TECHNISCHE
UNIVERSITÄT
DRESDEN

Faculty of Computer Science Institute of Systems Architecture, Chair of Computer Networks

Segment Routing and SRv6: Origins, State-of-the-Art and Outlook

Analyse eines Forschungsthemas

Dominik Pataky

2018-07-31

Contents

Author	Dominik Pataky <dominik.pataky@tu-dresden.de>
	PGP 0x80BF7C9C5B62468F
Supervisor	Dr.-Ing. Marius Feldmann
Last update	Friday 17th August, 2018 10:02
License	CC BY-SA 4.0

Abstract

Segment Routing is a new technology based on the approach of Source Routing. With Segment Routing, nodes participating in a network are able to send network traffic via pre-defined hops inside a network. This way traffic engineering algorithms for load balancing or, in case of failures, rerouting traffic can be used more efficiently. While Segment Routing is based on core networks in which MPLS is used for routing, SRv6 aims to be an IPv6 header extension which is compatible with any IPv6-enabled software stack.

This research task includes the current state of the conceptual design of SRv6 and the motivation behind the development of Segment Routing and SRv6. We will have a look at the technical implementations of SRv6, taking MPLS and IPv6 into account, and look at practical use cases and boundary points to other technologies. It concludes with an overview of the current status of SRv6 in terms of specifications, RFCs, existing implementations, projects and working groups.

1. Source Routing, Segment Routing and SRv6

As Segment Routing is an implementation of the so called Source Routing algorithm, we will first have a look at the concept of Source Routing to understand where the concepts and motivation for the development of Segment Routing came from.

1.1. Source Routing and other routing approaches

In general, Source Routing allows *the sender* of a packet to determine network hops a packet must be routed through on its path towards the destination. This stands in contrast with the more established algorithms, with which the routers take over the decision which links and hops are used to transmit the packet. Source Routing is therefore also known as Path Addressing, because the source not only defines the destination address but also the hops in between those two hosts.

Originally, the specification of Internet Protocol version 4 (IPv4) introduced a Source Routing mechanism. In IPv4 the header allows an appendix of „Internet Protocol Options“, but it is not widely used. Two options exist in the context of Source Routing, as specified in *RFC 791*: **Loose Source Routing (LSR)** and **Strict Source Routing (SSR)**.

With LSR the sender creates a list of IP addresses the packet must visit and appends this list to the packet itself, using the top address as the destination. As soon as the packet arrives at the destination, the router takes the next IP address from the list, replaces the destination address and sends the packet towards the new destination. Because the list does not have to include the whole route through the network, this algorithm is called „loose“, since there is some flexibility how to route the packet in between the destinations.

In contrast to LSR, SSR uses the same address list appended to each packet, with the difference that the complete route is determined in advance and included in the packet. There is no dynamic routing in between the hops.

Another alternative to Source Routing is **Policy-based Routing (PBR)**. PBR uses policies for routing decisions at each hop instead of using the destination as the primary routing decision factor.

In this context, **Software Defined Networking (SDN)** and **OpenFlow** need to be mentioned as well. The concept of SDN introduces new network architectures regarding routing policies. An SDN-designed network disconnects the Control Plane from the Data Plane, a central controller instance takes over the routing decisions all routers in a network execute. OpenFlow is an open protocol which allows these controllers to communicate with routers made by multiple vendors. The controller knows the whole network topology and state and creates routing policies dynamically, distributing the layer 3 routing tables on the routers via the OpenFlow protocol. See section 2.6 for further details.

What are reasons to use Source Routing? Being able to determine a fixed set of hops a packet must pass through, Source Routing can be used for **troubleshooting** and **performance**. In case of troubleshooting, network administrators can test multiple paths and hops to identify problems inside a network. Performance can be gained by routing packets through paths which have capacity, implementing load balancing and avoiding traffic congestion. On the other hand Source Routing also bares some problems. Deploying Source Routing in a network can for example be used to **reach protected hosts** behind a firewall or NAT gateway, in case an attacker is able to append the internal IP address of a host into the list of segments. Secondly, the approach of performance routing mentioned above can be abused to route packets over single links, overusing them and triggering a **Denial of Service (DoS)**. [Sys08]

We will come back to security related aspects in section 2.5.

1.2. Segment Routing

The original motivation to develop Segment Routing came from Cisco Systems and was based on the idea to extend Multi Protocol Layer Switching (MPLS) with a mechanism which reduces the complexity of MPLS tunnel configuration inside a network [Tec16a]. The developed standard (as *RFC 8402*) for Segment Routing brings an alternative to MPLS tunnelling, making the maintenance of thousands of MPLS Resource Reservation Protocol for Traffic Engineering (RSVP-TE) tunnels obsolete. The standardisation of this concept aims for a multi-vendor deployment, decoupling implementations from proprietary solutions only available in Cisco products.

Segment Routing is based on the concept of Source Routing. A network node which sends packets to another node uses a locally available policy or a central controller to request a path through this network, creating a list of hops which is appended to each packet. The packet is then routed via the pre-defined hops, following a pre-calculated path. Overall the aim of this approach is the creation of stateless network routing, encapsulating all flow information inside the packets.

The key feature used in this scenario is the central controller instance. This controller holds all topology information, even across multiple divided networks (or domains, e.g. datacenter, WAN and peering exchange points), aiming to use the most optimal path through all participating networks, a traffic flow will need to pass to reach its target.

In the context of SDN, this controller is seen as the „glue“ between applications – which need to use individually optimised paths for their traffic – and the network itself.

Participating bodies and committees

The initial development of Segment Routing came from inside Cisco, with Clarence Filsfils being the original creator and developer [Tec16a][Sysc]. Due to the declared aim to develop Segment Routing as an open multi-vendor implementation, the development and standardisation was continued in the IETF Source Packet Routing in Networking (SPRING) Working Group [SPRb] which is actively working on *RFC 8402*, „Segment Routing Architecture“. See appendix A for more information.

1.3. Deployment with IPv6: SRv6

Segment Routing for IPv6 is based on the existing Internet Protocol version 6 (IPv6) standard (*RFC 8200*). SRv6 uses a type of Routing Header Extension in IPv6, defined as the Segment Routing Header (SRH), to insert the list of hops a packet should pass through into the IPv6 header. This allows the implementation of SRv6 functionality into existing networks without introducing breaking changes, since network stacks which do not support the SRv6 header type can safely ignore it and continue packet parsing by reading the header referenced in the Next Header field.

The Linux kernel implementation is currently maintained by Dr. David Lebrun (Université Catholique de Louvain, Belgium; now Google) [Leb] [LB17a]. An alternative implementation is supported by Cisco in context of the FD.io project, making use of the Vector Packet Processing (VPP) technology [Sysc] [FDia].

Quoting Gaurav Dawra from Cisco Systems summarises the key features [Daw+17]:

> Segment Routing (SR) is an architecture based on the source routing paradigm that seeks the right balance between distributed (network-wide) intelligence and centralized (controller-based) programmability. [. . .] SRv6 enhances the properties of network simplification, strict SLA enforcement, automation and scaling (stateless fabric) pioneered by SR-MPLS. SRv6 extends the implementation of network programming by allowing ’any’ instruction to be bound to a segment; for example: for-

warding and user-defined instructions. SRv6 use-cases expand into Network Function Virtualization (NFV), Service Chaining, Spray, 5G mobile etc. [...] In addition, the support of SRv6 in open source projects (Linux Mainline and Vector Packet Processing (VPP) platform) provides Comcast, other network operators and researchers with powerful tools for the validation and deployment of the technology. SRv6 brings untapped potential and innovation to the network layer that translates into impactful opportunities.

The quote also mentions the terms „network programming“ and „user-defined instructions“. This references the idea of „Network as a Computer“ which is devised by Cisco and e.g. Comcast [Daw+17] [Day16]. The concept includes the idea of the routing of packets through multiple micro service instances inside a network which perform tasks on the packet, forwarding the result. This way for example a video streaming service can implement Segment Routing (or rather SRv6) to route from a video source via a transcoding service into a Content Delivery Network (CDN) for content delivery towards the clients. This topic will be picked up again in section 2.6.

