# Achieving Best-path Selection at Line Rate through the SRv6 Live-Live Behavior

Marco Polverini*, Antonio Cianfrani*, Tommaso Caiazzi†, Mariano Scazzariello‡,
Ahmed Abdelsalam§, Clarence Filsfils§, Pablo Camarillo§
*Sapienza University of Rome †Roma Tre University ‡KTH Royal Institute of Technology §Cisco Systems

***Abstract*—The network programming model of the Segment Routing (SRv6) architecture offers the possibility to define new functions aiming at improving the network performance. In this paper we introduce SRv6 Live-Live, a new behavior for the SRv6 data plane. SRv6 Live-Live is based on two primitives: i) traffic *duplication*, performed at the ingress node, and ii) the traffic *de-duplication*, executed at the egress node. The proposed behavior is suitable for the service provisioning of traffic flows having stringent requirements in terms of reliability, low delay and high throughput. Our preliminary performance evaluation, conducted in an emulated environment and realized by using a prototype implementation based on P4, shows that SRv6 Live-Live enhances the performance of the selected traffic flows in challenging network scenarios, characterized by high level of packet corruption/loss and large values of bandwidth-delay products.**

***Index Terms*—SRv6 behavior, network programmability, P4**

## I. Introduction

Segment Routing over IPv6 (SRv6) [1] is recently attracting large interest either from the industry, the standardization bodies and the research community [2]. It is an overlay architecture defined over an IPv6 substrate, which represents a practical realization of the source routing paradigm, where the source node (or the ingress node of the SRv6 domain) decides the path to use to deliver the traffic. SRv6 also supports a flexible and innovative network programming model [3], where standard or custom functions (*i.e.*, behaviors) instantiated in different network nodes, are addressed by specific identifiers (named Segment Identifiers, SIDs). Thus, the source node can specify either the path and a list of functions to apply (*i.e.*, the *segment list*) to packets of a traffic flow, then define a network program.

In the literature, it is possible to find different examples of SRv6 behaviors. A comprehensive list of standard behaviors is defined in [3]. Among them, the most common one is the *End* behavior, that is used to update the current SID with the next in the segment list, when the instruction associated to the current SID is completed. As for the custom behaviors, some interesting examples are reported in [4]–[6]. Xhonneux et al. [4] define a new behavior for the Linux kernel SRv6 implementation, named *End.BPF*. This function allows redirecting the packet to an eBPF program that realizes a network function. A behavior to handle the live migration of virtual machines with no packet loss is presented in [5]. Finally, [6] introduces the *MaxThroughput* behavior, which allows a traffic flow to transit for an SRv6 domain through the highest throughput path.

In this paper, we propose a novel SRv6 behavior, named SRv6 Live-Live, which realizes a traffic replication scheme, based on two main primitives: *duplication* and *de-duplication*. The solution exploits the capability of SRv6 to steer the traffic of a specific flow over multiple paths (spray policy) so that to improve the flow's performance in terms of end-to-end delay and throughput. In order to be transparent with the higher layer protocols that could be sensible to the reception of duplicate packets (*e.g.*, TCP), at the egress node the *de-duplication* primitive is applied. Our preliminary evaluation using a P4 [7] implementation highlights that SRv6 Live-Live is effective in improving the traffic flows performance in challenging network scenarios characterized by a high level of packet corruption/loss and high values of Bandwidth Delay Product (BDP). This finding makes SRv6 Live-Live particularly appealing for interesting use cases such as, service provisioning for Ultra Reliable and Low Latency Communication (URLLC) 5G network slices, traffic protection and TCP throughput acceleration in Wide Area Networks (WANs) [8], and transport protocols performance enhancement in the context of Non Terrestrial Networks (NTNs) [9].

The rest of the paper is organized as follows. In Sec. II we present the working principle of the SRv6 Live-Live behavior, while a preliminary performance evaluation is shown in Sec. III. Finally, Sec. IV concludes the paper and discusses the future work.

