Segment Routing: IPv6, Implementation and a Practical Use Case

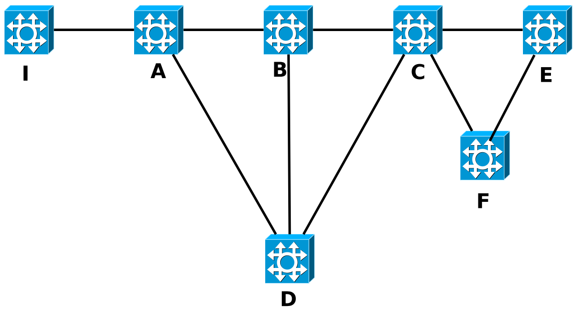
David Lebrun
<david.lebrun@uclouvain.be>

Université Catholique de Louvain
Louvain-la-Neuve, Belgium

Segment Routing

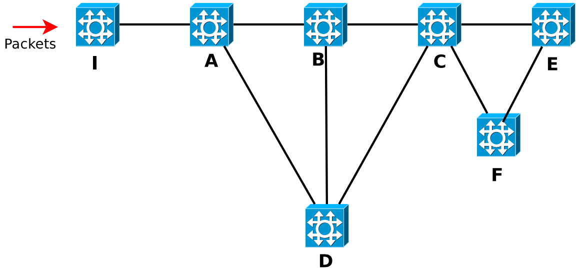
- Source routing
- Path encoded as stack of segments (IPv6 addresses)
- Node and adjacency segments
- Segments distributed through IGP

Illustration



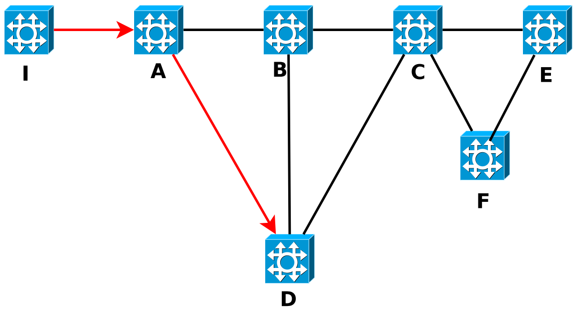
- Abstract SR Header
  - `Segments = SD, SB, SS, SF, SE`
  - `Ptr = Segments[0] (SD)`

Illustration



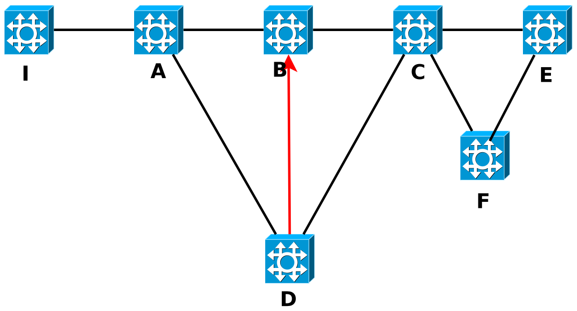
- Abstract SR Header
  - `Segments = SD, SB, SS, SF, SE`
  - `Ptr = Segments[0] (SD)`

Illustration



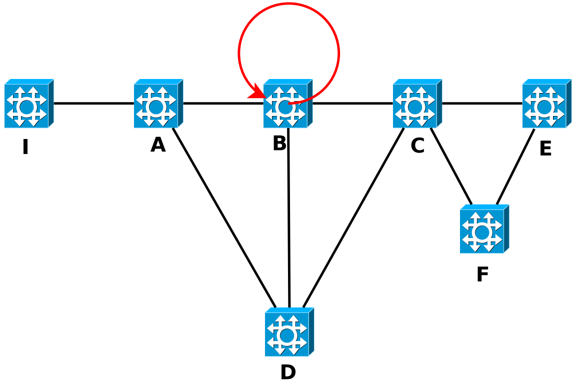
- Abstract SR Header
  - `Segments = SD, SB, SS, SF, SE`
  - `Ptr = Segments[0] (SD)`

