Investigating on Black Holes in Segment Routing Networks: Identification and Detection

Marco Polverini, Antonio Cianfrani, Marco Listanti, *Life Member, IEEE*, Giulio Siano, Francesco Giacinto Lavacca, and Carlo Candeloro Campanile

***Abstract*—Network Black Holes (BHs) are logical failures that create a service disruption for a subset of traffic flows, generally due to device misconfiguration. Detection of a BH is a hard task due to its specific nature: the infrastructure is up and the disconnection affects a limited number of flows. An example of BH is the one caused by the failure of the Path MTU Discovery procedure in IPv6. The Segment Routing (SR) Architecture is an overlay infrastructure that provides source routing support by exploiting the connectivity service offered by the underlay IPv6 (SRv6). Thus, SR inherits the problems related to BHs affecting IPv6. In SR this problem is even more stressed due to the encapsulation mechanism that is required to enforce the segment lists on packets. Even worse, existing active probing based tools to detect network BHs for IPv6 are not suitable in SR. In this paper we investigate the problem of detecting SR Black Holes in SR domains. As first, we provide an experimental demonstration of the creation of an *SR Black Holes*. Then we show that existing tools based on active probing are not suitable to detect SR BHs. Then, a passive framework named *Segment Routing Black Holes Detection* (*SR-BHD*) is introduced. *SR-BHD* makes use of specific traffic counters available in SR capable nodes to verify the validity of the flow conservation principle on each network element. Experimental evaluation carried out through simulation and emulation shows the effectiveness of *SR-BHD* in detecting the presence of SR BHs.**

***Index Terms*—Segment routing, network black hole, failure detection, network monitoring.**

I. Introduction

THE ADVENT of the Network Function Virtualization (NFV) paradigm and the need to create complex services by configuring the Service Function Chains (SFCs), have pushed forward routing technology enhancements. With reference to the control plane functionalities the introduction of the Software Defined Networking (SDN) has allowed the definition and implementation of efficient and flexible routing algorithms. Considering the data plane, this requirement is efficiently provided by the Segment Routing (SR) architecture [1]. SR leverages the concept of source routing to let the source node to declare the set of instructions (either topological or service based) to apply on each packet. In particular, SR defines a powerful network programming model [2] that offers an unprecedented expressiveness in the definition of network programs to be applied on traffic flows. An instruction, referred to as *segment* in the SR jargon, allows to specify the type of function to execute (and eventually to pass a set of arguments as input) and the node that has to perform it, also known to as *locator*. A network program is then represented by a *segment list*, i.e., an ordered list of segments.

To reduce the burden of network nodes and make the architecture scalable, network programs are directly inserted in the packet header. In particular, since SR uses the IPv6 data plane (SRv6, [3]), the segment list is included in a new extension header named *Segment Routing Header* (SRH). So, the flexibility of the SR architecture comes at the price of an overhead increase. If, on one hand, this cost is affordable considering the more and more growing of the backbone link capacities, on the other hand longer packets being transported over an IPv6 data plane causes potential creations of anomalies in the packet forwarding. In fact, it is well known that IPv6 has problems [4] with the correct handling of the Maximum Transmission Unit (MTU), since the fragmentation operation is allowed only at the source node. Communication failures due to silent discard of too big packets are known in IPv6 to as *black holes* [5].

The working principle of SRv6 further stresses the issues related on MTU constraint violation in IPv6. It is strongly suggested within an SR domain [3] to use a greater MTU value than the one at the ingress edges. Furthermore, efforts to reduce the overhead required to enforce a network program on packets have been recently made by defining the concept of microSIDs [6]. In a nutshell, a microSID is a special instruction able to encode a whole segment list into a single segment identifier. Nevertheless, while following the recommendation specified in [3] and using the microSIDs, the risk of network black hole events can be reduced, the problem is still present.

There are several factors that contribute to the creation of a black hole. Among those, the most critical one is that network programs can potentially be enforced at every node in the network on the basis of the verification of a logical condition (e.g., re-routing policy to bypass a failed link [7], or redirect traffic in case of Virtual Machine migration [8]). To better explain this concept, let Δ be the difference between MTU of the bottleneck link in the SR domain and the length of a packet that is entering the domain. A black hole can happen if the overhead $\mathcal{O}$ due to the enforcement of network programs on the packet becomes greater than Δ. It is evident that the

Manuscript received 17 May 2022; accepted 2 August 2022. Date of publication 9 August 2022; date of current version 7 March 2023. The associate editor coordinating the review of this article and approving it for publication was C. Avin. *(Corresponding author: Marco Polverini.)*

Marco Polverini, Antonio Cianfrani, and Marco Listanti are with the Department of Information Engineering, Electronics and Telecommunications, University of Roma "Sapienza," 00184 Rome, Italy, and also with the Consorzio Nazionale Interuniversitario per le Telecomunicazioni, Pisa, Italy (e-mail: marco.polverini@uniroma1.it).

Giulio Siano, Francesco Giacinto Lavacca, and Carlo Candeloro Campanile are with the Department of Information Engineering, Electronics and Telecommunications, University of Roma "Sapienza," 00184 Rome, Italy.

Digital Object Identifier 10.1109/TNSM.2022.3197453

recommendation specified in [3] aims at increasing Δ, while the use of microSIDs tries to reduce $\mathcal{O}$.

In [9] we have conjectured the existence of a silent network failure in SRv6 due to the MTU constraint violation, i.e., an *SR Black Hole*. Furthermore, we have discussed through an application example that classical detection tools using active approaches (i.e., the transmission of probes) fail in diagnosing the presence of an *SR Black Hole*. These methods work according to the fate sharing paradigm, i.e., they assume that probe and data packets share the same network "fate." Unfortunately, since SR architecture follows a policy routing approach, there is no guarantee that probes and data packets follow the same path. Consequently active detection tools are not suitable in case of *SR Black Holes*.

In this paper we address the problem of detecting MTU related black holes in an SRv6 network. In particular, we propose a passive monitoring framework that is able to accurately detect the presence of *SR Black Holes*, providing as output a short list of suspected link/flow pairs, i.e., the list of flows impacted by black holes and the links causing such black holes. The proposed framework, named *Segment Routing Black Holes Detection* (*SR-BHD*) uses a passive approach based on the observation of traffic counters available in SR capable nodes [10]. The present paper extends the seminal idea presented in [9], by achieving the following improvements:

- we prove, through an experimental demonstration, the existence of *SR Black Holes*, that was conjectured in [9];
- we show that *SR Black Holes* cannot be trustworthy detected by means of an active probing based tool;
- we extend the framework presented in [9] to make it work also in realistic scenarios where multiple sources of packet loss exist;
- we define a procedure based on the availability of extra information provided by specific SR traffic counters, to improve the performance of *SR-BHD*;
- we run an extensive evaluation, including a sensitivity analysis, to assess the performance of the proposed framework in terms of *SR Black Holes* detection;
- we present a prototype implementation to validate the effectiveness of the detection through the proposed passive approach.

The rest of the paper is organized as follows. We review the literature in Section II. The experimental demonstration of the existence of the *SR Black Hole* and the inability of detecting it through an active approach is presented in Section III. Section IV introduces the proposed passive monitoring framework, while the performance evaluation through simulation is presented in Section V. The validation over the real testbed is discussed in Section VI. Finally, Section VII concludes the paper.

II. Related Works

In this section we provide an overview of the research activities related to network black holes. In particular, we divide the literature in two categories: i) known types of network black holes and existing frameworks for their detection, and ii) performance measurements tools in the context of Segment Routing architecture.

A. Network Black Holes and Detection Frameworks

As defined in [11], network black holes are silent logical failures, often caused by events such as misconfiguration or software bugs. Among the different causes, the use of overlay architectures seems to be a common accelerator for the creation of black holes. A first example is reported in [12], where different failure modes that lead to the occurrence of a black hole are presented, in the context of an IP over MPLS infrastructure. In this scenario, failures affecting the Label Distribution Protocol (LDP) execution can create a black hole, due to the fact the underlying IGP domain is working correctly, while end to end reachability is compromised.

The present work is focused on black holes occurring in an SRv6 network due to the violation of the MTU constraint caused by the failure of the Path MTU Discovery (PMTUD). The different types of failure modes for the PMTUD procedure are described in [5]. Among those, the most common one is represented by unresponsive routers, that are configured to not send ICMP Packet Too Big (PTB) messages back to the source node whenever a packet with a length exceeding the MTU is received. Many different detection systems have been proposed to detect the presence of network black holes in different contexts. All of them rely on the active test of the network status through the sending of probes. In the next we describe some of the most relevant detection tools.

In [11] an active probing detection mechanism is defined; it is able to detect network black holes occurring in an IP/MPLS backbone. A full mesh of probes are periodically exchanged among the edge routers. The method is based on the concept of *failure signature*, that represents the set of probes that are lost in case a black hole occurs in a specific link. Then, spatial correlation is exploited to identify a set of suspicious links. In particular, all the links whose failure signature is close to the actual set of failed probes are inserted into an hypothesis set.

One of the most reliable tools to discover PMTUD failures is Scamper [5]. Scamper is a two steps procedure to determine either the largest MTU that can be used on a end to end path, and to discover (in case of a failure) what is the router that is not participating to the PMTUD. Both the phases of Scamper are based on the enforcement of probes along the end to end path to check. These probes consist in a set of UDP segments destined to an unused port, when performing the first step, whereas a set of ICMP Echo messages destined to intermediate routers are used in the second phase.

Netalyzr is presented in [13], it is a network measurement and debugging tool to monitor the Internet. The architecture is provided with a set of pre-installed applets; one of these aims at determining the path MTU toward a destination server. This search is based on a process that emits a set of UDP probes to the target destination.

Ripe Atlas [14] is a worldwide monitoring infrastructure based on the use hardware probes placed in the so called vantage points. In [4] Ripe Atlas has been used to discover path MTU black holes in the Internet, with the specific focus of assessing the main causes and the most affected data plane protocol. The obtained results show that black holes due to failure in the PMTUD procedure affect both the IPv4 and the

IPv6 data planes. Specifically, Ripe Atlas is able to detect the main causes and the location of these failures, such as PTB messages and fragmented packets filtered by firewalls.

In this paper we extend the seminal idea described in [9], i.e., a passive approach based on the elaboration of traffic-related data available in SRv6 network devices. This approach has already been successfully applied in the past to identify and detect network anomalies and failures. As an example, in [15] the network tomography is exploited to identify anomalous traffic flows, while [16] that defines a statistical analysis based on Signal Processing techniques and applies it over SNMP MIB data in order to detect different types of network anomalies, such as, file server failure due to abnormal user behavior or protocol implementation errors. Nonetheless, no passive monitoring tool has ever targeted the detection of *SR Black Holes*.

B. SR Performance Measurement Tools

SR architecture provides a set of Operation And Maintenance (OAM) tools that Network Operators can use to measure the performance of their infrastructure, execute troubleshooting operations, and so on. Here we report few examples. In [17] it is presented a scalable and topology-aware data-plane monitoring system for SR-MPLS. Ping and Traceroute for SR networks are defined in [18]. Bidirectional Forwarding Detection (BFD) to test the aliveness of Segment Routing Policies for Traffic Engineering is presented in [19].