## II. SRv6 Live-Live

We consider a SRv6 network domain where ingress routers are responsible for traffic steering and SRv6 encapsulation, and egress nodes perform SRv6 decapsulation. The ingress nodes apply control level policies (such as traffic engineering ones) to incoming traffic, by adding the proper SRv6 header to each incoming packet. For instance, explicit routing is realized by inserting a segment list, *i.e.*, an ordered list of routers identifiers (SIDs) to be crossed in the SRv6 header.

In this work, we define a novel network function for a SRv6 network, referred to as SRv6 Live-Live behavior. The main features of SRv6 Live-Live are:

- it provides an advanced network service to a specific traffic flow, *i.e.*, the Live-Live flow, crossing the SRv6 domain;

- it is a lightweight mechanism implemented only in the ingress and egress nodes associated to the Live-Live flow;
- it exploits all the $N$ available paths, chosen by the network administrator, from the ingress node to the egress one;
- it proactively selects, on a per-packet basis, the best path to be used to traverse the SRv6 domain.

More in detail, SRv6 LIVE-LIVE defines a duplication primitive in the ingress node, *i.e.*, all the packets of the Live-Live flow are replicated across all the available $N$ paths, while the egress node must support the de-duplication function, that is only one of the replicated packets, *i.e.*, the first one received by the egress node, must be forwarded out of the SRv6 domain, while other copies must be dropped. In other words, SRv6 LIVE-LIVE is offering a novel service to a selected (or a subset of selected) flow(s): a real-time packet-level optimization of the network performance, in terms of end-to-end delay and throughput. The path "followed" by each packet of the Live-Live flow will be the one providing the lowest SRv6 domain crossing time among the $N$ paths associated to the SRv6 LIVE-LIVE behavior. Actually, $N$ paths are traversed but, from an external perspective, only the one providing the better performance is used for packet delivery.

The implementation of SRv6 LIVE-LIVE behavior requires the duplication primitive at ingress node and the de-duplication one at egress node. Both primitives exploit the programming approach of SRv6, and in particular the possibility of defining explicit network paths by coding them as segment lists. Anyway, to efficiently implement SRv6 LIVE-LIVE behavior, an innovative feature must be defined: the detection and removal of duplicated packets. More in detail, considering that $N$ different packets are forwarded in the SRv6 domain for each incoming packet of the Live-Live flow, the egress node must be able to: (i) identify the Live-Live flow each received packet belongs to (since many Live-Live flows could co-exist) and (ii) detect the duplicated packets of a packet already received. To implement such operations at the egress node, the ingress node has to insert in the SRv6 header of the Live-Live packets two additional information: the Flow Identifier and the Sequence Number. The Flow Identifier can be computed as an hash function of the classical 5-tuple (IP source address, IP destination address, source port, destination port and protocol); the Sequence Number, associated to each Live-Live flow, is an integer counter to be incremented at each new packet received. The insertion of the Flow Identifier and of the Sequence Number in the SRv6 header is possible by exploiting the availability of the Type Length Value (TLV) optional field: TLV provides metadata for advanced SRv6 processing. Thus, we defined the Live-Live TLV as reported in Fig. 1.

