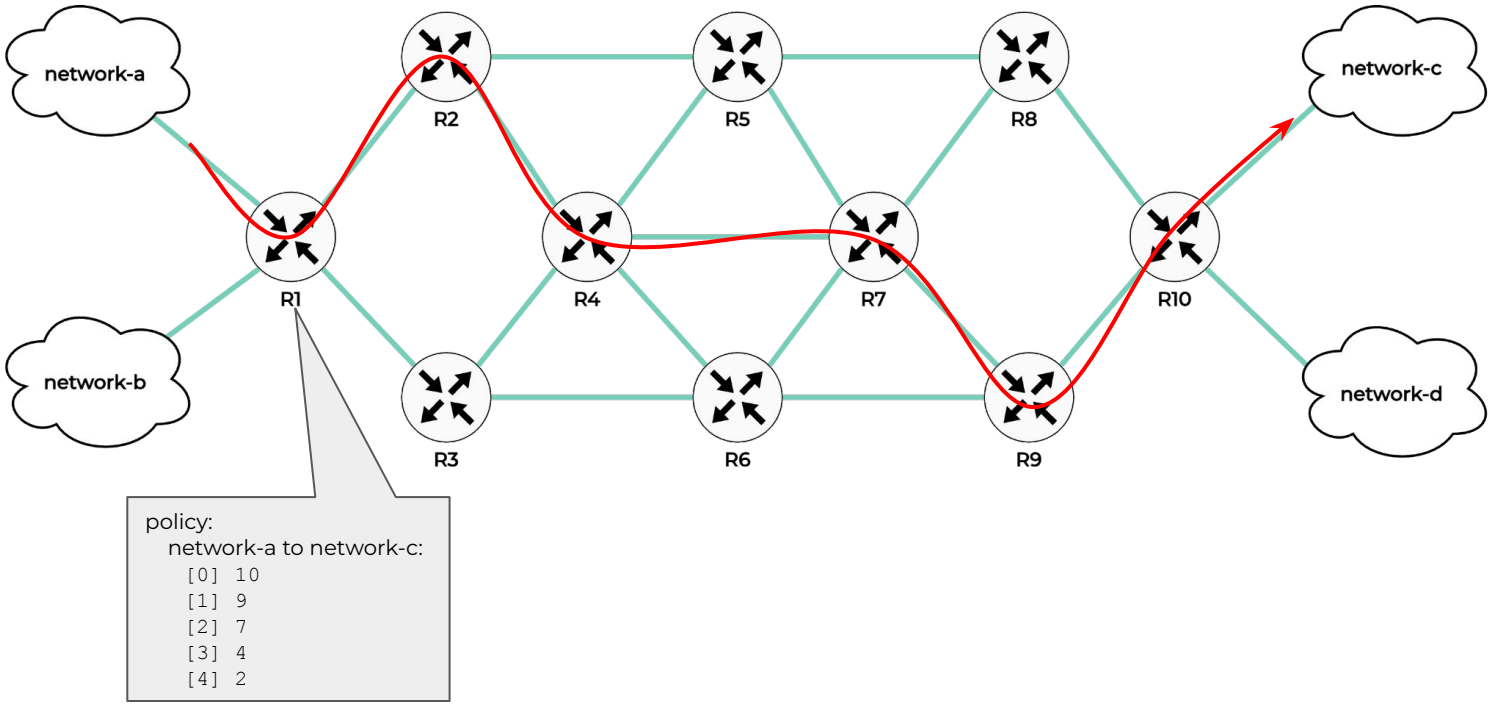
Leveraging eBPF for Intent-Driven Application-Centric End-to-End Segment Routing over IPv6

Master Thesis, Autumn 23

Content

- Simplified Introduction into SRv6
- Project Journey
- Implementation
- Traffic Flow Example
- Demo
- Recap and Outlook