Illustration



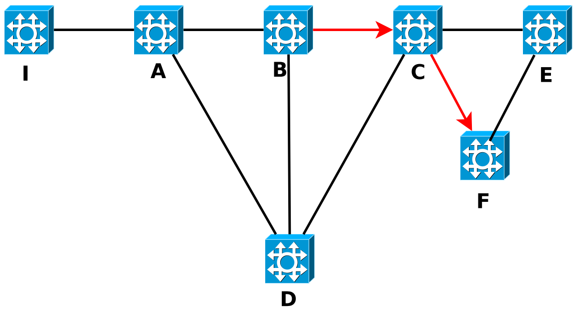
- Abstract SR Header
  - `Segments = SD, SB, SS, SF, SE`
  - `Ptr = Segments[1] (SB)`

Illustration



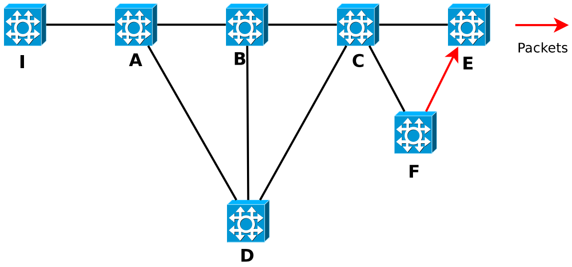
- Abstract SR Header
  - `Segments = SD, SB, SS, SF, SE`
  - `Ptr = Segments[2] (SS)`

Illustration



- Abstract SR Header
  - `Segments = SD, SB, SS, SF, SE`
  - `Ptr = Segments[3] (SF)`

Illustration



- Abstract SR Header
  - `Segments = SD, SB, SS, SF, SE`
  - `Ptr = Segments[4] (SE)`

Use cases

- Link/node disjoint paths
- Dynamic network reconfiguration
- Middleboxing (firewalls, etc)
- User/customer-level path selection
- ...

IPv6 Segment Routing

- Segment = IPv6 address
- New extension header: Routing Header type 4
- Security concerns of RH0 addressed with HMAC field

IPv6 Segment Routing extension header

```
0                   1                   2                   3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Next Header   |  Hdr Ext Len  | Routing Type  | Segments Left |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| First Segment |             Flags             |  HMAC Key ID  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
|            Segment List[0] (128 bits ipv6 address)            |
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
                              ...
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
|            Segment List[n] (128 bits ipv6 address)            |
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
|                       HMAC (256 bits)                         |
|                        (optional)                             |
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

IPv6 Segment Routing extension header

- Two flags currently defined: cleanup and fast reroute
- Cleanup is important
  - Penultimate hop removes SRH
  - Avoid data leak when packets exit network

SR-IPv6 forwarding algorithm

Algorithm 1 SR Segment Endpoint processing

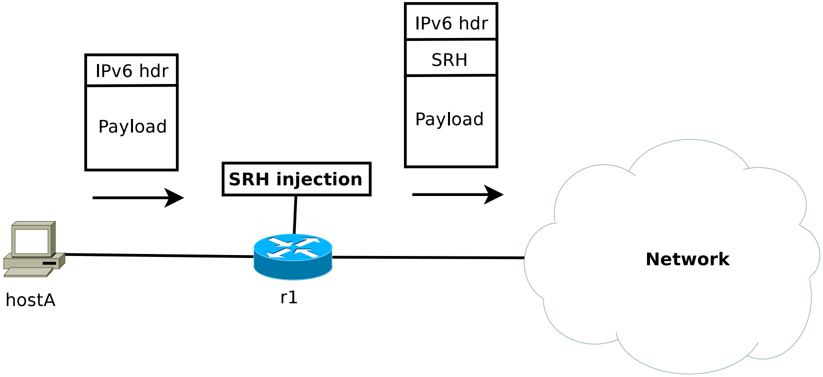
```
 1: if DA = myself (segment endpoint) then
 2:     if Segments Left > 0 then
 3:         Decrement Segments Left
 4:         Update DA with Segment List[Segments Left]
 5:         if Segments Left == 0 AND Clean-Up bit set then
 6:             Strip SRH
 7:         end if
 8:     else
 9:       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

SR-IPv6 implementation

- Linux kernel implementation, current branch: 3.14.x
- About 2,500 LoC as of latest commit
- Open-source
- Interfaces for SRH injection and control
- `http://github.com/segment-routing/`