SR capable nodes are able to collect statistics on the traffic by exploiting a set of traffic counters called SR Routing Traffic Counter (SRTCs), allowing to perform traffic measurements at different granularity. By means of SRTCs an SR node is able to collect statistics on the received traffic flows aggregating them according to the active segment. Three different types of SRTCs are used in the proposed framework: i) SR-INT, ii) PSID, and iii) POL. SR-INTs (also known to as link counts) account the traffic at link granularity, i.e., they measure the amount of SR traffic that is sent over a specific link. By means of the PSID counters, an SR capable node can account the amount of received traffic that is directed toward a specific node. Finally, POL counters keep track of the amount of traffic that has been steered through a specific SR policy. These counters are described in detail in [20], where the logical relations between them are captured by a mathematical model. Exploiting this model, the Authors show SRTCs can successfully used to improve the performance of existing algorithms in different networking problems (e.g., Traffic Matrix Assessment, Traffic Anomaly Detection, etc.). The present paper is strongly based on the findings reported in [20].

In [21] an SRv6 Performance Monitoring (SRv6-PM) framework is proposed, allowing to perform a deep performance monitoring on an SRv6 infrastructure. Three main components are defined: i) a set of data plane tools for performing traffic measurements (e.g., packet loss, delay) at line rate, ii) a control plane logic that requires to the network nodes to perform specific measurements (and a southbound interface for the data/control plane interaction), and iii) a Cloud Native Big Data Management system for data storage, processing and visualization. As use case for the validation of SRv6-PM, the fine grained measurement of the packet loss level affecting a single SR flow is considered. To measure the amount of packets that are lost for a specific flow, SRv6-PM exploits SR traffic counters instantiated at the ingress and egress nodes. Specifically, the difference among these two counters represents the overall number of lost packets for the target flow.

SCMon [22] is a network wide monitoring system that allows to check the health status of links. It exploits the source routing and the flexibility achieved by SR to create a set of monitoring cycles where to send probes to the aliveness of each them. By properly designing the different cycles it is possible to precisely localize a failed link. Furthermore, SCMon exploits adjacency SIDs of SR to test the status of IP links composed by bundle of connections at layer 2 (the inability to do that is a major drawback of the classical systems based on Bidirectional Forwarding Detection, BFD).

C. Segment Routing Background

Segment Routing (SR) [23] is a novel network paradigm based on source routing, i.e., the source node decides the path that each packet has to go through. The end to end paths are encoded as an ordered list of instructions, also referred to as segments. Thus the full list of segments is named Segment List (SL). Segments are expressed as labels, named Segment IDentifiers or SIDs. In the case of SRv6, the underlay data plane is based on IPv6 and a SID is an IPv6 address.

In SRv6 packet forwarding works as follows. When the border router receives a packet, it has to steer it over a specific SL. This last is chosen according to a given *SR Policy*. After a packet has been processed according to a matched *SR Policy*, its most external IPv6 header is extended by the inclusion of a SR Header (SRH). This last containing the SL and a pointer identifying the active segment, i.e., the current SID to be used for packet forwarding. In particular, the active segment is copied in the destination address field of the outer IPv6 header. Transit routers forward incoming packet by inspecting the IPv6 destination address of the outer header. Once the node having the same SID of the active segment is reached, the SID-related function must be applied (many functions can be defined). A common function is the END one, which implies that the active segment has to be updated, thus the pointer moves to the next SID in SL. A further action that can be performed is the enforcement of a SL by inserting a new policy. Finally, before the packet leaves the SR domain, the SRH has to be removed.

III. Experimental Demonstration of Possible Existence of SR Black Holes

The goal of this section is twofold: i) to experimentally demonstrate the existence of possible *SR Black Holes* in an SRv6 networks, and ii) to show that active probing based tools are not trustworthy for detecting such a type of failures.

The test is conducted over an emulated network, created through virtual routers supporting SRv6 as data plane technology. Vector Packet Processor (VPP) [24] with SRv6 plugin

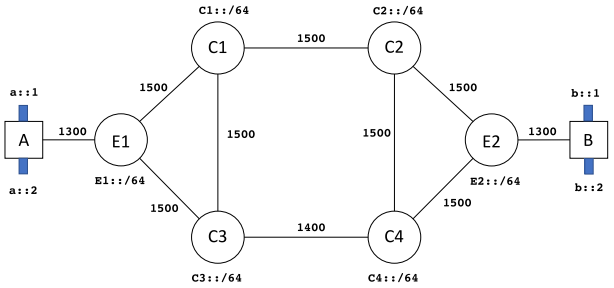


Fig. 1. Reference scenario.

TABLE I
MAIN FEATURES OF THE POLICIES CONFIGURED IN THE EMULATED NETWORK SCENARIO

Policy Name	Scope	Head end node	Segment List
*pol*_1	high reliability	E1	C1,C2,E2
*pol*_2	best effort	E1	C3,C4,E2
*pol*_3	link bypass	C1	C3,C4

```
Control connection MSS 1228
Time: Fri, 14 May 2021 12:25:20 GMT
Connecting to host b::1, port 80
        Cookie: 6iv4fwamqo52yirrfxmwqzqvbaneujryoy3h
        TCP MSS: 1228 (default)
[ 5] local a::1 port 48300 connected to b::1 port 80
Starting Test: protocol: TCP, 1 streams, 131072 byte blocks, omitting 0 seconds,
 1 blocks to send, tos 0
[ ID] Interval           Transfer     Bitrate         Retr    Cwnd
[  5]   0.00-0.00  sec  43.2 KBytes   321 Mbits/sec    0    12.0 KBytes
- - - - - - - - - - - - - - - - - - - - - - - - -
Test Complete. Summary Results:
[ ID] Interval           Transfer     Bitrate         Retr
[  5]   0.00-0.00  sec  43.2 KBytes   321 Mbits/sec    0    sender
        Sent 43.2 Kbyte / 10.2 Kbyte (424%) of /home/giulio/Desktop/BlackHolesFi
nal3/big_file
[  5]   0.00-0.01  sec  0.00 Bytes    0.00 bits/sec         receiver
CPU Utilization: local/sender 26.8% (0.0%u/26.8%s), remote/receiver 0.7% (0.0%u/
0.7%s)
snd_tcp_congestion       cubic
rcv_tcp_congestion       cubic

iperf Done.
```

Fig. 2. Snapshot of the Iperf window on client node.

```
-----------------------------------------------------------
Server listening on 80
-----------------------------------------------------------
Time: Fri, 14 May 2021 12:25:20 GMT
Accepted connection from a::1, port 48300
        Cookie: 6iv4fwamqo52yirrfxmwqzqvbaneujryoy3h
        TCP MSS: 0 (default)
[ 5] local b::1 port 80 connected to a::1 port 48300
Starting Test: protocol: TCP, 1 streams, 131072 byte blocks,
 1 blocks to send, tos 0
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Test Complete. Summary Results:
[ ID] Interval           Transfer     Bitrate
[  5] (sender statistics not available)
[  5]   0.00-0.01  sec  0.00 Bytes    0.00 bits/sec
rcv_tcp_congestion       cubic
iperf 3.7
```

Fig. 3. Snapshot of the Iperf window on server node.

has been used. The experimental topology is shown in Fig. 1. The SRv6 domain is composed of 6 nodes, identified by the locators reported in the figure. Two hosts (A and B) are connected through the SRv6 domain. MTUs of the access links are equal to 1300 Bytes, while MTUs of the internal links are equal to 1500 Bytes, except the link between nodes C3 and C4 that has a lower MTU, equal to 1400 Bytes. We highlight that the considered setting for the MTU is compliant with the recommendation reported in [25], which suggests to configure MTUs of the links internal to the SR domain bigger than those associated to the external links.

Two SR policies, named *pol*_1 and *pol*_2, are configured in the head end node E1. The two policies are reported in Tab. I. In the considered scenario, the two policies impose two levels of reliability of network paths: *pol*_1 is used for a high reliability traffic, while *pol*_2 is associated to the regular traffic. In particular, for each link crossed by the path imposed through the *pol*_1, a backup path allows the link bypass in case of failure. As an example, *pol*_3 reported in Tab. I is configured in node C1; in case of failure of the link between nodes C1 and C2, the policy imposes a re-routing of the packets over an alternative path (C1-C3-C4).

To verify the existence of the *SR Black Hole* two traffic flows are created between hosts A and B. The first one is a TCP connection that requires a reliable transfer through the SRv6 domain. This requirement is satisfied by adding a traffic classifier in the node E1 that steers the packets belonging to the TCP connection along the path specified by the policy *pol*_1. The second one is a series of ICMP Echo Requests, that are sent over the regular path by means of policy *pol*_2. The experiment is repeated two times: the first execution is performed without the introduction of link failure events, while in the second run the link between nodes C1 and C2 fails.

As expected, no black hole has been experienced in the first case. In this situation, the lowest MTU of the links belonging to the path followed by the TCP connection is equal to 1300 Bytes, while the overhead imposed by the use of SRv6 is equal to 96 Bytes.[1] Since the MTU of internal links of the SRv6 domain crossed by the TCP connection is equal to 1500 Bytes, then no MTU violation happens and the flow is correctly delivered to the destination.

A different situation happens when policy *pol*_3 is used to detour the traffic due to the failure status of the link between nodes C1 and C2. In this case the overhead due to SRv6 processing is equal to 176 Bytes. The smallest MTU among the links belonging to the overall path followed by packets of the TCP connection is still equal to 1300, which is the largest size of packets injected in the SRv6 domain. At the same time, the bottleneck MTU inside the SRv6 domain is now equal to 1400 Bytes (due to the detour, the link C3-C4 is now crossed by the TCP connection). In this situation the packets sent through the link C3-C4 have an overall length of 1476 Bytes, which exceeds the MTU, thus leading to a large quantity of packets to be silently dropped. Fig. 2 shows the *Iperf* window at the client side of the TCP connection, where it can be seen the amount of traffic sent and the considered MSS. As shown in Fig. 3, the traffic is not received by the server due to the *SR Black Hole*. It is worth noting that the TCP connection has been successfully established, due to the small size of the messages exchanged during the three way handshake procedure. Furthermore, the ICMP traffic, which is also crossing the link C3-C4, is correctly delivered to the destination.

Summarizing, the previous experiment has proven the existence of a *SR Black Hole*. Clearly, this is an anomalous event that could happen only in case the network is not correctly configured. For instance, in the proposed experiment the sending of ICMP PTB messages was disabled (default configuration in VPP). This suggests that the SRv6 domain must be carefully configured in order to avoid the creation of *SR Black*

[1] 40 Bytes for the outer IPv6 header, 8 Bytes for the SRH and three SIDs each one having a length of 16 Bytes.

Holes, either enabling the sending of ICMP PTB messages on the nodes, and to properly design the policy enforced on the incoming flows. Clearly, misconfiguration is an unplanned and unwanted event, and generally [26] it is extremely hard to be found and corrected. For this reason, in this work we define a framework to help Network Operator to detect *SR Black Holes*.

