Consortium GARR

8° BORSISTI DAY
"ORIO CARLINI"
MERCOLEDÌ 14 DICEMBRE 2016 - ROMA

MONSTER

Managing an Operator's Network with Software Defined Networking and Segment Routing

Ing. **Luca Davoli**

davoli@ce.unipr.it

Tutor: Prof. Ing. Luca Veltri

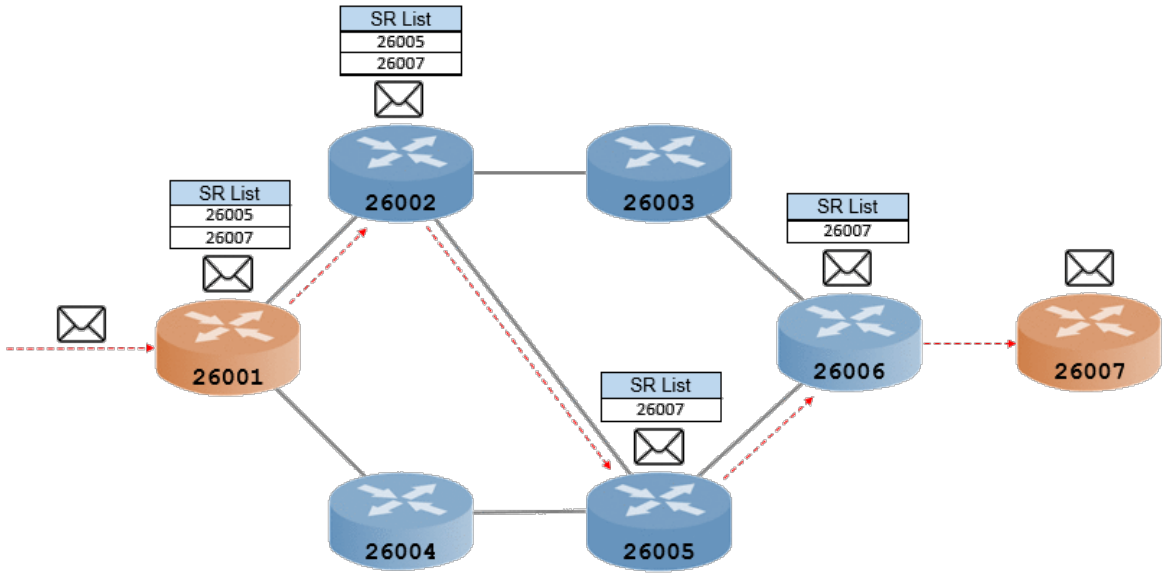
UNIVERSITÀ DEGLI STUDI DI PARMA
Dipartimento di Ingegneria dell'Informazione

Overview

- Segment Routing
- Traffic Engineering with SDN & SR
- IPv6 SR: Alternatives and Solutions
- Network Emulator (NEMO)
- Service Function Chaining & Network Function Virtualization
- SR/Virtual Function Chaining
- SR and Network Functions
  - SR-aware architecture
  - SR-unaware architecture