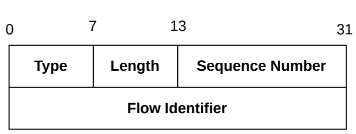
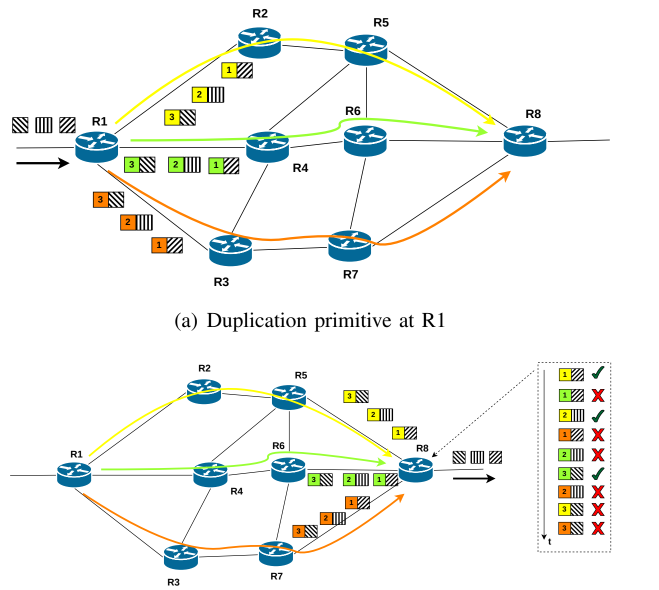



Fig. 1. TLV for LiveLive flows.




(a) Duplication primitive at R1

(b) De-duplication primitive at R8.

Fig. 2. Example of an SRv6 network domain with a LiveLive flow from ingress node R1 to egress node R8 using three network paths.

To summarize the steps of duplication and de-duplication primitives, we consider the example reported in Fig. 2(a) and Fig. 2(b). The SRv6 network domain is composed of eight routers (R1-R8) and we define a Live-Live flow having R1 as ingress router and R8 as egress one. In Fig. 2(a) three incoming packets of the Live-Live flow at router R1 are shown. The ingress node executes the following steps to implement the duplication primitive: (i) traffic steering to check if the packet belongs to a Live-Live flow, (ii) Flow Identifier and Sequence Number computation (a table where storing for each defined Live-Live flow the last used Sequence Number is needed), (iii) insertion of the SRv6 header with the Live-Live TLV field, and (iv) replication of the packet in $N$ copies, each one with a different segment list (a copy for each path to follow). In Fig. 2(a) three paths are used for the Live-Live flow: the different Segment Lists used to encode the paths are represented by three colors (yellow for path R1-R2-R5-R8, green for path R1-R4-R6-R8 and orange for path R1-R3-R7). As a result of the duplication primitive, three copies of an incoming packet are injected in the domain; they all have the same Flow Identifier and Sequence Number in the TLV field, while a different Segment List is encoded in the SRv6 header of each copy. On the egress side, for each packet received belonging to the Live-Live flow, the following steps of the de-duplication primitive are executed: (i) the presence of Live-Live behavior is checked (looking at the last SID of the segment list), (ii) Flow Identifier and Sequence Number are used to check if the packet has been already received (in this case the packet will be dropped) or not, and (iii) if the packet is a "new one", the classical egress operations are performed, *i.e.*, SRv6 header is removed, and the original incoming packet is forwarded out of the domain. In Fig. 2(b) the results of the de-duplication procedures at router R8 are shown: nine SRv6 packets are received and only the three original packets of the Live-Live flow are sent out of the domain. More in detail, the packets received on the yellow path having sequence numbers 1 and 2 and the packet with sequence number 3 received on the green path are accepted, decapsulated and forwarded, while the remaining six packets are discarded.

The SRv6 Live-Live behavior could be of interest for several use cases. It is clearly an additional method to provide different QoS levels: Live-Live flows can be considered as critical flows (such as URLLC ones) that should always receive the highest performance level from the network, without executing any complex and time consuming reservation procedure. Moreover, it can be considered as a mechanism to improve the network reliability experienced by the Live-Live flow, avoiding the adoption of ad-hoc protection schemes acting at lower layers. Finally, SRv6 Live-Live can be of interest in network scenarios where network performance has a significant variability due to the communication technology used (such as for Non Terrestrial Networks).

Regarding the impact of SRv6 Live-Live on the overall network performance, the packets' replication is clearly increasing the network load and thus it could have an impact on normal flows (*i.e.*, no Live-Live). However, we consider our proposal as applicable only for a small amount of network flows, *i.e.*, for an amount of traffic negligible with respect to the overall network traffic.