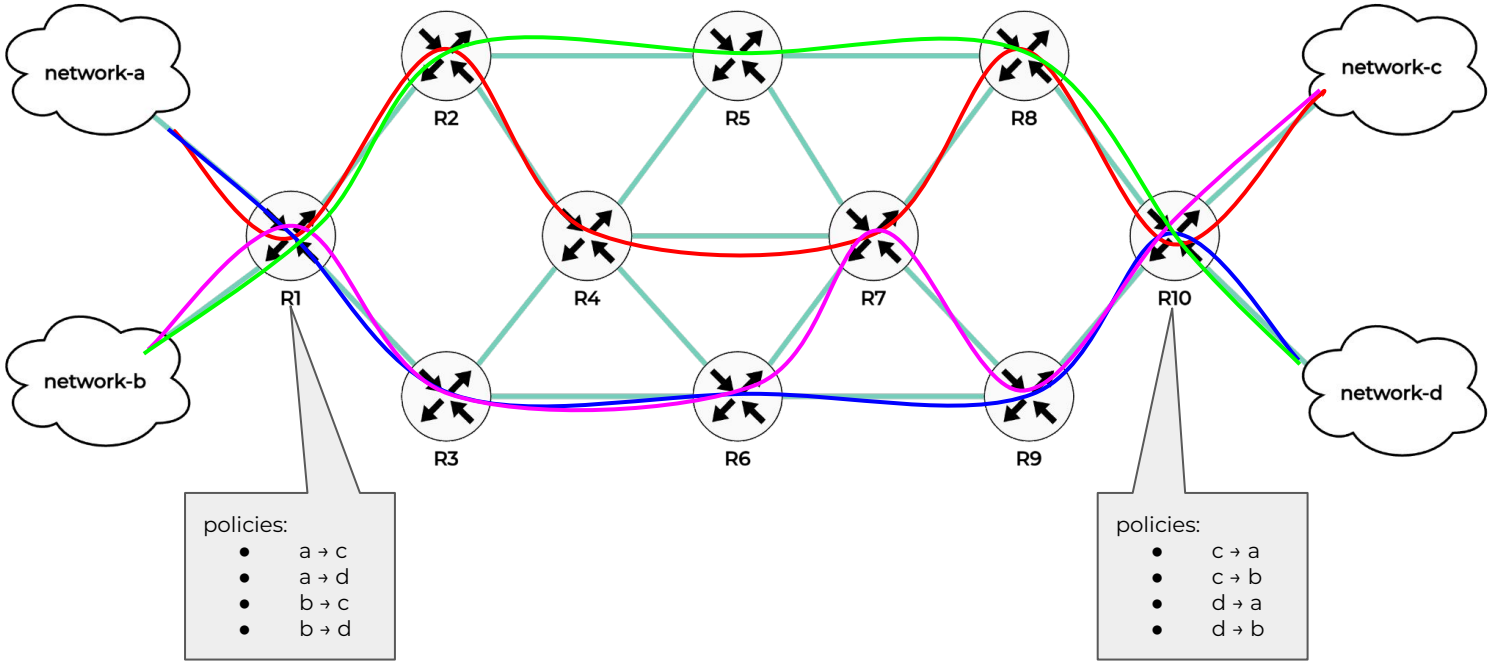
SRv6 a simplified Introduction



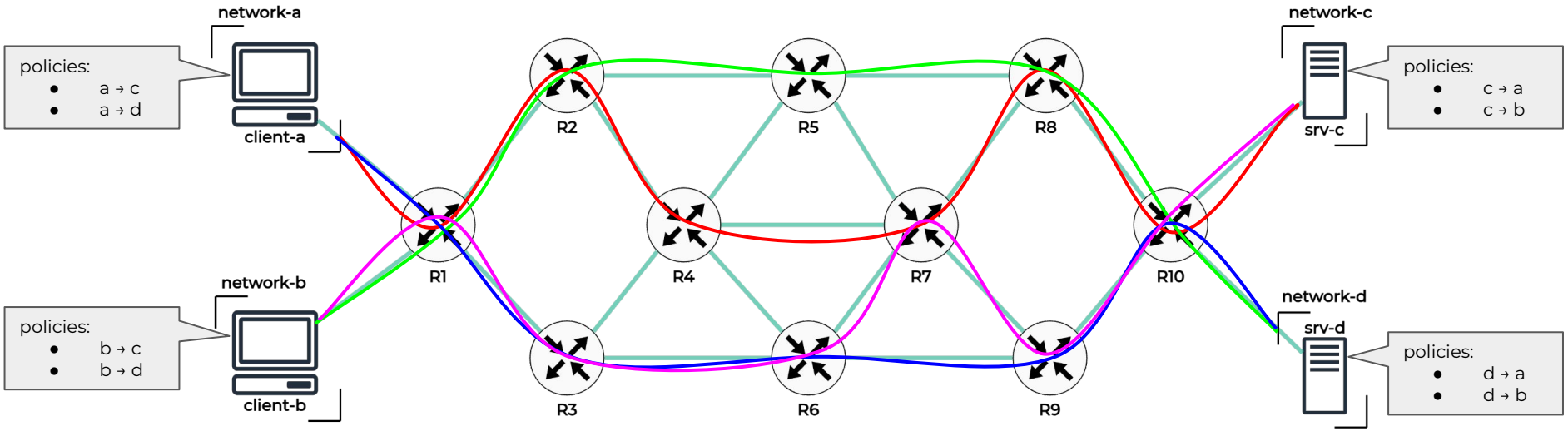
journey

to an intent-driven application-centric application

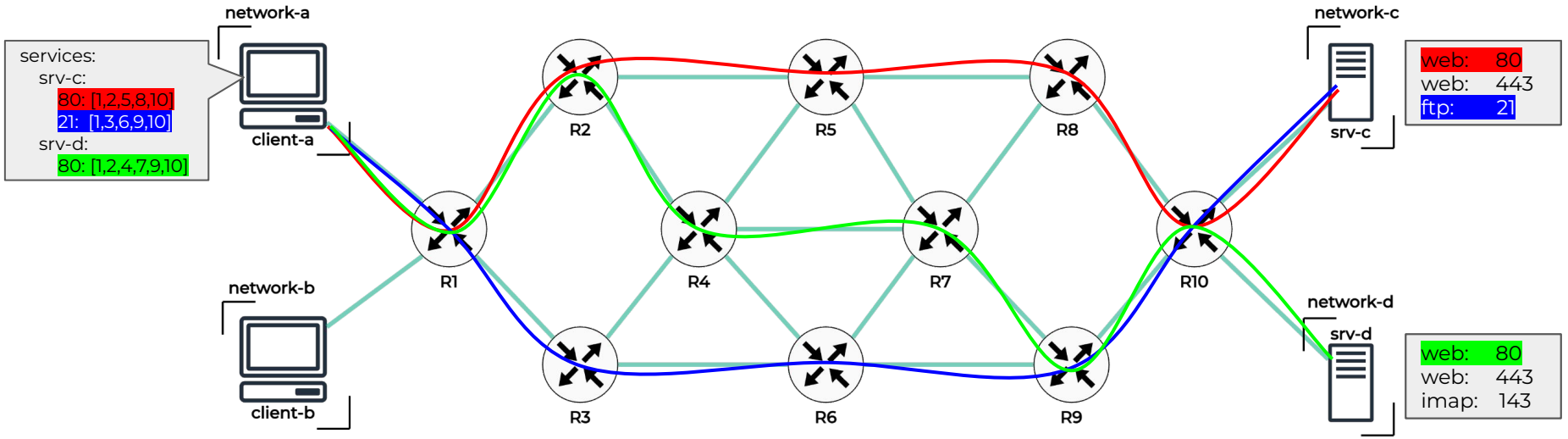
Network-Level Traffic Engineering



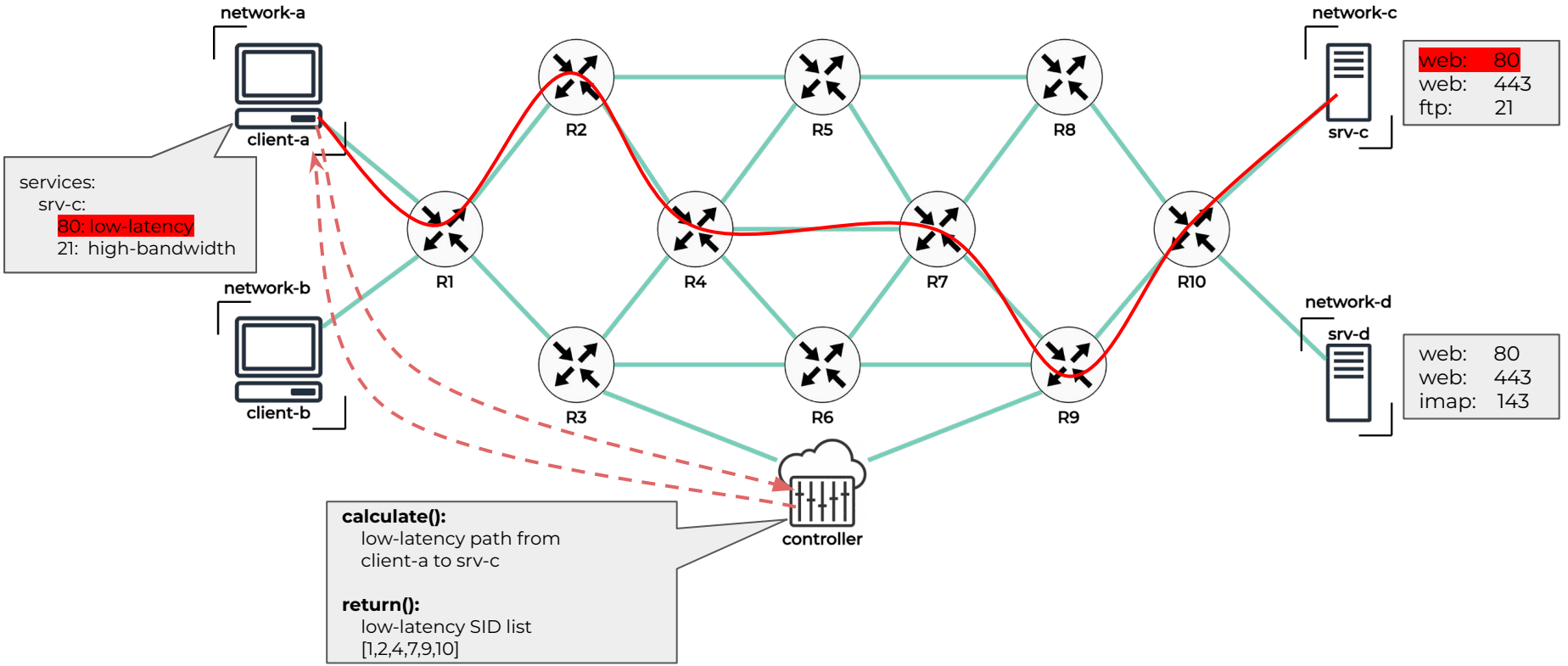