2. Technical implementations

2.1. Existing technologies used by Segment Routing and SRv6

2.1.1. MPLS

Multi Protocol Layer Switching (MPLS) is a packet switching/routing protocol which injects a label header (as shown in fig. 1) between the link layer and the network layer, e.g. between the Ethernet frame and the IP header. When an unlabelled packet reaches an ingress router of an MPLS network, a so called **Label Edge Router (LER)**, this router prefixes the packet with an MPLS prefix based on the configured **Forwarding Equivalence Class (FEC)**. The configuration of these classes and the **Label Information Base (LIB)** is distributed using the **Label Distribution Protocol (LDP)** or the **Resource Reservation Protocol for Traffic Engineering (RSVP-TE)**.

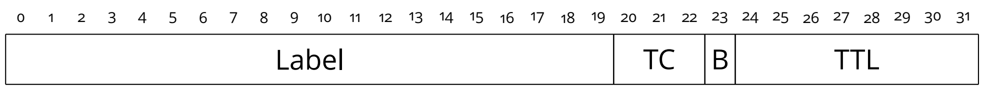

Figure 1: Header structure of the MPLS prefix.
TC: Traffic Class, B: Bottom-of-Stack flag, TTL: Time-to-Live

All routers inside the MPLS network, called **Label Switch Router (LSR)** or transit routers, then only evaluate the label of incoming packets, not the header of e.g. IPv4. While sending the packet towards the next hop, the LSR replaces the existing label with a new label according to the LIB. Exiting the MPLS network happens at the egress router. This router holds information about labels which should be cleared instead of being replaced. A packet which arrives at an egress router with a label indicating the end of the **Label Switch Path (LSP)** (also called **MPLS tunnels**) is cleared from the MPLS header and routed according to the underlying protocol, like IPv4 or IPv6. The removal of the MPLS label header can also happen at the second to last router, before passing the packet to the egress router. This is called Penultimate Hop Popping (PHP) and can be used to ease the workload of big egress routers.

Considering security and resilience of MPLS networks, LSPs can be ranked into three categories: primary, secondary and tertiary. Failure of a primary link can therefore be handled by replacing the label with the secondary label and forwarding the packet via another interface.

To provide additional traffic recovery mechanisms, **Fast Reroute (FRR)** is a technology in MPLS for traffic rerouting in case of link or node failures to recover flows in 50 ms. With FRR pre-defined alternative LSPs are announced in the network, protecting links and nodes.

The difference between using backup LSPs and FRR is the chosen alternative route. With secondary LSPs, the whole tunnel from Ingress to Egress is duplicated, using other nodes and links as the primary LSP. With FRR, failing nodes and links are bypassed ad-hoc, only leaving the primary LSP at the Point of Local Repair (PLR) and reentering at the Merge Point (MP). In combination, FRR can be used to fix a failing LSP until the secondary LSP can be activated, taking over the tunnel.

2.1.2. IPv6

Internet Protocol version 6 (IPv6) is the successor of the Internet Protocol version 4 (IPv4), defined in *RFC 2460* (obsoleted) and *RFC 8200*. IPv6 simplifies and embeds multiple implementation details found in IPv4 and its extensions, enlarges the available **address space** from 2^{32} to 2^{128} addresses and offers header extensions for future extensibility. The size of the IPv6 address space offers new possibilities to address hosts and services, making it possible to bind services to their own globally routed IPv6 address.

Figure 2 shows the format of the 40 Bytes header of IPv6. This header has a fixed size and is extendable through the usage of **Extension Headers (EHs)**, which contain more optionally usable fields and are inserted between the IPv6 header and the upper layer header (e.g. TCP/UDP). The following Extension Headers are included in all full implementations of IPv6 (indicating their type number):

Hop-by-Hop Options (0) Only header which is evaluated by each router on the routed path.

Routing (43) Defines a Routing Type and a list of nodes the packet should be routed through.

Fragment (44) The source fragments packets larger than the MTU and chains them with an Identification tag. Packets containing later fragments (second to last) use this EH only, since the first packet contained all other needed EHs.

Destination Options (60) Only examined by the destination node of the packet.

Authentication (51) See *RFC 4302*.

Encapsulating Security Payload (50) See *RFC 4303*.

All EHs which carry options, format them as Type/Length/Value (TLV) payloads. See [IANb] for a list of additionally specified Header Types and Options.

IPv6 uses **unicast, anycast and multicast** addresses. Broadcasting used in IPv4 is not specified for IPv6. Unicast addresses have their own **scope**: link-local (`fe80::/10`, only known to directly connected neighbours) and global. In multicasting the fourth four bits in the address (`ff0x::/8`) determine the scope from a larger list of specified scopes. The private address ranges in IPv4 (`10.0.0.0/8`, `172.16.0.0/12` and `192.168.0.0/16`) are defined as **Unique Local Addresses (ULAs)** in the range `fc00::/7`, from which `fd00::/8` can be chosen randomly and fall into the control of the local administrator.

Address, neighbour and route discovery in IPv6 is done with Internet Control Message Protocol version 6 (**ICMPv6**), Stateless Address Autoconfiguration (**SLAAC**), Dynamic Host Configuration Protocol (**DHCPv6**), Neighbor Discovery Protocol (**NDP**) and optionally IGPs like **OSPFv3**.

NDP replaces the use of Address Resolution Protocol (ARP) and improves the provided functionalities. It uses ICMPv6 packet types to automatically identify routers and nodes connected to a link. It also supports the discovery of address prefixes, unreachable neighbours, next-hops and duplicate addresses in case SLAAC is used.

SLAAC generates interface addresses from the interface MAC address, detecting possible collisions with Duplicate Address Detection (**DAD**) from NDP. If this auto-configuration can not be used, DHCPv6 can be used like in IPv4, receiving an address from an external DHCP server. Since the addresses generated by SLAAC always stay the same as long as the MAC address does not change, SLAAC Privacy Extensions were added to the specification. With enabled Privacy Extensions, a randomised address is additionally configured for the interface and changed periodically. Packets sent globally through the interface use the randomised source address.

As shown in the list above, the Extension Headers include Authentication and ESP. Both are used by **Internet Protocol Security (IPsec)** to provide authentication and encryption of packet payloads.

2.1.3. Other

Virtual Routing and Forwarding (VRF) provides virtual, co-existing routing tables within routers. Using VRF a provider is able to use the same hardware edge routers for multiple customers while isolating each customer routing table from each other. This way VPNs between customer sites can be routed solely based on the connected (logical) links.

OSPF and IS-IS are two link-state IP routing protocols used to automatically create network topologies on every router inside a network. They both use the Dijkstra algorithm to calculate the fastest paths towards other nodes, considering not only the hop count but also other metrics like bandwidth and latency.

Open Shortest Path First (OSPF) in IPv6 is defined as OSPFv3 in *RFC 5340*, in contrast to OSPFv2 for IPv4. With OSPF each router uses Link-State Advertisements (LSAs) to distribute its routing table to all connected neighbours. The receiving routers then distribute these tables towards their neighbours, so each participating router receives the neighbours of every router in the network. The topology and fastest paths can then be calculated based on these tables. OSPF creates IP packets for information exchange using Next Header type 89. For scaling and the reduction of route flooding, OSPF can be divided into multiple areas. A hierarchy of areas – with Area 0 being the central backbone area – are connected internally via Area Border Routers, reducing the load of all routers inside an Autonomous System (AS). The load inside an Area is futher decreased by choosing a Designated Router and its Backup Router. The DR receives all routing tables centrally and distributes them further in one batch.

Intermediate System to Intermediate System (IS-IS) exchanges routing information according to the OSI layer protocols, using Protocol Data Units (PDUs) to carry Type/Length/Value (TLV) entries. It is therefore applicable for IPv4 and IPv6. IS-IS is an ISO standard ISO/IEC 10589:2002(E) [Gur].

For inter-AS routing, **Border Gateway Protocol (BGP)** is used, standardised as version BGP-4 in *RFC 4271*. BGP is used to connect AS peers with each other, establishing a permanent TCP session between the two edge routers from within each AS, over which the connection state is periodically checked and information is exchanged. If BGP is used inside a single AS, to connect multiple OSPF zones together for example, it is called Internal BGP (iBGP), else it is called External BGP (eBGP).

2.2. Segment Routing

As stated in [FMT17] „the Segment Routing architecture does not assume a specific data plane implementation“. Nevertheless it is currently implemented for MPLS and IPv6, so technical details in this section might use MPLS or IPv6 as reference, but this does not exclude the use of other implementations.

To begin with the most basic term: what is a „Segment“? A Segment is defined as an „instruction“, which the active SR node executes. The term „instruction“ might sound confusing in context of routing and forwarding, but the list of instructions includes „forward packet through interface“ and „forward to node on shortest path“. It can also be used as an instruction to pass the packet to a specific application. As a whole, the decision to use the term „instruction“ is most probably based on the „network as a computer“ approach mentioned before.

