A Universal Protocol Mechanism for Network Function Virtualization and Application-Centric Traffic Steering

Changcheng Huang
Department of Systems and Computer Engineering
Carleton University
Ottawa, Canada
huang@sce.carleton.ca

Jiafeng Zhu
Huawei Technologies Inc.
Santa Clara, US
jiafeng.zhu@huawei.com

***Abstract*— New services such as network virtualization, service chaining, and application-centric traffic steering bring new opportunities for network providers and service providers. While Software Defined Network is poised to support new services, it is still in development stage. Mechanisms supporting new services are still missing. In this paper, we propose a new application driven mechanism called Service Forwarding Label that can be used as a universal group-based label to differentiate various services and forward packets based on different service requirements. Compared with existing solutions such as VXLAN, our SFL approach provides universal support for a variety of existing and new services with much less overhead.**

Keywords—Software Defined Network, Network Function Virtualization, Traffic Steering, Service Chaining, VXLAN

I. INTRODUCTION

Routing and forwarding in Software Defined Network (SDN) is based on a flow concept that is more general than the traditional 5-tuple IP flow. As stated in OpenFlow specification [1], each OpenFlow switch will have a flow table with each flow entry having a set of match fields for matching packet headers. A match field may be wildcarded (match any value) or bitmasked so that an arbitrary flow can be defined as matching all the match fields of a flow entry. While this may seem to be quite flexible, it is still under the constraint of the existing packet header fields. If a new service cannot be clearly defined by the existing header fields, the service is hard to be deployed. Although there may be numerous such new services, we only discuss three of them to demonstrate the issues in the following paragraphs.

One of the primary new services that have been envisioned is the network virtualization, which allows physical network provider to sell different virtual networks to different service network providers [2]. Each service network provider can use its virtual network just like the way it uses its own private network while sharing underlying physical network resources with other service network providers. The physical network provider, on the other hand, can enjoy new revenue growth through selling virtual networks with different granularities. In order to do so, each router/switch in a physical network must have the capability to differentiate packets belonging to different virtual networks and forward them differently as required. This is not an easy task with current packet header structure.

Service function chaining (SFC) is another area that SDN can play an important role. Traditionally an SFC consists of a set of dedicated network service boxes such as firewall, load balancers, and application delivery controllers that are concatenated together to support a specific SFC [3]. With a new service, new devices must be installed and interconnected in certain order. This can be a very complex, time-consuming, and error-prone process, requiring careful planning of topology changes and network outages and incurring high OPEX. This situation is exacerbated when a tenant requires different service sequences for different traffic flows or when multiple tenants share the same datacenter network. Through network function virtualization (NFV) service, SDN can dynamically create a virtual environment for a specific SFC and eliminate the complex hardware and labor work. However each virtual node in a service provider network must have the capability to differentiate packets belonging to different SFCs so that it can process and forward them differently. This can be very challenging because each virtual node is in a virtual network rather than a physical network.

The third area is application-centric traffic steering. Application service providers have tried various ways to differentiate their users so that they can maximize their revenues and minimize their costs [4]. The fact that a user today is likely to have multiple access devices and his/her location is typically mobile has made it very hard to route and steer packets based on application characteristics.

There are various proposals to address the abovementioned issues. A good example is VXLAN [5], which is designed with two major objectives: a) to extend Virtual LAN (VLAN) service from a local area network to a network that can span an IP network; b) to address the shortage of VLAN ID. It advocates the architecture that overlays virtualized Layer 2 networks over Layer 3 networks. A 24-bit VXLAN network identifier is added to allow identifications of more than 16M different VLANs. While VXLAN can be implemented relatively easily with existing technologies, it has several major disadvantages that make it difficult to meet the long term challenges. First, VXLAN is an extension of Layer 2 VLAN. Its main objective is to extend VLAN service rather than to

support arbitrary new services. Therefore, it inherits characteristics of Layer 2 technologies, which make it far away from application-centric concept. It is hard to generalize the VLANs supported by VXLAN to other services such as service chaining. Second, it is hard to support resource allocation and optimization for virtual networks due to its distributed nature. Third, the overhead for overlaying Layer 2 networks over Layer 3 networks is overwhelming. Fourth, it is hard to imagine how to support recursive services (discussed more in the next section) with VXLAN.

