flows anymore. In these cases, classification is achieved creating local virtual L2 networks between a function and its switch. Since network functions typically separate flows received from different L2 networks, a packet will not change its network after the function. Hence, a different (virtual) L2 network per each chain traversing the function helps in associating a packet with its chain. That is, packets belonging to a given chain are tagged with a VLAN tag, which is maintained unchanged when the packet traverses the function[2]. The VLAN tag is removed before sending a packet back to the SGi-LAN, since it is meaningful only on the switch-function link. However, the classification information is required also at the next function in the chain, thus, this

[2]In today's network functions this feature is usually called *VLAN separation*. The tag is maintained also for the new flows generated as a consequence of the reception of tagged packets. Further information can be found in network functions' manuals, e.g., https://techlib.barracuda.com/bwf/deplyvlan.

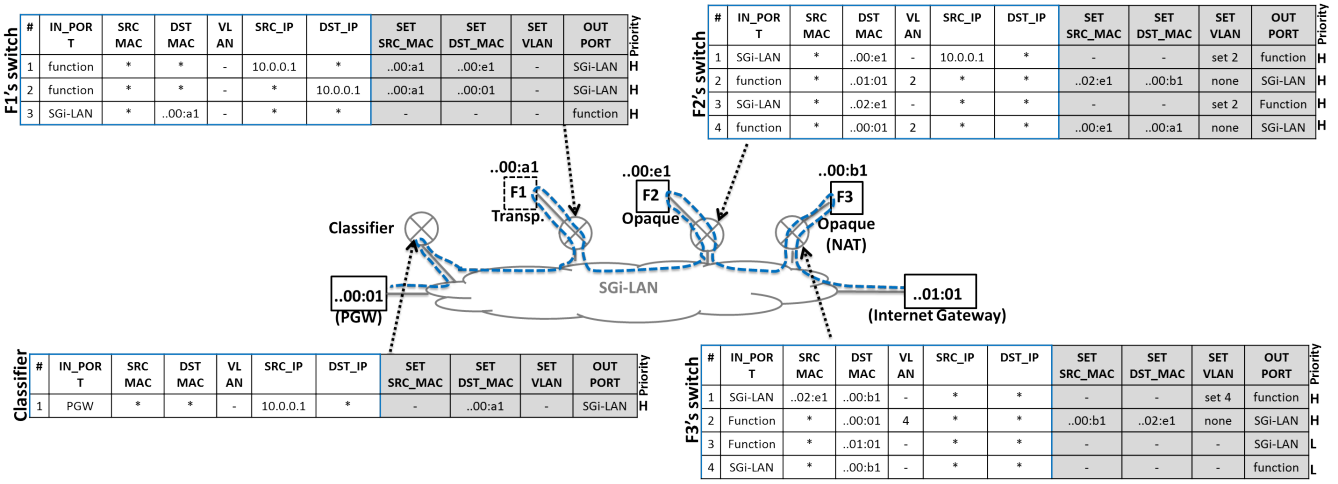

Fig. 3. Forwarding tables configuration example, for the steering of a flow with FID "src_IP=10.0.0.1", which traverses the function chain F1, F2, F3.

information is encoded in the packets' source Ethernet address. Such an address is generated to be unique for each couple chain/function, and it is generated when the next chain's function is attached to a different software switch. In fact, for functions attached to the same switch, it is enough to read the VLAN tag value. When packets are received at the next function's switches, instead, classification is performed looking at the source Ethernet address.

Downstream. Downstream flows are classified at the NATs deployed as chains' last functions. The function chain is then traversed in reverse order. CATENAE operations are again dependent on the type of function the packets traverse. Until there are opaque functions traversed by the downstream flow, the function's switch performs flows classification using VLANs. As in the upstream case, when required, the classification information is encoded in a MAC address value, which this time is written in the packet's Ethernet destination. Recall that this MAC address was generated already for each chain and function during the handling of the upstream flow. Hence, the location of the generated address was already learned by the SGi-LAN. After the last opaque function (i.e., the first one in the perspective of the upstream flow) has processed the downstream flow, the original FID is used to perform packets classification[3]. Here, we assume an opaque function restores the original packet header for the downstream flow. E.g., for downstream flows a NAT restores the original upstream flow headers, with switched source/destination addresses and transport ports. Thus, the downstream flow coming from an opaque function can be classified at a transparent function's switch that receives it, using the FID.