Segments are identified by **Segment Identifiers (SIDs)**, which format depends on the implementation (e.g. MPLS label or an IPv6 address). A list of these instructions is called a **Segment List** or **SID List**. Each entry in this list is used to route the packet through the network, or through a chain of services. Each packet in Segment Routing carries this Segment List individually, hence „the state is in the packet“ is one of the main features of Segment Routing. The currently executed instruction is called the **Active Segment**.

Besides routing packets based on their current Active Segment, SR nodes also maintain the list of Segments if necessary. Three operations are defined for managing the Segment List:

1. PUSH: Insert new Segment at top of the List, marked as Active Segment.
2. CONTINUE: Active Segment is not completed, remains active.
3. NEXT: Active Segment is completed, next Segments becomes the Active Segment.

The distribution of SIDs must ensure the uniqueness of SIDs within an SR domain. SIDs used by all nodes in a domain are called **Global Segments**. All nodes within a domain understand every instruction identified by their Global Segment, installing the instruction in their forwarding tables. In contrast, **Local Segments** exist as well. They are also advertised inside the domain, but only executed by one single node. For illustration, an example with routing over an interface: if a Segment instructs nodes to forward the packet to NodeX, multiple nodes would support this Global Segment. But if one wants to enforce the execution on NodeY, the Local Segment of NodeY is used instead of a Global routing instruction.

A **Segment Routing Policy** is an ordered list of Segments. The list can either be created locally or by an external Path Computation Engine (PCE). The policy can be used for Traffic Engineering, Operations and Management (OAM) or Fast Reroute.

Regarding the Control Plane of Segment Routing, the usage of a routing protocol is the preferred method of Segment distribution. Here a differentiation is done between **IGP Segments** (using IS-IS or OSPF) and **BGP Segments**. The detailed implementation will be explained further later.

2.3. Topology Independent Loop-free Alternate Fast Reroute (TI-LFA)

In section 2.1.1 we learned about Fast Reroute (FRR) in MPLS. As explained in [FMT17], FRR was later extended and „ported“ to fit to other technologies. First, IPFRR took the features of IP networks into account, introducing Loop Free Alternate (LFA) and later Remote LFA (RLFA). Adding Segment Routing to the list of available technologies resulted in Topology Independent LFA (TI-LFA), which uses Segment Routing for FRR. [FMT17] uses the terms „pre-convergence“ and „post-convergence“ paths to reference the pre-failure and post-IGP-recalculation paths.

A short recap of **FRR** in MPLS: MPLS uses Label Switch Paths (LSPs) to steer labelled traffic on paths through multiple nodes inside a network. Using FRR, rerouting traffic after a link or node failure can temporarily be done via the precomputed FRR paths between **Point of Local Repair (PLR)** and **Merge Point (MP)** until the IGP recalculations take over the new topology.

But using this mechanism in an IP-based network has drawbacks. The paths are calculated to circumvent the failed link or node, enforcing the rerouting of all traffic towards the MP (seeing the paths as circuits), not acknowledging Equal Cost Multi-Path (ECMP) or simply leaving out the MP node in the traffic path (paths as IP routes). With **IPFRR**, the FRR calculations are based on address prefixes and therefore accommodate the routing of IP packets towards different destinations.

With IPFRR also came LFA and RLFA. The **Loop Free Alternate (LFA)** defines a node N which is a direct neighbour to the PLR node and which shortest path towards a destination node D is smaller than the shortest path from the PLR to D. Equation 1 shows the used formula.

$$dist(N, D) < (dist(N, PLR) + dist(PLR, D)) \tag{1}$$

This calculation is needed, because routing traffic via N, while N has a longer shortest path towards D than the PLR, N would then forward the traffic via the original PLR node, resulting in a loop. If the topology allows the PLR to choose a LFA, this node is called the „release point“ for D. Forwarding traffic for D via the LFA guarantees a loop-free reroute. Not having an LFA available results in lost traffic until the IGP recalculations are finished. The availability of LFAs in a given topology can be expressed as „**coverage**“, meaning the percentage of all pairs (PLR, D) which can route traffic via an available LFA node.

Solving the problem of low coverage of LFAs in FRR *RFC 7490* introduces another extension, **Remote LFA (RLFA)**. With RLFA a PLR can choose a node inside the network as a „virtual“ LFA. This node does not have to be a directly connected neighbour node anymore, but acts the same: as the release point of rerouted traffic. The needed calculations to find optimal RLFAs for each (PLR, D) pair is of course more complex than for LFAs, but the coverage $\text{Coverage}_{LFA} + \text{Coverage}_{RLFA}$ can be as high as 99% (see *RFC 7490*).

To further improve the coverage and operation handling, **Topology Independent LFA (TI-LFA)** was developed. For TI-LFA, the deployment of Segment Routing is needed as TI-LFA uses SR for repair paths (for nodes, links and Shared Risk Link Groups (SRLGs)). The features of TI-LFA include 100% coverage, simple operation and integration into the IGP. The integration of repair path route calculation into the IGP closes the gap between the temporary FRR and the post-convergence paths – they are the same. The calculation of repair paths for protected links and nodes by the IGP for TI-LFA *and* post-convergence means there is no intermediate repair path routing after failure of a link or node, the repair path is the same as the post-convergence path.

Segment Routing enables PLR nodes to route traffic via neighbours which would fail the criteria displayed in equation 1, because the packets are not routed by their IP prefix anymore but by their Segment Endpoints. The routing via a „longer“ shortest path is therefore enforceable, preventing loops. The PLR encodes a list of Segments and PUSHes the new hops onto the Segment List. Since the computation of these Segment Lists are done per destination, the routes are also optimal.

TI-LFA can also be used to **protect unlabelled IP traffic**. Traffic which is not routed with a Segments List can be encapsulated at the PLR and unpacked at the release point (which must also be a Segment Endpoint), continuing as before. This technique is an example of a partially deployed Segment Routing topology.

The usage of Segment Routing additionally allows to enforce to not only route via determined nodes but also which link (Adjacency Segment) a node must use for traffic forwarding. With this, TI-LFA enables **microloop avoidance**. Microloops can appear between nodes in some network topologies when the timing between pre-convergence (and FRR) and the installation of the new post-convergence routes in the Forwarding Information Bases (FIBs) of all nodes leads to unstable routes. An example is shown in figure 3: node A routes traffic via B to D. C also has a link to A and D, but the route has a higher metric (weight). If the link between A and B fails and the IGP distributes the new shortest path from A to D via C, but C still has the old FIB installed, C routes incoming traffic back to A (aiming for a better metric via A and B) and therefore creates a loop until C installs the IGP update. This error can be completely avoided by using SR to steer traffic to a stable release point which would not consider loops even with an old FIB. This approach is similar to RLFA.

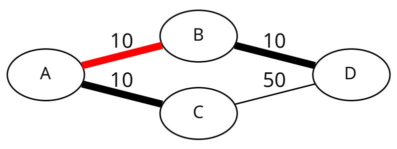

Figure 3: Illustration of a microloop-prone topology. The numbers indicate the metric of each link.

2.4. SRv6

Even though the concept of Segment Routing is designed without a specific protocol or software stack in mind, the current implementations exist for MPLS and IPv6. In this section we will have a closer look into the IPv6 implementation of Segment Routing, called SRv6 [LB17a]. Please note that the usage of address prefixes like `C1:` is not technically usable, since this prefix is not available in the current IANA IPv6 address space. For technical correctness, one could add the ULA `fd00::/8` prefix to each `interface address` (e.g. `fd00:C1::`).

2.4.1. Segment Routing Header

The Segment Routing Header (SRH) is a new type of Routing Header, used in the Routing Extension Header in IPv6 [SPRa]. This allows the usage of SRv6 in an IPv6 network with full compatibility to non-SR enabled routers.

Figure 4 shows the format of the SRH packet header. The packet format underwent multiple design change iterations, for example the „First Segment“ field was renamed to „Last Entry“, the flags were more or less removed from originally five available flags (Clean-up, Protected, OAM, Alert and HMAC) to none and the tag field was introduced, replacing a formerly „reserved“ area.

Following the three fields the list of Segments is appended to the packet. This list contains the SR Policy, meaning a list of 128 bit IPv6 addresses. Optionally this list is followed by a list of Type/Length/Value (TLV) objects.