OpenADN [4] is another option that has been proposed recently. With OpenADN, two new labels are added, one at Layer 3.5 and another at Layer 4.5. The label at Layer 3.5 is used for hop-by-hop transport between waypoints in a segment and the label at Layer 4.5 is used for segment switching. Both labels cannot be used as an end-to-end identifier of a service. Furthermore, issues such as recursive services still cannot be supported.

Similar to OpenADN, Serval [6] tries to address the issues related to mobility and multi-homed services by adding a Service Access Layer (SAL), sitting between the network and transport layer, that gives a group-based service abstraction. The SAL includes a source flow ID, a destination flow ID, and a service ID. The two flow IDs are used to de-multiplex traffic at the receiving host. The service ID is used to identify a service. However, Serval assumes packets are still forwarded based on their IP addresses, which requires dynamic binding between service IDs and IP addresses through control plan. It may also be hard to maintain the service ID after going through a middle box. Serval also does not support recursive service.

In this paper, we propose a new Service Forwarding Label (SFL) mechanism at Layer 5 that can be used to identify a group-based service instance for packet forwarding so that NFV and application-centric traffic steering can be realized with very little overhead. Our label creation process is application driven, allowing close integration of applications and forwarding functions. Different from existing approaches, we do not require the binding between SFL and IP address. By utilizing the flow concept in SDN, packets may be forwarded directly based on their SFLs.

The paper is organized as follows. In Section II, we will discuss in more detail about the issues to be resolved and the recursive service concept. In Section III, we will propose our SFL scheme. This will be followed by several use case scenarios in Section IV. We will finish the paper with some concluding remarks in Section V.

II. RECURSIVE SERVICES

In a real world, it is quite common that users, service providers, and network providers are separate entities. They may all have their own objectives that may conflict with each other. Take the example of enterprise network. With the fast growth of datacenters, enterprises are becoming increasingly interested in outsourcing their enterprise networks to datacenters. Under this scenario, the owner of the physical network now is the owner of the datacenter, such as Amazon. The service network providers are the enterprises who outsource their enterprise networks to the datacenters. Therefore a service provider is independent from the network provider as well as independent from other service network providers who share the same physical network. Recognizing this need, the pioneers of SDN advocate a layer called FlowVisor [7] that plays the same role as the hypervisor for virtual machines. FlowVisor allows a physical network provider to partition its physical network into slices for various

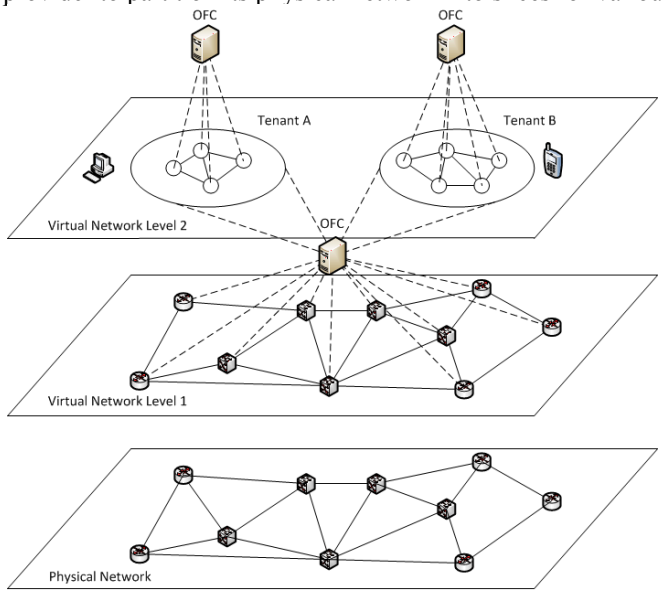

Fig. 1. The architecture of virtual network service.

service network providers. Each service network provider can virtually own one slice which has its own virtual network topology and related resources generated through a topology abstraction process. The service network provider can then optimize its usage of the slice which it owns. A physical network provider can sell network virtualization service to service providers with different granularities at different prices. This will change the situation that physical network providers today can only sell pipes and equip physical network providers a new venue for revenue growth.

