Leveraging IPv6 Segment Routing for Service Function Chaining

David Lebrun
ICTEAM, Université catholique de Louvain
Louvain-La-Neuve – Belgium
firstname.lastname@uclouvain.be

ABSTRACT

Network operators seek more flexibility with solutions like Network Function Virtualization and Service Function Chaining (SFC). Thanks to these techniques operators can define on-the-fly which services have to be applied to which packets. We propose to use IPv6 Segment Routing to support SFC and describe and evaluate the performance of our implementation in the Linux kernel.

1. SEGMENT ROUTING

Segment Routing is a new source routing architecture that is being developed within the Internet Engineering Task Force [1]. It allows packets to follow non-shortest paths towards the destination by specifying a list of detours, i.e. waypoints. These waypoints are called *segments*. The packets are forwarded along the shortest path from the source to the first segment, then through the shortest path from the first segment to the second segment, and so on. These segments can refer to a link (*adjacency segment*), i.e. force the traversal of a specific link, or to a node (*node segment*), i.e. force the traversal of a specific network node. Segment Routing currently exists in two flavors: MPLS and IPv6 [2] [4] We focus on the IPv6 version of Segment Routing (SR-IPv6).

The IPv6 flavor of Segment Routing uses an IPv6 Routing Header having its Routing Type field set to 4. [4] A segment is a routable IPv6 address announced through an IGP (OSPF, IS-IS, etc.). Furthermore, only the nodes that actually process SR-enabled packets need to support SR-IPv6. Intermediate nodes that are on the path of a segment forward the packets as regular IPv6 packets. A listing of the Segment Routing forwarding pseudocode is shown in Algorithm 1

This algorithm describes how a node processes an IPv6 packet with the SR header. The first part of the algorithm is the standard processing of the SR header defined in [4]. If there are still segments in the SR header, the header is modified and the next segment is placed as destination address of the packet before forwarding it. We modify this algorithm to allow a node to deliver a received packet to an application to runs on this node. This is the building block of using Segment Routing for Service Function Chaining.

Algorithm 1 SR header processing

```
1: if DA = myself (segment endpoint) then
2:    if Segments Left > 0 then
3:       Decrement Segments Left
4:       Update DA with Segment List[Segments Left]
5:       if Segments Left == 0 AND Clean-Up bit set then
6:          Strip SRH
7:       end if
8:    else
9:       Continue regular IPv6 processing of the packet
         e.g. deliver packet to next PID (application)
10:      End of processing
11:   end if
12: end if
13: Forward the packet out
```

2. SERVICE FUNCTION CHAINING

With the advent of Network Function Virtualization, it has become increasingly important to be able to specify on-the-fly network functions to be applied to network packets, in an arbitrary order. The Segment Routing architecture allows to do this. We already have node segments to force the traversal of a specific network node. We extend this definition to create a *service segment* representing a service that the packets have to traverse. See Fig. 1 for an illustration.

Network operators can then configure the network (e.g. through an SDN controller or any other suitable configuration scheme) to map different traffic classes to specific services chains. This configuration can be either static or dynamic, enabling the network to automatically react to network events such as congestion, intrusion detection, and so on.

A service segment can represent an appliance such as carrier-grade firewalls, IDS, etc. However, as network functions are increasingly virtualized, such service might be handled by some virtual machine in the network. With this framework, virtually any network function can be provided. For example, an operator may provide a site-to-site VPN service to its clients, within the same Segment Routing domain. The operator can implement this by chaining a compression service, then an encryption service. At the other side, there would be a decryption service followed by a decompression service. Furthermore, the operator might simply expose the

DOI: http://dx.doi.org/10.1145/2842665.2843568

different services to the clients. They can then use those services by adding the appropriate SR header at the source.