The TLV objects contain additional information which can be examined by the destination router of the packet. TLVs can also be changed on route, depending on the first byte of the type field (`0` indicating that no change is possible, `1` indicating possible value changes). This implementation is used in context of the IPv6 Authentication Header, a computation of the Integrity Check Value (ICV) over a mutable field would not be possible and is therefore ignored. Two TLVs are defined so far: Padding and HMAC. While the Padding TLV is simply used for padding the packet to a multiple of 8 octets, the Hash-based Message Authentication Code (HMAC) TLV can provide packet integrity and authentication.

Three types of nodes are defined: Source, Transit and Endpoint. The following attributes and processing rules apply:

Source An SR Source Node sends an IPv6 packet with an SID as the destination address. The packet can be either new and without an SRH or it is an encapsulated IPv6 packet with an SRH, if the node is an ingress router for example. A Source Node decides which SR Policy must be applied. Creating only a single Segment can cause the Source Node to ommit the SRH. Otherwise the full SRH with the Segment List and optionally the TLV as well as all header fields is created and added to the packet.

Transit Transit Nodes forward IPv6 packets according to their destination address. These routers do not have to be capable of parsing the SRH. Transit Nodes do not examine the Routing Header, as their interface addresses never match the Destination Address of the packet. Therefore Transit Nodes simply forward the packet according to their IPv6 routing table towards the Destination Address.

Endpoint A Segment Endpoint Node receives an IPv6 packet with a destination address equal to one if its own interface addresses. These interface addresses can be either a configured IPv6 address or a Segment Identifier (SID) (which have the same format as IPv6 addresses). Further processing depends on the address type and the SRH.

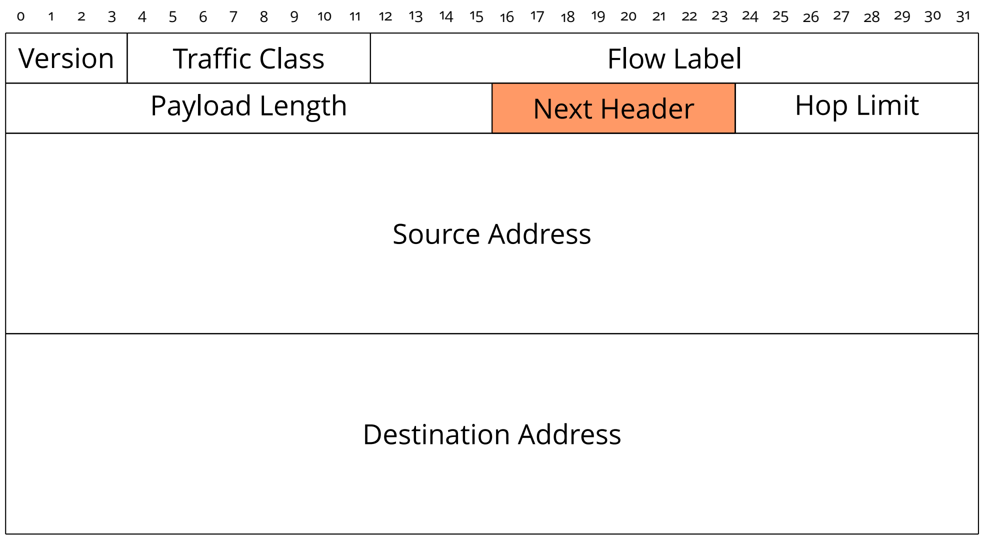

Figure 2: Header structure of the fixed IPv6 main packet header. Highlighted is the Next Header field which indicates the type of the next header field, e.g. TCP 0x06, UDP 0x11 if no additional IPv6 header extension field is used or – like in the case of SRv6 – the Routing Extension Header with type number 43 (0x2B).

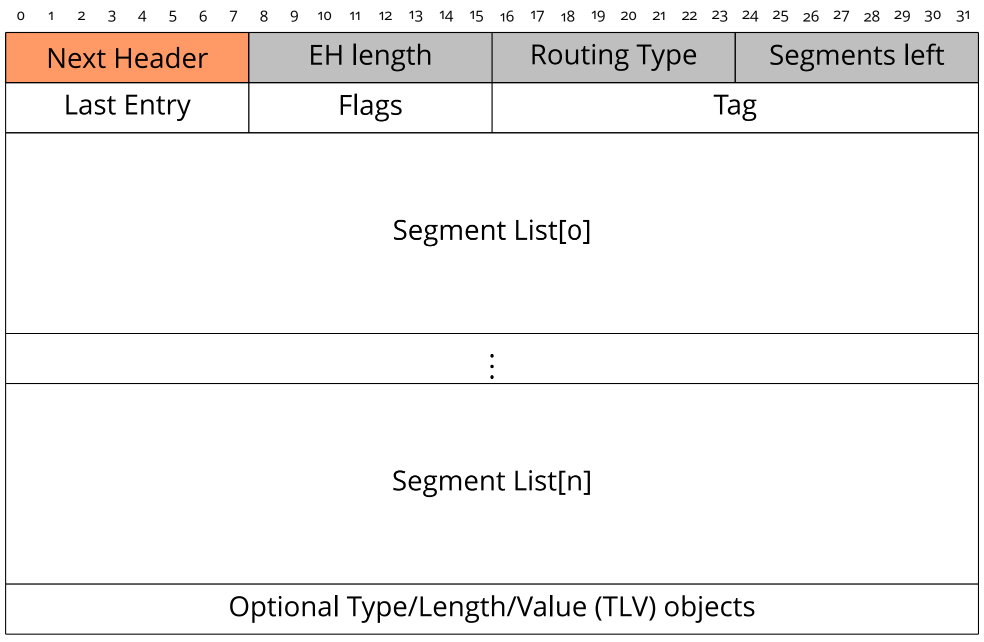

Figure 4: Header structure of the Segment Routing Header (SRH) according to `draft-ietf-6man-segment-routing-header-14` (earlier drafts called the „Last Entry“ field „First Segment“ and used up to five flags). The structure is based on the IPv6 Routing Header Extension which defines the first 32 bits. The Routing Type for SRH is not yet registered with IANA, number 4 is proposed. The *Segment List[i]* fields are 128 bit IPv6 addresses. EH: Extension Header.

To illustrate the procedure, let's say we have a fully meshed network with nine routers: the Source Node S, three Segment Endpoints E1, E2, ED and some Transit Nodes. See figure 5. A packet from the Source Node S shall be sent to Segment Endpoint Node ED, through the Segment Endpoint Nodes E1 and E2. The possible paths are displayed as directed arrows.

1. Source Node S originates a packet with destination E1.
2. The packet is received by E1, which is also the Destination Node and examines the SRH.
3. E1 uses the DA of E2 as the new IPv6 DA and forwards the packet to either T3 or T5.
4. Both T3 and T5 are Transit Nodes and forward the packet according to their routing tables to E2.
5. E2 is again a Destination Node and uses the address of ED as the new packet DA.
6. T4 forwards the packet to its destination, ED.
7. ED is the last Segment in the Segment List and acts accordingly to the instruction.

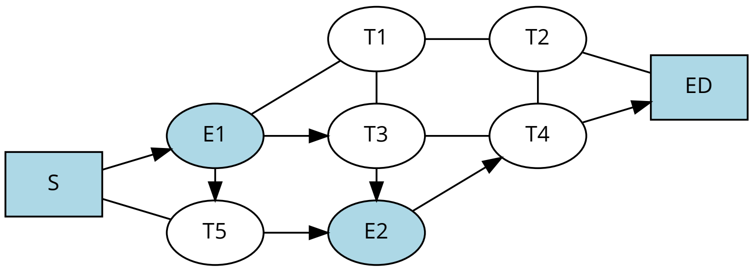

Figure 5: Illustration of a meshed network with some Endpoint nodes. All links have the same metric.

2.4.2. Network Programming with SRv6

In this section we will have a detailed look into the concept of Segment Identifiers (SIDs) and their use as instructions. Details are based on *draft-filsfils-spring-srv6-network-programming-05* and [Daw+17].

As mentioned in section 2.2 above, the term „Segment“ defines an encoded „instruction“ which is a function to be executed at a specific location in the network. These instructions can simply be packet forwarding towards a new destination address or the passing of the packet to a local application for further processing. The combination of SIDs in the Segment List (contained in the SRH) then allows for tasks more complex the basic routing algorithms.