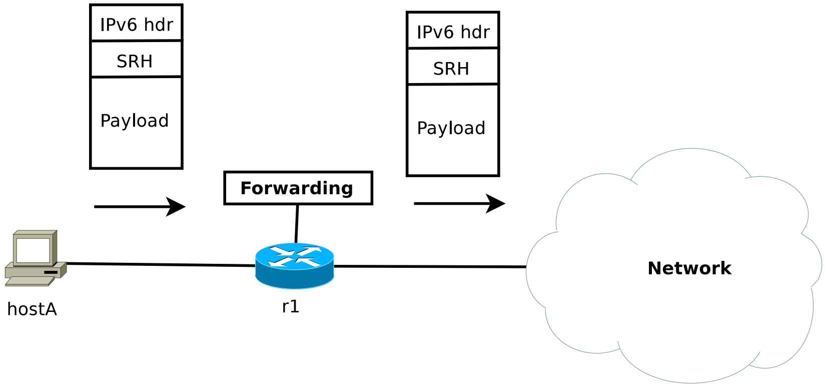
SRH injection: router-level

- Currently: per destination prefix



SRH injection: host-level (1)

- Per socket, through `setsockopt()`



SRH injection: host-level (2)

- SRH reversal
  - For TCP connections
  - Ensure outbound flow uses same path as inbound flow
  - Per-socket control

Interface

- Kernel exposes netlink interface
- Sysctl for global control of some variables
- Userland tool to control kernel structures (`seg6ctl`)

Configuration example: injection

Injection

```
# seg6ctl --prefix
2a03:2880:2130:cf05:face:b00c:0:1/128 --add
2a00:1450:4007:808::100e,2001:67c:2e8:22::c100:68b
```

Configuration example: injection

Injection

```
# seg6ctl --prefix
2a03:2880:2130:cf05:face:b00c:0:1/128 --add
2a00:1450:4007:808::100e,2001:67c:2e8:22::c100:68b
```

- "When a packet with DA = Facebook is forwarded, inject an SRH containing two segments: first Google, then RIPE."
- Segments list is comma-separated

Configuration example: injection

Injection

```
# seg6ctl --prefix
2a03:2880:2130:cf05:face:b00c:0:1/128 --add
2a00:1450:4007:808::100e,2001:67c:2e8:22::c100:68b
```

- "When a packet with DA = Facebook is forwarded, inject an SRH containing two segments: first Google, then RIPE."
- Segments list is comma-separated

With cleanup

```
# seg6ctl --prefix
2a03:2880:2130:cf05:face:b00c:0:1/128 --add
2a00:1450:4007:808::100e,2001:67c:2e8:22::c100:68b
--cleanup
```

Configuration example: injection

Injection

```
# seg6ctl --prefix
2a03:2880:2130:cf05:face:b00c:0:1/128 --add
2a00:1450:4007:808::100e,2001:67c:2e8:22::c100:68b
```

- "When a packet with DA = Facebook is forwarded, inject an SRH containing two segments: first Google, then RIPE."
- Segments list is comma-separated

With cleanup

```
# seg6ctl --prefix
2a03:2880:2130:cf05:face:b00c:0:1/128 --add
2a00:1450:4007:808::100e,2001:67c:2e8:22::c100:68b
--cleanup
```

- Same thing, but penultimate SR hop (i.e. RIPE, in this case) must remove SRH before forwarding to final destination (i.e. Facebook)

Configuration example: table dump

Show table

```
# seg6ctl --show
> 2a03:2880:2130:cf05:face:b00c:0:1/128 via 2 segs
[2a00:1450:4007:808::100e 2001:67c:2e8:22::c100:68b]
id 0 hmac 0x0
> fc00:42::/64 via 2 segs [fc00:1::2 fc00:1::7] id 0
hmac 0x0 cleanup
> 2001:db8::/32 via 1 segs [2a01::12] id 0 hmac 0x0
```

Configuration example: misc

Delete

```
# seg6ctl --prefix
2a03:2880:2130:cf05:face:b00c:0:1/128 --delete
```

Flush

```
# seg6ctl --flush
```

Code example: per-socket injection (1)

```
struct ipv6_sr_hdr *hdr;
int sock, tot_len;
struct sockaddr_in6 sin6;

sock = socket(AF_INET6, SOCK_STREAM, 0);
sin6.sin6_family = AF_INET6;
sin6.sin6_port = htons(80);
inet_pton(AF_INET6, "2a03:2880:2130:cf05:face:b00c:0:1",
          &sin6.sin6_addr.s6_addr);
```

