Reaping the Benefits of IPv6 Segment Routing

Public PhD thesis defense

David Lebrun

Université catholique de Louvain

October 19, 2017

Table of Contents

Introduction

Segment Routing

Implementing IPv6 Segment Routing in Linux

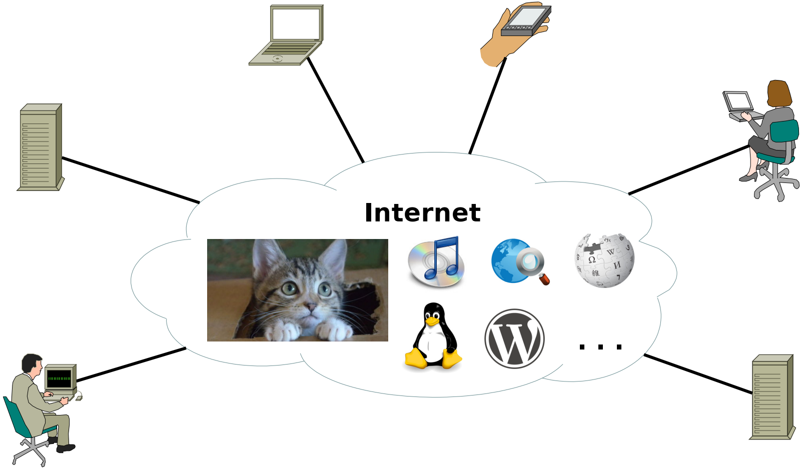
Exploring IPv6 Segment Routing

Rethinking IPv6 Enterprise Networks

Conclusion

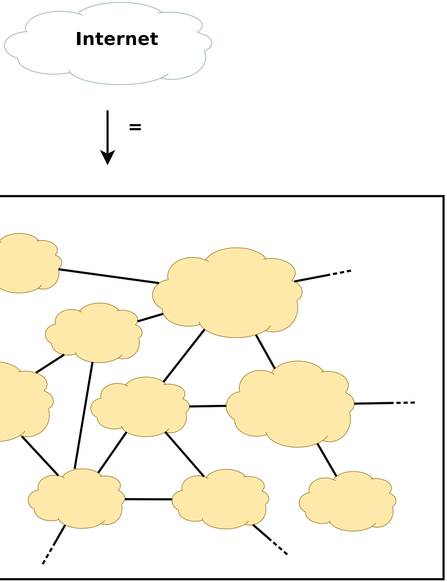
Introduction

- Networks connect devices and transport information



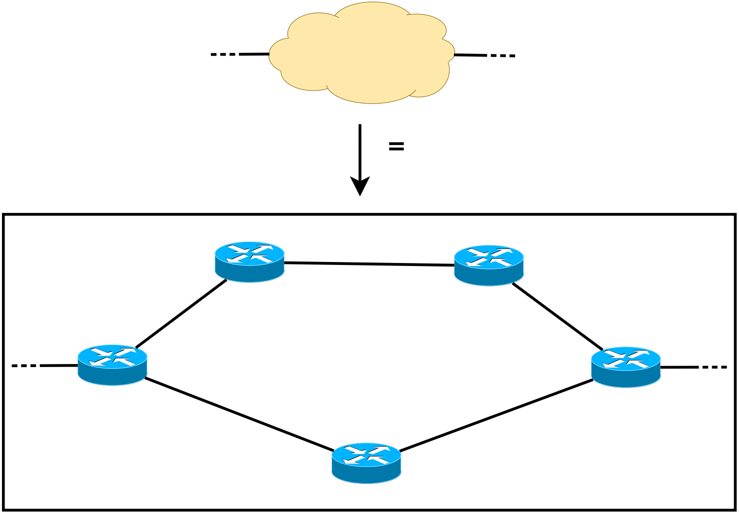
Introduction

- Networks are interconnected



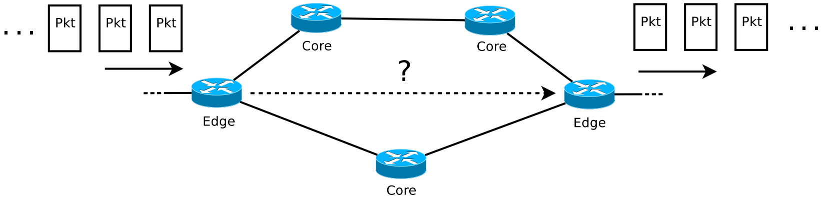
Introduction

- Basic building blocks of networks are routers



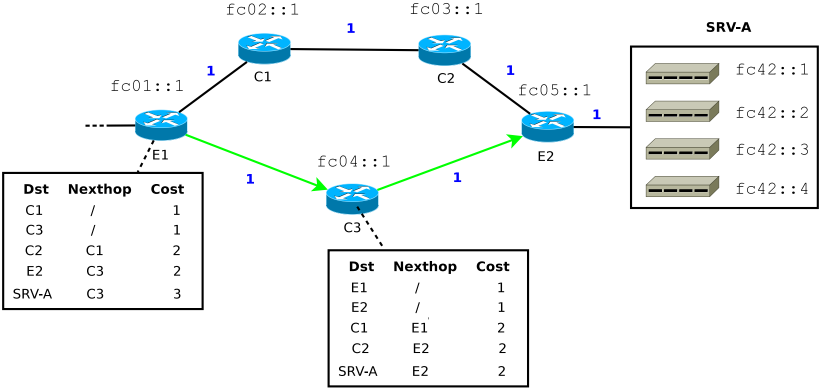
Packet-based forwarding

- Information is chunked into *packets*
- How are packets exchanged ?
- The faster the better: *shortest-path forwarding*
- What is the shortest path ?



Routing tables

- Routing table: instructions on how to forward packets
- Each router computes its routing table