Figure 3 shows a chain example and the switches's forwarding entries generated to implement such chain for a network flow. The entries are expressed in an OpenFlow-like format, with a match part, which identifies the flow, and an action part, which specifies the actions that should be applied to the matched packets. A few details can be captured looking at these entries. First, notice that after a function, the packet's Ethernet source is rewritten to the function's MAC address. This rewriting is required to guarantee the correct SGi-LAN's MAC learning. Second, when flows are received from an opaque function, the flow's direction is detected looking at the destination MAC address. We assume that any opaque function is configured to always use IGW and PGW as forwarding gateways for the upstream and downstream directions, respectively [2]. Thus, if the value is the IGW's MAC address, then the direction is upstream; if the value is the PGW's MAC address, the direction is downstream.

C. Deployment

CATENAE can be deployed in legacy SGi-LANs. The deployment process requires the configuration of SDN software switches in the servers connected to the SGi-LAN, the deployment of the SFC controller and the redirection of the user traffic to the CATENAE's classifier. While the former activities are a matter of software configuration on the servers, the redirection of the traffic to the classifier is the actual hook of the SFC system in the SGi-LAN. Such operation is as easy as changing the default IP gateway address in the PGW's configuration. In fact, the classifier is implemented as a software switch running on a general purpose server connected to the SGi-LAN.

IV. Evaluation

This section describes a CATENAE's proof of concept implementation and its evaluation.

Prototype. We implemented the SFC Controller on top of *Ryu*[4]. The core traffic steering algorithm is implemented in

[3]Actually, the FID is modified to switch source address and transport ports with the destination ones, to match the downstream flow.

[4]http://osrg.github.io/ryu

less than 100 lines of python code. We use *OpenvSwitch* (OVS) as VNFs' switches, and OpenFlow as protocol for the switches configuration. We emulate VNFs running either *click* [17] or *node.js* in Linux *containers*. In all the tests, the SFC Controller runs on a single core of an Intel i5-2540M CPU @ 2.60GHz, using the Python 2.7.3 interpreter shipped with the Ubuntu 12.04.5 LTS distribution. OVS (v. 2.3) and VNFs instances run on servers equipped with an Intel CPU E31220 (4 cores @ 3.10GHz).

Number of chains. CATENAE generates new MAC addresses to support opaque functions. It is unlikely to define a number of chains that could consume the entire MAC address space, however, there is an actual limitation on the number of distinct MAC addresses one can use in an Ethernet network. In fact, Ethernet switches have limited memory to store the associations (address $\leftrightarrow$ switch's port) generated during the MAC learning process [29]. For instance, consider chains that include 4 opaque functions on average (excluding the NAT function at the end, for which no MAC address is generated), and assume that a switch can learn 100k associations (e.g., this is the case of the Broadcom Trident switching chip [29]). In this case, the system could support 25k chains (actually slightly less, considering that some MAC addresses are required for, e.g., physical servers and VNFs). Also, each opaque function can be traversed by 4095 chains at most, since VLAN tags are used to correlate function's entering and exiting flows. While this is a strict limitation, one should consider our initial assumption of supporting VNFs in the number of thousands and notice that the same chain may be applied to several network flows. In fact, operators typically define a single chain for a group of users (e.g., premium users), or services (e.g., web traffic). Furthermore, the actual total number of possible distinct chains is perhaps limited to only few thousands in practice. In fact, consider the case in which a user can pick her services out of a bucket of 10 possible services. If the operator will define a predefined order for the application of such services, such as, anomaly detection is applied before the web proxy, only 1024 distinct chains could be defined (i.e., 2^{10} chains, since each function can be either included or not). Finally, notice that there is no such limitation when dealing only with transparent functions. In such cases, CATENAE does not need to generate any additional MAC address. Moreover, if multiple opaque functions are connected to the same switch, no additional MAC addresses are generated. With K representing the average number of chain's functions attached to the same switch, and recalling that after the first opaque function all the remaining chain's functions are handled as opaque ones, in Fig. 4 we show the number of required MAC address for a chain's implementation. Notice that an early positioning of an opaque function requires more MAC addresses, while the co-location of functions reduces such requirement.