The SDN architecture [1] for virtual network service is shown in Fig. 1. As shown in the figure, each entity has its own OpenFlow Controller (OFC). The OFC of a service provider is in charge of routing user flows and optimizing resource usage within its own virtual topology. The OFC of the network provider will execute the topology abstraction process through a topology virtualizer based on a global topology map of the underlying physical network.

It is envisioned in SDN architecture that a service provider can further partition its virtual network and sell virtual network service to other service providers. This kind of recursive network virtualization is illustrated in Fig. 2. As shown in the figure, a network provider sells a virtual network to a tier-1 service provider who in turn sells a virtual network to a tier-2 service provider. This process can continue several iterations making service providers become network providers in a recursive manner. This kind of recursive service is becoming popular today. For example, Amazon sells virtual server service to Netflix and Netflix in turn sells video streaming service to its clients.

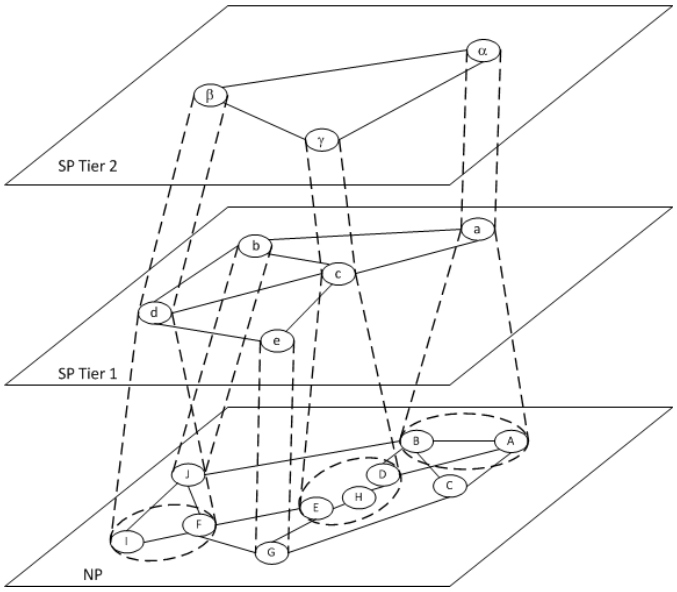

Fig. 2. Illustration of recursive network virtualization.

While virtual network is a powerful concept, it also faces several challenges to be addressed. These include overlapping address spaces, middle-box traversal, SDN network migration, multiple tenants, etc. We will briefly discuss these problems in the following paragraphs.

As we mentioned earlier, each service provider treats its virtual network as its private property. They decide how to assign VLAN IDs and IP addresses within their own virtual networks without consulting each other. Most likely they will use private IP address blocks to avoid the shortage of address space. This creates a high possibility of overlapped address spaces being used by multiple virtual networks sharing the same physical network. Using Fig.1 as an example, if the two virtual networks owned by Tenant A and Tenant B respectively use the same address space, switches in the physical network will not be able to differentiate packets for the two virtual networks. Clearly some mechanism is required to identify packets belonging to different virtual networks.

In a service chain application, packets need to traverse multiple middle-boxes before reaching their destinations. Each middle-box such as firewall or load balancer provides functions that may require processing packets up to the transport layer. Therefore the header information including all layers up to the transport layer will not be maintained end-to-end. This will make it difficult for physical switches to steer traffic to the correct next middle-box on the path to the destination. This issue suggests that an end-to-end ID above the transport layer is required to identify traffic belonging to the same service chain.

SDN as a new architecture is attracting many vendors and network providers. Given the size of the Internet today, the evolution towards SDN networks will likely be a long and slow process. It is easy to see that SDN networks will be deployed initially as islands within Internet. Datacenters, for example, are the first place where SDN will be adopted early. When datacenters provide virtual network services to clients, most of the clients are unlikely to be attached directly to the datacenters. Therefore, traffic from these clients has to travel through legacy IP networks before they can reach SDN based datacenters. Any information used to identify virtual networks in the lower three layers may be lost when traffic reaches the datacenters. Inter-datacenter traffic may also span multiple legacy IP networks making it difficult to support virtual networks across multiple datacenters.

Recursive network virtualization raises another new challenge. As we mentioned earlier, each virtual network may be owned by a different entity. This leads to the situation that multiple entities may be involved in recursive network virtualization. Each entity has the right to control the resources within its own virtual network and decide how to route traffic based on its own policy. Orchestration among multiple OFCs is calling for simple mechanisms that help streamline business relationship among different entities.