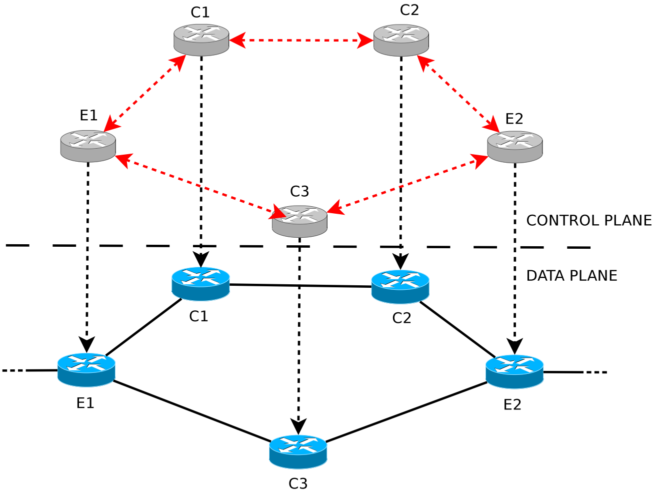


Dst	Nexthop	Cost
C1	/	1
C3	/	1
C2	C1	2
E2	C3	2
SRV-A	C3	3

Dst	Nexthop	Cost
E1	/	1
E2	/	1
C1	E1	2
C2	E2	2
SRV-A	E2	2

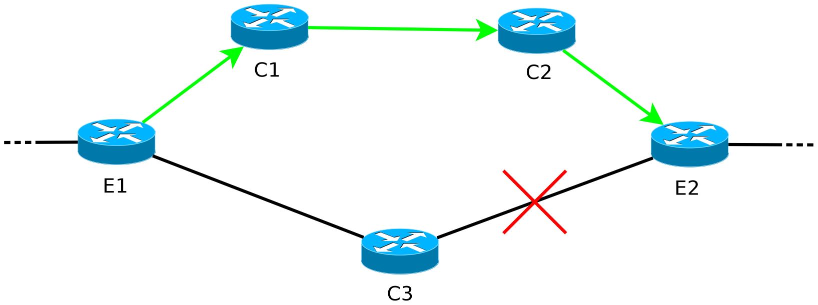
Control plane

- Routers exchange view of network
- *Interior Gateway Protocols* (IGP)
- Convergence to coherent global network state



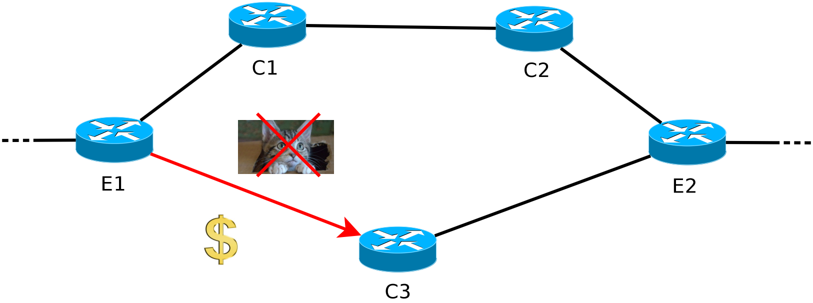
Fault tolerance

- IGP recomputation triggered on link/node failure
- Network state converges, best paths change



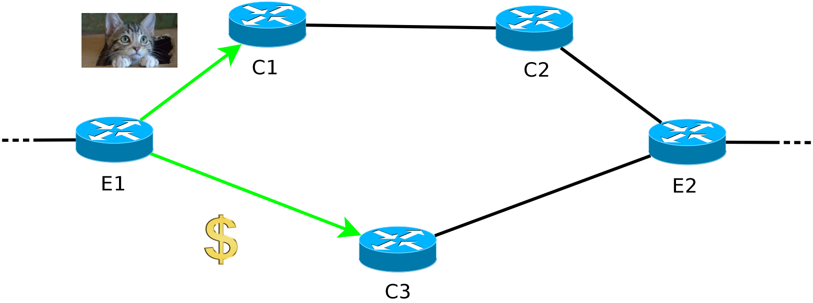
Quality of Service

- Not all traffic is equal
- Prioritization of some classes of traffic (QoS)
- Congestion may occur ⇒ drop of low-prio traffic



Traffic engineering

- QoS only ⇒ inefficient resource utilization
- Traffic steering: make a detour



Traffic engineering

- Difficult to achieve TE solely with IGPs

- Traffic will follow shortest path

- Existing solutions not scalable (MPLS/RSVP-TE)

Table of Contents

Introduction

Segment Routing

Implementing IPv6 Segment Routing in Linux

Exploring IPv6 Segment Routing

Rethinking IPv6 Enterprise Networks

Conclusion

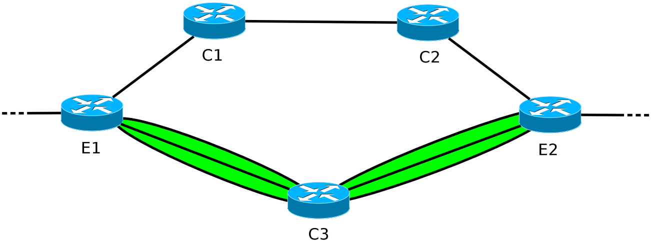
Segment Routing

- **Source routing paradigm**
- Path defined at source as list of *segments*
- List of segments embedded in each packet
- Segment ⇒ instruction (steering through node, link, ...)
- IPv6 Segment Routing (SRv6)[1] ⇒ segment = IPv6 address
- Runs on top of existing IGP: **sequence of shortest paths**

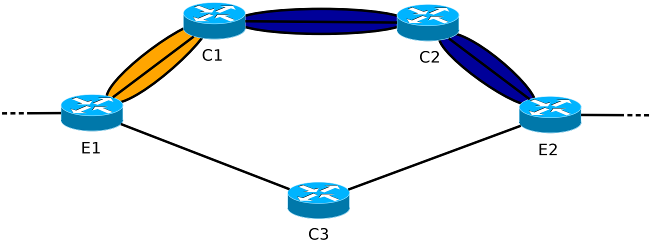
[1] Stefano Previdi, Clarence Filsfils, David Lebrun, et al. *IPv6 Segment Routing Header (SRH)*. . Internet-Draft draft-ietf-6man-segment-routing-header-07. Work in Progress. Internet Engineering Task Force, July 2017. 34 pp.

