Pushing Network Programmability to the limits with SRv6 uSIDs and P4

Ahmed Abdelsalam[1], Angelo Tulumello[2], Marco Bonola[3,4], Stefano Salsano[2], Clarence Filsfils[1]
[1]Cisco Systems, [2]University of Rome Tor Vergata, Italy, [3]CNIT, Italy, [4]Axbryd S.r.l., Italy.

ABSTRACT

P4 is a domain-specific programming language for expressing how packets are processed by network devices. P4 is used to program P4-enabled devices with a custom forwarding pipeline. SRv6 is a network architecture that encodes a list of instructions in the IPv6 packet header to define a *network-wide* packet processing *program*. Each instruction defines a node to process the packet and the behavior to be applied to that packet by that node. Recently, the SRv6 architecture has been extended to support a set of new instructions, known as uSID instructions, that provide a better scalability and MTU efficiency. In this demo paper, we provide a P4 forwarding pipeline that supports the SRv6 uSID instructions. Our implementation leverages the P4 BMV2 behavioral model and extends the ONOS controller to support the new SRv6 behaviors. Finally, we show two scenarios where we use our implementation to provide a fully programmable network fabric.

CCS CONCEPTS

• **Networks → Routing protocols**; **Programmable networks**.

KEYWORDS

Software Defined Networking, Segment Routing, Programmable Networks, Routing Protocols

ACM Reference Format:
Ahmed Abdelsalam[1], Angelo Tulumello[2], Marco Bonola[3,4], Stefano Salsano[2], Clarence Filsfils[1] [1]Cisco Systems, [2]University of Rome Tor Vergata, Italy, [3]CNIT, Italy, [4]Axbryd S.r.l., Italy. . 2020. Pushing Network Programmability to the limits with SRv6 uSIDs and P4. In *3rd P4 Workshop in Europe (EuroP4'20), December 1, 2020, Barcelona, Spain.* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/3426744.3431331

1 INTRODUCTION

The Segment Routing (SR) architecture is based on presence of a list of instructions, known as segments, in the packet headers. These instructions influence the forwarding and processing of packets along their path. In SRv6 (Segment Routing over IPv6 data plane) the Segment IDs (SIDs) are represented with IPv6 addresses, which are 16 bytes long. When an SRv6 service requires a long list of instructions/SIDs, the overhead introduced by the long headers could be significant.

EuroP4'20, December 1, 2020, Barcelona, Spain
© 2020 Association for Computing Machinery.
ACM ISBN 978-1-4503-8181-9/20/12…$15.00
https://doi.org/10.1145/3426744.3431331

The Micro SID (uSID in short) solution is meant to drastically reduce the length of the SID list in SRv6. In particular, in this solution a network program can be represented with a set of micro instructions encoded in the 16 bytes of a plain SRv6 SID. Such micro instructions are called Micro SIDs (or uSIDs). In the most typical configuration, uSIDs are represented with 2 bytes, while the SRv6 SID in which they are carried belongs to a /32 address space, named Locator Block. A segment list composed of 6 instructions can be inserted in just one 16 bytes SRv6 SID, resulting in large saving of packet overhead. The basic idea behind Micro SID is that when a uSID enabled node receives a packet with its uSID ID immediately after the Locator Block in the IPv6 Destination Address, it *consumes* it: it pops its uSID from the segment and shifts left by 16 bits the remaining part of the list. More details on the SRv6 network programming model and the uSID solution are described in [1][2][3].

In the following, we show the implementation of uSID in P4 and present our demo with two use cases involving P4 software switches (BMV2) and the ONOS controller.

2 P4 IMPLEMENTATION

We implemented the uSID solution in P4 starting from an existing SRv6 implementation [4] and enhanced its functionalities by implementing the following four behaviors:

- **uN** is responsible for shifting by 16 bits the uSID list to extract the next end router uSID
- **uA**, similar to uN, shifts by 32 bits the uSID list and cross connects the packet to a neighbor router, skipping the normal IPv6 routing
- **End** is the legacy SRv6 End action, responsible for the advancement of the pointer in the SRv6 segment list
- **uDX** is responsible for the decapsulation of the packet

These behaviors are the actions of a Longest Prefix Match (LPM) table, named `my_sid_table`, that matches on the destination IPv6 address of the packet. All the logic to support both uSID and legacy SRv6 resides in this table.

Another table, named `srv6_encap_table` is responsible for the encoding of the Micro SID policy into the packets entering the Micro SID domain. If the number of uSIDs is less or equal to 6, the packet will undergo a simple IPv6-in-IPv6 encapsulation (the uSID policy will be encoded in the outer IPv6 destination address). If the number of uSIDs is more than 6, the packet will undergo the same encapsulation, but with a Segment Routing Header (SRH) containing the remaining list of uSIDs in the SRH segment list.

The ONOS controller is programmed to configure the four behaviors mentioned above, taking the configuration from a JSON file containing the assigned micro instructions for each router. Moreover, an ONOS command has been implemented to configure arbitrary

Micro SID policies in the ingress routers. The different encapsulation depending on the number of uSIDs is handled by the controller logic. Also the uA behavior can be configured with a special ONOS command.

We were able to implement the SRv6 uSID behaviors used for this demo by simply leveraging the $P4_{16}$ primitive actions.

3 DEMO

The topology in Figure 1 represents a service provider SRv6-enabled network. Blue links represent low latency links and red double-lined links represent high-bandwidth links. Site A and Site B of a customer are connected through the SRv6 network. The topology is emulated with Mininet and all the nodes in the testbed, including the ONOS controller, run in docker containers.