End-to-End Traffic Engineering



Application-Centric Traffic Engineering



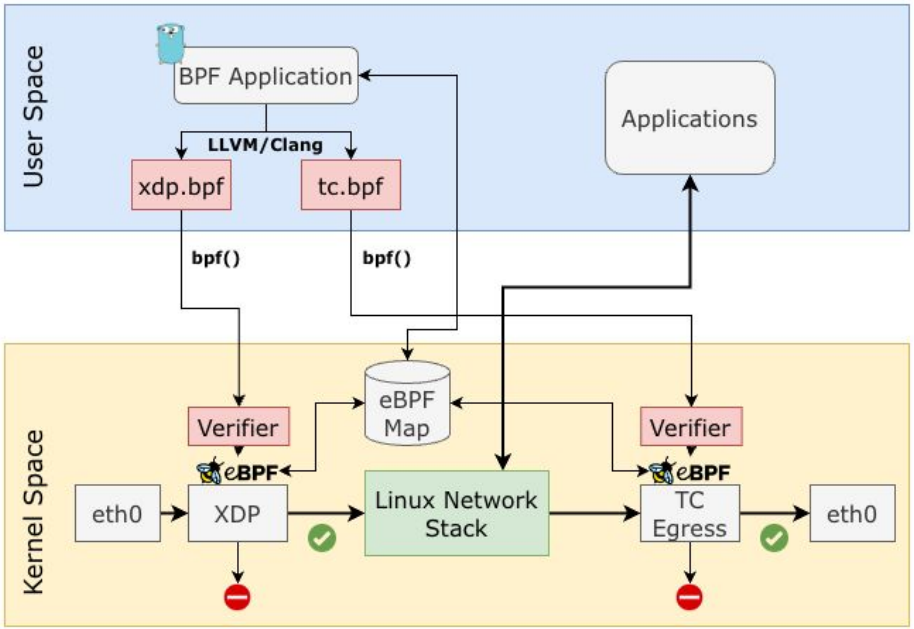
Intent-Driven Traffic Engineering



implementation

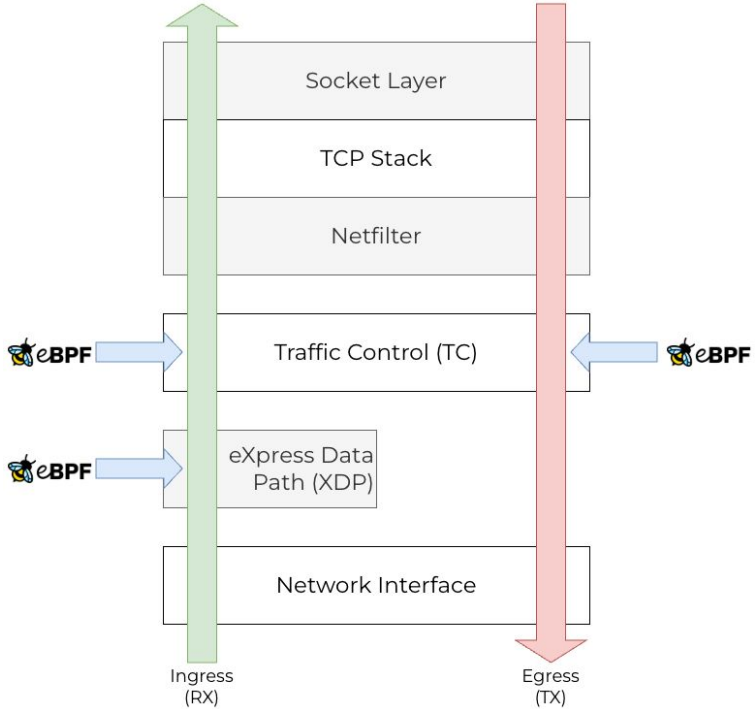
of an intent-driven application-centric application