Number of flows. The total number of flows supported by the system defines the number of supported users and how granular their policies can be. CATENAE assigns flows to

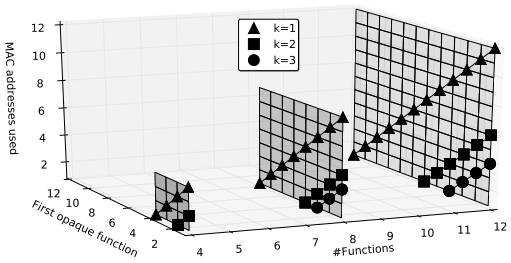

Fig. 4. Number of required MAC address for the implementation of a chain, when varying the number of chain's functions and the position of first opaque function in the chain, for different values of K. Where K is the average number of chain's functions attached to the same switch.

chains performing classification at the SDN switches. The switch's entries are installed in advance, when a chain is first configured, thus, a switch has to host the entries for all the flows that may traverse it. The number of forwarding entries required to configure a flow in CATENAE scales linearly with the number of functions contained in the chain assigned to the flow. In particular, transparent and opaque functions require 2 and 4 entries each, respectively, per flow. Since we rely on software switches, we can easily scale to millions of entries per switch. Assuming that an entry requires 50B of memory (including all the header values [23] and rewriting actions), storing 10 million entries requires 500MB of RAM. Such numbers should be sufficient to support millions of users, even considering several policies per user, e.g., distinct chains per users and per user's flows carrying web, voice, video, etc.

Configuration time. The system configuration time depends on the number of entries the SFC controller has to install. The number of entries scales with the product of the number of flows and number of functions per flow's chain. Our SFC controller prototype is developed in python and can send only about 2200 entry configuration messages per second, limiting the flow configuration performance. Figure 5 shows the rate of flow configurations per second, for chains of lengths between 2 and 5 functions, when functions are either all transparent (but the last one, which is anyway a NAT) or all opaque. In order to confirm that this poor performance is a limitation of the Ryu-based implementation, we re-implemented the core algorithm of the SFC controller using the faster Beacon controller [5]. This second implementation achieved, on the same hardware, a flow configuration rate of more than 16k flows per second, in case of chains with 5 opaque functions.

Flow forwarding delays. Forwarding entries in CATENAE are installed beforehand, thus, no delay is introduced by the traffic steering method, even when new flows are initiated. Notice that alternative solutions (e.g., [7], [14]) may instead introduce delays on (few) flows' packets.

Overheads. It is well known that tunneling protocols increase the cost of processing packets at VNFs' switches [16] and VNFs themselves [7]. Furthermore, it is expected that the average packet size in mobile networks will decrease to

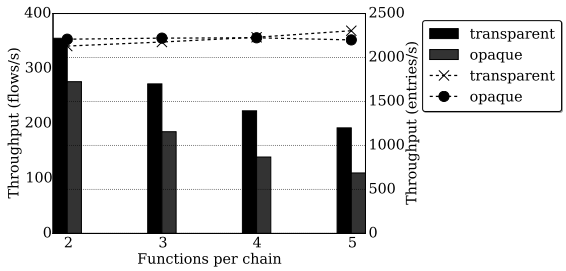

Fig. 5. SFC Controller throughput in configured flows/s (bars) and generated switch's entries/s (lines).

384B [30] in future. For some tunneling technologies, such as VXLAN, this would mean the introduction of more than 14% overhead in terms of on wire transferred bytes (54B are required for VXLAN encapsulation over IPv4). CATENAE does not use any extra header in the packets, avoiding these overheads which are common to other solutions (e.g., NSH [26]).

Data plane scalability. The main CATENAE's bottleneck for the system's data plane scalability is the classifier. In fact, the servers running SDN switches and VNFs, which also handle data plane traffic, could be increased in number to scale with the offered load. Scaling the classifier, instead, would require the introduction of additional components, such as load balancers, between the PGW and the classifier. Such components would increase the deployment complexity of CATENAE and work against our aim of minimizing the impact on the legacy infrastructure. Therefore, we built a trace-driven simulator for the classifier, in order to analyze its performance under different traffic loads. We validated our simulator by comparing the reported performance with the one measured with our prototype, when running a small scale experiment with synthetic traffic. The validation test shows that for relevant performance metrics, such as the system's throughput, the simulator reports values with a general difference below 1% from those measured on the real system.