In summary, the above mentioned challenges must be addressed before the benefits of SDN can be fully realized. In the next section, we will describe in detail how our SFL works.

III. SERVICE FORWARDING LABEL

As discussed in the last section, we need an ID that can be used to identify a group-based service instance (In our context, a virtual network or a service chain instance is considered as a single service instance. A mobile user with multiple devices can also be treated as a service instance). This ID needs to sit above Layer 4 so that it can stay intact while a packet traverses legacy IP networks and middle-boxes. This naturally points to an ID at Layer 5.

In OSI model, Layer 5 is called the session layer which is designed to establish, manage, and terminate connections between local and remote applications. A good example is a video conference session where multiple parties join and leave dynamically. This bears similarity to a service instance such as a virtual network that has many nodes and carries a large number of dynamic traffic flows. This similarity motivates us to define an ID at Layer 5 for the identification of a service instance. We call this ID as Service Forwarding Label (SFL).

SFLs will be created and maintained by a service provider and used by its clients and OpenFlow switches to identify and steer traffic belonging to different service instances. SFLs can be stacked for applications such as recursive service where each level of the stack is administered by the owner of the service level in a recursive business relationship as discussed in the last section. This allows easy scale to multiple levels of services with multiple ownerships nested in the SFL stack.

An SFL must be unique within the space of the service provider who administers the SFL. Multiple service providers at the same level will be differentiated by their SFLs at the lower level. The combination of the SFLs across different levels in a label stack uniquely identifies a service at any layer in a physical substrate domain.

We now discuss SFL format and the process to establish and terminate an SFL. As shown in Fig.3, each SFL is represented by 4 octets. Starting from bit 0 of the 4 octets, the

first 30 bits hold the label, bit 30 is reserved for experimental use, bit 31 is the top-of-stack bit (S). The S bit is set to one for the last entry in a label stack, and zero for all other label entries in the stack. As the header at Layer 5, SFL can run either over UDP or TCP making it applicable to all kinds of traffic belonging to the same service instance.

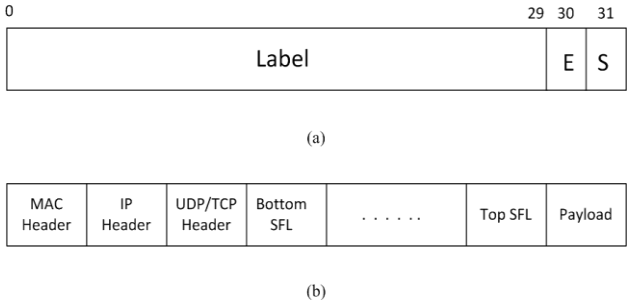

Fig. 3. (a) Format of Service Forwarding Label; (b) Illustration of a packet header with SFL stack.

Each SFL is associated with a lifetime. When the lifetime expires, the SFL will be terminated or be renewed. This dynamic mechanism allows a service provider to maintain a smaller pool of SFLs.

We propose to add the SFL field as a match field in the flow table of an OpenFlow switch. Different from other existing approaches, an SFL does not have to be mapped to an IP address. The SFL match field can be used alone or in combination with other match fields to define a flow. When used alone, it allows an OpenFlow switch to forward traffic only based on the SFL in a packet. With our approach, transport and IP layers will stay the same.

There are various scenarios that may happen during the lifetime of a SFL. The procedures for establishing and terminating SFL depend on the actual scenario encountered. A typical scenario is shown in Fig. 4. We describe the procedures step by step in the following part.