eBPF in a Nutshell

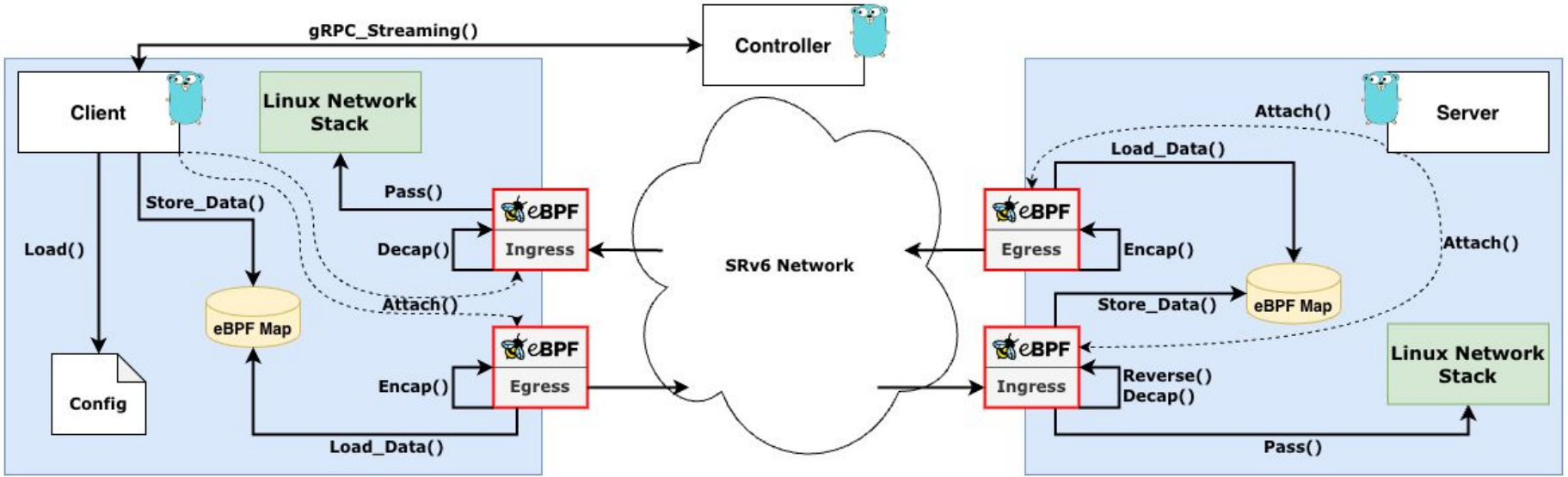


Linux Kernel Hooks - Network Subsystem

- Ingress (RX)
  - XDP
  - TC Ingress
- Egress (TX)
  - TC Egress



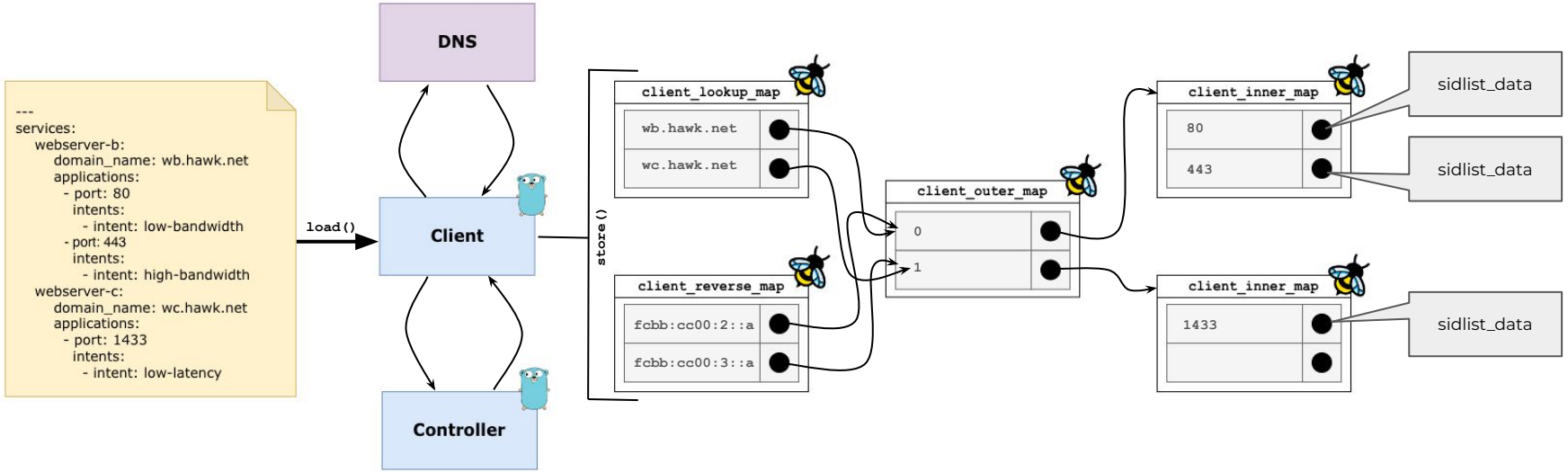
Application Architecture



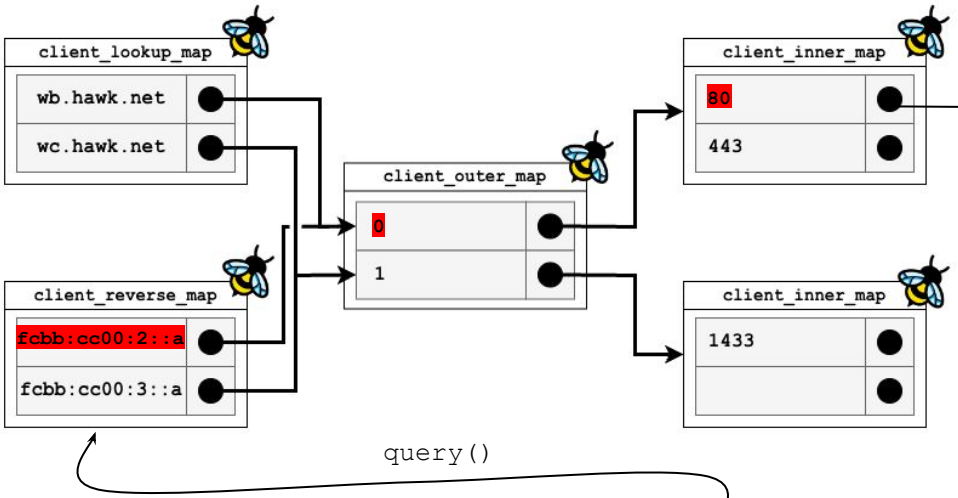
example

of a simplified traffic flow

Startup Procedure



Client Traffic Matching and Encapsulation



Outgoing packet:

IPv6: {dest: **fcbb:cc00:2::a**}
TCP: {dest_port: **80**}

encap()