With IPv6 the SIDs encoded in the Segment Routing Header (SRH) Segments List are 128-bit values (addresses). Each SRv6-capable node maintains a local SID table in which all locally instantiated Segments are contained, while the node itself is called the „parent node“ for these SIDs. The local SIDs do not have to be configured interface addresses on the node.

For illustration, an example: node N has a virtual loopback interface with address `C1::1/40` and the two SIDs `C1::100` and `C2::101` in its local SID table. The prefix for `C1::/40` is announced via IGP for this node, so a packet for SID `C1::100` would be routed towards this node. The `C2::` prefix however is not distributed by the IGP since the prefix is not used as an interface address. A packet destined for `C2::101` therefore would have to be routed to `C1::100` first, followed by the SID `C2::101`.

The local SID table contains a list of local SIDs, their bound instruction and possibly the parameters associated with this instruction. With a total length of 128 bits, SIDs are built as shown in figure 6. The parameter field might be ommitted if no parameters are needed for the instruction. The SID table matches on the longest-prefix basis to counter possible collisions when parameters are used.

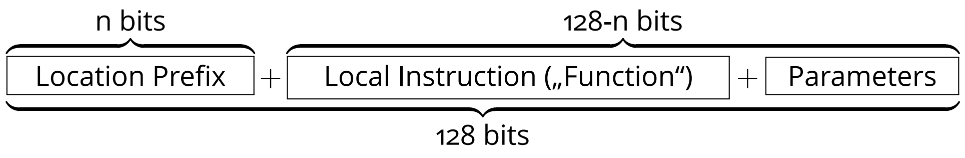

Figure 6: Structure of a SID entry in the Segments List contained in the SRH

Now that the structure and addressing of instructions is explained, we'll have a look at the list of currently specified Endpoint instructions. For an overview see the table in figure 7.

End The most basic instruction is the `End` function. It only serves as an Endpoint identifier for this node, a packet which Active Segment is destined for a SID bound to this instruction is routed to the node. The node decrements the Segments Left field, updates the Destination Address and forwards the packet.

End.X Using this function is similar to `End` but with the difference that forwarding the packet towards the new DA is not done by looking up the routing table but by enforcing the use of a specific outgoing link (adjacency).

End.T The function `End.T` is a second variant of the `End` function. With it a forwarding router sets the packets Destination Address to the next SID, but additionally uses a local table T to lookup the forwarding policy („The End.T is used for multi-table operation in the core.“).

End.DX2 Evaluates the Next Header of the inner payload. If the Next Header is of type 59 (No Next Header) forwards the packet payload on layer 2. L2VPN use-case.

End.DX2V Similar to DX2, but forwards according to the encapsulated VLAN tags. EVPN flexible cross-connect use-cases.

End.DT2U Similar to DX2, but forwards according to the encapsulated MAC Source and Destination Addresses. EVPN Bridging unicast use-cases.

End.DT2M Similar to DX2, but forwards inner layer 2 payload on all interfaces except the one specified in the optional argument `Arg.FE2`. EVPN Bridging BUM use-cases with ESI filtering.

End.DX6 Forwards on layer 3 according to the encapsulated IPv6 payload (type 41). IPv6 L3VPN use (equivalent of a per-CE VPN label).

End.DX4 Similar to DX6, but with an encapsulated IPv4 payload (type 4).

End.DT6 Similar to DX6, but forwards the inner IPv6 payload according to a lookup table T. IPv6 L3VPN use (equivalent of a per-VRF VPN label).

End.DT4 Similar to DT6, but with an encapsulated IPv4 payload.

End.DT46 A combination of DT6 and DT4, forwarding handled depending on the inner type.

End.B6 Does not change the received SRH, adds a new SRH to the packet and sets the IPv6 Destination Address according to the SRv6 Policy.

End.B6.Encaps Decrements the SRH SL field and updates the IPv6 Destination Address. Then encapsulates the packet with a new IPv6 and SRH header and sets the outer DA to the inner Source Address, the outer DA according to the SRv6 Policy.

End.BM Decrements the SL and updates new IPv6 DA, but also pushes an MPLS label stack into the packet and forwards according to MPLS.

End.S Looks up the last SID in a specified table T and forwards accordingly.

T Forwarding without inspecting the SRH, if the DA is neither a local address nor a local SID.

T.Insert Prepending of an additional SRH, with a list of Endpoints according to an SRv6 Policy. Then forwarding according to the new DA in the outer SRH.

T.Encaps Similar to `T.Insert`, but the packet is encapsulated into a new IPv6 header with its own SRH. Used for example for L3VPN.

T.Encaps.L2 Similar to `T.Encaps`, but the whole layer 2 frame (e.g. Ethernet) is encapsulated.

Each Endpoint Type has its own identifying value, used in the „Local Instruction (Function)" bit range inside the IPv6 address. The draft proposes the usage of values in the range `0x0000` to `0xFFFF`, meaning up to 65535 Types, with specified and private ranges.

Instruction	Description of Instruction	Routing* BGP-LS and . . .
End	Instantiation of a prefix SID	IGP
End.X	Layer-3 cross-connect. Instantiation of an adjacency SID.	IGP
End.T	Specific IPv6 table lookup	IGP
End.DX2	Decapsulation and Layer-2 cross-connect.	BGP
End.DX2V	Decapsulation and VLAN L2 table lookup.	BGP
End.DT2U	Decapsulation and unicast MAC L2 table lookup.	BGP
End.DT2M	Decapsulation and L2 table flooding.	BGP
End.DX6	Decapsulation and IPv6 cross-connect.	IGP, BGP
End.DX4	Decapsulation and IPv4 cross-connect.	BGP
End.DT6	Decapsulation and IPv6 table lookup.	IGP, BGP
End.DT4	Decapsulation and IPv4 table lookup.	BGP
End.DT46	Decapsulation and IP table lookup.	BGP
End.B6	Endpoint bound to an SRv6 policy. Instantiation of a Binding SID	
End.B6.Encaps	Endpoint bound to an SRv6 encapsulation Policy. Instantiation of a Binding SID	
End.BM	Endpoint bound to an SR-MPLS Policy. SRv6/SR-MPLS instantiation of a Binding SID	
End.S	Endpoint in search of a target in table T.	
T.S	Transit behaviour	
T.Insert	Transit behaviour with insertion of an SRv6 policy	
T.Insert.Red	Transit behaviour with reduced insertion of an SRv6 policy	
T.Encaps	Transit behaviour with encapsulation in an SRv6 policy	
T.Encaps.Red	Transit behaviour with reduced encaps in an SRv6 policy	
T.Encaps.L2	T.Encaps behaviour of the received L2 frame	
T.Encaps.L2.Red	Transit with reduce encaps of received L2 frame	

Figure 7: List of currently specified Endpoint instructions and Transit Behaviours. *Note: all Endpoint SIDs use Border Gateway Protocol Link State (BGP-LS) for routing signalling, additionally to the one mentioned in column „Routing".

2.4.3. Distributing SIDs and Capabilities with OSPFv3

As already mentioned in section 2.2 Segment Routing uses established Routing Protocols (IS-IS, OSPF) for label and SID distribution. This section will examine the SRv6 implementation of Open Shortest Path First (OSPF) for Internet Protocol version 6 (IPv6) (OSPFv3) in more detail, based on *draft-li-ospf-ospfv3-srv6-extensions-01*.

To recap the Segment Routing Header (SRH): an SRH-enabled IPv6 packet includes the IPv6 source and destination addresses and a Next Header with type 43 (Routing Extension Header). This EH contains the SRH (proposed as type 4) with all fields as shown in figure 4.

Routers which implement the handling of packets with an SRH need a way to advertise their capabilities, meaning their Segment Identifiers (SIDs). In OSPFv3 three types of extensions are used to implement SRv6: the Capabilities TLV, the Node SID TLV for nodes and the Link Attribute TLV for adjacencies. See *RFC 7770* and *RFC 8362* for further details about TLVs in OSPF. The **Capabilities TLV** (a type of OSPFv3 Router Information LSA TLV) is used to announce the general SRv6 support as well as the supported SRv6 features by each router. Currently the draft requires the flooding of these LSAs in the area scope. The Capabilities TLV has a fixed header and uses Sub-TLVs as payload:

Maximum SL Sub-TLV Maximum allowed value of the Segments Left header field. Default 0.

Maximum End Pop SRH Sub-TLV Maximum number of SIDs in the top SRH in an SRH stack to which the router can apply Penultimate Segment Popping (PSP) or Ultimate Segment Popping (USP) flavors. If 0 or not advertised, the application of PSP/USP must be considered unsupported by this router.