- A client sends a service request to the OFC of a service provider with its user ID and requested service type using HTTP request message. Metadata can be sent through HTTP Post message;
- The OFC of the service provider decides whether it can accept the request by applying optimization process which determines how to route traffic and allocate resources for the requested service within the service provider domain;
- If the request is admissible, the OFC will create a new SFL which is unique to the service provider and send the SFL and associated lifetime to the client, and relevant OpenFlow switches or middle-boxes that need to steer or process traffic based on the SFL through an OpenFlow OFPT_FLOW_MOD message;
- Upon receiving the message from the OFC, the OpenFlow switches or middle-boxes will set the SFL and its lifetime into their flow tables as part of a rule set;
- The OFC will send HTTP response message with the SFL and associated lifetime to the client confirming the acceptance of the request;
- The client will add the label as Layer 5 header to its packets destined for the requested service and send them out;
- When the packets reach the switches or middle-boxes within the service provider network, the service provider will match the Layer 5 header (and other headers in other layers if necessary) to its rule set and decide how to forward or process the packets based on their service requirement;
- The switches or middle-boxes will then process those packets and steer them to the next switch or middle-box if necessary;
- When the lifetime of the SFL expires, the client can choose either to renew the service or leave. If it decides to renew, it will send a HTTP request message with the SFL to the OFC, the above procedures will be repeated except that the original SFL will be used instead of generating a new SFL.

In the next section, we will discuss some use cases that will help illustrate how the SFL can be used in real applications.

IV. SAMPLE USE CASES

There are numerous use cases that the proposed SFL can be applied to. We will discuss some common use cases briefly in this section.

The first use case is the virtual network service we mentioned earlier. Here a physical network provider will serve as the service provider and service network providers will serve as clients. Service network providers request virtual networks from the physical network provider. Each service network provider will have full control over its virtual network.

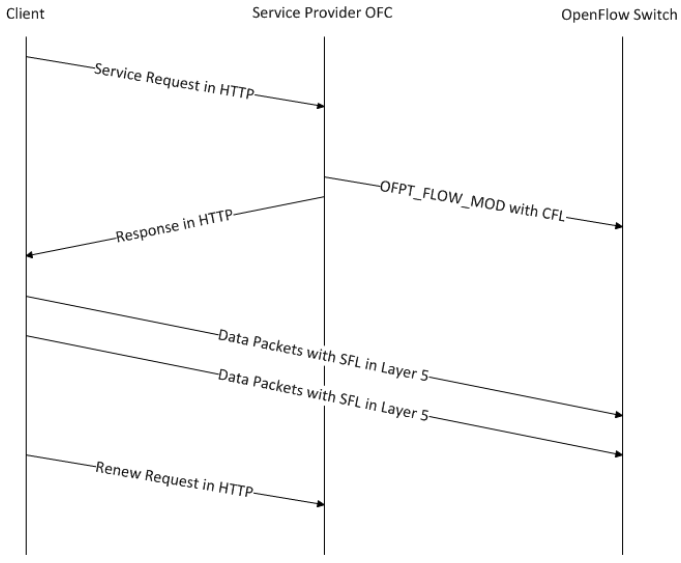

Fig. 4. Flowchart of labeling process.

One issue we mentioned earlier is that the address space used by service providers can be overlapped. An example is shown in Fig. 5 where Client Network 1 owns Virtual Network 1 and Client Network 2 owns Virtual Network 2. Both Virtual Network 1 and Virtual Network 2 share a physical network owned by a SDN network provider. When a packet reaches a switch in the SDN network, the switch needs to decide which virtual network the packet belongs to.

Through the procedures discussed in the last section, each client network will receive an SFL assigned by the SDN network as an identifier of its virtual network. The client network will inform its users of adding the SFL for all packets that need to use the virtual network it owns. When packets reach the switches in the SDN network, they can be differentiated using their SFLs even though their IP address spaces may be overlapped. This can be done by a simple match in the flow table. Without SFL, multiple header fields may need to be matched in order to identify packets belong to a virtual network, which will likely cause flow table fragmented and bloated.

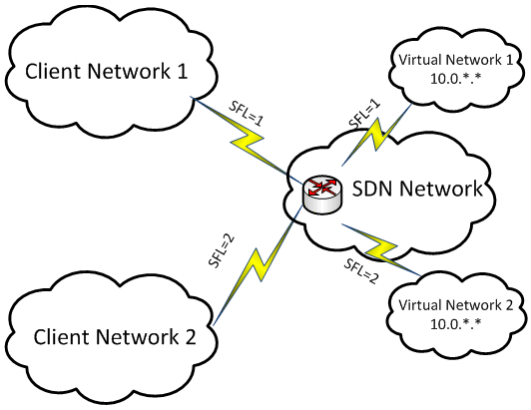

Fig. 5. Virtual networks with overlapped address space.