Segment Routing

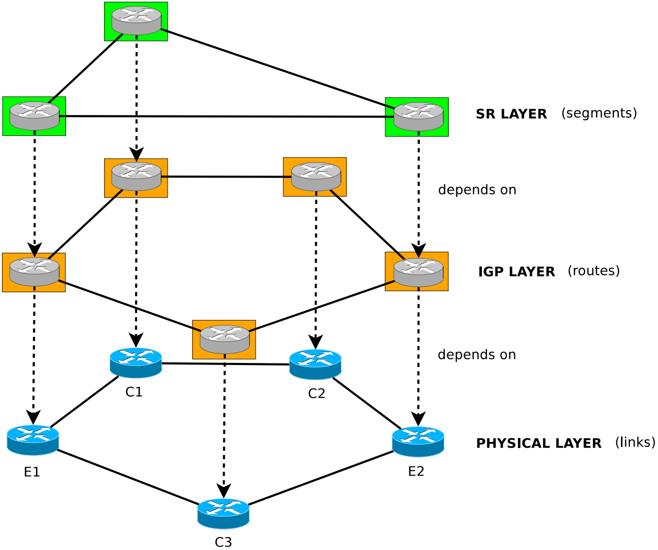
- From E1 to E2, segments: E2



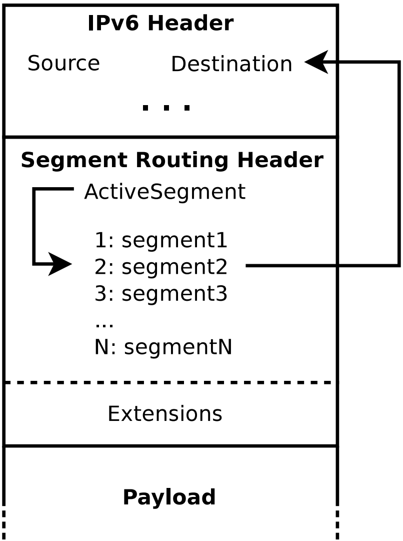
- From E1 to E2, segments: C1, E2



Segment Routing layers



Segment Routing Header



SRv6 operations: encapsulation and insertion

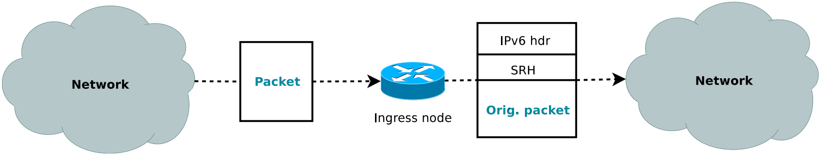


Figure: SRH encapsulation by ingress node.

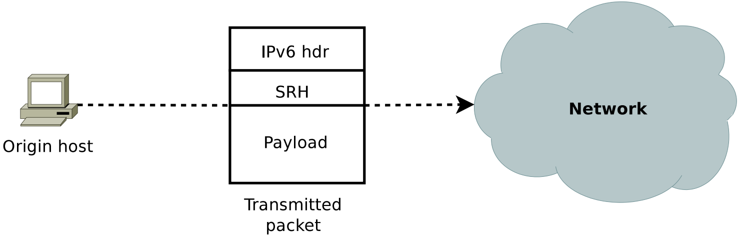


Figure: SRH insertion by source.

SRv6 operations: processing and decapsulation

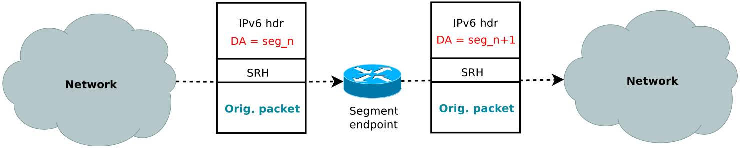


Figure: SRH processing by segment endpoint.

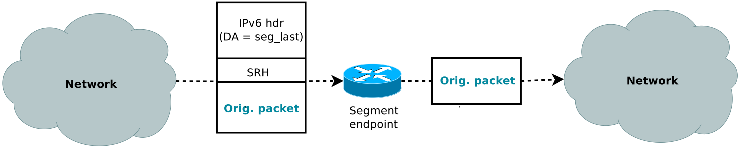


Figure: SRH decapsulation by egress node.

Table of Contents

Introduction

Segment Routing

Implementing IPv6 Segment Routing in Linux

Exploring IPv6 Segment Routing

Rethinking IPv6 Enterprise Networks

Conclusion

Benefits of a Linux implementation

- Mainline integration: widespread availability[2]

- Feedback loop for a developing technology

- Research opportunities for the scientific community

[2] David Lebrun and Olivier Bonaventure. "Implementing IPv6 Segment Routing in the Linux Kernel". In: *Proceedings of the 2017 Applied Networking Research Workshop*. ACM. 2017.

Routing engine

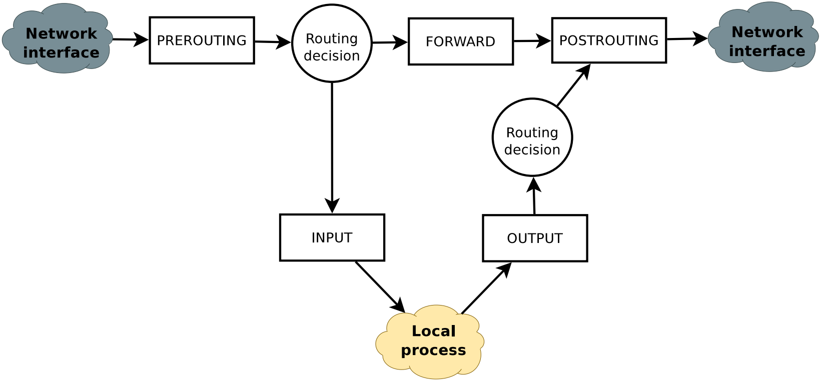


Figure: High-level overview of Linux routing process.

SRH insertion/encapsulation (forwarded packet)

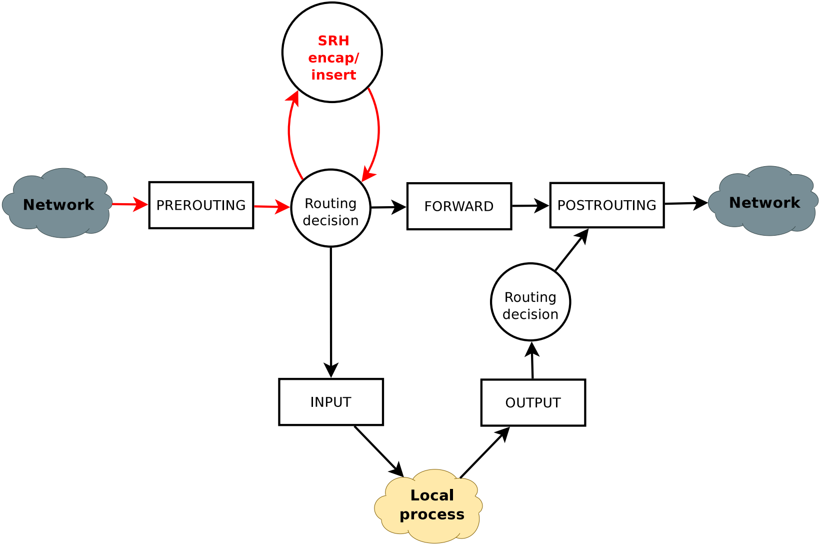


Figure: SRH insertion codepath for forwarded packets.