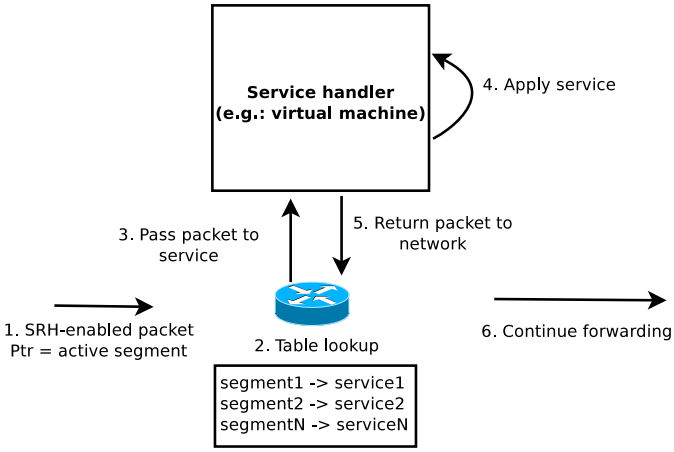

Figure 1: Local service processing

3. IMPLEMENTATION AND EVALUATION

We implemented SR-IPv6 in the Linux kernel. At the time of writing and to the extent of our knowledge, this is the sole open-source Linux implementation of Segment Routing with IPv6. [3] Our implementation provides an API to map a segment to an application, meaning that if the kernel receives an SR-enabled packet with the corresponding active segment, the whole packet will be delivered to the specified application through a `netlink` socket. The application has to register beforehand to the kernel, through the `netlink` socket, specifying the segment to bind to.

This packet delivery can be synchronous, i.e. the kernel awaits for the application to transfer back the packet, modified or unmodified, or asynchronous, i.e. the kernel delivers the packet to the application and immediately forwards it into the network. Asynchronous transfers can be used for accounting or monitoring purposes. Synchronous transfers are obviously more convenient for services that need to modify the packet header or payload (e.g. NAT, compression, encryption, etc.).

We performed lab measurements to evaluate the performance of our implementation. We provide results for plain-packet forwarding (i.e. non-SR), SR-enabled forwarding, asynchronous service segment, and synchronous service segment (the service is simply a packet counter). The simulations were performed over three servers (HP ProLiant DL120 with Intel Xeon X3440 CPU and 8GB RAM). The network is shown in Fig. 2. The servers are connected with 10G Ethernet links, but the links were capped to 100M for the service segment forwarding tests. For higher throughput, transfers between userspace and kernelspace severely impact the performances and the services should be performed within the kernel or through a tool such as DPDK. Performances were measured through the `iperf3` tool, using UDP packets with a payload of 64 bytes[1]. Table 1 shows a summary of our simulations.

[1]The iperf3 command used is `iperf3 -c <client> -u -b 100M -l 64 -Z -A 0 -t 60`

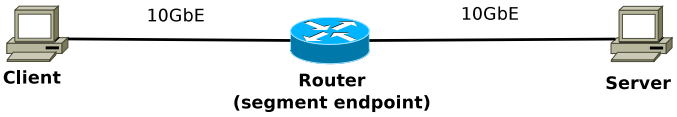

Figure 2: Virtual network

Experiment	Line rate	Packets/s	Bytes/s
Non-SR	10G	683 Kpps	330 Mbps
Non-SR	100M	205 Kpps	100 Mbps
SR (fwd)	10G	680 Kpps	328 Mbps
SR (fwd)	100M	201 Kpps	99 Mbps
SR (sync)	100M	195 Kpps	98 Mbps
SR (async)	100M	176 Kpps	90 Mbps

Table 1: This table shows the throughput in packets per second, for each experiment

Our experiments show that our implementation has good relative performances, the impact of SR forwarding and service segment processing is acceptable. However, the synchronous service segment experiment shows better performance than the asynchronous experiment, which is counter-intuitive. The mechanisms causing these performances discrepancies will be investigated. Moreover, we see that the actual throughput for the uncapped experiments is about 330 Mbps. This is the limit at which `iperf3` is able to generate packets.

4. FUTURE WORK

We presented our work in progress on Service Function Chaining using Segment Routing with IPv6. Our implementation of SR-IPv6 itself is quite complete. We now focus on (i) designing different services that can be chained, such as encryption and compression, and dealing with specific issues for each of them, such as TCP sequence number adjustment for the compression service; and (ii) profiling the code and analyzing bottlenecks in order to reach performances as close as possible to a line-rate performance, using tools such as DPDK.

Acknowledgements

This work was partially supported by the ARC grant 13/18-054 (ARC-SDN) from Communauté française de Belgique.

5. REFERENCES

[1] C. Filsfils et al. Segment Routing Architecture. Internet draft, draft-ietf-spring-segment-routing-05, work in progress, Sept. 2015.

[2] C. Filsfils et al. Segment Routing with MPLS data plane. Internet draft, draft-ietf-spring-segment-routing-mpls-01, work in progress, May 2015.

[3] D. Lebrun. Supporting IPv6 Segment Routing in the Linux kernel. http://www.segment-routing.org/, 2015.

[4] S. Previdi et al. IPv6 Segment Routing Header (SRH). Internet draft, draft-previdi-6man-segment-routing-header-07, work in progress, July 2015.