Lacking access to real traffic traces, we extracted relevant traffic properties from [14], [12] and designed a flow-level trace generator to feed our simulator. The generated traffic trace reproduces the distributions of flow sizes and rates, for the network traffic seen at the PGW, as extracted from [12]. Fixing these parameters, we derive corresponding flow durations. As a correctness check, we verify that the CDFs of the generated flow durations as well as the flow's correlation coefficients between size, rate and duration match the ones reported in [12]. The dynamics of the network flows, e.g., flows arrival rate and number of concurrent flows per second, are extracted from [14], which provides base station's statistics in terms of average active users and data connections created per second. As a last check, we compared the numbers of concurrent flows reported in [12] with the numbers counted in our trace. Here, notice that the numbers of concurrent flows in our trace depend both from the generated flow durations, computed earlier, and the flows dynamics reported in [14].

We fed our simulator with the generated traffic trace, to verify if the classifier is able to handle the offered load with zero packet loss. Notice that, in the scenario presented in [12], the PGW is connected to 22 base stations and handles an aggregated traffic of less than 1 Gbit/s. Considering an average packet size of 512B [30], the system has to handle $\sim$0.23 million packets per second (Mpps). We configured the simulator to cap the software switch forwarding performance at 1 Mpps. This is a very conservative assumption, since current software switches can forward several Mpps [24], [10]. With this configuration, we simulated 30 minutes of system operations, generating 4.6M flows, in which the classifier achieved zero packet loss.

Notice that a 10x increase in load is expected in 2014-2019 [4], which would correspond to an aggregated throughput of $\sim$10 Gbit/s in our simulation. Thus, we performed new simulations to understand the limits of our system. scaling the offered load in two directions: varying the number of base stations connected to the PGW and the per-flow load. The number of base stations affects the rate of new flows created per second as well as their total number (with 40 base stations, we create up to 8.6M flows). This may impact the distribution of the system load peaks. Our test results show that the classifier can handle up to 29Gbit/s of aggregated PGW's throughput: a value three-times bigger than the 2019's forecast. In fact, the classifier handles only the upstream flows, which in the worst case account for the 15% of the overall throughput, in our trace. I.e., 4.35 Gbit/s, which is about 1 Mpps if the packet size is 512B.

V. Discussion

This section discusses the implication of our design choices and provides a few consideration stemming out from our evaluation results.

Legacy infrastructures. CATENAE matches our original aim of minimizing the impact on legacy infrastructures in several ways. First, it can be seamlessly deployed in the LTE architecture, requiring only the installation of software components in the general purpose server attached to the SGi-LAN (cf. Sec. III), and without requiring any architectural change. This is a unique feature when compared to the related work presented earlier. Second, it does not use any tunneling protocol, be it a L2 tunneling protocol, such as VLAN, or a higher level ones, such as VxLAN. This provides a number of advantages and it is another clear distinction point in comparison with the previously mentioned related work. When considering tunneling protocols at the higher network layers, the introduced processing overheads in the servers may be high, unless hardware offloading mechanisms are implemented in the network interface cards (NICs). While it is fair to expect that the most successful protocols, e.g., VxLAN, will be soon offloaded by the majority of the NICs, this is still not the case [8]. Thus, we expect CATENAE will be more efficient in using the server's processing power in the next few years,

when servers will be facing a limited tunneling offloading support. In the case of protocols such as VLAN, for which the offloading is already well established in the NICs, CATENAE provides perhaps an even bigger advantage. In fact, VLAN-like protocols are extensively used to perform logical network separation by a number of systems. In effect, as it became clear in several discussions with network operators, using, e.g., VLANs, is in most of the cases not an option, since it would require very complex, time-consuming and error-prone integrations with the systems that deal with the VLANs management. With CATENAE the coordination with such systems is not required, in fact CATENAE operations deal with VLANs only on the link between software switches and VNFs. Third, while other solutions require modifications to the network functions [7], [26], CATENAE supports current network functions with no modifications, leveraging features that are already extensively used, such as VLAN separation. That is, VNFs are considered as black boxes, helping in decoupling the deployment and configuration of network functions from their composition in a chain [15]. Finally, CATENAE nicely integrates with systems that provide the VNFs deployment, such as OpenStack, which in turn can perform, e.g., optimal VNFs placement.