SRH insertion/encapsulation (local packet)

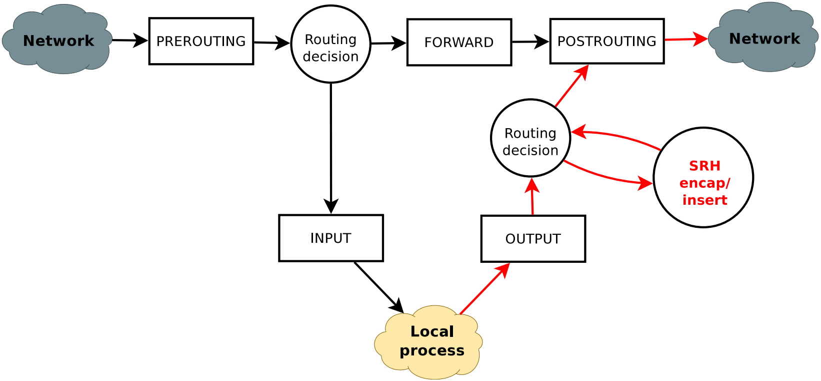


Figure: SRH insertion codepath for locally generated packets.

SRH processing/decapsulation

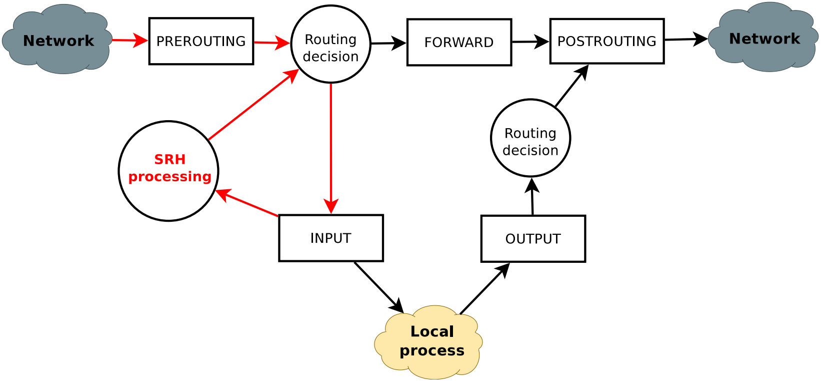


Figure: SR-enabled packet codepath.

Per-socket SRH insertion

- An application can (partially) *program* the network

Listing 1: Application code defining a per-socket SRH.

```
struct ipv6_sr_hdr *srh;
int fd, srh_len;

srh_len = build_srh(&srh);

fd = socket(AF_INET6, SOCK_STREAM, IPPROTO_TCP);

setsockopt(fd, IPPROTO_IPV6, IPV6_RTHDR, srh, srh_len);
```

Performance evaluation: hardware setup

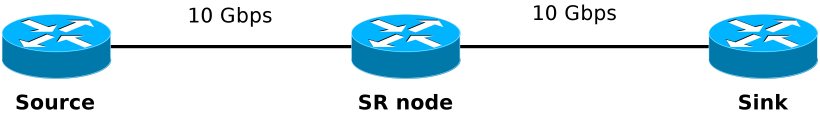


Figure: Physical testbed.

- Intel Xeon X3440 @ 2.53 GHz (4 cores / 8 threads)
- Intel 82599 10 Gbps Ethernet cards
- 16 GB RAM

Single-core performance

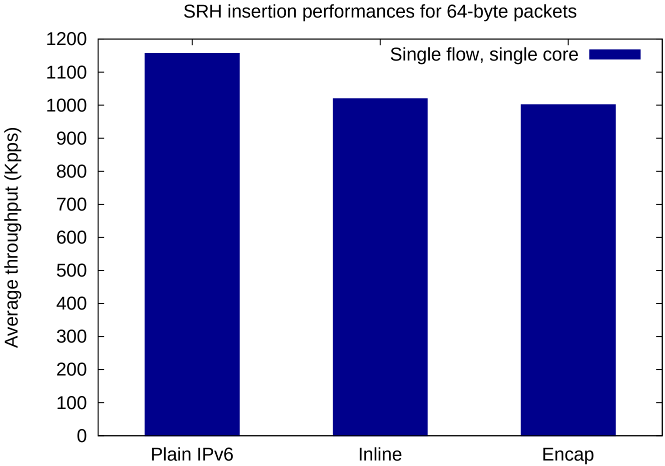


Figure: Performance with a single core.

4-core performance

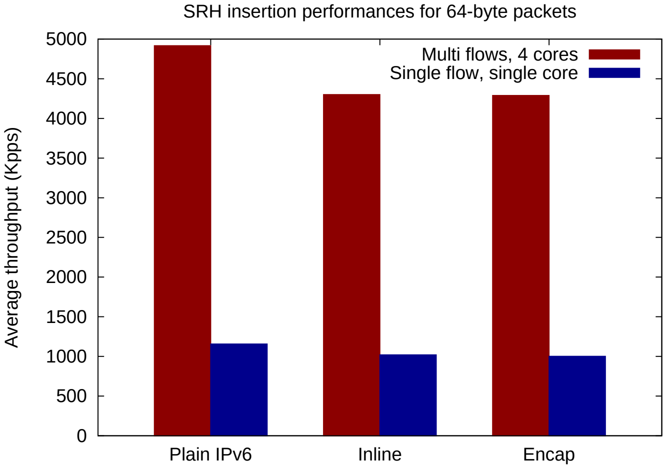


Figure: Performance comparison between single core and four cores.

SRv6 in Linux: conclusion

- Available in official Linux kernel (about 3,000 LoC)
  - Accepted and merged in November 2016
  - Linux 4.10 (Feburary 2017): first release
  - Linux 4.12 (July 2017): performance improvements
  - Linux 4.14 (November 2017): new features
- Good and scalable performances
- Anyone can contribute

Table of Contents

Introduction

Segment Routing

Implementing IPv6 Segment Routing in Linux

Exploring IPv6 Segment Routing

Rethinking IPv6 Enterprise Networks

Conclusion

Overview

- Traffic duplication for latency-critical application[3]

- **Fine-grained and scalable network monitoring**[4]

[3] François Aubry, David Lebrun, Yves Deville, and Olivier Bonaventure. "Traffic duplication through segmentable disjoint paths". In: *IFIP Networking Conference (IFIP Networking), 2015*. IEEE. 2015, pp. 1–9.

[4] Francois Aubry, David Lebrun, Stefano Vissicchio, Minh Thanh Khong, Yves Deville, and Olivier Bonaventure. "SCMon: Leveraging segment routing to improve network monitoring". In: *35th Annual IEEE International Conference on Computer Communications, INFOCOM 2016*. IEEE. 2016, pp. 1–9.