## III. Evaluation

In this section, we evaluate and discuss the performance of our SRv6 Live-Live proof-of-concept implementation. Source code and scripts to perform the experiments are available at [10].

In the following, we aim to answer the following questions:

Q1. *"What are the benefits of* SRv6 Live-Live *while experiencing packet losses?"*

Q2. *"How does* SRv6 Live-Live *impact TCP performance?"*

Q3. *"What are the gains of* SRv6 Live-Live *with respect to a solution without the de-duplication primitive?"*

**Testbed.** We implemented the algorithm described in Sec. II using the P4 [7] language, and we run it in the `bmv2` software switch [11]. All the experiments are repeated 5 times (average and confidence intervals are shown in the figures), and they are performed on a workstation equipped with Intel® Core-i7™ 10700 CPU and 16 GB of RAM, using the Kathará [12] network emulator.

**Topologies and workload generation.** We generate topologies with an increasing number of paths on the ingress node $e_1$, ranging from 2 to 6 paths, as depicted in Fig. 3. On each path, we use the `tc` tool to set a fixed latency of $40\,ms$ and a variable packet loss, extracted from a $LogN$ distribution with

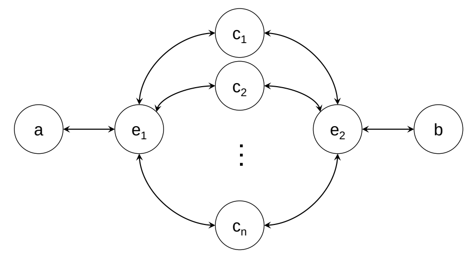



Fig. 3. Evaluated topology.

the following parameters: $\mu = 1; \sigma = 0.7$. We set a fixed random seed so that the experiments are repeatable.

We use the `iperf3` tool to generate traffic, with node $b$ as the server and node $a$ as the client, respectively. Since `iperf3` tries to saturate all the available bandwidth, we limit the maximum throughput to 10 Mbps in order to avoid rate fluctuations during the experiments.

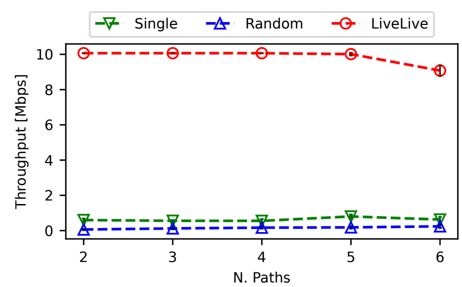
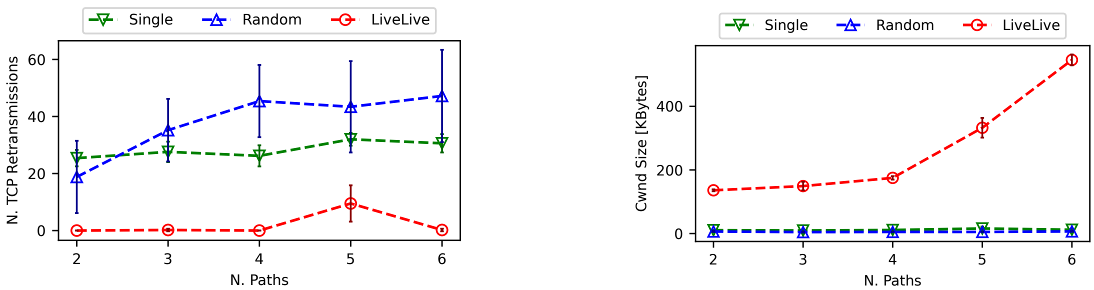



Fig. 4. Output Throughput varying the number of paths.