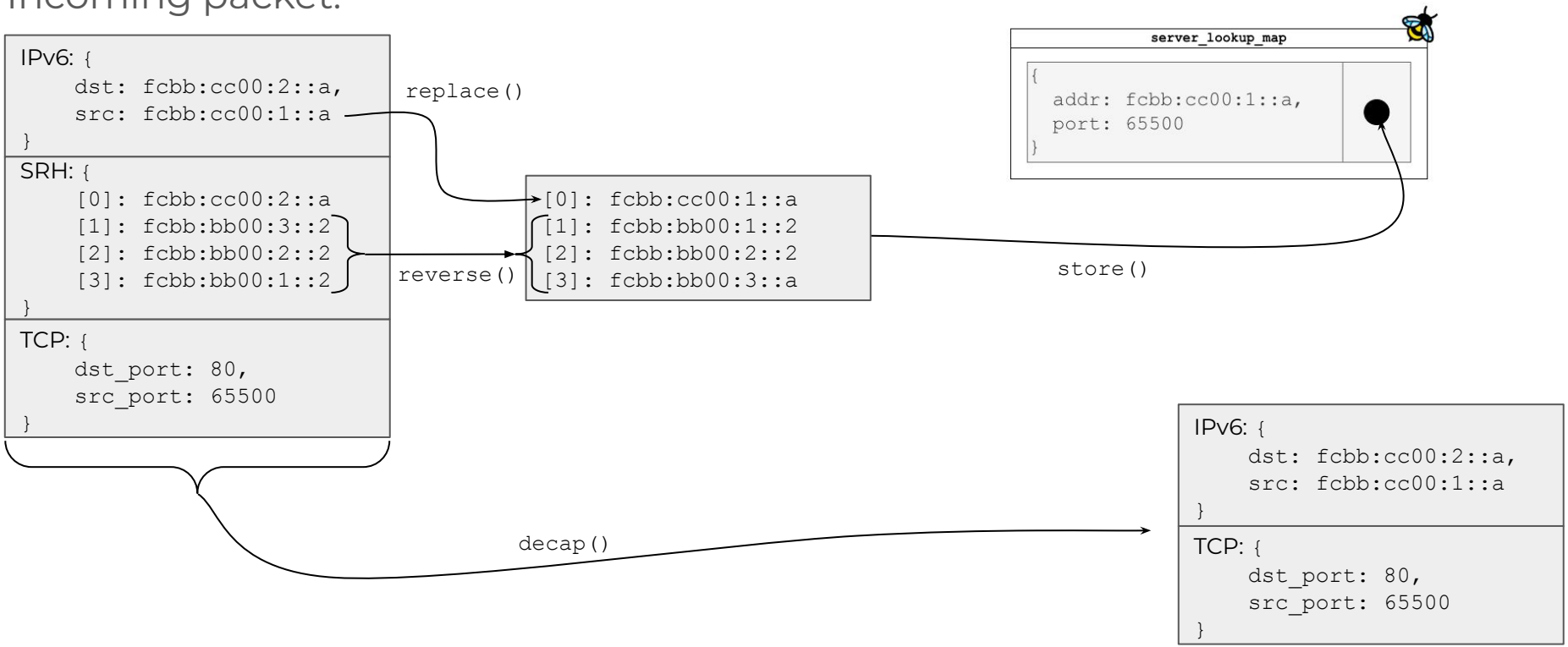
IPv6: {dest: fcbb:bb00:1::2}

SRH: {
[0]: fcbb:cc00:2::a
[1]: fcbb:bb00:3::2
[2]: fcbb:bb00:2::2
[3]: fcbb:bb00:1::2
}

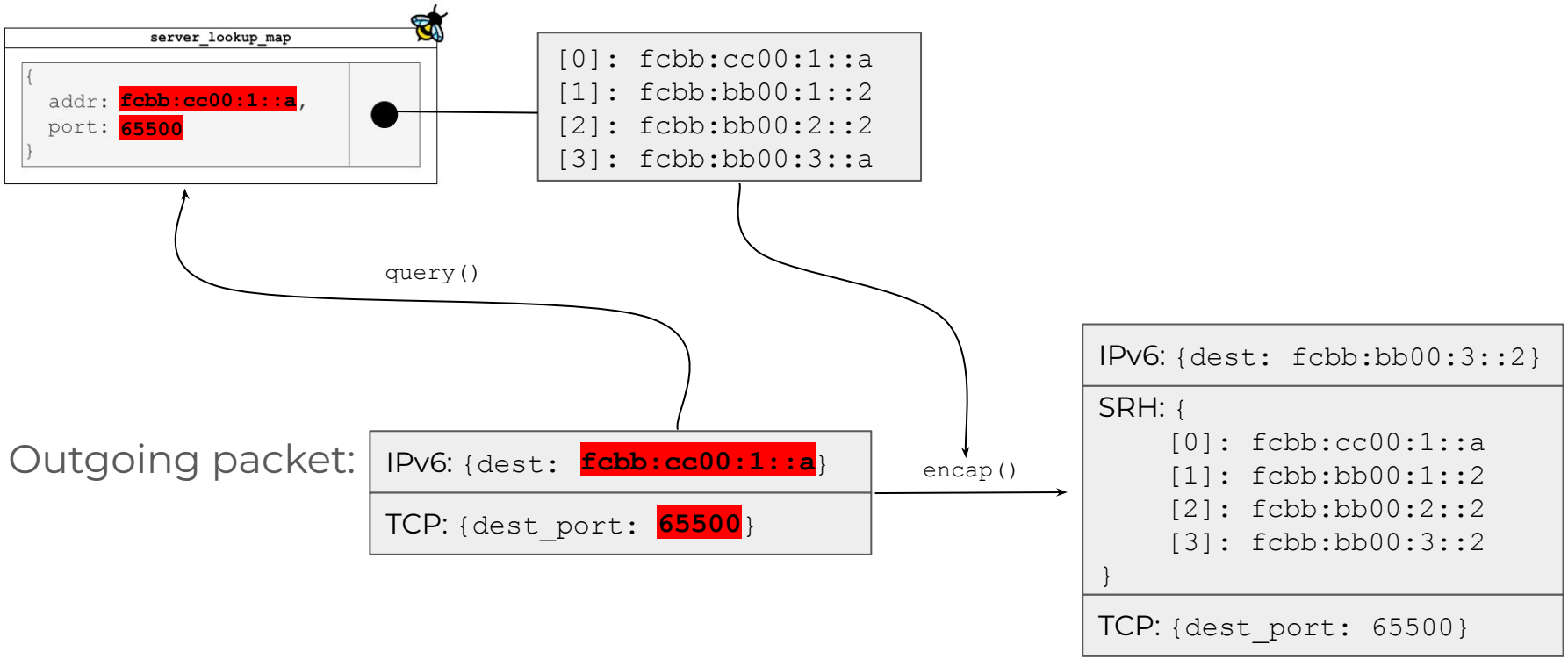
TCP: {dest_port: 80}

Server Data Processing and Decapsulation

Incoming packet:

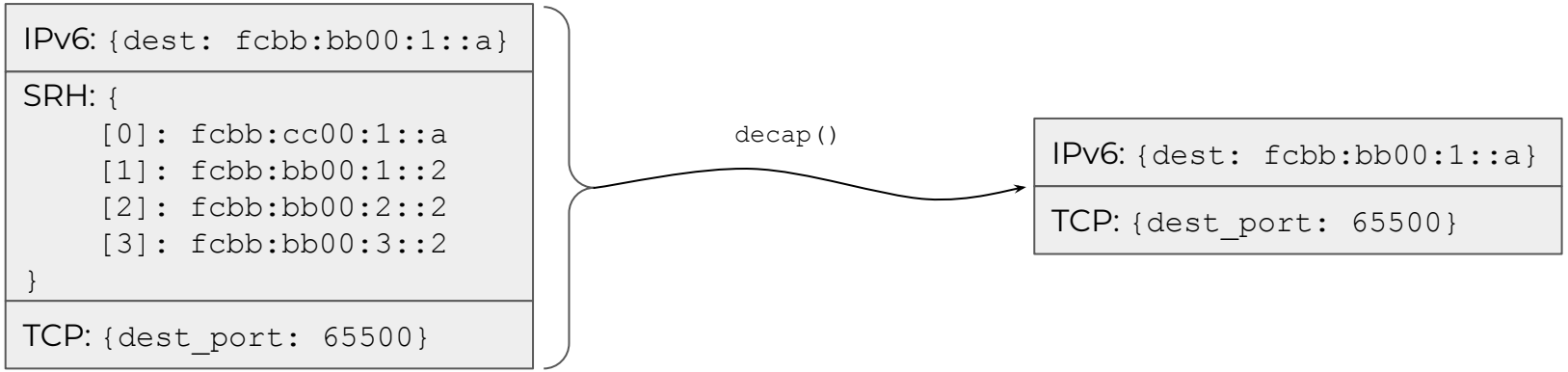


Server Traffic Matching and Encapsulation



Client Decapsulation

Incoming packet:



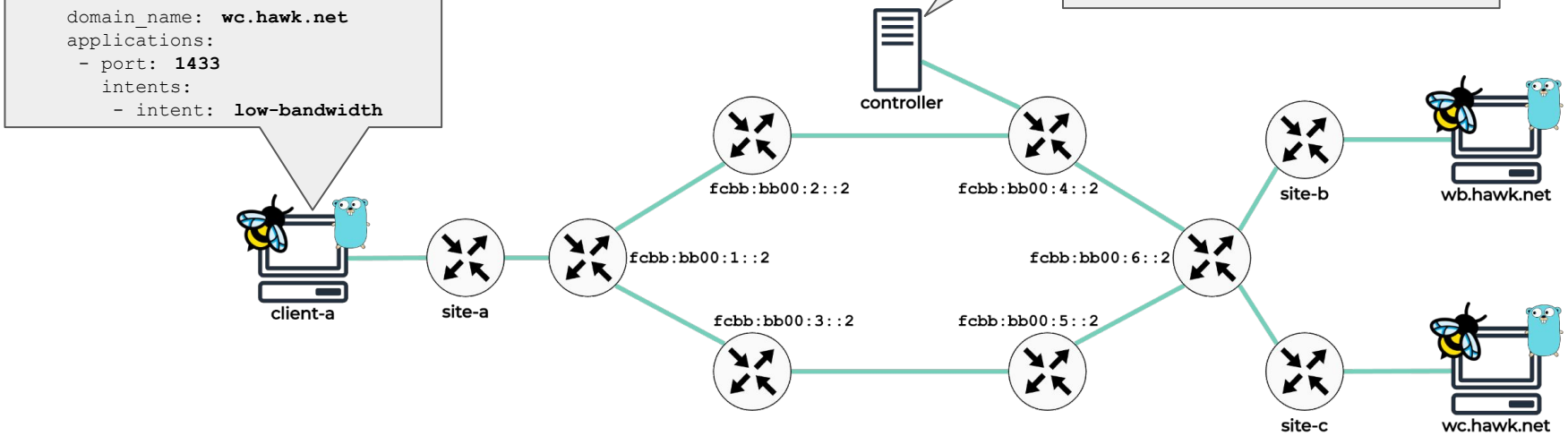
demo

of the finished application

Demo Infrastructure

```
services:
  webserver-b:
    domain_name: wb.hawk.net
    applications:
     - port: 80
       sid:
        - fcbb:bb00:1::2
        - fcbb:bb00:3::2
     - port: 443
       intents:
        - intent: low-latency
  webserver-c:
    domain_name: wc.hawk.net
    applications:
     - port: 1433
       intents:
        - intent: low-bandwidth
```

```
to:
  - webserver-b
    intents:
     - intent: low-latency
       sid_list:
        - fcbb:bb00:1::2
        - fcbb:bb00:2::2
        - fcbb:bb00:4::2
        - fcbb:bb00:6::2
  - webserver-c
    intents:
     - intent: low-bandwidth
       sid_list:
        - fcbb:bb00:1::2
        - fcbb:bb00:4::2
        - fcbb:bb00:5::2
        - fcbb:bb00:1::2
        - fcbb:bb00:4::2
```