Network monitoring

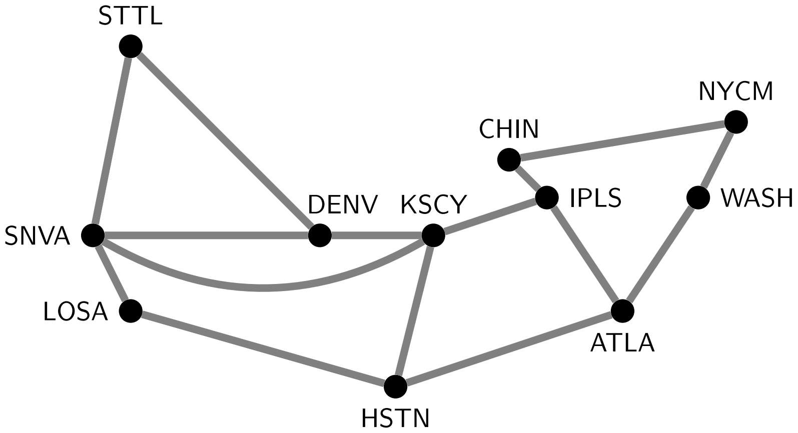


Figure: Abilene network.

Link bundles

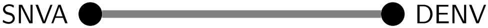


Figure: Routing perspective.

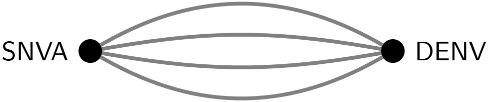


Figure: Physical perspective.

Backup links

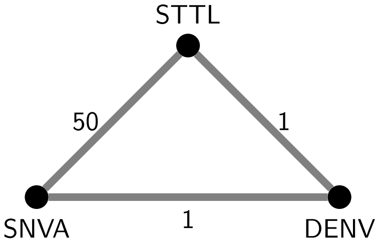


Figure: Topology with backup link.

Equal-Cost Multi-Path

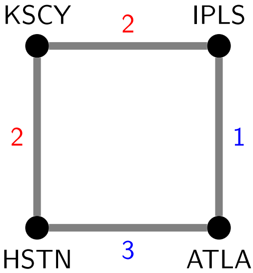


Figure: ECMP topology.

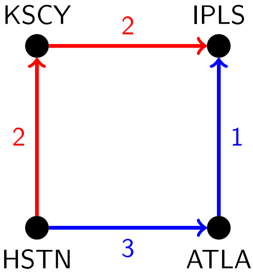


Figure: ECMP routing.

Monitoring mechanisms

- Protocol-based [BFD]: per-link heartbeat

- Probe-based [IPSLA]: dataplane probe

Protocol-based monitoring

- Per-link, per-router configuration
- Miss forwarding failures

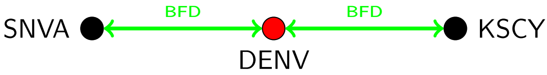


Figure: Undetected forwarding failure.

Probe-based monitoring

- Shortest-path forwarding
- Multiple vantage points
- Cannot traverse backup links
- Miss ECMP and bundle failures

SCMon

- Create cycles with segments
- Send probes over those cycles
- Single vantage point

Cycles (1)

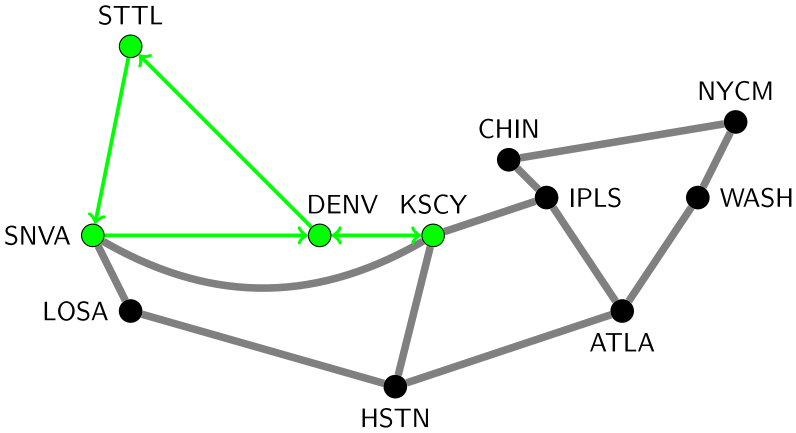


Figure: Abilene network.

Cycles (2)

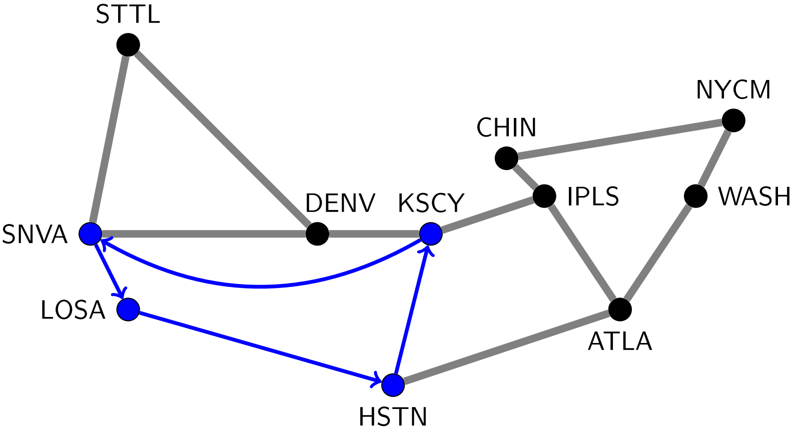


Figure: Abilene network.

Cycles (3)

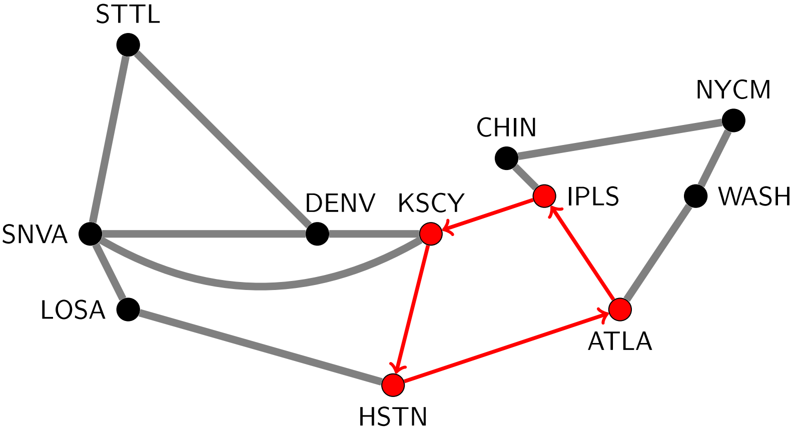


Figure: Abilene network.