A. Applying Active Probing Tools to Detect the SR Black Hole

The experimental demonstration about the existence of the *SR Black Hole* has been carried out in a non collaborative network environment, i.e., nodes that do not send ICMP PTB messages. In this subsection we use the Scamper tool [5] to determine its effectiveness in the correct detection of the path MTU. Scamper has been though to detect MTU related black holes in a non collaborative network. It exploits a traceroute like mechanism to accomplish two different tasks: i) find the link where the black hole occurs, and ii) determine the bottleneck MTU. UDP probes are sent along the path between the source and the destination nodes. At first, small UDP probes are sent to test whether the destination is actually reachable or not. Next, a PMTUD process is executed by sending probes with increasing size. In order to deal with unresponsive nodes, Scamper uses a timeout mechanism: if an answer is not obtained at the expiration of the timer, it assumes that the packet has been silently dropped due to the MTU constraint violation. After two consecutive timeout events, Scamper starts determining the maximum size for packets that is supported. In particular it exploits a table of well known MTU values to speed up the process: when a given packet length is detected to exceed the MTU (through the timeout mechanism), the next MTU is fixed equal to the smallest value among the previously tested packet lengths. Once the path MTU value is determined, a traceroute like procedure is used to determine the hop where the bottleneck link is located. Probes size is set bigger than the path MTU and the Time To Live (or Hop Limit in case of IPv6) is incrementally increased. As soon as ICMP Time Exceeded messages are not received, the bottleneck link is found: it is the one connecting the last responding node with its next hop.

A first observation arising from using Scamper in our scenario is that, due to the IP in IP encapsulation performed by the ingress node of the SRv6 domain, the Hop Limit based mechanism for the detection of the location of the bottleneck link does not work whenever this node belongs to the SRv6 domain. In fact, after the encapsulation the Hop Limit of the outer header is set to the default value, that is larger than the one reported in the inner header.

Next we detail the results that we have obtained by using Scamper in the network scenario shown in Fig. 1. The MTU values tested by Scamper at each iteration of its execution are shown in Fig. 4. The value assumed in the last iteration is the one reported as output to the source host, that will generate packets accordingly.

Two experiments are conducted. In the first one, we leave the network configuration unchanged with respect to the setting used in the previous experiment where link between nodes C1 and C2 fails. In this scenario we have already commented that the TCP traffic is encapsulated twice, with an overall overhead due to SRv6 processing of 176 Bytes. Despite of the fact that the link having the smallest MTU in the overall path followed by the TCP traffic is the one connecting the source node to E1, due to the SRv6 overhead, the limit on the packet size is imposed by the link between nodes C3 and C4. As a consequence the maximum allowed size for packets injected in the SRv6 domain is 1224 Bytes. Unfortunately, due to the policy based routing used in SR, the probes sent by Scamper follow a different path with respect to the data traffic; in particular, they are handled through the policy *pol_2*. The final MTU size obtained by Scamper in this case is the one reported by the blue curve in Fig. 4. The returned MTU size is equal to 1300 Bytes, as a consequence, the TCP source will generate packets having a size larger than the one that is supported, thus creating a black hole.

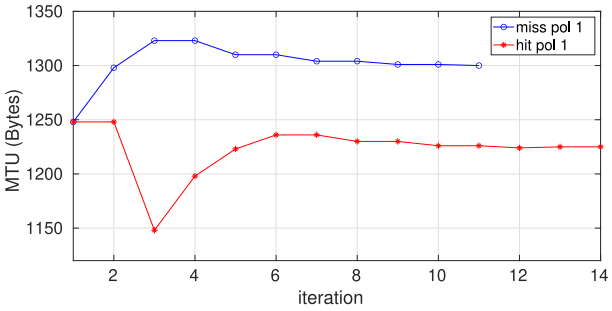


Fig. 4. MTU assessment procedure followed by Scamper.

The previous test has confirmed the intuition that active tools fail in detecting the *SR Black Hole* due to the policy based routing used in SR. To further stress this point, we have performed a second experiment by including in the node E1 a classification rule that forces Scamper probes to follow the same path of the TCP traffic. The outcome of the Scamper execution is represented by the red line reported in Fig. 4. As expected, under this setting Scamper correctly determine that the value of the maximum supported MTU is 1224. In fact, by injecting in the SR domain packets with such length, the enforcement of the *pol_1* and *pol_3* (that determine an SR related overhead equal to 176 Bytes) leads to the maximum size of 1400 Bytes, which is the MTU of the bottleneck link.

To summarize, the presented experiments have shown that an active probing mechanism can fail in detecting an *SR Black Hole*, leading to the creation of *false negatives* which are unacceptable for a detection tool.

IV. Segment Routing Black Holes Detection Algorithm (*SR-BHD*)

SR-BHD algorithm is a passive monitoring system developed to detect *SR Black Holes* in SR networks. In the following, the system model and the notation are presented, then the working principle of *SR-BHD* is introduced in two phases: the ideal model is firstly discussed and some modifications are then introduced in order to make *SR-BHD* robust with respect to "noise signals" (e.g., packet loss due to congestion). Finally,

TABLE II
SYMBOL DESCRIPTION

Symbol	***Description***
$\mathcal{G}(\mathcal{N},\mathcal{L})$, N, L	Network graph - number of nodes - number of links
l, $l.t$, $l.h$	link - link tail - link head
$y_L(l)$, $y_B(i,a)$, $y_P(i,e,c)$	link count - PSID counter - POL counter
$<i,e,c>$, $\Pi_{i,e,c}$,$\sigma_{i,e,c}$	policy between nodes i and e with color c - set of application flows steered through the policy - segment list
$g_{i,a}(l)$	fraction of flow sent over the link l if one unit of traffic is sent using the IGP path between nodes i and a
$r_{i,e,c}(l)$	fraction of flow sent over the link l if one unit of traffic is steered through the SR policy $<i,e,c>$
$b^{n,a}_{i,e,c}$	fraction of flow accounted by PSID counter at node n with active segment a if one unit of traffic is steered through the SR policy $<i,e,c>$
$\mu(l)$,C_l	- margin over the link l - capacity of link l

an enhanced version of *SR-BHD*, called *SR-BHD*$^+$, based on the use of advanced traffic counters, is proposed to improve the precision of *SR-BHD*.

A. System Model

Let $\mathcal{G}(\mathcal{N},\mathcal{L})$ be the graph representing the topology of the considered SR domain, where $\mathcal{N}$ and $\mathcal{L}$ are the set of N nodes and L links respectively. Considering a link l, the head node is indicated with the notation $l.h$ and the tail node to as $l.t$ (the link leaves the tail and enters the head). With reference to the node $i \in \mathcal{N}$, its node-SID is represented with the same symbol. A summary of the notation is reported in Tab. II.

In this scenario three different types of traffic counters are available. The first one is named *link count*, $y_L(l)$, and accounts the amount of traffic (expressed in number of bytes) transmitted over the link l. The collection of all the link counts is represented by the vector $\mathbf{y}_L$. Furthermore, Prefix Segment IDentifiers (*PSID*) and POLicy (*POL*) counters [20] are used to get statistics on the SR traffic. The PSID counter instantiated at node i for the segment identifier a, referred to as $y_B(i,a)$, accounts all the packets received at node i having a as active segment. The collection of all the PSID counters is represented by the vector $\mathbf{y}_B$. SR capable nodes can enforce the segment list on each incoming packet on the basis of an SR policy. An SR policy is represented by the tuple $<i,e,c>$, where i and e are the ingress and egress points of the SR tunnel and c is the color, that encodes the scope of the policy (i.e., low latency, best effort, high reliability, etc.). Some examples of SR policies are reported in Tab. I, which includes two policies between the same ingress and egress nodes ($E1$ and $E2$ in the example) having two different colours (high reliability and best effort). Each policy has an associated traffic counter that accounts for the traffic that is steered through it. This quantity is indicated with the symbol $y_P(i,e,c)$. The collection of all the POL counters is represented by the vector $\mathbf{y}_P$. Some practical examples of PSID, POL and link counters are reported in Fig. 5.

Let us refer with the symbol $\mathcal{F}$ to the set of application flows (e.g., HTTP, SSH, etc.) injected in the SR domain. In order to cross the SR domain, each application flow f should be steered

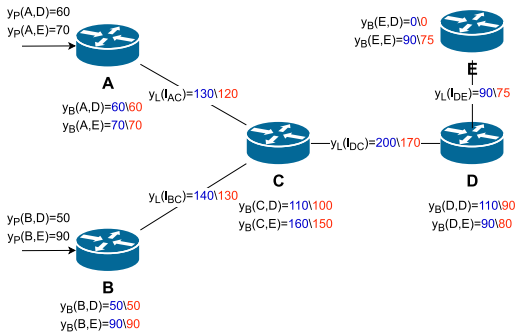


Fig. 5. The toy case example.

through an SR policy.[2] Considering the policy $<i, e, c>$, the set of application flows that are handled through it is referred to as $\Pi_{i,e,c}$. We define an *SR flow* to as the aggregated traffic that is represented by the summation of all the application flows that are steered through the same policy. An SR flow is indicated with the symbol $x_{i,e,c}$. The collection of all the SR flows is represented by the vector $\mathbf{x}$.

In the IPv6 underlay layer the shortest path policy is used to compute the path between every pair of nodes of the SR domain. The underlay path to go from node i to node a is referred to as $\mathbf{g}_{i,a}$ and is represented by a vector of length L, whose l-th component is equal to the percentage[3] of traffic that goes over the link l if the node i sends one unit of traffic toward node a. With reference to the overlay, the routing is determined by the segment lists that are currently configured. The segment list associated to the SR policy $<i, e, c>$ is referred with the symbol $\sigma_{i,e,c}$ and consists of an ordered sequence of node-SIDs. Then, the vector $\mathbf{r}_{i,e,c}$ representing the overlay path followed by the SR flow $x_{i,e,c}$ can be calculated using the following Eq. (1):

$$\mathbf{r}_{i,e,c} = \sum_{k=1}^{|\sigma_{i,e,c}|-1} \mathbf{g}_{\sigma_{i,e,c}[k],\sigma_{i,e,c}[k+1]} \tag{1}$$

The Eq. (1) shows that the overlay routing of an SR flow is given by a linear combination of the vectors representing the paths in the underlay between nodes included in the segment list. The routing matrix $\mathbf{R}$ is the collection of the vectors $\mathbf{r}_{i,e,c}$ for all the existing policies, having L rows and as many columns as the number of flows.

With reference to the PSID counter instantiated at node n for the active segment a, the variable $b^{n,a}_{i,e,c}$ represents the percentage of the SR flow $x_{i,e,c}$ that it accounts. The matrix $\mathbf{B}$ is given by the collection of the $b^{n,a}_{i,e,c}$ variables for all the PSID counters and SR flows. Then, the relation between the SR flows and the value assumed by each PSID counter can be expressed through the Eq. (2).

$$\mathbf{y}_B = \mathbf{B} \cdot \mathbf{x} \tag{2}$$

[2]In practice a set of application flows are merged and steered using the same SR policy.

[3]SR exploits the presence of Equal Cost Multi Paths (ECMP) in the underlay.