The Locator Block `FCBB:BB00::/32` is assigned for uSID allocation. The uSIDs that can be allocated within such block are divided in two sets: (1) the Global Information Base (GIB) is the set of IDs for global uSID allocation and (2) the Local Information Base (LIB) for local uSID allocation. An uSID from the first set typically identifies a shortest-path to a node in the SR domain that is unique and globally routable. On the contrary, an uSID from the LIB can be reused and has local significance, e.g. it may identify a cross-connect to a direct neighbor over a specific interface.

For the uN behavior, each router is allocated with a 16-bit ID from the GIB, e.g. R_01 will be allocated `0x0001`. This results in two entries in the `my_sid_table`, one with prefix length /48 for uN action and the other with prefix length /64 for End action. For example, R_1 will be configured with `FCBB:BB00:0001::` /48 and /64. Since the table has an LPM match, the /64 entry is matched with higher priority when a `0x0000` End of Container instruction is immediately after the uSID ID. This means that the uSID list in that specific segment has been "consumed". Thus, the End behavior will be applied to the packet instead of uN, copying the next segment in the SRH list to the destination address.

Some routers are configured to apply the uA behavior, which is assigned with an ID from the LIB preceded by the ID of the uN behavior of the hosting node. For example, R_{09} will allocate 0xFA94 for uA behavior that will cross-connect the packets to node R_{04}.

The SRv6 uDX behavior is assigned an ID from the LIB (`0xfd00` in this Demo) and can be reused across all nodes as it has local significance. However, it has to be preceded by the uSID ID of the hosting node.

3.1 Use case 1: Low latency network slice.

The first use case is creating a low latency network slice for the traffic between Site A and Site B. To achieve such a low latency network slice, traffic has to go through only blue links which represent low latency links. The candidate path for the low latency slice is R_{01}, R_{04}, R_{05}, R_{08}, R_{07}, R_{06}, R_{03}, R_{02}. This path will be enforced by R_{01} for traffic classified as low latency with an uSID policy. The policy will push a SID list that represents that path, but it is not needed to encode all the nodes of the path.

In fact, the set of nodes needed in this case are R_{08} (uN), R_{07} (uN) and R_{02} (uN and uDX) resulting into a single uSID container: `FCBB:BB00:8:7:2:FD00::`. In this case all uSID IDs of the SRv6

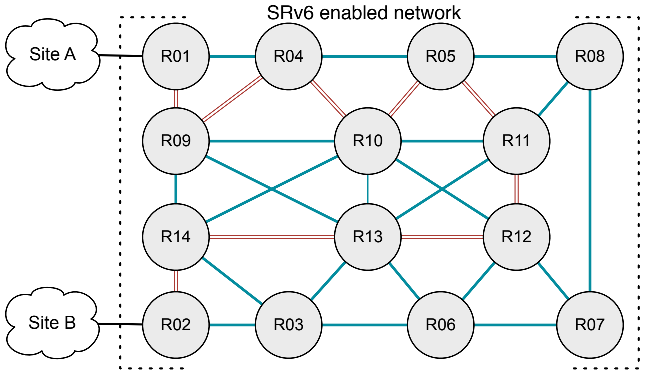

Figure 1: DEMO topology

policy fits in one uSID container which will be carried in the destination address of the outer IPv6 header of the SRv6 packet.

3.2 Use case 2: High bandwidth network slice.

The second use case is creating a high bandwidth network slice for the traffic between Site A and Site B. Thus, traffic has to go through only red double-lined links which represent high bandwidth links. The candidate path for the high bandwidth slice is R_{01}, R_{09}, R_{04}, R_{10}, R_{05}, R_{11}, R_{12}, R_{13}, R_{14} and R_{02}. This path will be enforced by R_{01} for traffic classified as high bandwidth with an uSID policy. Again, the set of nodes needed are less than the actual path: R_{09} (uN and uA),R_{10} (uN and uA), R_{11} (uN), R_{12} (uN), R_{14} (uN) and R_{02} (uN and uDX). The resulting policy will fit in two uSID containers: `FCBB:BB00:9:FA94:A:FAA5:B:C` and `FCBB:BB00:E:2:FD00::`. The first will be encoded in the outer destination address while the second in the SRH list of the SRv6 packet.

The source code, instructions on how to reproduce the DEMO and a video showing the two use cases can be found in our repository [5].

4 CONCLUSIONS

In this work, we demonstrated that our Micro SID implementation adds minimal processing and resource utilization with respect to a legacy SRv6 P4 implementation. Moreover, the Micro SID solution is built upon SRv6 and it is fully compatible with it, making it possible to run in mixed scenarios where only some routers support Micro SID.

ACKNOWLEDGMENT

This work has received funding from the Cisco University Research Program Fund and the EU H2020 5G-EVE project.

REFERENCES

[1] C. Filsfils, P. Camarillo (eds.) et al., *Network Programming extension: SRv6 uSID instruction*. IETF, Internet-Drafts, May 2020, url: https://tools.ietf.org/html/draft-filsfils-spring-net-pgm-extension-srv6-usid.
[2] Clarence Filsfils, *SRv6 micro-instructions*, url: https://blog.apnic.net/2020/05/15/srv6-micro-instructions.
[3] C. Filsfils et al, *SRv6 Network Programming*, IETF, Internet-Drafts, October 2020, url: https://tools.ietf.org/html/draft-ietf-spring-srv6-network-programming-24

[4] C. Cascone et al., *Next-Gen SDN Tutorial*, Open Networking Foundation, url=https://github.com/opennetworkinglab/ngsdn-tutorial

[5] *Micro SID DEMO repository*, url: https://github.com/netgroup/europ4_2020