When recursive network virtualization is deployed, each service network will serve as client as well as service provider at the same time. As a client, it receives a SFL from the service provider one level below it. As the service provider, it administers the SFLs that identify the virtual networks it sells. A physical switch can use multiple levels of the label stack to steer packets for the correct virtual networks they belong to.

Now we look at the second use case that demonstrates how SFC can be supported. It is easy to see that an SFC instance can be realized using NFV where each virtual node represents a specific service such as firewall that can be dynamically mapped to a physical node in the lower level. By the virtualization of a service chain, dynamic sharing of physical resources can be achieved. Traffic flows for different service chain instances can be uniquely identified and steered by the combinations of the multiple SFLs in their label stacks. This enables great flexibility and leads to significant cost reduction in OPEX. An example is illustrated in Fig.6.

One of the key issues introduced by the hierarchy of recursive SFC relationship is the relationship between different levels. There are three types of relationship that can be envisioned. The first one is called independent relationship where the lower level is agnostic of the SFCs created by upper levels. Therefore all the service functions created by an upper level will be implemented and enforced at the upper level SFC modules while the lower level modules are completely unaware. When traffic arrives at a lower level module, the module processes the incoming traffic based on its service function requirements and de-multiplexes the traffic to the right upper level module using the SFLs it assigned. The lower level module does not execute the service functions of upper level. The upper level applies different service functions based on the SFLs it assigned. In this case, the upper level module does not have to be the same type as the lower level module.

Another type is the opaque relationship where service functions defined by the upper level require collaboration from lower level. For example, Enterprise B may inform Cloud Provider A about some service functions it needs and ask Cloud Provider A to help implement those service functions. When traffic arrives at Cloud Provider A, it will identify traffic flows using both the SFL it assigned and the SFL assigned by upper level as a concatenated ID and then apply associated service functions. The traffic stream will be delivered to upper level module for extra service functions. In this case, upper level functions inherit properties from lower level functions. They are also constrained by the functions available from lower level. However the upper level can create new properties such as new firewall rules as long as it doesn't violate the constraint posed by the lower level. Whenever service functions at lower level are changed, upper level service functions will also be changed. However changes made to the upper level may not apply to lower level. In reality, a tenant is more likely to retain some functions as independent (e.g. encryption function) and some functions as opaque (e.g. load balance).

Other than the two types mentioned above, it is also possible that Enterprise C, as a customer of Enterprise B, may delegate its administrative role to Enterprise B, which in turn delegates its authority to Cloud Provider A. The benefit of this single administrative domain approach is that Enterprises B and C do not need to handle the administrative work. However, both B and C need to disclose all information to A, forming a transparent relationship. With transparent relationship, Cloud Provider A has to implement all SFCs with a hierarchical structure that satisfies the roles and responsibilities distributed according to the organizational structure of Enterprises B and C. This kind of structured service is likely to become one of the SFC deployment paradigms.

Application service providers such as Google are increasingly interested in providing different treatments [8] to different types of customers, e.g. subscribers vs. casual users. Based on the SFLs they are carrying, user traffic flows can be steered to different environments with different networking and computing resources provisioned. Under this context, SFL provides a simple and effective hook that connects applications to physical layer devices directly and enables application-centric traffic steering. For example, there are many existing Quality of Service (QoS) schemes such as VLAN and DiffServ. But they are Layer 2 or 3 mechanisms that are hard to scale to end-to-end applications. As mentioned earlier, it is

difficult to maintain any code points in headers up to Layer 4 for end-to-end services due to middle boxes and different domains a packet may traverse. By sitting at Layer 5, our SFL can travel through networks and middle boxes easily and therefore provide a very strong support for various end-to-end applications.

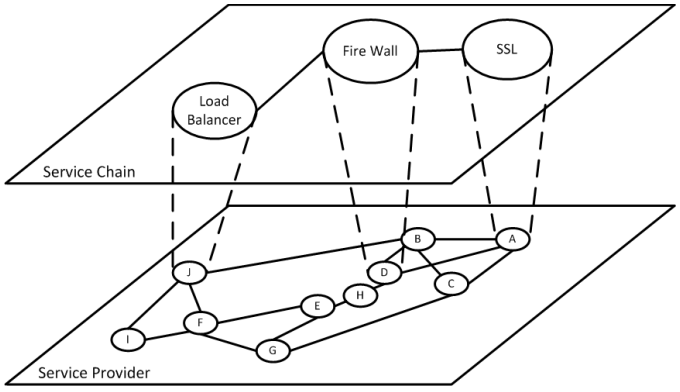

Fig. 6. Service chain as network function virtualization.