For the ease of comprehension, in Fig. 5 we introduce a simple toy case scenario. The toy case scenario is composed of a 5 nodes network and four SR flows ($x_{A,D}$, $x_{A,E}$, $x_{B,D}$ and $x_{B,E}$).[4] The SR traffic flows are steered over the shortest paths by the enforcement of segment lists of length 1, in which the only reported SID is the one that identifies the destination node. In the toy case scenario we assume that $x_{A,E}$ is lost in a black hole experienced on the link between nodes *C* and *D*. Furthermore, due to congestion, 5 units of traffic are dropped for each flow over each link. The numbers reported in blue and red in Fig. 5 represent the traffic sent over each link in the congestion free (no packet loss due to congestion) and real cases, respectively. Finally, in Fig. 5 the PSID counters of each node in the congestion free and real cases are also reported.

With reference to Eq. (2) and the example reported in Fig. 5, we have that $x_{A,D}$ and $x_{B,D}$ cross node *C* having *D* as active segment, thus $b_{A,D}^{C,D} = 1$ and $b_{B,D}^{C,D} = 1$. Then, according to Eq. (2), the counter $y_B(C, D)$ is equal to the summation of $x_{A,D}$ and $x_{B,D}$. The validity of this condition can be easily verified by inspecting Fig. 5.

B. SR-BHD Principle

SR-BHD is inspired by the flow conservation principle [27] which imposes that, considering a network node, the difference between the incoming and outgoing flows is equal to the local demand. Clearly, in case of a *SR Black Hole* event on a link, the flow conservation principle is violated. In fact, a portion of the flow that should leave the node through the link affected by the *SR Black Hole*, is dropped due to the MTU constraint. Consequently, the balance between the incoming and outgoing traffic is not more satisfied. Clearly, the occurrence of a black hole is not the only event that induces a violation of the flow conservation principle. Congestion, binary errors and route missing (i.e., destination unknown) are examples of other causes of packet loss leading to a violation of the equilibrium between traffic entering and leaving a node. We first show the equations that regulate the flow conservation principle; then, we describe a method to include the different sources of packet loss into the mathematical model.

SR-BHD periodically monitors the traffic statistics collected by the network devices and verifies the validity of the flow conservation principles, and eventually raises an alarm for a possible *SR Black Hole*. Two different types of equations are defined in *SR-BHD* to check the validity of the flow conservation principle:

$$\sum_{a \in \mathcal{N}} g_{i,a}(l) \cdot y_B(i, a) = y_L(l) \forall l \in \mathcal{L} \tag{3}$$

$$y_B(i, a) = \sum_{l \in \delta_i^+} g_{l.t,a}(l) \cdot y_B(l.t, a) \forall i, a \in \mathcal{N} \tag{4}$$

Eq. (3) imposes the flow conservation principle at the link level; the overall amount of traffic received at node *i* and to be forwarded over the output link *l* is equal to the amount of traffic actually sent over link *l*. In fact, each PSID counter accounts the traffic received at node *i* and having *a* as active segment. This traffic is routed according to the underlay path between the nodes *i* and *a*, that is represented by the vector $\mathbf{g}_{i,a}$. The multiplication of the *l*-th component of this vector with the value assumed by the PSID counter returns the portion of the traffic that is routed over the link *l*. Summing up the contribution of each possible active segment it is possible to calculate the amount of the received traffic that node *i* forwards over the output link *l*. In the ideal case (no losses) this quantity coincides with the link count $y_L(l)$.

As a concrete example, let us consider the application of Eq. 3 over the link between nodes *C* and *D* of the toy case scenario. As reported in Fig. 5 (see blue counters), in absence of congestion the traffic forwarded by node *C* toward the next hop *D* is given by the sum of the PSID counters $y_B(C, D)$ and $y_B(C, E)$, whose value is 110 and 160 respectively, i.e., 270 units of traffic. On the contrary, looking at the link count $y_L(l_{\mathrm{C,D}})$, only 200 units of traffic are actually sent. This situation leads to the violation of the flow conservation principle and to the detection of a potential black hole over link $l_{C,D}$. Anyone of the 4 flows crossing the link could have been potentially lost in the black hole.

Eq. (4), where the symbol δ_i^+ represents the set of incoming links of node *i*, imposes the flow conservation principle at the active segment level. In particular, with reference to the node *i*, the amount of traffic that it receives having *a* as active segment must be equal to the traffic that its neighbors send toward him with the same active segment.

Eq. (4) can be used to reduce the set of suspicious flows. In particular, let us focus the attention on the PSID counter instantiated at node *D* for the active segment *E*. In this situation, Eq. (4) imposes that the PSID $y_B(D, E)$ must be equal to the sum of the PSID counters, related to the same active segment, instantiated in the neighbor nodes having *D* as next-hop in the shortest path to the active segment. In the considered example the condition is that $y_B(D, E)$ should be equal to $y_B(C, E)$. From Fig. 5 (see blue counters) it is evident that Eq. (4) is not satisfied. On the contrary, considering the active segment *D*, the condition holds. The outcome is that the suspicious flows are the ones crossing link $l_{\mathrm{C,D}}$ having *E* as active segment.

It is important to highlight that checking the validity of Eq. (4) is a critical task in a real network, mainly due to the fact that the traffic counters involved in the formula are instantiated at different network nodes. Consequently, lack of alignment of the traffic counters can compromise the validity of the flow conservation principle. On the contrary, the condition imposed by Eq. (3) is robust with respect to alignment errors, since all the involved counters are stored at the same node. Furthermore, the previous equations are valid if only node-SIDs are considered, i.e., adjacency-SIDs are not defined. Nonetheless, the framework can be extended to support the use of adjacency-SIDs.

C. Framework Overview

Fig. 6 shows the main functional blocks of the proposed *SR-BHD* framework, to be integrated in a centralized monitoring

[4]The color is not considered here.

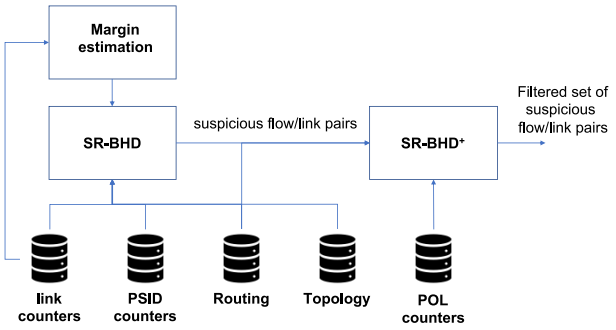


Fig. 6. Scheme of the *SR-BHD* monitoring framework.

Algorithm 1 *SR-BHD* Algorithm Pseudo Code

Require: the network graph: $\mathcal{G} = (\mathcal{N}, \mathcal{L})$, the routing matrix: $\mathbf{R}$, the traffic counters: $\mathbf{y}_L$ and $\mathbf{y}_B$
1: initialize $\mathcal{L}_S \leftarrow \emptyset$, $\mathcal{F}_S \leftarrow \emptyset$ and $tmp \leftarrow \emptyset$
2: **for all** $l \in \mathcal{L}$ **do**
3: **if** Eq. 3 for link l does not hold **then**
4: $\mathcal{L}_S \leftarrow l$
5: $tmp \leftarrow l.h$
6: **end if**
7: **end for**
8: **for all** $n \in tmp$ **do**
9: **for all** $a \in \mathcal{N}$ **do**
10: **if** Eq. 4 for node n and active segment a does not hold **then**
11: $\mathcal{F}_S \leftarrow \left\{ x_{i,e,c} : r_{i,e,c}(l) > 0, l \in \mathcal{L}_S \cap \delta_n^+ \wedge a \in \sigma_{i,e,c} \right\}$
12: **end if**
13: **end for**
14: **end for**
15: **return** the set of suspicious flows $\mathcal{F}_S$ and links $\mathcal{L}_S$.

system. The central monitoring system queries the network devices, through a southbound interface, to collect the traffic statistics, which are stored in different databases. When new traffic measurements are available, the monitoring system triggers the execution of the *SR-BHD* block. It takes as input the link and PSID traffic counters, the network topology and the current routing configuration (both IPv6 paths and SRv6 segment lists). *SR-BHD* block also receives as input the value of the margin,[5] that consists in a vector containing the estimation of the packet loss due to different causes (e.g., congestion, transmission errors, etc.) for each network link. The margin estimation block needs as input the current link utilization.

The output of the *SR-BHD* block is a list of link/flow pairs that are suspected to be affected by an *SR Black Hole*. In order to improve the precision of the output, a refinement step can be performed through the adoption of the *SR-BHD*$^+$ block. Its execution requires the availability of a further set of traffic counters, i.e., the POL ones. Since these counters are not always available in SR capable nodes, the refinement performed by means of *SR-BHD*$^+$ is optional.

D. Algorithm Explanation

The pseudo code of the *SR-BHD* is reported in Algorithm 1. It takes as input the network graph, the overlay routing matrix and the link count and PSID counter vectors. As first, three data structures are initialized as empty (line 1): i) the two sets $\mathcal{L}_S$ and $\mathcal{F}_S$ of suspicious links and flows (potentially involved in a black hole), and ii) the set *tmp* storing the head nodes of suspicious links. After that, the validity of Eq. (3) is tested for each link (lines 2–7). In the case Eq. (3) is not respected, the current link is declared as suspicious and its head node is included in the *tmp* set (lines 4–5). After the identification of the potential *SR Black Hole* location, *SR-BHD* detects the affected SR flows (lines 8–14): for each node in *tmp*, the flow conservation law for all active segments is checked (line 10). If the condition is not respected for active segment a, the set of suspicious flows is updated (line 11). In particular, all SR flows crossing the suspicious link l having a as active segment are included in the set $\mathcal{F}_S$. The outcome of *SR-BHD* is the list of suspicious flows and links (line 15).

It is worth noting that the previously presented implementation of *SR-BHD* works only if *SR Black Holes* are the unique source of packet loss in the network. In a real situation, losses due to different events would be interpreted by *SR-BHD* as the proof of the existence of an *SR Black Hole*, leading to the creation of a large number of false positives. To cope with such events, we introduce a tolerance term in Eqs. 3 and 4, that is referred to as *margin*. Specifically, the margin of link l is indicated by the symbol $\mu(l)$. With this new logic, the flow conservation conditions are violated only if the difference between the traffic sent and the traffic actually transmitted is greater than the margin.

As a concrete example of the importance of the margin, let us refer to the toy case scenario and consider the link $l_{\mathrm{A,C}}$, not experiencing a black hole in the considered case. Nonetheless, if we consider the real case traffic counters (red ones in Fig. 5) which include packet loss events due to congestion, we can immediately see that Eq. 3 is violated, thus leading to the creation of a false positive alarm from the framework. On the other hand, if we assume that the margin over the considered link is $\mu(l_{\mathrm{A,C}}) = 20$ units of traffic (which is an over estimation of the packet loss due to congestion over the considered link), then no alarm is raised, since the difference between the traffic received by node *A* that is forwarded over the link $l_{\mathrm{A,C}}$ (130 traffic units) and the traffic that is actually sent (120 traffic units) is 10, which is lower than the considered margin. In this way the false positive alarm is avoided.