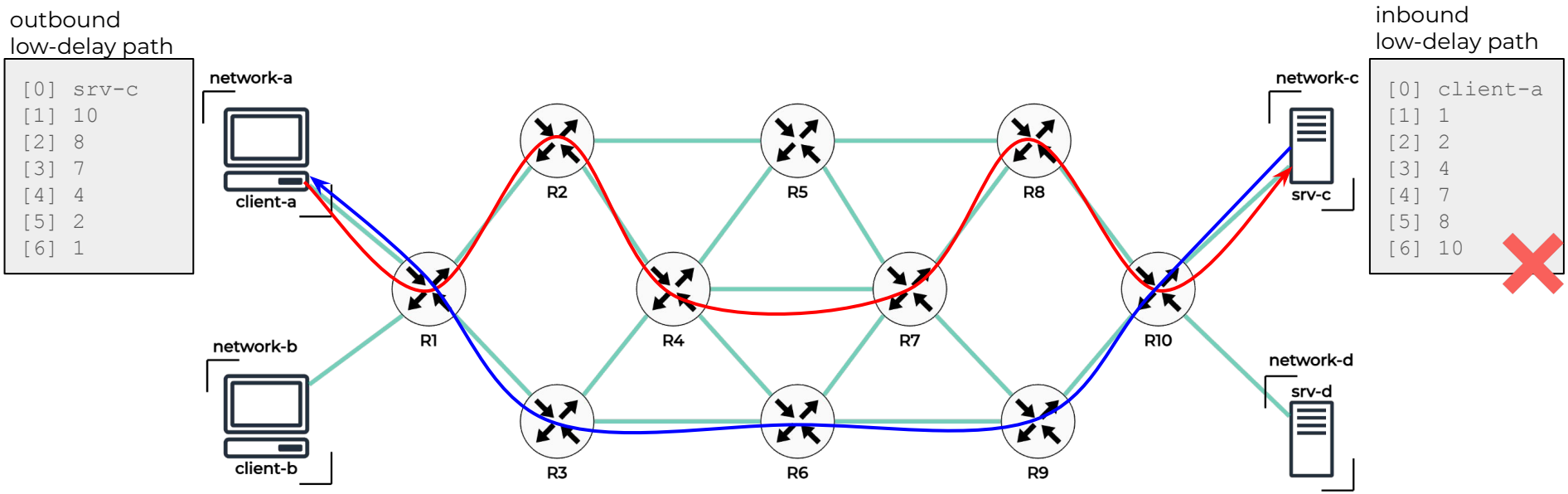


recap and outlook

improvements and future research directions

Asynchronous Paths

Problem: outbound low-delay path ≠ inbound low-delay path

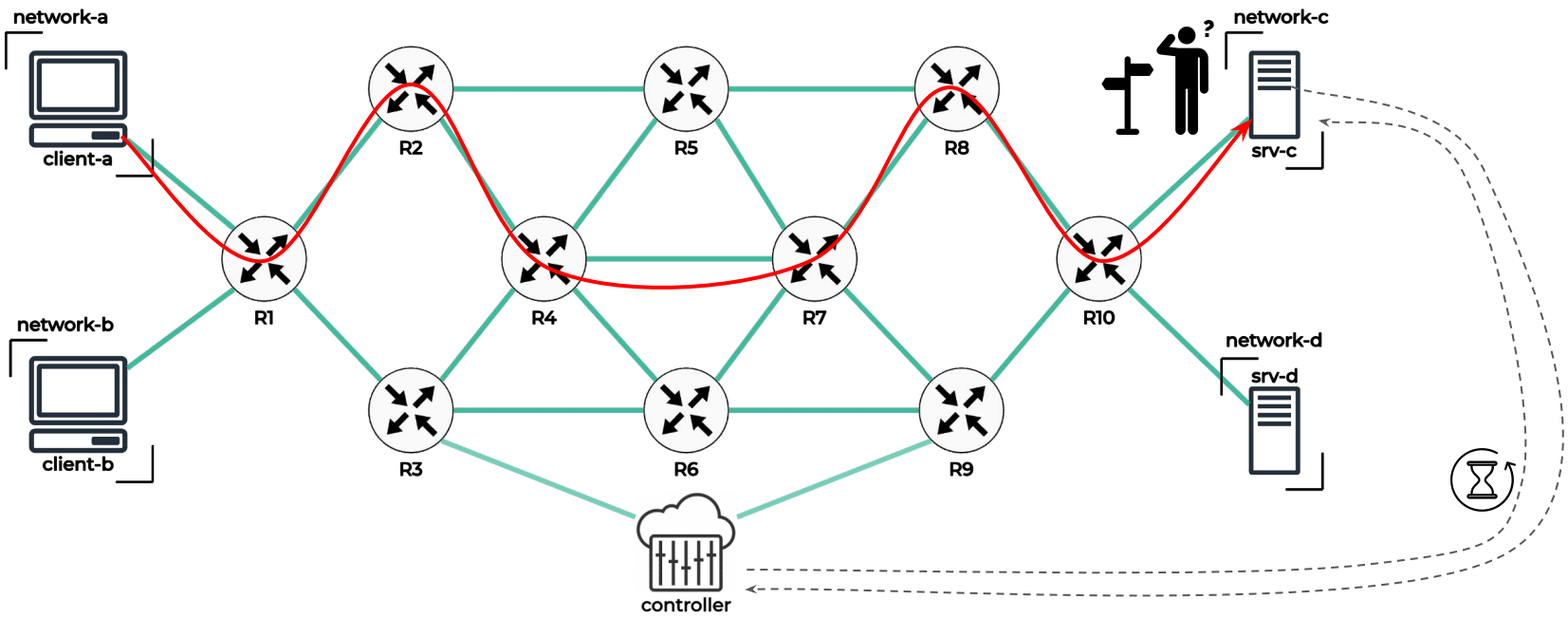


Asynchronous Paths

Possible solution: Request controller for inbound low-delay path
Problem: How to encapsulate packets when waiting on controller reply?