**Q1. Spreading packets increases overall throughput in case of packet loss.** Fig. 4 shows the output throughput in Mbps (y-axis) over a varying number of paths on the node $e_1$ (x-axis). We performed the experiments using three different configurations on the ingress ($e_1$) and egress ($e_2$) nodes: (i) for each packet, $e_1$ only chooses the path traversing $c_1$, we call it *Single*. *Single* represents solutions that steer the traffic over the best path that has been pre-computed; (ii) for each packet, $e_1$ selects a random path traversing a node $c_n$, we call it *Random*. This approach represents a set of solutions that perform load balancing over multiple paths; and (iii) SRv6 Live-Live. It is possible to notice that the throughput in both *Single* and *Random* configurations is dramatically affected by the packet loss in the network, reaching a maximum of 0.80 Mbps and 0.24 Mbps, respectively. In contrast, SRv6 Live-Live can sustain the 10 Mbps rate in all scenarios, showing that our approach is already effective in delivering packets, even in the presence of high packet loss, exploiting only two distinct paths. Note that the performance decrease in the 6-paths scenario is due to inefficiencies in the `bmv2` software switch implementation, that cannot process/forward incoming packets in a timely manner, resulting in unwanted delays that reduce throughput. We expect to achieve line rate performance in every scenario using programmable hardware.




(a) Number of TCP Retransmissions.

(b) TCP Congestion Window Size.

Fig. 5. SRv6 Live-Live evaluation on TCP performance with a variable number of paths.

**Q2. SRv6 Live-Live improves TCP performance.** To evaluate the impact of SRv6 Live-Live on TCP performance, we measured how the congestion window size and the number of retransmissions varies with respect to a varying number of paths, using the same configurations of Q1. SRv6 Live-Live effectively improves the overall performance. Fig. 5(a) confirms that SRv6 Live-Live avoids TCP retransmissions, effectively preventing packet losses with respect to *Single* and *Random* configurations. Moreover, Fig. 5(b) shows that the congestion window in the SRv6 Live-Live configuration is higher than the other two cases, reaching a size 500× greater in the 6-paths scenario. This finding also demonstrates that an increasing number of paths brings increasing benefits for the selected SRv6 Live-Live flows. Overall, it is worth noticing that using multiple paths requires a careful trade-off to mitigate the impact of SRv6 Live-Live on the other flows transmitted on the same links. Also, using `bmv2` can lead to unexpected behaviors under heavy load, as already described in Q1. Indeed, the 5-paths scenario presents some unwanted variations in the number of TCP retransmissions in the SRv6 Live-Live case. We will further investigate if the SRv6 Live-Live mechanism introduces such fluctuations.

**Q3. The de-duplication primitive is crucial for SRv6 Live-Live.** One may wonder if an alternative design that only implements the duplication primitive brings the same benefits of SRv6 Live-Live, that also entails the de-duplication primitive on the SRv6 egress node. To prove that, we compared our system with an implementation (called *No De-duplication*) that does not de-duplicate packets on $e2$. Fig. 6 shows the output throughput in Mbps over a varying number of paths on the node $e_1$. Differently from SRv6 Live-Live, the *No De-duplication* configuration can sustain the 10 Mbps throughput only in the 2-paths scenario. Indeed, the performance rapidly degrades with more paths, due to the increasing number of copies of the same packet (with the same TCP sequence number) received by node $b$. Such copies are eventually dropped by the receiver, but they still congest the link, degrading performance. This experiment demonstrates that the de-duplication primitive is a fundamental mechanism for SRv6 Live-Live to exploit packet spreading effectively.

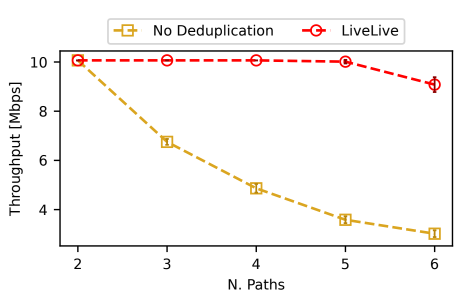



Fig. 6. Comparison between SRv6 Live-Live and an implementation that does not deduplicate packets.