The goal of the margin is to make *SR-BHD* robust with respect to sources of packet loss different than the targeted *SR Black Hole*. Considering the link l, the best value to use for the margin is to set it equal to the amount of traffic that is lost due to congestion, errors, etc. Unfortunately, this quantity is unknown in advance, thus an estimation method is needed. As first, a dataset $\mathcal{D}$ of observations for each network link, considering a time horizon T, is collected. The hypothesis is that no *SR Black Hole* occurred during the creation of the dataset. The observation related to the link l at time t is represented by the tuple $< y_L(l)[t], C_l, y_E(l)[t] >$, which includes: i) the value of the link count on link l at time t, ii) the capacity of the link l, and iii) the amount of traffic dropped on link l at time t. The measurement of this quantity is performed by error counters widely deployed in current network cards, and that accounts for packets discarded due to different reasons (e.g., congestion, checksum errors, TTL expired, etc.). The

[5]Formally defined in the following.

first two parameters represent the features of the observation while the third one is the label. Once the dataset is built, it is used to train a Neural Network (NN) having the goal of learn the relation between the link utilization and the amount of traffic lost. The input layer of the NN is composed of two nodes, one receiving the value of the link count for the link *l* and the other one for its capacity, while the output layer contains a single node, reporting the targeted amount of traffic lost. This last quantity is used as the value for the margin. Before concluding it is worth highlighting that the propose structure for the NN is purely illustrative. As it will be outlined in Section VI, in a working environment the structure of the NN can be considerably different.

E. Exploit POL Counters to Improve the Precision

In the next we introduce an enhancement of the proposed algorithm, named *SR-BHD*$^+$, that allows for a consistent improvement of the precision. In particular, the idea is to exploit the extra information provided by the POL counters to reduce the size of the set of suspicious flows. In fact, as previously explained, POL counters allow to measure the volume of each SR flow. Starting from this information, by means of Eq. (5), it is possible to calculate the amount of traffic flowing over each network link in the ideal case ($\mathbf{y}_L^I$), i.e., no packet loss occurred (either due to black holes or for other causes).

$$\mathbf{y}_L^I = \mathbf{R} \cdot \mathbf{y}_P \tag{5}$$

The goal is to determine the amount of traffic lost in the black hole, starting from the knowledge of the ideal link count vector. To perform such evaluation, it is required to determine the packet loss experienced by the traffic flows along the network paths. Specifically, focusing on a specific link, two different loss-related effects can be associated to an SR flow: i) *localised* on that link, and ii) *cumulative* over links that the SR flow has already crossed. The *localised* effect is due to the packet loss events happening on a single link, such as, congestion, transmission errors, TTL expired, etc. The *cumulative* effect concerns the overall volume reduction that a given SR flow has experienced due to packet loss events *localised* on links that the flow has already crossed.

To better explain the difference between these two contributions, we refer the toy case example. In the ideal case (no congestion and no black hole), the expected traffic over the link $l_{\mathrm{C,D}}$ is equal to 270 traffic units. On the other hand, from the Fig. 5 we know that the actual amount of traffic sent over this link is equal to 170 traffic units. The difference among the two numbers represents the overall amount of traffic lost, which is equal to 100 traffic units. This quantity includes three contributions: i) the black hole, ii) the *cumulative* effect, and iii) the *localized* effect of congestion. The *cumulative* contribution of the packet loss due to congestion is equal to 20 traffic units: 5 traffic units for each of the four flows have been lost along the path from the sources to the intermediate node *C*. The *localized* contribution is equal to 15 traffic units, since 5 units of traffic are lost for each of the 3 flows (one has fallen in the black hole) that cross the link $l_{\mathrm{C,D}}$. Consequently $100 - 20 - 15 = 65$ traffic units are lost in the black hole, that is exactly the intensity of the flow between nodes *A* and *E* received by node *C*.

To filter out the contribution of other sources of packet loss from the ideal link count of a generic link *l*, calculated through the Eq. (5), we need to compensate for both the localized and the cumulative effects. With reference to the first one, it can be easily removed by means of the margin ($\mu(l)$), since it represents an estimation of the packet loss due to congestion occurring at link *l*. The cumulative effect can be removed by deleting from the ideal link count the value calculated according to Eq. (6):

$$m(l) = \sum_{a \in \mathcal{N}} g_{i,a}(l) \cdot \left(\mathbf{y}_B^I(i,a) - \mathbf{y}_B(i,a) \right) \tag{6}$$

where the vector $\mathbf{y}_B^I$ contains the ideal values of the PSID counters, that can be calculated through the Eq. (2). The intuition behind Eq. (6) is that since PSID counters instantiated in a node allow to measure the overall amount of traffic that the node has received, then they can separate the *cumulative* contribution from the *localised* one. Then, the difference between the ideal value of the PSID counters and the actual one allows to estimate the amount of traffic that has been lost along the way (before reaching a given node). Considering the toy case scenario of Fig. 5, the cumulative effect of the packet loss due to congestion at link $l_{C,D}$ can be calculated as the sum of the differences between the ideal (blue) and real (red) PSID counters instantiated at node *C*: it is equal to 20 units of traffic. This quantity is exactly the over amount of packet loss due to congestion experienced by the four traffic flows before crossing link $l_{C,D}$.

Then, the vector representing the effects (in terms of packet loss) of the *SR Black Hole* over the network links is given by the Eq. (7).

$$\Delta_L = \mathbf{y}_L^I - \mu - \mathbf{m} - \mathbf{y}_L \tag{7}$$

The principle of the *SR-BHD*$^+$ algorithm is to reduce the set of suspicious flows by comparing the amount of traffic lost due to the black hole with the size of each SR flow. In case the two quantities are comparable, then the flow is suspected to be affected by an *SR Black Hole*. More in detail, considering an SR flow $x_{i,e,c}$ that is suspected to fall in a black hole located at link *l*, then *SR-BHD*$^+$ uses the condition reported in Eq. (8) to determine if it is likely that the flow is actually affected by an *SR Black Hole*.

$$\frac{\Delta_L(l)}{y_P(i,e,c)} \in [s_{\min}, s_{\max}] \tag{8}$$

The numerator of the left term of the Eq. (8) represents the estimated amount of traffic that is lost in a black hole, while the denominator is the volume of the suspected SR flow. The right side of the Eq. (8) represents an interval that indicates the tolerance of *SR-BHD*$^+$. It is composed of two parameters, namely $s_{\min}$ and $s_{\max}$. The presence of the tolerance interval is fundamental, since the output of Eq. (7) is just an estimation of the traffic lost in the black hole. Furthermore, also in case the estimation error is negligible the tolerance interval is still required. To better understand this aspect, let us consider the toy case scenario reported in Fig. 5. We have already seen that

Algorithm 2 *SR-BHD*$^+$ Algorithm Pseudo Code

Require: $\mathbf{y}_L$, $\mathbf{y}_B$, $\mathbf{y}_P$, μ, $\mathbf{G}$, $\mathbf{B}$, $\mathbf{R}$, $s_{\min}$, $s_{\max}$, $\mathcal{F}_S$, $\mathcal{L}_S$
1: initialize $\mathcal{L}_S^+ \leftarrow \emptyset$ and $\mathcal{F}_S^+ \leftarrow \emptyset$
2: calculate $\mathbf{y}_B^I$ according to Eq. 2
3: calculate $\mathbf{y}_L^I$ according to Eq. 5
4: calculate $m(l)$ according to Eq. 6
5: **for all** $l \in \mathcal{L}_S$ **do**
6: calculate $\Delta(l)$ according to Eq. 7
7: **for all** $x_{i,e,c} \in \mathcal{F}_S$ **do**
8: **if** $r_{i,e,c}(l) > 0$ **AND** Eq. 8 is *true* **then**
9: $\mathcal{F}_S^+ \leftarrow x_{i,e,c}$ and $\mathcal{L}_S^+ \leftarrow l$
10: **end if**
11: **end for**
12: **end for**
13: **return** the set of suspicious flows $\mathcal{F}_S^+$ and links $\mathcal{L}_S^+$.

the traffic that is lost due to the black hole is equal to 65 units of traffic (which represents an estimation with no error). At the same time, it can be seen that the intensity of the flow between nodes *A* and *E* is equal to 70 traffic units. The discrepancy between these two quantities is due to the fact that part of the flow has been lost due to congestion (*cumulative* effect), not in the black hole. Thus, the tolerance interval allows to deal with this phenomenon. In the performance evaluation it is shown how, by properly setting these two values, it is possible to make *SR-BHD*$^+$ robust also in critical situations (e.g., heavy congestion, margin estimation errors, etc.).

The main steps regarding the *SR-BHD*$^+$ execution are summarized in Algorithm 2. Before concluding it is important to stress that the use of *SR-BHD*$^+$ requires the support of POL counters in network devices. Furthermore, we point out that all the presented algorithms have a computational complexity that is polynomial in the size of the input.

V. Performance Evaluation

In the following we provide a simulation based analysis of *SR-BHD* performance. Three different real networks taken from [28] are considered: Abilene ($N = 12$, $L = 30$), Geant ($N = 22$, $L = 72$) and Germany ($N = 50$, $L = 176$). For these networks, real traffic matrices are available. The shortest path policy is used for traffic demand routing: it implies that the segment lists configured in the network contain only a single SID associated to the destination node. Each traffic flow defined in the traffic matrix is steered by means of an SR policy installed at the source node.

Capacity planning is performed as follows: i) the peak traffic matrix is routed according to the shortest path policy, and ii) each link is assigned with a capacity that is randomly (according to a uniform distribution) selected so that the utilization falls in the range [50%, 99%]. Packet loss due to congestion on a link l, referred to as Q_l, is simulated by dropping a percentage of the traffic flowing over a link. Q_l is randomly selected according to a Normal distribution, whose mean is calculated according to Eq. (9):

$$\overline{Q_l} = \alpha_C \cdot \frac{y_L^I(l)}{C_l - y_L^I(l)} \cdot 10^{-3} \tag{9}$$

where α_C is the congestion amplification factor that allows to tune the level of packet loss due to congestion, C_l is the capacity of the link l, and $y_L^I(l)$ is the amount of traffic flowing over link l in the ideal case (no packet loss), calculated according to Eq. (5). Furthermore, we set for the considered Normal distribution a variance equal to the root square of the mean ($\overline{Q_l}$). Finally, the packet loss is shared among the flows routed over link l, proportionally to their intensity.

In order to determine the margin, for each of the considered networks and for each of the considered values of α_C, we have trained a fully connected feed forward NN composed of[6]: i) an input layer with two neurons, one representing the traffic over the link and the other that represents the link capacity; ii) a single hidden layer with 5 neurons using ReLu as activation function, and iii) an output layer with a single neuron that uses ReLu as activation function and representing the assessed value of the margin. The learning process is performed according to the Mean Squared Error minimization policy. For each of the considered NN, the training set is obtained by running 24 simulations (we consider one Traffic Matrix per hour among those reported in [28]) without considering the presence of *SR Black Holes* and collecting for each link the tuple $< y_L(l), C_l, Q_l >$.