Hardware network functions. While CATENAE seamlessly supports software legacy functions, hardware network functions can be only supported if directly attached to a SDN switch. Hence, two options are actually viable. In a first case, the hardware function may be attached back-to-back to a server running a software switch. However, the network function may overload the software switch, which, unlike the case of the classifier, should handle both upstream and downstream flows. An alternative solution is to deploy a hardware SDN switch. In this second option, a limitation could be the size of the hardware switch forwarding table. To address this issue, architectures like the one presented in [1] may help. Anyway, please notice that this is a common issue for all the solutions presented in Sec. II, furthermore, unlike other solutions that modify L3 headers [19], CATENAE only rewrites MAC addresses, which is an operation commonly supported in hardware switches.

Classification. In our proof of concept prototype, we implemented the classifier using a software OpenFlow switch. Such a decision may limit the ability of CATENAE to perform complex classification functions that may require Deep Packet Inspection (DPI). However, please notice that CATENAE design does not limit the options for the implementation of a more complex classifier, provided that it exposes an SDN-like interface for configuring the MAC address rewriting operations. In effect, in the evaluation of Sec. IV, we performed our data plane scalability simulations using a particularly low forwarding capacity for the classifier, with the purpose of evaluating the system in the case in which the classifier is performing complex operations. In fact, the 1 Mpps throughput cap is better suited for a complex network function [20], while a software switch is usually capable of forwarding packets in the order of 10 Mpps [10].

Metadata. CATENAE does not support the delivery of metadata to the network functions. For instance, a user's wireless link quality information has to be delivered to the network functions that may need it, e.g., transcoders, using out-of-bound channels. Other solutions support metadata delivery requiring modifications to the VNFs [26], [7].

VI. Conclusion

We presented CATENAE, an SFC system for the SGi-LAN. CATENAE can be deployed on legacy infrastructures, introducing effective SFC without paying the overheads of additional packet header fields, but still scaling to provide fine grained policies for millions of network flows.

Acknowledgment

This work has been partly funded by the EU in the context of the "CleanSky" project (Grant Agreement: 607584).

References

[1] R. Bifulco and A. Matsiuk. Towards scalable sdn switches: Enabling faster flow table entries installation. In *ACM SIGCOMM*, 2015.

[2] J. Blendin, J. Ruckert, N. Leymann, G. Schyguda, and D. Hausheer. Position paper: Software-defined network service chaining. In *IEEE EWSDN '14*.

[3] M. Boucadair, C. Jacquenet, Y. Jiang, R. Parker, and Kengo. Requirements for service function chaining (sfc). Internet-Draft draft-boucadair-sfc-requirements-06, IETF Secretariat, February 2015.

[4] Cisco. Visual networking index: Global mobile data traffic forecast update 20142019 white paper.

[5] D. Erickson. The beacon openflow controller. In *ACM SIGCOMM HotSDN '13*.

[6] ETSI. Network functions virtualisation - white paper.

[7] S. K. Fayazbakhsh, L. Chiang, V. Sekar, M. Yu, and J. C. Mogul. Enforcing network-wide policies in the presence of dynamic middlebox actions using flowtags. In *USENIX NSDI '14*.

[8] S. Guenender, K. Barabash, Y. Ben-Itzhak, A. Levin, E. Raichstein, and L. Schour. Noencap: Overlay network virtualization with no encapsulation overheads. In *ACM SOSR*, 2015.

[9] J. Halpern and C. Pignataro. Service function chaining (sfc) architecture. RFC 7665, October 2015.

[10] M. Honda, F. Huici, G. Lettieri, and L. Rizzo. mSwitch: A highly-scalable, modular software switch. In *ACM SOSR '15*.

[11] M. Honda, Y. Nishida, C. Raiciu, A. Greenhalgh, M. Handley, and H. Tokuda. Is it still possible to extend tcp? In *ACM IMC '11*.

[12] J. Huang, F. Qian, Y. Guo, Y. Zhou, Q. Xu, Z. M. Mao, S. Sen, and O. Spatscheck. An in-depth study of LTE: Effect of network protocol and application behavior on performance. In *ACM SIGCOMM '13*.