outbound
low-delay path

```
[0] srv-c
[1] 10
[2] 8
[3] 7
[4] 4
[5] 2
[6] 1
```

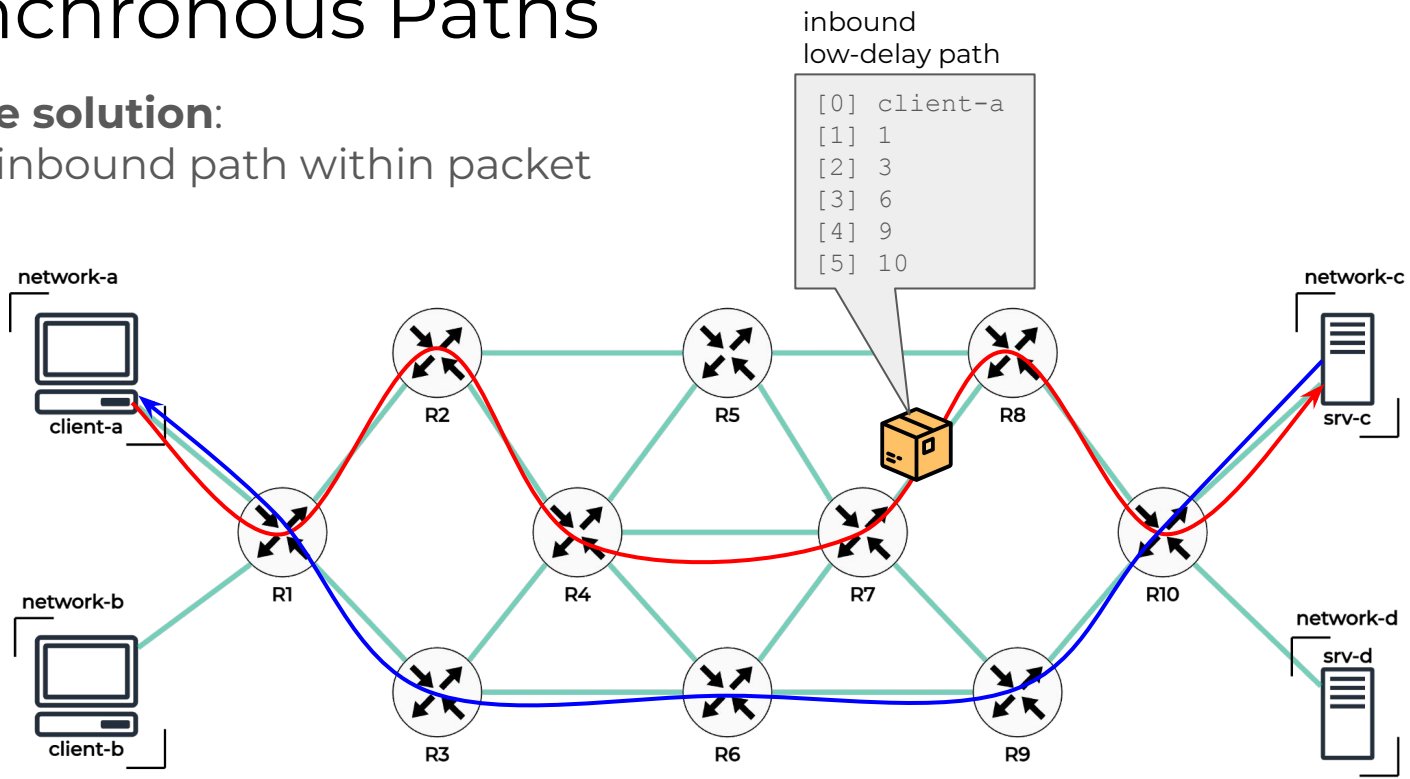


Asynchronous Paths

Possible solution:
Deliver inbound path within packet

outbound
low-delay path

```
[0] srv-c
[1] 10
[2] 8
[3] 7
[4] 4
[5] 2
[6] 1
```



Implementing C-SID

Traditional SID List:

16 * 3 = 48 bytes

```
[0]: fcbb:bb00:1::1
[1]: fcbb:bb00:2::1
[2]: fcbb:bb00:3::1
```

needs SRH:
IPv6 + SRH + SID list
40 + 8 + 96 = **96 bytes**

Compressed C-SID:

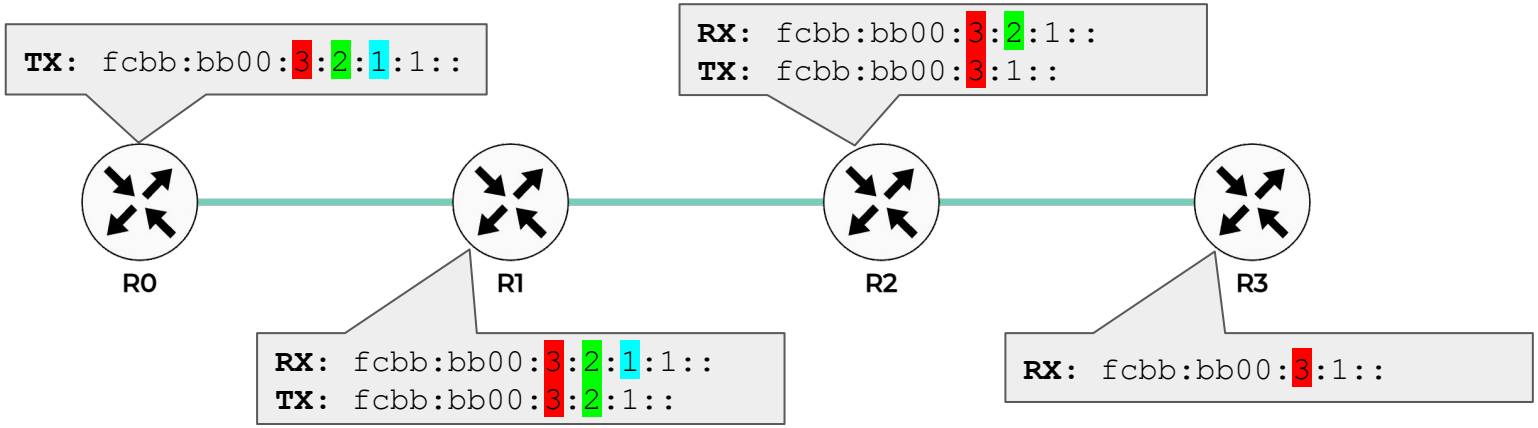
16 bytes

```
fcbb:bb00:3:2:1:1::
```

no need for an SRH:
IPv6
40 = **40 bytes**

Implementing C-SID

Problem: Bits are shifted away when segment is processed
→ Lost information for the inbound path

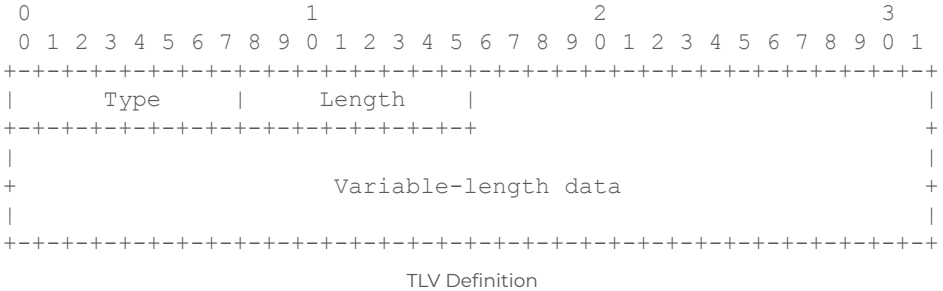


Solution: Deliver inbound path within packet

Delivering Metadata

SRH Type Length Value (TLV)

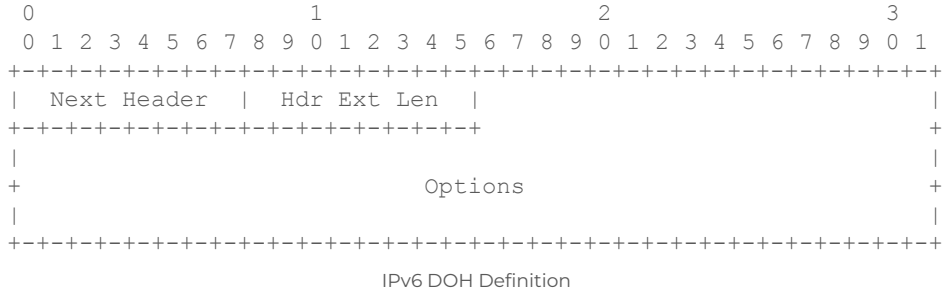
- Carry additional metadata within SRH
- Metadata for segment processing
- Every node on path must handle TLVs correctly



TLV Definition

IPv6 Destination Option Header (DOH)

- Carry additional metadata as IPv6 extension header
- Only examined at the packet's destination



IPv6 DOH Definition

questions?