Before presenting the results of the performance evaluation, we introduce the formulas of the *Precision* and *Recall* parameters that are widely considered in the next:

$$Precision = \frac{\text{true positives}}{\text{true positives} + \text{false positives}} \tag{10}$$

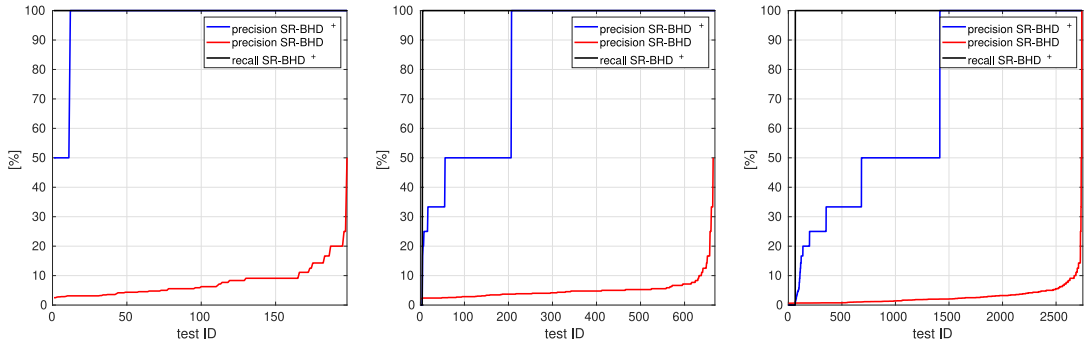
$$Recall = \frac{\text{true positives}}{\text{true positives} + \text{false negatives}} \tag{11}$$

where a *true positive* is a correctly detected black hole, a *false positive* represents a false alarm, and a *false negative* is an undetected black hole.

The first analysis we propose aims at evaluating the precision and the recall of the proposed algorithms in detecting the black holes. During the experiment we assume the presence of a single black hole. All possible events are considered: for each link and for each flow, an MTU violation is generated and the algorithms performance are evaluated. The congestion amplification factor is set equal to $\alpha_C = 10^{-3}$, and it is assumed that a black hole determines the loss of all the packets of the affected flow. Fig. 7 shows the obtained results for different networks. In the figure, the x axis represents the different performed tests (i.e., the link/flow pair involved in the black hole) sorted in increasing order with respect to the precision. In case of *SR-BHD*$^+$, $s_{\min} = 0.97$ and $s_{\max} = 1.05$. We use a narrow interval for the tolerance to maximize the precision. Both algorithms use the same neural network to assess the margin.

Looking at the results reported in Fig. 7 it can be seen that the two methods (*SR-BHD* and *SR-BHD*$^+$) achieve a very high value of recall. In all the experiments *SR-BHD* obtained 100% of recall in all the networks for all possible configurations of link/flow pair involved in the black hole (for this reason we do not report it on the figures). This means that *SR-BHD* is always able to detect the presence of an *SR Black Hole*. Similar results

[6]The parameters of the NN, such as the number of input layer neurons and the structure of the hidden layer/s, are strictly related to the network scenario considered.



(a) Abilene. (b) Geant. (c) Germany.

Fig. 7. Precision and Recall analysis of the *SR-BHD* and *SR-BHD*$^+$ in different networks.

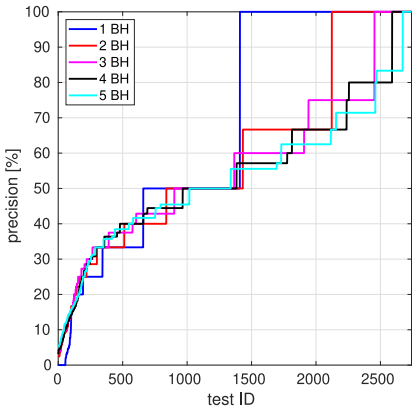


Fig. 8. CDF of the precision of *SR-BHD*$^+$ in presence of multiple *SR Black Holes*.

are obtained for *SR-BHD*$^+$ in the Abilene network whereas, in few tests for Geant and Germany networks, the algorithm is not able to correctly detect the black hole, i.e., the link and flow affected by the black hole are not in the suspicious sets (in these tests the recall is 0). This behavior is due to the estimation error in the evaluation of the amount of traffic lost in the black hole, performed using Eq. (7).

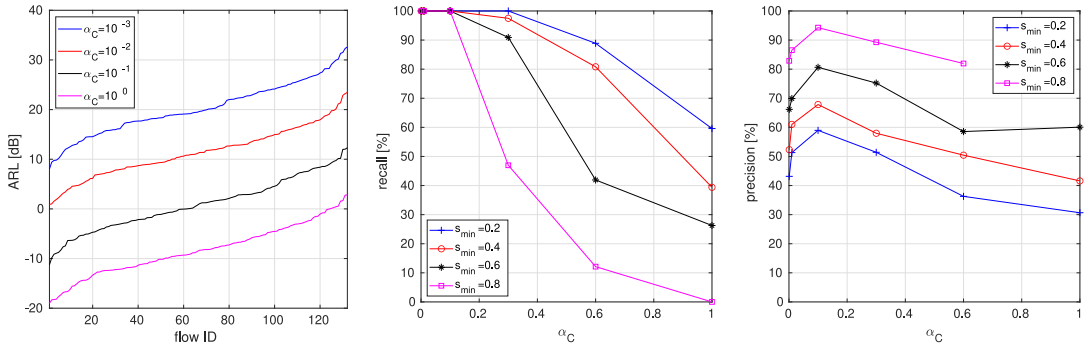
As far as the precision is concerned, results reported in Fig. 7 prove the huge performance improvement achieved by *SR-BHD*$^+$ due to the use of POL counters information. In particular, considering the Abilene network (Fig. 7(a)) *SR-BHD*$^+$ achieves a mean value of the precision equal to 97.22, while the mean value obtained by *SR-BHD* is 7.71. The relevance of the improvement on the precision is better visible in the bigger networks (Geant and German). In fact, while in these situations *SR-BHD* basically insert all the flows crossing the link affected by the black hole in the suspicious set (making impossible to fix the misconfiguration), *SR-BHD*$^+$ detects at most 3 flows (33% of precision) in more than 90% of the cases. This improvement makes the proposed tool actually useful for troubleshooting in productive environments.

In the next we propose an evaluation of the impact of multiple *SR Black Holes* on the performance of *SR-BHD*$^+$. The results for Germany network are reported in Fig. 8 in terms of precision achieved by *SR-BHD*$^+$. The number of contemporary *SR Black Holes* varies from 1 to 5, and their location (link/flow pair) is randomly selected. The main outcome of the analysis reported in Fig. 8 is that the presence of multiple *SR Black Holes* has a limited negative impact on the precision. Furthermore, it is interesting to see that independently than the number of *SR Black Holes*, the precision is above the 50% in about the 50% of the performed tests.

The next two evaluations represent a sensitivity analysis of *SR-BHD*$^+$. In the first one the aim is to test the capability of *SR-BHD*$^+$ in differentiating between regular packet loss events (caused by congestion, transmission errors, etc.) and anomalous packet loss events, due to the presence of a black hole. In the evaluation, the amount of the regular packet loss events is tuned by changing the value of the congestion amplification factor (α_C). To quantify the relationship between anomalous events and regular ones, we introduce the *Anomalous over Regular Losses* (ARL) ratio. In Fig. 9(a) the ARL for different flows, considering different values of α_C, is shown. In particular, for each flow the ratio between its volume and the amount of packet loss due to congestion on each link crossed is computed; then, the average value over all the links is considered. Due to the large number of tests, only results for the Abilene network are shown.

Figs. 9(b) and 9(c) show the average recall and precision obtained by *SR-BHD*$^+$ as a function of the congestion amplification factor, for different values of the $s_{\min}$ tolerance parameter (the $s_{\max}$ is kept constant and equal to 1.1). The main outcomes of this analysis are three: i) the performance decreases as the α_C parameter increases, ii) by tuning the $s_{\min}$ tolerance parameter is possible to make *SR-BHD*$^+$ robust with respect to regular packet loss events, and iii) 100% of recall is feasible also for very low ARL values (in the order of -10 dB).

With reference to the first two points, the performed experiment has highlighted an interesting relationship between the α_C parameter and the performance of the algorithm. On one hand, as expected the recall monotonically decreases as the congestion amplification factor increases (see Fig. 9(b)). In particular, by reducing $s_{\min}$ it is possible to keep the recall over the 90% also for high values of α_C, while keeping a good level of precision. For instance, the value $s_{\min} = 0.2$ allows for a 90% of recall when the congestion amplification factor is equal to 0.6, while having a precision of 38%. Consider that this level of precision implies that there are on average less



(a) ARL. (b) Recall. (c) Precision.

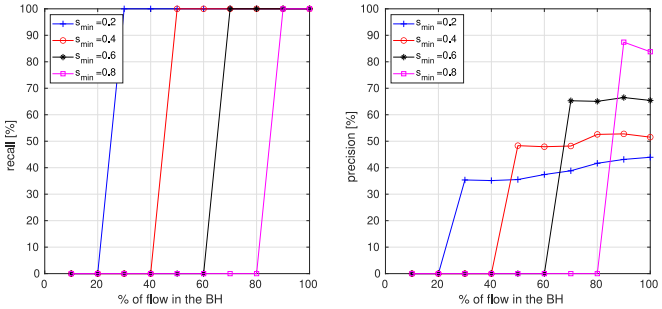
Fig. 9. Sensitivity analysis of *SR-BHD*$^+$. Precision and recall as a function of the congestion level.

than three suspected link/flow pairs. Then, although *SR-BHD*$^+$ does not provide the highest precision, it drastically reduces the set of link/flow pairs that require further investigation.

A second interesting aspect is the relationship between the congestion amplification factor and the precision of *SR-BHD*$^+$ (Fig. 9(c)). In particular, the precision does not monotonically decrease if α_C increases: i) initially the precision increases as the congestion level increases, then ii) once a maximum value is reached, it starts decreasing. This behavior is due to the working principle of *SR-BHD*$^+$, which selects the suspected flows by comparing their volume with the estimated amount of traffic that is lost in the black hole. If there are many flows having a comparable intensity, then the resulting precision of *SR-BHD*$^+$ is small. Also in the ideal case of no regular packet loss, if all the flows had the same volume, the resulting precision would be poor. In these situations, the presence of the regular packet loss creates a distance between the intensity of the different flows, helping *SR-BHD*$^+$ in correctly discriminating the flow that has actually fallen into the black hole. Clearly, when the congestion level overcomes a threshold value, the beneficial effect is lost and the performance starts to decrease as α_C increases.

With reference to the third point (100% of recall is feasible also for very low values of ARL), *SR-BHD*$^+$ has shown a high tolerance to the presence of regular packet loss events on the traffic counters. In particular, Fig. 9(b) shows that the highest value of α_C at which *SR-BHD*$^+$ gets 100% of recall is 0.3. From Fig. 9(a) it can be seen that, for such a value of congestion amplification factor, the lowest level of ARL reached is of the order of -10 dB. It means that *SR-BHD*$^+$ is able to correctly determine the flow lost in the black hole, also in the case its volume is 10 times smaller than the level of packet lost due to congestion. Furthermore, it is worth highlighting that, in these circumstances, *SR-BHD*$^+$ is still able to obtain an acceptable precision (see Fig. 9(c)). In particular, for $\alpha_C = 0.3$ and $s_{\min} = 0.2$ the average precision is close to 50%, meaning that the suspicious link/flow pairs in output are 2.

The last analysis we propose aims at evaluating the precision and the recall of *SR-BHD*$^+$ if only a portion of a flow is lost in the black hole. In fact, as explained in Section IV, an SR flow represents the aggregation of many application flows, then it



(a) Recall. (b) Precision.