Cycles (4)

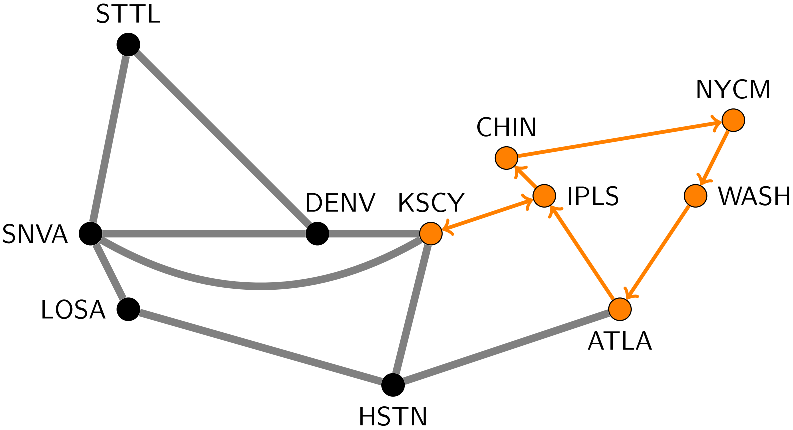


Figure: Abilene network.

SCMon evaluation

Topology	Nodes	Links	Cycles	Avg RTT	Max RTT
OVH Europe	57	216	87	18 ms	28 ms
RF AS1239	153	1010	195	83 ms	360 ms
RF AS1755	67	248	34	49 ms	130 ms
RF AS3257	103	484	76	48 ms	127 ms
RF AS3967	57	208	24	109 ms	206 ms

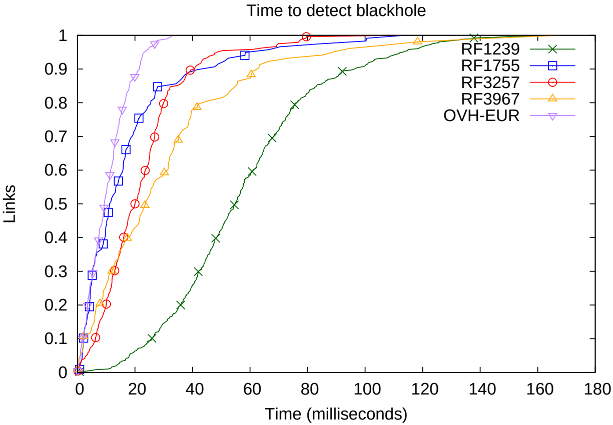


Figure: Link failure detection time for each topology.

Network monitoring: conclusion

- SCMon: Single-box monitoring
- Data plane probes over cycles
- Prototype implementation
- Detect and locate link failure within milliseconds

Table of Contents

Introduction

Segment Routing

Implementing IPv6 Segment Routing in Linux

Exploring IPv6 Segment Routing

Rethinking IPv6 Enterprise Networks

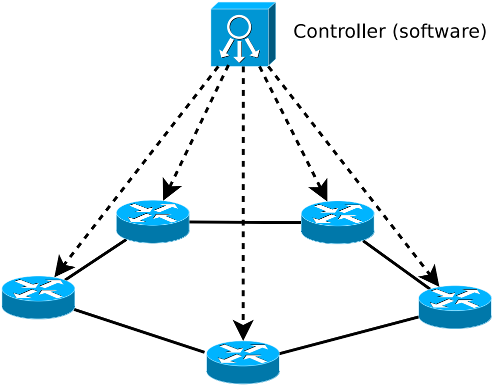
Conclusion

Enterprise networks

- Complex networks, various business policies
- Operator needs fine-grained traffic engineering (→ SR)
- Fast reaction to failures (→ underlying IGP)
- Best place for control: traffic sources (→ `setsockopt()`)
- How do sources (applications) know the segments to use ?

Software-Defined Networking

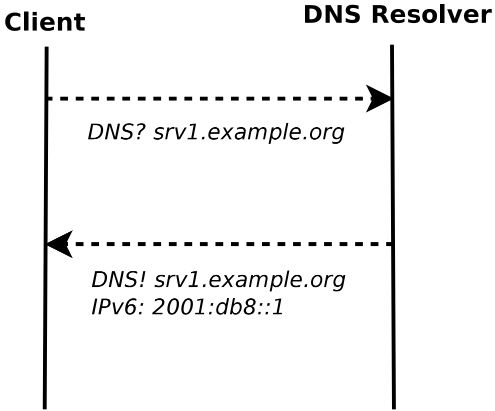
- Central controller knows the network state and configures the devices
- Application ↔ controller communication ?



DNS protocol

- Domain Name System
- Resolve names to IP addresses
- Example: `google.com` → `2a00:1450:4009:815::200e`
- Used virtually everywhere
- **Idea**: piggyback app flow control on DNS messages

Regular DNS request



Software Resolved Network

- Use DNS as network signaling protocol → *Software Resolved Network*
- Resolver = Controller → *SDN Resolver*

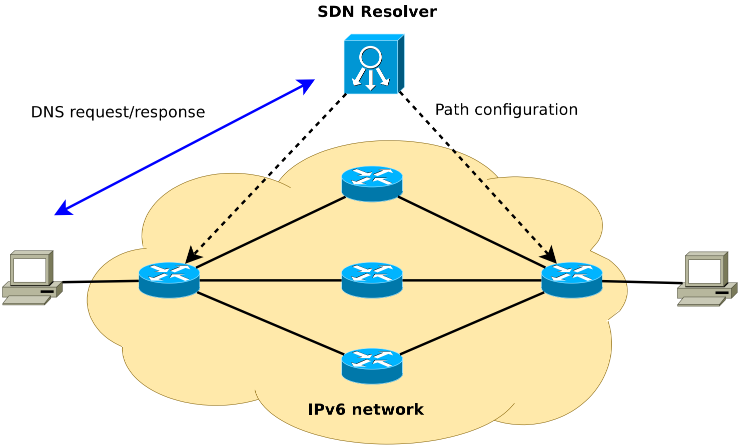
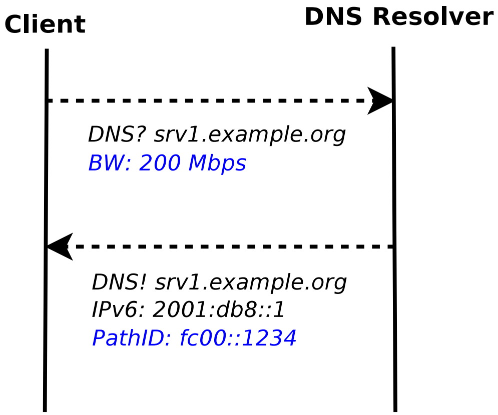


Figure: Software Resolved Network.