Maximum T.Insert SRH Sub-TLV Maximum number of SIDs which can be inserted as part of the T.Insert instruction. If 0 or not advertised, the support for T.Insert must be considered unsupported.

Maximum T.Encap SRH Sub-TLV Maximum number of SIDs which can be included as part of the T.Encap instruction. Requires the E-flag to be set in the TLV header.

Maximum End D SRH Sub-TLV Maximum number of SIDs in an SRH when applying End.DX6 and End.DT6 functions.

The **Node SID TLV** extension is used to advertise the locally available SIDs (installed in the local SID table) and their endpoint functions. See section 2.4.2 for further details about functions and transit behaviours. The OSPFv3 Router Information LSA header for this extension is composed of type and flags as well as the function code (see table 7) and a variably-sized SID linked to the function (up to 128 bits).

While Node SID TLVs are used to identify and advertise node-local functions, **SID Link Attribute Sub-TLV** identify adjacencies - connected links which can be referenced for traffic flows. For example the End.X function uses a adjacency SID to specify the interface to be used. Here, two versions exist:

SID Link Attribute Sub-TLV For point-to-point, point-to-multipoint or broadcast adjacencies.

SID LAN Link Attribute Sub-TLV For links connected to broadcast and NBMA networks.

Regarding security the draft mentiones that no new security considerations are introduced and references *RFC 5340* and *RFC 8362* for further details.

2.4.4. Linux Kernel implementation

Handling the Segment Routing Header (SRH) in the Linux Kernel is possible since the official release version 4.10 (released 2017-02-19, added in `net/ipv6/seg6.c`) and was documented in [LB17a] by David Lebrun and Olivier Bonaventure. Their work began in April 2010 and is seen as the only open-source implementation of SRv6 which supports both endhost and router functionalities (whereas FD.io with VPP focuses on router support only).

According to the paper the implementation supports receiving and parsing of IPv6 packets with SRH in the `INPUT` of the Linux network stack. Optionally the HMAC validation is done for SHA-1 and SHA-256 based HMACs. The requirement of the presence of HMAC is configurable via the `net.ipv6.conf.*.seg6_require_hmac` sysctl value.

A Segment Left value of 0 indicates that the local machine is an egress node. In this case the Next Header is inspected to determine the next destination of the packet. Since the 4.10 kernel did not support handling encapsulation of different types other than IPv6 in IPv6, the authors bypassed the default processing (and possible failure of decapsulation) by re-injecting inner IP headers back into the ingress interface of the local machine.

Additionally, if both the Destination Address and the Next Segment exist on the local machine, the packet is directly looped back into the SRH handling function after the first SRH handle iteration (decrementing of SL and updating the DA) to avoid using the `ip6_input` function again.

For routing, the local machine must support adding and removing SRH in IPv6 packets. The userspace `iproute2` tool was modified to insert and remove system-wide routes which allow en- and decapsulation of packets destined for the route prefix. Example: `ip -6 route add fc42::/64 encap seg6 mode encap segs fc00::1,2001:db8::1,fc10::7 dev eth0`. The outgoing interface must not be a virtual loopback interface.

Measurements presented in section 3 of the paper show that with some optimisations, SRH processing without HMAC (insertion as both inline and as encapsulation) reduces the throughput performance to around 90% compared to plain IPv6 forwarding. In contrast, software HMAC validation causes a much bigger performance penalty. This drawback is also mentioned in [SPRa], aiming for a hardware based solution to this implementation.

2.5. Security

Since some aspects of Segment Routing are based on the Source Routing concept, the disadvantages of Source Routing needed to be addressed during the development of Segment Routing. The following security issues of Source Routing are described in *RFC 4942* and *RFC 5095*:

Amplification Causing loops among a set of routers, resulting in bandwidth exhaustion (DoS).

Reflection Hiding the real source of attacks by routing attacks via intermediate hops.

Bypass Using an intermediate node to access a connected private network.

For Segment Routing the following two node types have to be differentiated to further look into the threat model: trusted nodes within an SR domain and untrusted nodes which are not part of the SR domain. As stated in [SPRa]:

> non-trusted nodes do not participate as either SR processing is not enabled by default or because they only process SRH from nodes within their domain. Moreover, all SR nodes ignore SRH created by outsiders based on topology information (received on a peering or internal interface) or on presence and validity of the HMAC field.

Therefore intermediate nodes should only act on valid and authorised and possibly HMAC-authenticated SRH. Other security threats listed in [SPRa] include:

Service stealing Non-participating nodes need to be prevented from using information or services they might receive by examining SR packets.

Topology disclosure The SRH contains the full list of nodes the packet is routed through. Stealing this information might expose routing configurations. This information might also be gathered by issuing a traceroute on an SR-enabled path.

ICMP generation As defined in *RFC 8200* nodes which receive packets with an unknown Routing Header and with a Segments Left value bigger than 0 are required to reply with an ICMP Parameter Problem to the packet's sender. This could be used by an attacker to cause any non-SR node to send ICMP packets to a single router, called a reflection attack. This attack is not SRH-specific, since attackers could use any Routing Type which is not understood by standard routers. ICMP rate-limiting should be applied.

SRH Security Fields The HMAC TLV contains a signature of the whole SRH to be used for validity verification, integrity checks and authorisation. An attacker would need to have access to the currently used pre-shared secret to forge the HMAC TLV. The trust model of Segment Routing foresees the evaluation of the HMAC TLV only at the ingress router. All following nodes might ignore the TLV because they trust the upstream router and it speeds up packet routing.

Hash algorithm The HMAC field has a size of 256 bits, so shorter or longer output of hash implementations must adjust their output before insertion.

HMAC performance Enabling HMAC causes performance loss. Some aspects lower this impact, e.g. using HMAC only for domain-external packets and only validating the header at the ingress router.

Pre-shared key management The `Key-id` field allows the usage of multiple pre-shared secrets and the choice of hash algorithm per secret. Segment Routing does not introduce any new approaches to key distribution.

Regarding the deployment of SR onto a network of nodes, the differentiation mentioned above is taken into account again: SR nodes within the SR domain are trusted to generate IPv6 packets with an SRH. Endpoint nodes which receive SRH on interfaces which are part of the SR domain may process the packet and route it to a local segment if needed. Nodes outside of the SR domain are generally untrusted. Ingress routers which receive packets which contain an SRH with a list of internal SIDs should discard these packets to avoid any attacks against the SR domain originating outside of the trusted network. For this, infrastructure Access Control Lists (iACLs) can be applied. To extend the domain across ingress routers, nodes outside of the domain might request an HMAC-signed SRH for its originating traffic. Valid packets arriving at the ingress router might then be routed, even though the sending router does not belong to the trusted SR domain.

With the possibility to create valid SRHs outside of the SR domain, the threat of topology disclosure gains new weight. Routers which forward packets from an external Source Node might intercept the content of the SRH, since HMAC does not encrypt the content and encryption mechanisms like IPsec are not applicable, due to the order of headers in IPv6 (with SRH as a Routing Header type coming before any IPsec related header fields). The exposure of Segments and TLV content should therefore be considered in Segment Routing deployments and might be limited by reducing non-domain SR traffic and by restricting the set of receiving SIDs to nodes which enforce HMAC verification (using iACLs).

Lastly, the approach of using Loose Source Routing (LSR) might conflict with Best Current Practice BCP-38 „Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing“. Routers which receive a packet routed by Segment Routing algorithms might drop valid SR packets due to their unexpected routing paths. To prevent this BCP-38 should only be enforced on SR-domain ingress routers.

2.6. Practical use cases in SDN, Cloud Computing and further contexts

With **Software Defined Networking (SDN)**, **Network Function Virtualization (NFV)** and **Service Function Chains (SFCs)** new architectural designs and software deployment approaches rise in popularity. While concepts like the TCP/IP network stack motivated the separation of network configuration and the applications running on the connected nodes, newer Cloud Computing infrastructure designs merge networking and software stacks together.

With a strong focus on flexibility and service modularity, combining traffic routing and the provisioning of services enables software stacks to automatically scale and reorganise. The overall achievements SDN and NFV provide include: a more flexible infrastructure, cost reduction by using standard off-the-shelf hardware, reduced energy consumption by utilising hardware resources more efficiently, independence from proprietary firmware implementations and improved efficiency in network operations.