Fig. 10. Sensitivity analysis of *SR-BHD*$^+$. Precision and recall as a function of the percentage of flow that gets lost in the *SR Black Hole*.

can happen that only a subset of application flows falls into the black hole. The result of this sensitivity analysis is reported in Figs. 10(a) and 10(b), showing the precision and the recall of *SR-BHD*$^+$ as a function of the percentage of flow that gets lost in the *SR Black Hole*, for different values of $s_{\min}$ parameter ($s_{\max}$ is kept constant and equal to 1.1). The main outcome of the performed evaluation is that *SR-BHD*$^+$ is able to handle this situation. In fact, also in case only 30% of a flow is lost in the black hole, the obtained recall is 100% (meaning that it is always correctly detected) and the precision is approximately 33%, i.e., on average there are 3 link/flow suspicious pairs.

VI. *SR-BHD* Prototype

In this section a description of the experimental prototype of *SR-BHD* that we have realized is provided. The goal of the prototype is to prove the effectiveness of the proposed approach. Three different aspects are discussed: i) the description of the design implementation of *SR-BHD* prototype, ii) the methodology adopted to run the tests, iii) the results obtained in the different types of performed tests.

A. Prototype Description

The proposed *SR-BHD* algorithm is designed to be integrated into a centralized monitoring system, according to the SDN paradigm. In the prototype, this task is accomplished by a bundle of scripts whose execution is triggered by a timer. By a

logical point of view, the prototype is composed by two main building blocks: i) the *Stats Collector* module, which is in charge of collecting the SRTCs in each node, and ii) the *Black Hole Detection* module, that implements the logic of *SR-BHD* on each link to detect potential black holes. Concerning the *Stats Collector* module, it is implemented as a bash script that automatically queries the network routers to get the required traffic statistics. On the contrary, the *Black Hole Detection* module is a Python script that takes as input the traffic measurements and verifies the validity of the equations that are at the basis of *SR-BHD* approach. It is also in charge of evaluating the margin. This module requires the presence of a *configuration file* that specifies the routing, i.e., the segment lists that are enforced on packets in the data plane as well as the underlay paths. A *timer* component triggers the execution of the two blocks.

In our prototype implementation, VPP virtual routers are used to instantiate an SRv6 capable network. With respect to the functionality available in an ideal SRv6 router, VPP lacks of the presence of the full suite of SRTCs. In particular, the availability of traffic counters in VPP is as follows: i) INT counters, i.e., interface level traffic counters that account the number of packets TX/RX to/from each interface, are available; ii) POL counters are not supported, and iii) PSID counters are updated only in case an SR operation is performed. Given the POL counter unavailability, only *SR-BHD* can be implemented in the prototype. Regarding the lPSID limitation, it represents a critical difference for the actuation of the proposed *SR-BHD* framework. In fact, it relies on statistics coming from PSID counters that are collected at each node. On the contrary, in VPP this type of SRTC counters are maintained only at nodes that perform SR related operation (e.g., a router that accounts the packets on which it applies the END operation). In order to finalize our prototype, we have applied the strict source routing policy, i.e., for each flow the segment list contains the explicit set of intermediate nodes to go through. As a future step we aim at implementing the PSID counters on every node (regardless the application of SR functions).

B. Adopted Methodology

The reference topology used in the evaluation is shown in Fig. 1, with the only difference that in the performed experiments the initial MTU is set equal to 1500 Bytes in all the links. All the links have the same capacity, equal to $R = 10$ Mbps. Nodes $E1$ and $E2$ represent the edge of the considered SR domain and implement SR policy enforcement on the incoming traffic flows. The main features (source, destination, path) of the four traffic flows included in the scenario are reported in Tab. III. These are generated by using the traffic generator included in VPP, that allows to create constant bit rate UDP flows, with configurable packet size and data rate. The traffic flows generated have the same data rate of 100 Kbps.

In each test a target flow and the link where the black hole occurs are selected. In order to create the black hole in the desired link and to hit the target flow, two actions are performed: i) packets of the target flow are generated with a higher size with respect to those of the other flows, and ii) the MTU of the link where the black hole happens is reduced. Then, all the traffic flows are simultaneously started and ended 120 seconds later. The *Stats Collector module* is triggered every 60 seconds, meaning that in each run the *Black Hole Detection module* is executed 2 times.

TABLE III
MAIN FEATURES OF THE TRAFFIC FLOWS INCLUDED IN THE EMULATED ENVIRONMENT

source	destination	segment list
a::1	b::1	C1,C2,E2
a::2	b::2	C1,C3,C4,C2,E2
b::1	a::1	C4,C3,E1
b::2	a::2	C4,C2,C1,C3,E1

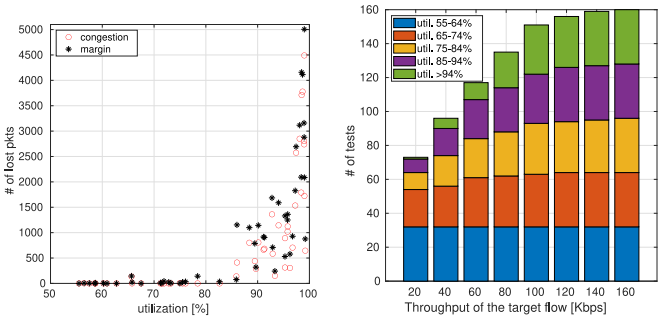
TABLE IV
DIFFERENT CONFIGURATIONS OF THE DTMP SOURCE

ID	*1*	*2*	*3*	*4*	*5*
P_{ON}	0.2	0.3	0.4	0.5	0.6
P_{OFF}	0.6	0.5	0.4	0.3	0.2
link util.	$55-64\%$	$65-74\%$	$75-84\%$	$85-94\%$	$>94\%$

In order to create controllable congestion events, we generated different level of background traffic. More in detail, a source-destination traffic flow is generated at the end nodes of each oriented link. Thus, considering the link *i*, *j*, a source of traffic is attached at node *i* and the related destination is included in node *j*. The source node is modeled as a Discrete Time Markov Process (DTMP) that at each time slot can be in ON or OFF state, and is characterized by transition probabilities p_{ON} and p_{OFF}. When the source is in ON state, it generates packets of length 1000 Bytes at a constant bit rate equal to R_a and, in the following time slot it remains active with probability p_{ON}, while it stops sending traffic with probability $1-p_{\mathrm{ON}}$. Conversely, when the source is in OFF state, packets are not generated and, in the following time slot, it remains inactive with probability p_{OFF}, while it starts sending traffic with probability $1-p_{\mathrm{OFF}}$. In the different tests we have set the duration of a time slot equal to 100 milliseconds and $R_a = 15$ Mbps (greater than link capacity so that to create congestion), while five different combinations of transition probabilities (reported in Table IV) have been considered.

C. Experimental Evaluation

In the next we describe the Neural Network (NN) structure used for the experimental evaluation of the prototype and the training procedure. We used a feed forward NN having two hidden layers with 32 and 16 neurons respectively, ReLu as activation function, and trained with the aim of minimizing the Mean Squared Error (MSE). In order to generate the training set, we performed the following steps: i) a given combination of transition probabilities for the DTMP sources is selected, ii) a 60 seconds experiment is executed collecting link utilization every 6 seconds, and iii) the final amount of lost packets is obtained. Each execution has been repeated 100 times for each of the 5 combinations of transition probabilities, thus having a final training set composed of $100 \times 5 \times 16 = 8000$ observations (16 is the number of links). Each observation is



(a) Margin VS packet loss due to congestion.

(b) Sensibility of *SR-BHD*.

Fig. 11. Performance evaluation of the developed prototype carried out over an emulated environment.

represented by 10 measurements of link utilization and a label consisting in the number of lost packets. As a consequence, the NN has an input layer with 10 neurons and an output layer with 1 neuron. With respect to the NN used for the simulation based analysis, the input is represented by a vector of utilization values (one each 6 seconds) and not by a single value (one for the overall time period): the reason is that the experimental test shows a great variability of the utilization during the execution period, and it should be taken into account when setting the margin.

In Fig. 11(a) the performance evaluation of the margin assessment over a test set of 50 points (10 for each combination of the transition probabilities) is reported. The x axis represents the link utilization averaged over 60 seconds. From the figure it is evident that similar values of average link utilization lead to highly different values of packet loss, thus motivating the choice of several utilization measurements for the NN input layer. As it can be seen, the NN always provide a satisfying estimation of the actual amount of packet loss due to congestion. Clearly, the quality of the estimation can be improved by feeding the NN with more measurements of the link utilization (i.e., with higher granularity), at the cost of increasing the signaling overhead. Anyway, as shown in the next analysis, the prototype is able to obtain good performance in terms of black hole detection with this NN setting; in other words, a rough estimation of the packet loss due to congestion is acceptable for black hole detection and identification.

The final evaluation, reported in Fig. 11(b), regards the sensibility of *SR-BHD* in detecting the SR flow affected by a black hole (the target flow from now on) as a function of flow throughput. As first, a combination of transition probabilities for the DTMPs (i.e., background traffic level) is selected and the throughput of the target flow is fixed. Then, the target flow (among the ones reported in Table III) and the link experiencing the black hole, are chosen. Summarizing, 16 combinations are tested, once at a time. Furthermore, in each test *SR-BHD* is applied two times (once every 60 seconds), thus for each pair of level of background traffic/value of the target flow, 32 observations (referred to as tests in the y-axis of Fig. 11(b)) are obtained. Fig. 11(b) shows, for different levels of background traffic, the number of tests where the black hole is correctly detected as a function of the size of the target flow. By inspecting Fig. 11(b) it can be seen that, for low values of background traffic, *SR-BHD* is able to detect also very small flows (20 Kbps over a 10 Mbps link) lost in the black hole. In this cases, the margin approximate very well the amount of packet loss due to congestion. On the contrary, as the level of background traffic increases, small flows are not correctly detected. Interestingly, when the rate of the target flow is equal to 160 Kbps (approximately 1.6% of the link bandwidth), the black hole is always correctly detected. This result highlights the significant capability of *SR-BHD* in detecting black holes even if the target flow is significantly lower than the link bandwidth.

VII. Conclusion

In this paper we have addressed the problem of logical failures in Segment Routing networks due to the violation of the MTU constraint, named *SR Black Hole*. We have experimentally proven that: i) in misconfigured SR domains *SR Black Holes* can occur, and ii) classical detection methods based on active probing fail to detect the failure. Then, by exploiting specific SR Traffic Counters we have introduced a passive monitoring framework, named *SR-BHD*, that is able to detect the presence of black holes, also in presence of multiple sources of packet loss (e.g., congestion, transmission errors, etc.). In particular *SR-BHD* imposes the flow conservation principle at different levels in the network, being able to tolerate the presence of other sources of packet loss (e.g., congestion, transmission error, etc.) by the introduction of a safety margin. An estimation framework based on the use of Neural Networks is presented as a method to select the proper value for the margin. The framework has further been extended in case specific traffic counters (POL counters) are available in SR capable nodes. The enhanced version of *SR-BHD*, named *SR-BHD*$^+$, allows to greatly improve the precision in the detection of the black hole. Specifically, a tolerance interval can be set to make *SR-BHD*$^+$ robust against critical situations, such the case of high level of packet loss due to congestion, or the case in which only a portion of a traffic flow is lost in the black hole. The performance evaluation has shown how, by properly tuning the tolerance interval of *SR-BHD*$^+$, it is possible to find a good trade off between the precision of the algorithm and its robustness with respect to the aforementioned critical situations. Finally, a prototype of *SR-BHD* has been realized and tested over an emulated environment. The conducted experiments have confirmed the effectiveness of the proposed framework in detecting the presence of *SR Black Holes*. As future works we aim at design a tool to help Network Operators to configure their SR domain by avoiding the creation of *SR Black Holes*, and the integration of the full suite of SR traffic counters in VPP and others programmable data planes.