## IV. Conclusion

In this paper, we have defined and implemented a new behavior for the SRv6 architecture, named SRv6 Live-Live, which performs the best path selection at line rate. Our preliminary performance evaluation highlighted that SRv6 Live-Live is effective in boosting the performance of TCP connections in challenging network scenarios, and that the proposed *de-duplication* primitive plays a key role in the defined procedure. As a next step, we aim at performing a deeper analysis of the performance, in terms of end-to-end delay, packet loss and throughput, by considering realistic application scenarios, such as the case of URLLC slices or communications over NTNs. Furthermore, we will study the scalability of SRv6 Live-Live using programmable hardware.

**Acknowledgements.** We would like to thank Dr. Andrea Chiarato for the support in the design and implementation of SRv6 Live-Live.

## References

[1] C. Filsfils, S. Previdi, L. Ginsberg, B. Decraene, S. Litkowski, and R. Shakir, "Segment Routing Architecture," RFC 8402, Jul. 2018. [Online]. Available: https://rfc-editor.org/rfc/rfc8402.txt

[2] P. L. Ventre, S. Salsano, M. Polverini, A. Cianfrani, A. Abdelsalam, C. Filsfils, P. Camarillo, and F. Clad, "Segment routing: A comprehensive survey of research activities, standardization efforts and implementation results," *IEEE Communications Surveys & Tutorials*, 2020.

[3] C. Filsfils, P. Camarillo, J. Leddy, D. Voyer, S. Matsushima, and Z. Li, "Segment Routing over IPv6 (SRv6) Network Programming," RFC 8986, Feb. 2021. [Online]. Available: https://rfc-editor.org/rfc/rfc8986.txt
[4] M. Xhonneux, F. Duchene, and O. Bonaventure, "Leveraging ebpf for programmable network functions with ipv6 segment routing," in *Proceedings of the 14th International Conference on Emerging Networking EXperiments and Technologies*, 2018, pp. 67–72.
[5] Y. Desmouceaux, M. Townsley, and T. H. Clausen, "Zero-loss virtual machine migration with ipv6 segment routing," in *2018 14th International Conference on Network and Service Management (CNSM)*. IEEE, 2018, pp. 420–425.
[6] M. Polverini, D. Aureli, A. Cianfrani, F. G. Lavacca, and M. Listanti, "Enhancing the srv6 network programming model through the definition of the maximize throughput behavior," in *NOMS 2022-2022 IEEE/IFIP Network Operations and Management Symposium*, 2022, pp. 1–5.
[7] P. Bosshart, D. Daly, G. Gibb, M. Izzard, N. McKeown, J. Rexford, C. Schlesinger, D. Talayco, A. Vahdat, G. Varghese, and D. Walker, "P4: Programming Protocol-Independent Packet Processors," *SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 3, p. 87–95, jul 2014. [Online]. Available: https://doi.org/10.1145/2656877.2656890
[8] J. Lee, P. Sharma, J. Tourrilhes, R. McGeer, J. Brassil, and A. Bavier, "Network integrated transparent tcp accelerator," in *2010 24th IEEE International Conference on Advanced Information Networking and Applications*. IEEE, 2010, pp. 285–292.
[9] M. Giordani and M. Zorzi, "Non-terrestrial networks in the 6g era: Challenges and opportunities," *IEEE Network*, vol. 35, no. 2, pp. 244–251, 2020.
[10] "Github: Srv6 live-live p4," https://github.com/Skazza94/srv6_live_live_p4/tree/main, accessed: 2023-10-27.
[11] p4lang, "Behavioral Model (bmv2)," https://github.com/p4lang/behavioral-model. [Online]. Available: https://github.com/p4lang/behavioral-model
[12] M. Scazzariello, L. Ariemma, and T. Caiazzi, "Kathará: A Lightweight Network Emulation System," in *NOMS 2020 - 2020 IEEE/IFIP Network Operations and Management Symposium*, 2020.