Code example: per-socket injection (2)

```
tot_len = sizeof(*hdr) + 2*sizeof(struct in6_addr);
hdr = malloc(tot_len);

hdr->hdrlen = 0; /* computed by the kernel */
hdr->type = 4;
hdr->first_segment = 1; /* offset */
sr_set_flags(hdr, SR6_FLAG_CLEANUP);
```

Code example: per-socket injection (3)

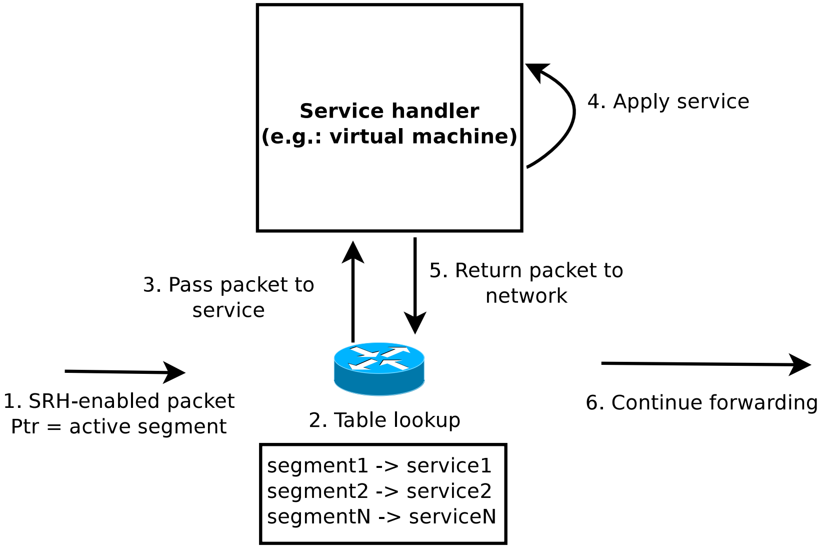
```
inet_pton(AF_INET6, "2a00:1450:4007:808::100e",
          hdr->segments);
inet_pton(AF_INET6, "2001:67c:2e8:22::c100:68b",
          hdr->segments + 1);

setsockopt(sock, IPPROTO_IPV6, IPV6_RTHDR, hdr, tot_len);

connect(...);
```

Services with Segment Routing

- On SRH processing: segment represents next hop
- It can also represent service to apply

Services with Segment Routing

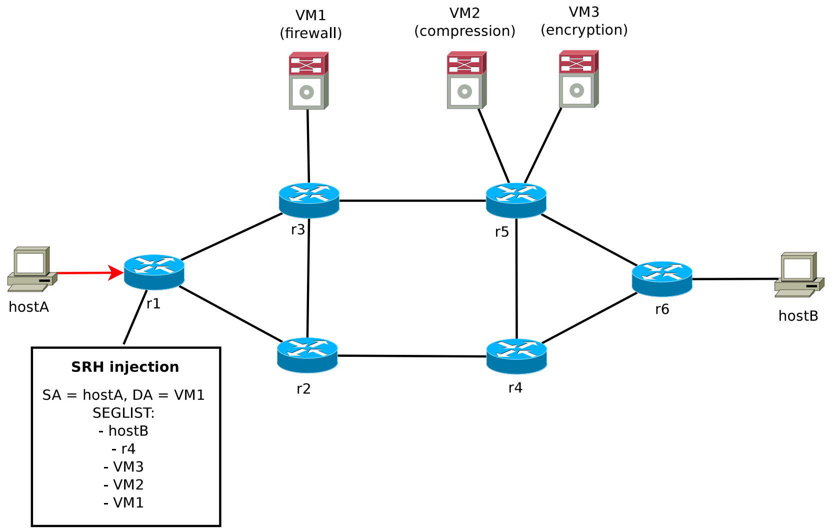
- On SRH processing: segment represents next hop
- It can also represent service to apply



Services with Segment Routing

- Multiple services can be designed:
  - Encryption
  - Compression
  - Firewalling
  - Netflow
  - DPI
  - NAT
  - etc...
- What if we need to firewall, then compress, then encrypt ?