SRN-augmented DNS request

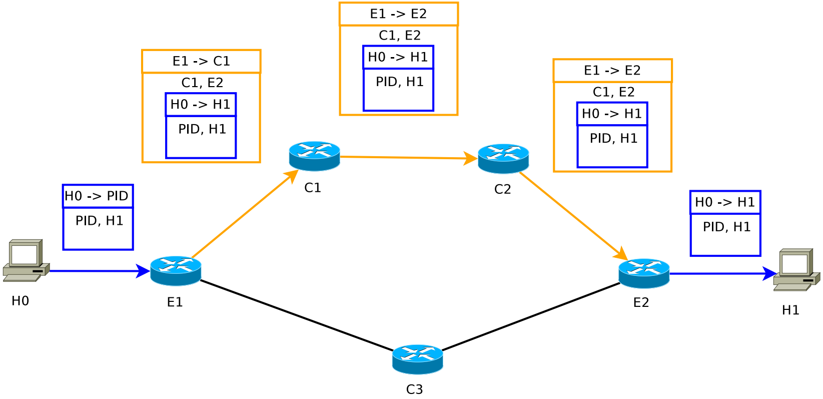


Conversations

- Conversations: bidirectional flow between applications
- Identified by a unique **PathID**
- Mapping PathID ⇒ network path
- Applications use only PathID

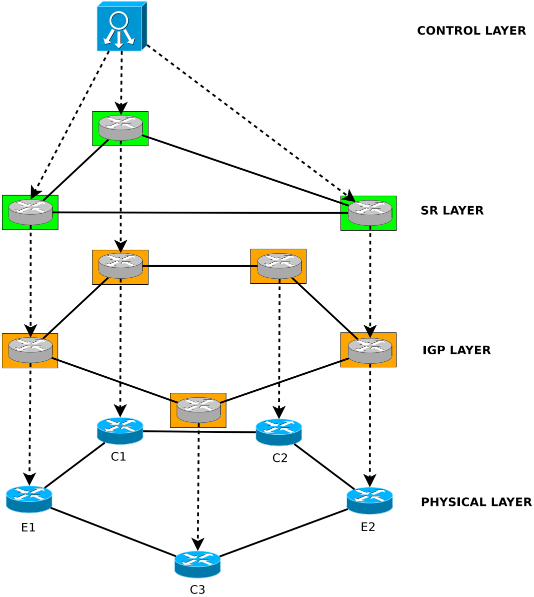
Implementing network paths

- Edge maps PathID → list of segments
- ⇒ Additional state only in edge

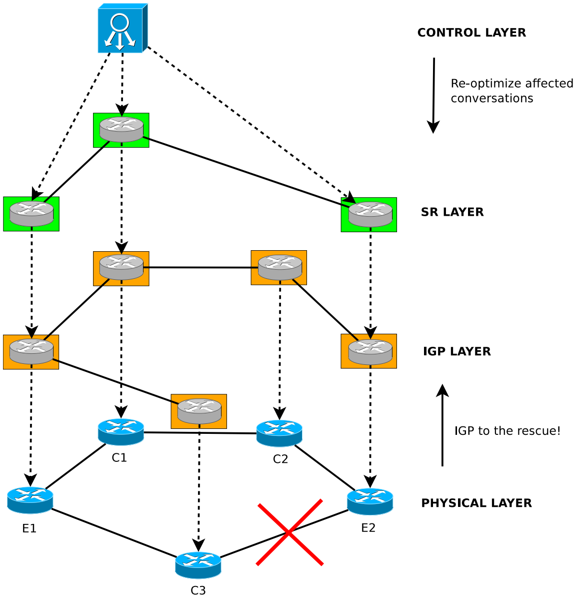


)

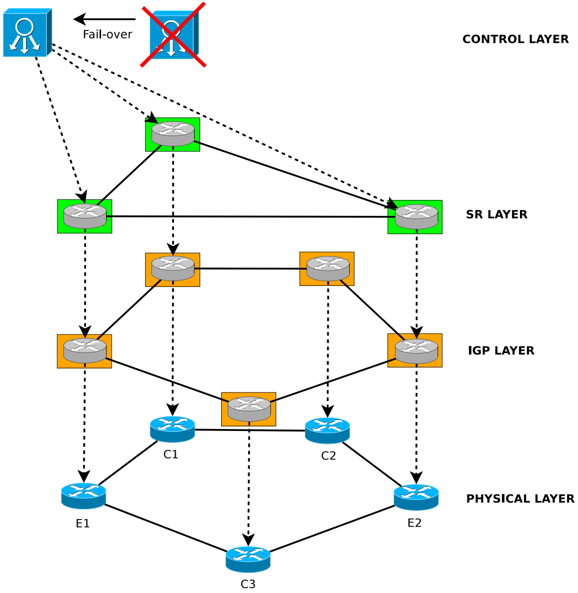
SRN layers


CONTROL LAYER
SR LAYER
IGP LAYER
PHYSICAL LAYER
C1
C2
E1
E2
C3

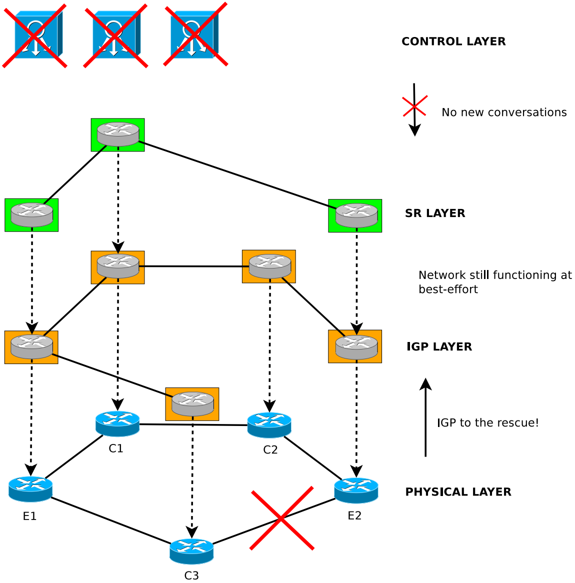
Fault tolerance: link failure



Fault tolerance: controller failure



Fault tolerance: full controller outage



Controller implementation

- Complete prototype in about 10,000 lines of C code[5]
- Microbenchmarks
- Virtual network experiment

[5] David Lebrun. *SDN Resolver controller code*. https://github.com/target0/thesis-data/sdnres-src.