[13] IETF. Service Function Chaining working group. SFC.

[14] X. Jin, L. E. Li, L. Vanbever, and J. Rexford. Softcell: Scalable and flexible cellular core network architecture. In *ACM CoNEXT '13*.

[15] W. John, K. Pentikousis, G. Agapiou, E. Jacob, M. Kind, A. Manzalini, F. Risso, D. Staessens, R. Steinert, and C. Meirosu. Research directions in network service chaining. In *IEEE SDN4FNS '13*.

[16] R. Kawashima, S. Muramatsu, H. Nakayama, T. Hayashi, and H. Matsuo. Sclp: Segment-oriented connection-less protocol for high-performance software tunneling in datacenter networks. In *IEEE NetSoft '15*.

[17] E. Kohler, R. Morris, B. Chen, J. Jannotti, and M. F. Kaashoek. The click modular router. *ACM Trans. Comput. Syst.*, 18(3):263–297, Aug. 2000.

[18] D. Levin, M. Canini, S. Schmid, and A. Feldmann. Incremental sdn deployment in enterprise networks. In *SIGCOMM*. ACM, 2013.

[19] V. Manetti, P. Di Gennaro, R. Bifulco, R. Canonico, and G. Ventre. Dynamic virtual cluster reconfiguration for efficient iaas provisioning. In *Proceedings of the 2009 International Conference on Parallel Processing*, Euro-Par'09, pages 424–433, Berlin, Heidelberg, 2010. Springer-Verlag.

[20] J. Martins, M. Ahmed, C. Raiciu, V. Olteanu, M. Honda, R. Bifulco, and F. Huici. Clickos and the art of network function virtualization. In *USENIX NSDI '14*.

[21] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner. Openflow: enabling innovation in campus networks. *SIGCOMM Comput. Commun. Rev.*, 2008.

[22] S. Mehraghdam, M. Keller, and H. Karl. Specifying and placing chains of virtual network functions. In *IEEE CloudNet*, 2014.

[23] J. C. Mogul, J. Tourrilhes, P. Yalagandula, P. Sharma, A. R. Curtis, and S. Banerjee. Devoflow: cost-effective flow management for high performance enterprise networks. In *ACM SIGCOMM HotNets '10*.

[24] B. Pfaff, J. Pettit, T. Koponen, E. Jackson, A. Zhou, J. Rajahalme, J. Gross, A. Wang, J. Stringer, P. Shelar, K. Amidon, and M. Casado. The design and implementation of open vswitch. In *USENIX NSDI '15*.

[25] Z. A. Qazi, C.-C. Tu, L. Chiang, R. Miao, V. Sekar, and M. Yu. Simplefying middlebox policy enforcement using sdn. In *ACM SIGCOMM '13*.

[26] Quinn. Network Service Header. draft-quinn-sfc-nsh-07, 2015.

[27] P. Quinn and T. Nadeau. Problem statement for service function chaining. RFC 7498, April 2015.

[28] P. Richter, M. Allman, R. Bush, and V. Paxson. A primer on ipv4 scarcity. *ACM SIGCOMM Comput. Commun. Rev.*, 2015.

[29] B. Stephens, A. Cox, W. Felter, C. Dixon, and J. Carter. Past: Scalable ethernet for data centers. In *ACM CoNEXT '12*.

[30] Stoke, Inc. LTE equipment evaluation: Considerations and selection criteria, 2012.

[31] S. Vissicchio, L. Vanbever, and J. Rexford. Sweet little lies: Fake topologies for flexible routing. In *ACM HotNets'14*.

[32] Z. Wang, Z. Qian, Q. Xu, Z. Mao, and M. Zhang. An untold story of middleboxes in cellular networks. In *ACM SIGCOMM '11*.

[33] X. Xu, Y. Jiang, T. Flach, E. Katz-Bassett, D. Choffnes, and R. Govindan. Investigating Transparent Web Proxies in Cellular Networks. In *PAM*. Springer, 2015.

[34] Y. Zhang, N. Beheshti, L. Beliveau, G. Lefebvre, R. Manghirmalani, R. Mishra, R. Patneyt, M. Shirazipour, R. Subrahmaniam, C. Truchan, and M. Tatipamula. Steering: A software-defined networking for inline service chaining. In *IEEE ICNP '13*.