There are many other application scenarios that can demonstrate the usage of SFL. For example, a service provider may want some of its user traffic be protected from server or link failures while other traffic not. When a server or link failure happens, the traffic that needs protection is steered to a protection path. In OpenFlow switches, packets that require protection will be matched at a group table instead of the regular flow table. Therefore incoming packets must be de-multiplexed into regular flow table or group table based on whether they need protection or not. The proposed SFL provides an excellent option to achieve this function. Specifically, we can assign one SFL to identify traffic requiring protection and another SFL for traffic not requiring protection. As shown in Fig.7, when packets arrive at a switch, it first goes to a regular flow table. If the SFL matching indicates a packet without protection requirement, other header fields will be matched as regular case; otherwise, the packets will be forwarded to a group table for protection matching.

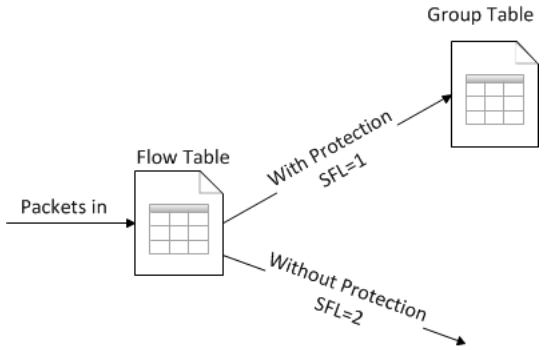

Fig. 7. Forwarding packets with or without protection.

V. CONCLUSION

By separating control plane from data plane and centralizing resource allocation, SDN has the potential to allow network and service providers to create a variety of new services. Existing SDN products have been focused on some basic functions such as flow setup and teardown. The potential to create new services has not been fully explored.

In this paper, we proposed a universal group-based SFL as an identifier for service instance. It is controlled by service providers and used by clients and OpenFlow switches to steer traffic to different services. The format of SFL is simple enough to minimize overhead. Through SFL stacking, recursive services such as recursive network virtualization can be supported easily while allowing different entities to exercise their controls over their own resources. With SFL as a Layer 5 mechanism, it can traverse middle-boxes and legacy networks without any changes so that the relationship between clients and service providers can be maintained end-to-end.

We have demonstrated various use cases ranging from network virtualization, service chaining, to application-centric traffic steering. Through these use cases, we can see that the proposed mechanism is simple to implement with existing protocols and technologies and can effectively enable various new services. In specific, we introduced recursive service as an important requirement for scaling up business relationship. We illustrate how the SFL can be used to support recursive service through stacking.

REFERENCES

[1] https://www.opennetworking.org/

[2] M.M.M.K. Chowdhury and R. Boutaba, "Network virtualization: state of the art and research challenges," IEEE Communications Magazine 47 (7),20-26.

[3] D. Jacobs, "How SDN and NFV simplify network service chain provisioning," http://searchsdn.techtarget.com/tip/How-SDN-and-NFV-simplify-network-service-chain-provisioning.

[4] S. Paul, R. Jain, J. Pan, J. Iyer, D. Oran, "OpenADN: A Case for Open Application Deliver Network," Proceedings of ICCCN 2013, July 2013, Nassau, Bahamas.

[5] M. Mahalingam, et al., "VXLAN: A Framework for Overlaying Virtulized Layer 2 Networks over Layer 3 Networks," IETF draft, http://datatracker.ietf.org/doc/draft-mahalingam-dutt-dcops-vxlan/.

[6] E. Nordstrom, et al., "Serval: an end-host stack for service-centric networking," Proceeding of 9th USENIX Symposium on Networked Systems Design and Implementation, April 25-27, 2012, San Jose, US.

[7] R. Sherwood, et al., "FlowVisor: A Network Virtualization Layer," OPENFLOW-TR-2009-1, OpenFlow Consortium, October 2009

[8] S. Jain, et al., "B4: Experience with a Globally-Deployed Software Defined WAN," ACM SIGCOMM 2013, August 12-16, 2013, Hong Kong