Microbenchmark evaluation

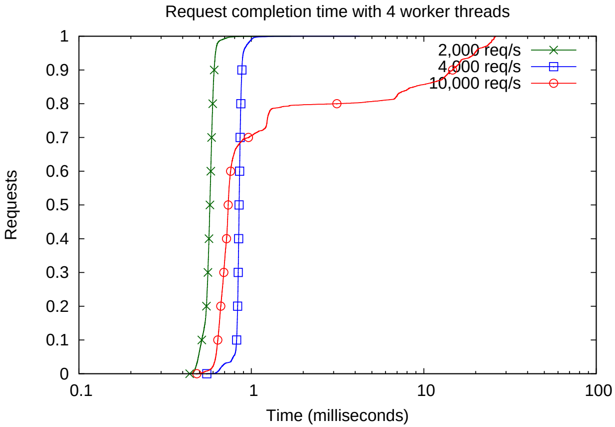
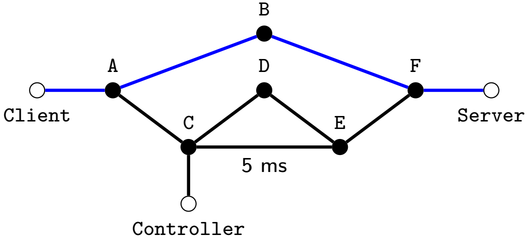
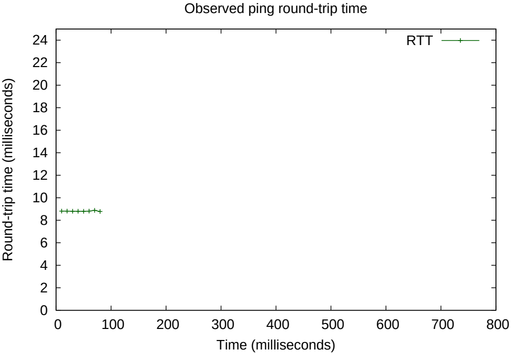


Figure: Request completion time with four worker threads for various loads.

Virtual network experiment: initial setup

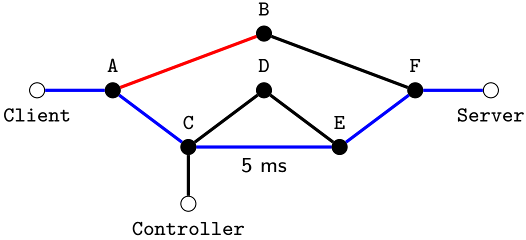
Segments: F

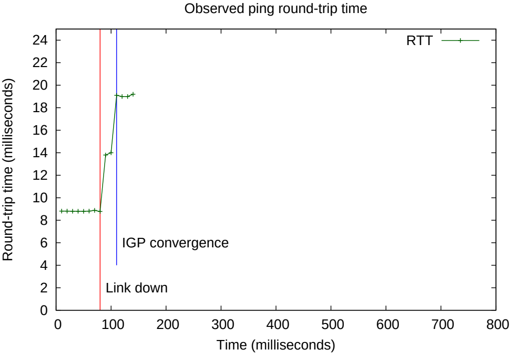




Virtual network experiment: link down and IGP convergence

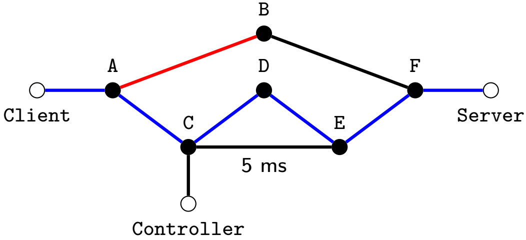
Segments: **F**

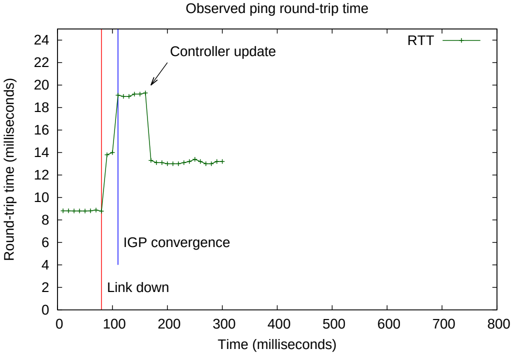




Virtual network experiment: controller update

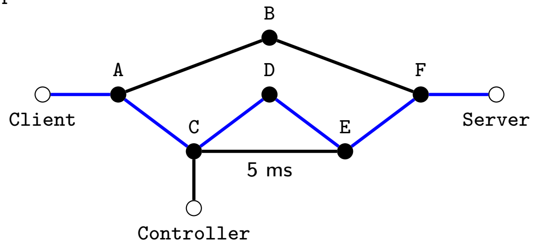
Segments: `D, F`

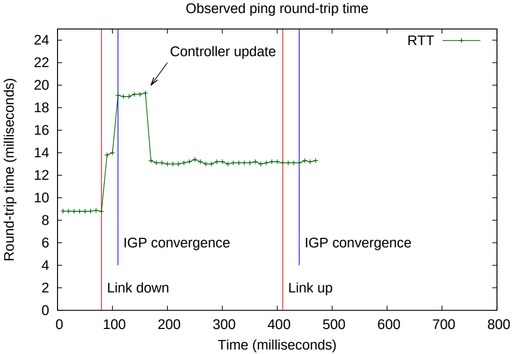




Virtual network experiment: link up and IGP convergence

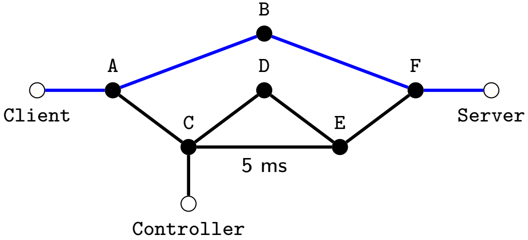
Segments: `D, F`

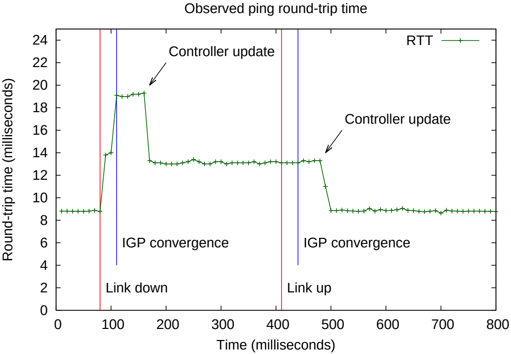




Virtual network experiment: controller update

Segments: `F`





Software Resolved Networks: conclusion

- SDN-like architecture for enterprise networks
- Traffic engineering through SRv6
- Applications interact with controller through DNS
- Complete prototype implementation
- Evaluation meets performance expectations

Table of Contents

Introduction

Segment Routing

Implementing IPv6 Segment Routing in Linux

Exploring IPv6 Segment Routing

Rethinking IPv6 Enterprise Networks

Conclusion

Conclusion

- Linux kernel implementation of SRv6
- Exploration of SRv6 applications
- Software Resolved Networks
- Fully reproducible: all code and data open-source and available
  - https://www.kernel.org (Linux kernel code)
  - https://github.com/target0/thesis-data