References

[1] P. L. Ventre *et al.*, "Segment routing: A comprehensive survey of research activities, standardization efforts and implementation results," *IEEE Commun. Surveys Tuts.*, vol. 23, no. 1, pp. 182–221, 1st Quart., 2021.

[2] C. Filsfils, P. Camarillo, J. Leddy, D. Voyer, S. Matsushima, and Z. Li, "Segment routing over IPv6 (SRv6) network programming," Internet Eng. Task Force, RFC 8986, Feb. 2021. [Online]. Available: https://rfc-editor.org/rfc/rfc8986.txt
[3] C. Filsfils, D. Dukes, S. Previdi, J. Leddy, S. Matsushima, and D. Voyer, "IPv6 segment routing header (SRH)," Internet Eng. Task Force, RFC 8754, Mar. 2020. [Online]. Available: https://rfc-editor.org/rfc/rfc8754.txt
[4] M. de Boer, J. Bosma, and W. Toorop, *Discovering Path MTU Black Holes in the Internet Using RIPE Atlas*, Univ. Amsterdam, Amsterdam, The Netherlands, 2012.
[5] M. Luckie, K. Cho, and B. Owens, "Inferring and debugging path MTU discovery failures," in *Proc. 5th ACM SIGCOMM Conf. Internet Meas.*, 2005, pp. 1–6.
[6] A. Tulumello *et al.*, "Micro SIDs: A solution for efficient representation of segment IDs in SRv6 networks," in *Proc. 16th Int. Conf. Netw. Service Manage. (CNSM)*, 2020, pp. 1–10.
[7] S. Litkowski, A. Bashandy, C. Filsfils, P. Francois, B. Decraene, and D. Voyer, "Topology independent fast reroute using segment routing," Internet-Draft draft-ietf-rtgwg-segment-routing-ti-lfa-07, Internet Eng. Task Force, Fremont, CA, USA, Jun. 2021. [Online]. Available: https://datatracker.ietf.org/doc/html/draft-ietf-rtgwg-segment-routing-ti-lfa-07
[8] Y. Desmouceaux, M. Townsley, and T. H. Clausen, "Zero-loss virtual machine migration with IPv6 segment routing," in *Proc. 14th Int. Conf. Netw. Service Manage. (CNSM)*, 2018, pp. 420–425.
[9] M. Polverini, A. Cianfrani, and M. Listanti, "Snoop through traffic counters to detect black holes in segment routing networks," in *Proc. Int. Conf. Commun. Netw.*, 2020, pp. 337–350.
[10] C. Filsfils, Z. Ali, M. Horneffer, D. Voyer, M. Durrani, and R. Raszuk, "Segment routing traffic accounting counters," Internet-Draft draft-filsfils-spring-sr-traffic-counters-01, Internet Eng. Task Force, Fremont, CA, USA, Apr. 2021. [Online]. Available: https://datatracker.ietf.org/doc/html/draft-filsfils-spring-sr-traffic-counters-01
[11] R. R. Kompella, J. Yates, A. Greenberg, and A. C. Snoeren, "Detection and localization of network black holes," in *Proc. 26th IEEE Int. Conf. Comput. Commun.*, 2007, pp. 2180–2188.
[12] L. Fang, A. Atlas, F. Chiussi, K. Kompella, and G. Swallow, "LDP failure detection and recovery," *IEEE Commun. Mag.*, vol. 42, no. 10, pp. 117–123, Oct. 2004.
[13] C. Kreibich, N. Weaver, B. Nechaev, and V. Paxson, "Netalyzr: Illuminating the edge network," in *Proc. 10th ACM SIGCOMM Conf. Internet Meas.*, 2010, pp. 246–259.
[14] R. Staff, "Ripe Atlas: A global Internet measurement network," *Internet Protocol J.*, vol. 18, no. 3, pp. 1–31, 2015.
[15] M. Mardani and G. B. Giannakis, "Estimating traffic and anomaly maps via network tomography," *IEEE/ACM Trans. Netw.*, vol. 24, no. 3, pp. 1533–1547, Jun. 2016.
[16] M. Thottan and C. Ji, "Anomaly detection in IP networks," *IEEE Trans. Signal Process.*, vol. 51, no. 8, pp. 2191–2204, Aug. 2003.
[17] R. Geib, C. Filsfils, C. Pignataro, and N. K. Nainar, "A scalable and topology-aware MPLS data-plane monitoring system," Internet Eng. Task Force, RFC 8403, Jul. 2018. [Online]. Available: https://rfc-editor.org/rfc/rfc8403.txt
[18] N. Kumar, C. Pignataro, G. Swallow, N. Akiya, S. Kini, and M. Chen, "Label switched path (LSP) ping/traceroute for segment routing (SR) IGP-prefix and IGP-adjacency segment identifiers (SIDs) with MPLS data planes," Internet Eng. Task Force, RFC 8287, Dec. 2017.
[19] Z. Ali, K. Talaulikar, C. Filsfils, N. K. Nainar, and C. Pignataro, "Bidirectional forwarding detection (BFD) for segment routing policies for traffic engineering," Internet-Draft draft-ali-spring-bfd-sr-policy-07, Internet Eng. Task Force, Fremont, CA, USA, May 2021. [Online]. Available: https://datatracker.ietf.org/doc/html/draft-ali-spring-bfd-sr-policy-07
[20] M. Polverini, A. Cianfrani, and M. Listanti, "A theoretical framework for network monitoring exploiting segment routing counters," *IEEE Trans. Netw. Service Manag.*, vol. 17, no. 3, pp. 1924–1940, Sep. 2020.
[21] P. Loreti *et al.*, "SRv6-PM: A cloud-native architecture for performance monitoring of SRv6 networks," *IEEE Trans. Netw. Service Manag.*, vol. 18, no. 1, pp. 611–626, Mar. 2021.
[22] F. Aubry, D. Lebrun, S. Vissicchio, M. T. Khong, Y. Deville, and O. Bonaventure, "SCMon: Leveraging segment routing to improve network monitoring," in *Proc. 35th Annu. IEEE Int. Conf. Comput. Commun.*, 2016, pp. 1–9.
[23] C. Filsfils, S. Previdi, L. Ginsberg, B. Decraene, S. Litkowski, and R. Shakir, "Segment routing architecture," Internet Eng. Task Force, RFC 8402, Jul. 2018.
[24] "CSIT REPORT: The Fast Data I/O Project (FD.io) Continuous System Integration and Testing (CSIT) Project Report for CSIT Master System Testing of VPP-18.04 Release." Aug. 2022. [Online]. Available: https://docs.fd.io/csit/master/report/_static/archive/csit_master.pdf
[25] C. Filsfils, D. Dukes, S. Previdi, J. Leddy, S. Matsushima, and D. Voyer , "IPv6 segment routing header (SRH)," Internet-Draft draft-ietf-6man-segment-routing-header-21, Internet Eng. Task Force, Fremont, CA, USA, Jun. 2019.
[26] A. Dhamdhere, R. Teixeira, C. Dovrolis, and C. Diot, "NetDiagnoser: Troubleshooting network unreachabilities using end-to-end probes and routing data," in *Proc. ACM CoNEXT Conf.*, 2007, pp. 1–12.
[27] A. Itai, "Two-commodity flow," *J. ACM*, vol. 25, no. 4, pp. 596–611, 1978.
[28] S. Orlowski, M. Pióro, A. Tomaszewski, and R. Wessäly, "SNDlib 1.0—Survivable network design library," in *Proc. 3rd Int. Netw. Optim. Conf. (INOC)*, Apr. 2007, pp. 1–15.

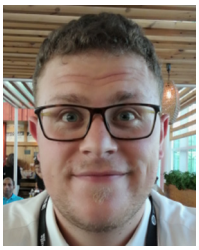
Marco Polverini received the master's degree in telecommunications engineering and the Ph.D. degree in information and communication engineering from the University of Rome La Sapienza, in 2010 and 2014, respectively, where he is currently a Research Fellow with the Department of Information, Electronic and Telecommunications Engineering. His main research interests are routing protocols for energy saving in IP networks, network traffic monitoring, and measurement in next generation routing technologies.

Antonio Cianfrani received the master's degree in telecommunications engineering and the Ph.D. degree in information and communication engineering from the University of Rome Sapienza in 2004 and 2008, respectively, where he is currently an Assistant Professor with DIET Department. His fields of interests include routing algorithms, network protocols, and performance evaluation of software routers and green networks. His current research interests are focused on segment routing and traffic matrix computation. He serves on the Editorial Boards of the IEEE TRANSACTION ON GREEN COMMUNICATIONS AND NETWORKING.

Marco Listanti (Life Member, IEEE) received the Dr.Eng. degree in electronics engineering from the University of Rome "Sapienza" in 1980. In 1981, he joined Fondazione Ugo Bordoni, where he was the Leader of the group TLC Network Architecture until 1991. In 1991, he joined the INFOCOM Department, University of Roma La Sapienza, where he is a Full Professor of Switching Systems. He participated in several international research projects supported by EEC and ESA. He has authored several papers published on the most important technical journals and conferences in the area of telecommunication networks. His current research interests focus on traffic control in IP networks and on the evolution of techniques for optical networking.

Giulio Siano received the bachelor's degree in computer engineering from Sapienza University in 2020. He is currently pursuing the master's degree in communication engineering with the Politecnico di Milano. His research interests are in the field of Programmable Networks and Network Automation.

Carlo Candeloro Campanile received the master's degree in electronics engineering from the Politecnico di Bari in November 2000. He is currently pursuing the Ph.D. degree with the University of Rome "Sapienza." His main research interest is anomaly detection in segment routing networks.



Francesco Giacinto Lavacca received the Laurea (M.Sc.) degree (*cum laude*) in electronic engineering in 2013 and the Ph.D. degree in information technology from the Sapienza University of Rome, Italy, in 2017. In 2018, he joined Fondazione Ugo Bordoni as a Researcher with the TLC Group. He is currently a Researcher with the DIET Dipartment, University of Roma "Sapienza," where he started his postdoctoral research activities. In 2016, he has been a Visiting Researcher with the College of Computing, Georgia Institute of Technology, Atlanta, GA, USA. Furthermore, he was involved in the framework of national and international projects, like Advanced Avionic Architecture and Nano Micro Launch Vehicle with Italian Space Agency and European Space Agency, respectively. His current research interests are in the fields of all-optical networks and switching architectures, 5G networks planning and design, network function virtualization, time-triggered and deterministic Ethernet.

Open Access funding provided by 'Università degli Studi di Roma "La Sapienza"' within the CRUI CARE Agreement