As mentioned in section 1.1 **SDN** aims for a change in network routing, separating the control plane from the data plane. The routing logic applied by all routers inside a network is shifted from a decentralised coordination (e.g. by using OSPF) towards a centralised instance, the SDN Controller. SDN and the OpenFlow protocol are promoted by the Open Networking Foundation (ONF) founded in 2011.

In contrast, **NFV** is meant to replace existing hardware-based network functionality with software implementation and virtualisation. Examples include firewalls, Intrusion Detection Systems (IDSs) or load balancers which would be migrated from standalone hardware devices into virtual instances. [SC12] (published in 2012) provides more details about the reasoning behind the development of NFV architectures:

> Network Operators' networks are populated with a large and increasing variety of proprietary hardware appliances. To launch a new network service often requires yet another variety and finding the space and power to accommodate these boxes is becoming increasingly difficult; compounded by the increasing costs of energy, capital investment challenges and the rarity of skills necessary to design, integrate and operate increasingly complex hardware-based appliances.
>
> Moreover, hardware-based appliances rapidly reach end of life, requiring much of the procure-design-integrate-deploy cycle to be repeated with little or no revenue benefit. Worse, hardware lifecycles are becoming shorter as technology and services innovation accelerates, inhibiting the rollout of new revenue earning network services and constraining innovation in an increasingly network-centric connected world.
>
> Network Functions Virtualisation aims to address these problems by leveraging standard IT virtualisation technology to consolidate many network equipment types onto industry standard high volume servers, switches and storage, which could be located in Datacentres, Network Nodes and in the end user premises. We believe Network Functions Virtualisation is applicable to any data plane packet processing and control plane function in fixed and mobile network infrastructures.

Further development of NFV specifications and platform tools was done at first by the European Telecommunications Standards Institute (ETSI) and is now continued in the **Open Platform for NFV (OPNFV)**, started by the Linux Foundation in 2014. In this context also the Linux Foundation Project **Cloud Native Computing Foundation (CNCF)** should be mentioned. Since some of the conceptual developments in NFV are oriented on the OpenStack and Kubernetes implementations, both projects of the CNCF.

As mentioned in some sources, the developers of Segment Routing had the concept of „Network as a Computer“ in mind. Additionally, according to [FMT17] and the talks on [Day16] Segment Routing was designed to be implemented in any fitting protocol stack (though MPLS and

IPv6 are the most prominent currently) and to be used in the context of SDN. Reading [Daw+17] and [CA18] is also strongly recommended as it explains SRv6 and Service Chaining with SRv6 further.

At last, to now come back to the connection between SRv6 and SDN/NFV, here is an example of an SRv6 SFC used by John Leddy in [Day16] called a „Video Service Chain“: SRv6 enables the routing of packets from a video source node through a transcoder node and a JIT packaging node towards a CDN distribution server node. The packet would therefore be „programmed“ by encoding this processing path into the SRH Endpoint List. Extending this approach one could also built „network libraries“ and use abstract code for routing.

This example can be ported to firewalls and other network utilities like the ones mentioned above in context of NFV and should illustrate the possibilities of combining SRv6 with SDN and NFV.

Appendices

A. Current implementations, standards and related projects

The following list is a collection of related projects and groups which were mentioned in this paper or in referenced resources.

segment-routing.net Cisco website of the Segment Routing project (focusing MPLS) [Sysc]

segment-routing.org Website for SRv6 in the Linux Kernel, maintained by David Lebrun [Leb]

Internet Engineering Task Force (IETF) Working (WG) and Research (RG) Groups

- **SPRING** Source Packet Routing in Networking WG
- **6MAN** IPv6 Maintenance WG
- **RTG** Routing Area WG
- **OSPF** Open Shortest Path First IGP WG
- **NFVRG** Network Function Virtualization RG
- **PANRG** Path Aware Networking Proposed RG

Linux Foundation Projects

- **FD.io, DPDK** The Fast Data Project is dedicated to multiple sub projects surrounding universal data planes. Hosting the Vector Packet Processing (VPP) stack. Makes use of the Data Plane Development Kit (DPDK). [FDia]
- **Open Platform for NFV (OPNFV)** Projects regarding Network Function Virtualization (NFV).
- **Cloud Native Computing Foundation (CNCF)** Projects regarding orchestration of containers and microservices (Kubernetes, Prometheus).

Open Networking Foundation (ONF) Umbrella organisation for the development of SDN, OpenFlow and further technologies

A.1. Implementations

According to [SPRa] the following implementations of Segment Routing exist to date:

Products	Status	SRH	END	TLV
Linux Kernel (since v4.14)	Production	x	x	x
Cisco Systems IOS XR and IOS XE	Pre-production	x	x	
FD.io VPP/Segment Routing for IPv6	Production	x	x	
Barefoot Tofino NPU	Prototype		x	
Juniper Trio and vTrio NPUs	Prototype	x	(x)[1]	

Figure 8: Current implementations of the SRH according to [SPRa].
SRH: adds SRH, END: performs END processing, TLV: supports TLV processing

[1]Only processing SIDs where the SID is an interface address

A.2. Standards and drafts

The following list consists of referenced Request For Comments (RFCs), RFC drafts and Best Current Practices (BCPs).

RFC 791	Internet Protocol Version 4, 1981
RFC 792	ICMP, 1981
RFC 1256	ICMP Router Discovery Messages, 1991
RFC 1322	A Unified Approach to Inter-Domain Routing, 1992
RFC 1812	Requirements for IP Version 4 Routers, 1995
RFC 1940	Source Demand Routing: Packet Format and Forwarding Specification (v1), 1996
RFC 3031	Multiprotocol Label Switching Architecture, 2001
RFC 3209	RSVP-TE: Extensions to RSVP for LSP Tunnels, 2001
RFC 4090	Fast Reroute Extensions to RSVP-TE for LSP Tunnels, 2005
RFC 4271	A Border Gateway Protocol 4 (BGP-4), 2006
RFC 4302	IP Authentication Header, 2005
RFC 4303	IP Encapsulating Security Payload (ESP), 2005
RFC 4861	Neighbor Discovery for IP version 6 (IPv6), 2007
RFC 5286	Basic Specification for IP Fast Reroute: Loop-Free Alternates, 2008
RFC 5340	OSPF for IPv6, 2008
RFC 6275	Mobility Support in IPv6, 2011
RFC 7490	Remote Loop-Free Alternate (LFA) Fast Reroute (FRR), 2015
RFC 7665	Service Function Chaining (SFC) Architecture, 2015
RFC 7770	Extensions to OSPF for Advertising Optional Router Capabilities, 2016
RFC 7855	SPRING Problem Statement and Requirements, 2016
RFC 8200	Internet Protocol, Version 6 (IPv6) Specification, 2015
RFC 8300	Network Service Header (NSH), 2018
RFC 8354	Use Cases for IPv6 Source Packet Routing in Networking, 2018
RFC 8355	Resiliency Use Cases in SPRING Networks, 2018
RFC 8362	OSPFv3 Link State Advertisement (LSA) Extensibility, 2018
RFC 8402	Segment Routing Architecture, 2018
draft	SRv6 Network Programming, 2018
draft	Service Programming with Segment Routing, 2018
draft	IPv6 Segment Routing Header (SRH) (6man WG)
BCP-38	Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing

B. Sources, References, List of Figures, Glossary

References

[Hag09] Silvia Hagen. *IPv6 Essentials*. O'Reilly Media, 2009. ISBN: 9780596105730.

[SC12] SDN and OpenFlow World Congress. *Network Functions Virtualisation. An Introduction, Benefits, Enablers, Challenges and Call for Action*. See https://portal.etsi.org/NFV/NFV_White_Paper.pdf. Oct. 2012.

[Daw+17] Gaurav Dawra et al. *SRv6: Network as a Computer and Deployment use-cases*. See https://pc.nanog.org/static/published/meetings/NANOG71/1445/20171005_Dawra_Segment_Routing_Ipv6_v1.pdf. Oct. 2017.

[FMT17] Clarence Filsfils, Kris Michielsen, and Ketan Talaulikar. *Segment Routing*. CreateSpace, 2017. ISBN: 978-1-5423-6912-1.

[LB17a] David Lebrun and Olivier Bonaventure. "Implementing IPv6 Segment Routing in the Linux Kernel". In: *Applied Networking Research Workshop 2017*. See https://irtf.org/anrw/2017/anrw17-final3.pdf. July 2017.