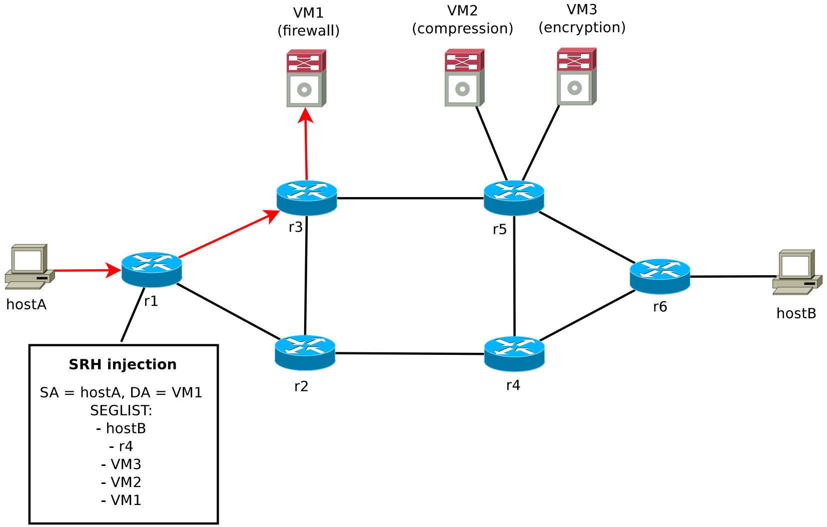
Service Function Chaining



SRH injection

SA = hostA, DA = VM1
SEGLIST:
- hostB
- r4
- VM3
- VM2
- VM1

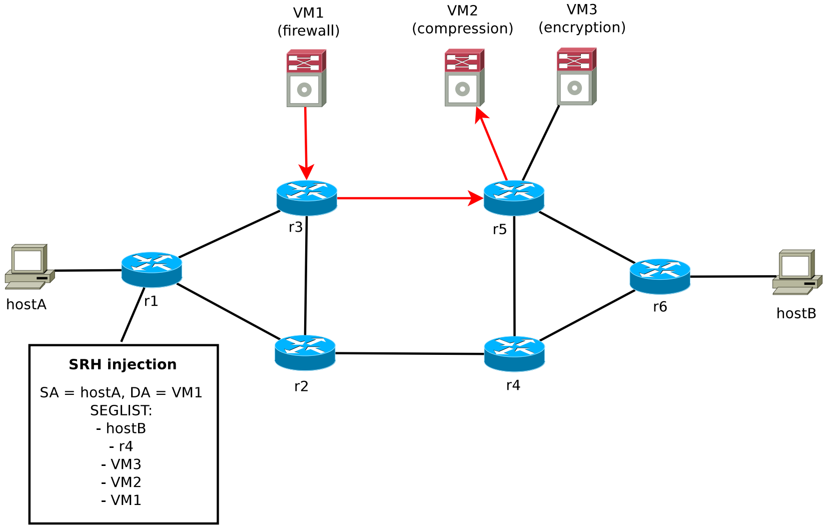
Service Function Chaining



SRH injection

SA = hostA, DA = VM1
SEGLIST:
- hostB
- r4
- VM3
- VM2
- VM1

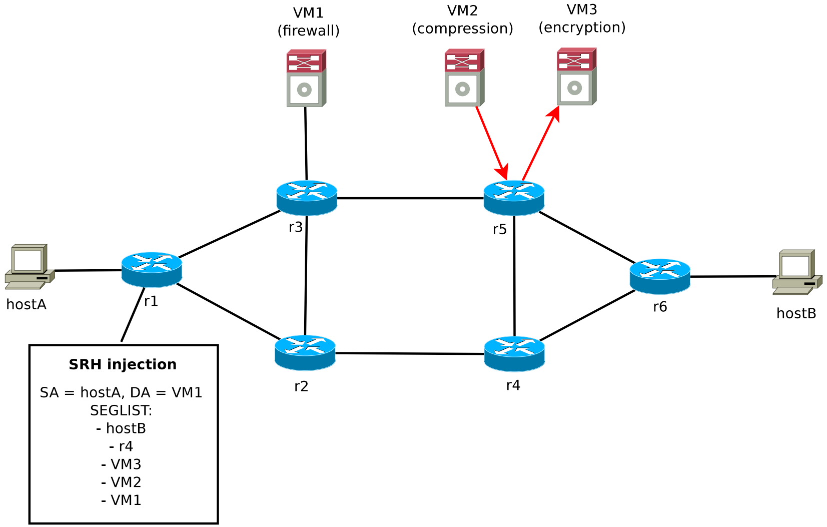
Service Function Chaining



SRH injection

SA = hostA, DA = VM1
SEGLIST:
- hostB
- r4
- VM3
- VM2
- VM1

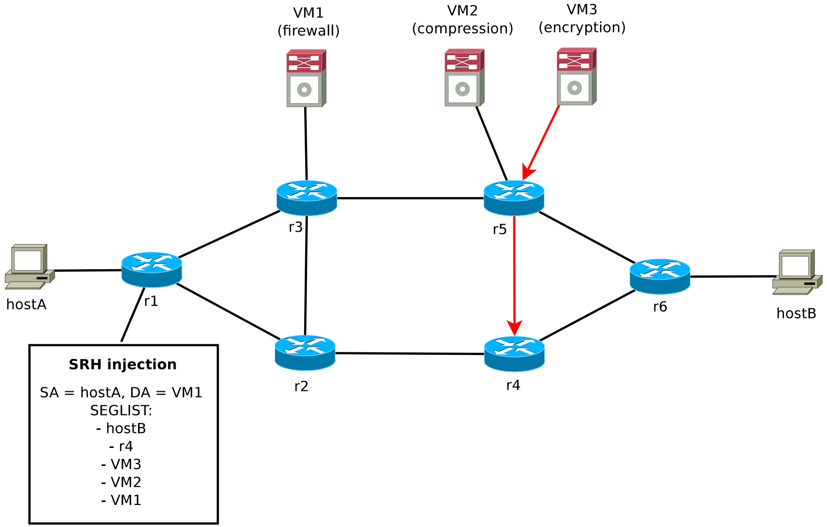
Service Function Chaining



SRH injection

SA = hostA, DA = VM1
SEGLIST:
- hostB
- r4
- VM3
- VM2
- VM1

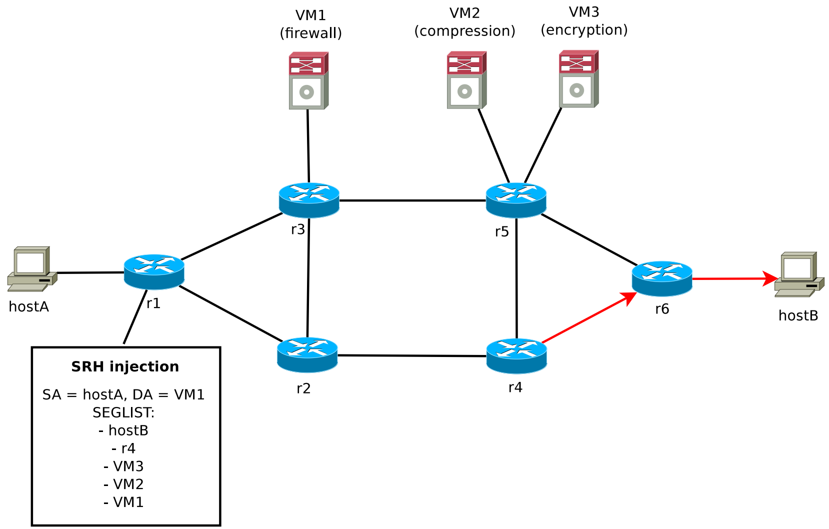
Service Function Chaining



SRH injection

SA = hostA, DA = VM1
SEGLIST:
- hostB
- r4
- VM3
- VM2
- VM1

Service Function Chaining



SRH injection

SA = hostA, DA = VM1
SEGLIST:
- hostB
- r4
- VM3
- VM2
- VM1

IETF drafts

- draft-ietf-spring-segment-routing-01
- draft-previdi-6man-segment-routing-header-06
- draft-vyncke-6man-segment-routing-security-02

Pointers

- UCL (SR-IPv6) website: http://www.segment-routing.org
- Cisco website: http://www.segment-routing.net
- Implementation: http://github.com/segment-routing/
- Technical report on SR-IPv6 implem (being updated): http://www.segment-routing.org/sr6-doc.pdf
- **Virtual Machine** to play around with SR-IPv6: http://www.segment-routing.org/sr6-vm.vdi.bz2

That's all folks !