[LB17b] David Lebrun and Olivier Bonaventure. *Implementing IPv6 Segment Routingng in the Linux Kernel*. See https://inl.info.ucl.ac.be/system/files/ANRW-SR.pdf. July 2017.

[CA18] Pablo Camarillo and Ahmed Abdelsalam. *SRv6 Network Programming - FD.io VPP and Linux*. See https://fosdem.org/2018/schedule/event/srv6/attachments/slides/2231/export/events/attachments/srv6/slides/2231/SRv6_Network_Programming_FOSDEM2018.pdf. FOSDEM'18, Mar. 2018.

Online resources

[Pro02] Joseph Z Provo. *Source-routing not considered harmful*. Dec. 6, 2002. URL: http://www.gweep.net/~crimson/network/lsrr.html (visited on 06/10/2018).

[Sys08] Internet Security Systems. *Source Routing*. Feb. 24, 2008. URL: https://web.archive.org/web/20080224174623/http://www.iss.net/security_center/advice/Underground/Hacking/Methods/Technical/Source_Routing/default.htm.

[Fil13] Clarence Filsfils. *Segment Routing - Simplifying the Network*. June 5, 2013. URL: https://youtu.be/371DUGHjPwk.

[Cis16] Cisco. *Introduction to Segment Routing*. Nov. 23, 2016. URL: https://www.youtube.com/watch?v=wGYXbL3upBg.

[Day16] Tech Field Day. *Segment Routing Field Day Roundtable*. 2016. URL: https://www.youtube.com/playlist?list=PLinuRwpnsHacUlfUCrVstvpzURnK_M3iI.

[Rou16] Segment Routing. *Topology Independent LFA (TI-LFA)*. Sept. 27, 2016. URL: http://www.segment-routing.net/tutorials/2016-09-27-topology-independent-lfa-ti-lfa/.

[Tec16a] TechWiseTV. *Segment Routing for Service Providers*. Jan. 14, 2016. URL: https://www.youtube.com/watch?v=pDIRXLajRXo.

[Tec16b] TechWiseTV. *Segment Routing for the Data Center*. June 2016. URL: https://youtu.be/m2l77Ny7GvY.

[Cla] Cisco Systems Clarence Filsfils. *Segment Routing - Deployment Experience and Technology Update*. URL: https://www.youtube.com/watch?v=EUxm9EH1wyQ.

[FDia] FD.io. *FD.io - The Fast Data Project*. URL: https://fd.io/.

[FDib] FD.io. *FD.io VPP SRv6 implementation*. URL: https://wiki.fd.io/view/VPP/Segment_Routing_for_IPv6.

[Gur] Fortinet Guru. *Intermediate System to Intermediate System Protocol (IS-IS)*. URL: https://www.fortinetguru.com/2016/06/intermediate-system-to-intermediate-system-protocol-is-is/2/.

[IANa] IANA. *Internet Protocol Version 4 (IPv4) Parameters*. URL: https://www.iana.org/assignments/ip-parameters/ip-parameters.xhtml.

[IANb] IANA. *Internet Protocol Version 6 (IPv6) Parameters*. URL: https://www.iana.org/assignments/ipv6-parameters/ipv6-parameters.xhtml.

[Kap] Amar Kapadia. *Understanding OPNFV Starts Here*. URL: https://www.linux.com/blog/opnfv/2017/8/understanding-opnfv-starts-here.

[Leb] David Lebrun. *SRv6 - Linux Kernel implementation*. URL: http://www.segment-routing.org.

[LB] David Lebrun and Olivier Bonaventure. *Implementing IPv6 Segment Routing in the Linux Kernel*. URL: https://inl.info.ucl.ac.be/publications/implementing-ipv6-segment-routing-linux-kernel.

[Net] Cisco Learning Network. *Introduction to Segment Routing*. URL: https://learningnetwork.cisco.com/blogs/vip-perspectives/2018/03/23/introduction-to-segment-routing.

[RTG] IETF WG RTG. *Topology Independent Fast Reroute using Segment Routing*. URL: https://datatracker.ietf.org/doc/draft-bashandy-rtgwg-segment-routing-ti-lfa/.

[SPRa] IETF WG SPRING. *IETF datatracker: IPv6 Segment Routing Header (SRH)*. URL: https://datatracker.ietf.org/doc/draft-ietf-6man-segment-routing-header/.

[SPRb] IETF WG SPRING. *Source Packet Routing in Networking (SPRING) WG documents*. URL: https://datatracker.ietf.org/wg/spring/documents/.

[Sysa] Cisco Systems. *Introduction to Segment Routing*. URL: https://www.cisco.com/c/en/us/td/docs/ios-xml/ios/seg_routing/configuration/xe-3s/segrt-xe-3s-book/intro-seg-routing.html.

[Sysb] Cisco Systems. *MPLS Traffic Engineering (TE) - Fast Reroute (FRR) Link and Node Protection*. URL: https://www.cisco.com/c/en/us/td/docs/ios/12%5C_0s/feature/guide/gslnh29.html.

[Sysc] Cisco Systems. *Segment Routing Project (Links and papers)*. URL: http://www.segment-routing.net (visited on 06/20/2018).

List of Figures

Glossary

Control Plane Protocols and data tables used to collect and calculate the topology of a network. 4

Data Plane Routing of packets based on routing information collected via the Control Plane. 4

Software Defined Networking (SDN) Network architecture with separated Data and Control Planes (using centralized controllers). 4, 5, 19

virtual loopback interface A Cisco-related term for a virtual router interface with a unique address which is always up. 13, 17

Acronyms

ARP Address Resolution Protocol. 7
AS Autonomous System. 8

BCP Best Current Practice. 18, 22
BGP Border Gateway Protocol. 8
BGP-LS Border Gateway Protocol Link State. 15

CDN Content Delivery Network. 6
CNCF Cloud Native Computing Foundation. 19, 21

DAD Duplicate Address Detection. 7
DHCP Dynamic Host Configuration Protocol. 7
DoS Denial of Service. 4
DPDK Data Plane Development Kit. 21

ECMP Equal Cost Multi-Path. 9
EH Extension Header. 7
ETSI European Telecommunications Standards Institute. 19

FEC Forwarding Equivalence Class. 6
FIB Forwarding Information Base. 10
FRR Fast Reroute. 7, 9

HMAC Hash-based Message Authentication Code. 11

iACL infrastructure Access Control List. 18
ICMP Internet Control Message Protocol. 7
ICV Integrity Check Value. 11
IDS Intrusion Detection System. 19
IETF Internet Engineering Task Force. 21
IPsec Internet Protocol Security. 8
IPv4 Internet Protocol version 4. 4, 7
IPv6 Internet Protocol version 6. 5, 7, 16
IS-IS Intermediate System to Intermediate System. 8, 9, 16

LDP Label Distribution Protocol. 6
LER Label Edge Router. 6
LFA Loop Free Alternate. 9
LIB Label Information Base. 6
LSA Link-State Advertisement. 8
LSP Label Switch Path. 6, 9
LSR Loose Source Routing. 4, 6, 18

MP Merge Point. 7, 9
MPLS Multi Protocol Layer Switching. 5, 6

NBMA Non-Broadcast Multiple Access. 16
NDP Neighbor Discovery Protocol. 7
NFV Network Function Virtualization. 6, 19, 21

OAM Operations and Management. 9, 11
ONF Open Networking Foundation. 19, 21
OPNFV Open Platform for NFV. 19, 21
OSPF Open Shortest Path First. 8, 9, 16

PBR Policy-based Routing. 4
PCE Path Computation Engine. 9
PDU Protocol Data Unit. 8
PHP Penultimate Hop Popping. 6
PLR Point of Local Repair. 7, 9
PSP Penultimate Segment Popping. 16

RFC Request For Comments. 22
RLFA Remote LFA. 9, 10
RSVP-TE Resource Reservation Protocol for Traffic Engineering. 5, 6

SFC Service Function Chain. 19
SID Segment Identifier. 9, 11, 13, 16
SLAAC Stateless Address Autoconfiguration. 7
SPRING Source Packet Routing in Networking. 5, 22
SRH Segment Routing Header. 5, 11–13, 16, 17
SRLG Shared Risk Link Group. 10
SSR Strict Source Routing. 4

TE Traffic Engineering. 9
TI-LFA Topology Independent LFA. 9, 10
TLV Type/Length/Value. 7, 8, 11, 12

ULA Unique Local Address. 7
USP Ultimate Segment Popping. 16

VPP Vector Packet Processing. 5, 21
VRF Virtual Routing and Forwarding. 8