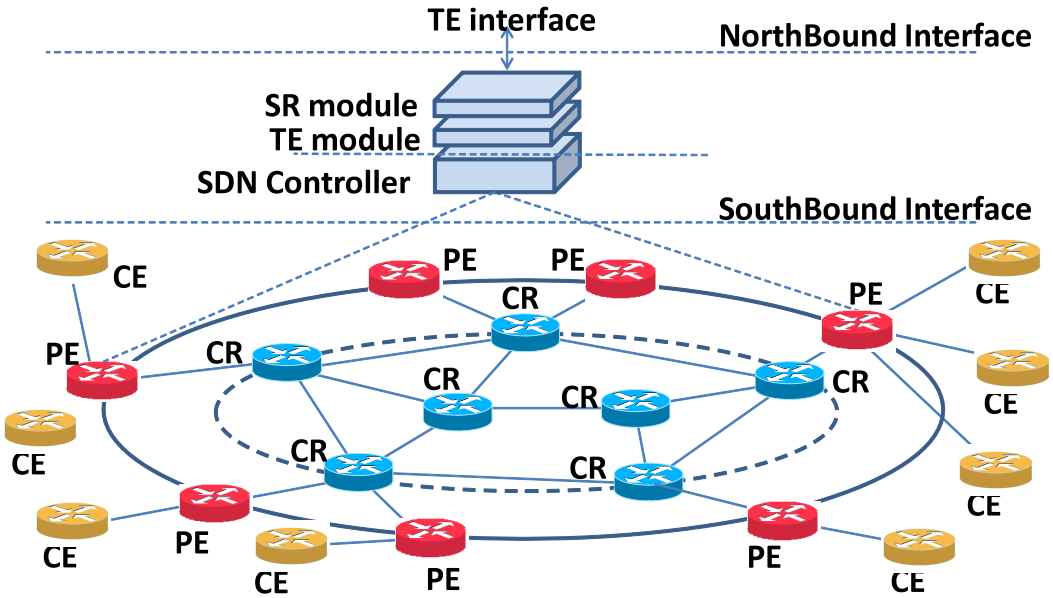
Segment Routing

- Segment Routing (SR): based on source routing
  - Enhanced packet forwarding without any topological restrictions and additional signaling requirements
  - Segments based on MPLS or IPv6



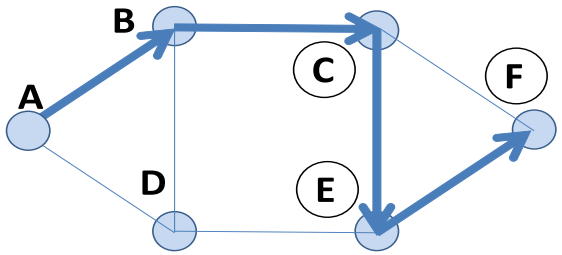
Traffic Engineering with SDN

- ISP network managed by a (logically) centralized SDN controller
- Provider Edge (PE) routers and Core Routers (CR) are hybrid IP/MPLS/ SDN nodes
- MPLS is used for TE
  - TE paths enforced by using Segment Routing (SR)
  - no change to the MPLS forwarding plane is required
  - no MPLS control plane has to be used



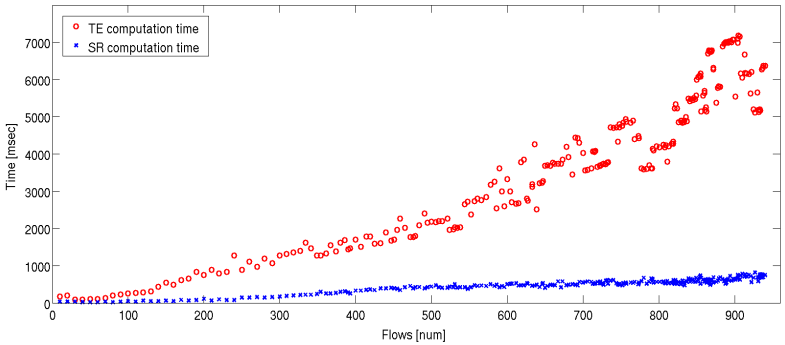
Traffic Engineering with Segment Routing

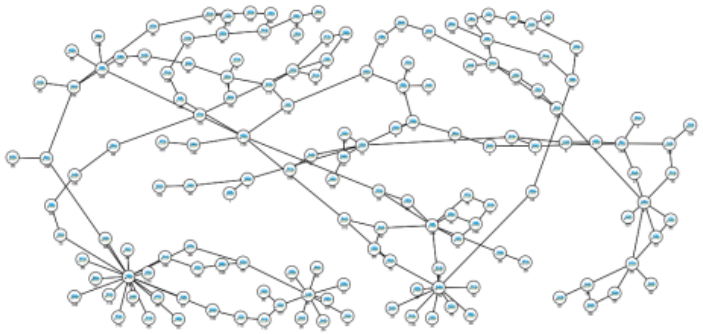
- Enhancement of a SDN controller with TE/SR modules
- The SDN controller is requested to allocate a set of traffic flows with a specified bit rate
  - The *flow assignment* algorithm is first executed in order to compute *TE paths*
    - Minimization of the overall network crossing time
  - For each *TE path*, the corresponding *SR path* is calculated
    - SR path is the list of *SIDs* that should be added to incoming packets for instructing them through the assigned *TE path*
    - Simple SR assignment algorithm that minimizes the number of required *SIDs*
- Studied different SR algorithms



Implementation and tests

- Proposed network architecture has been implemented and tested
- Experimental analysis with two main goals:
  - testing the SR assignment algorithm
    - 153 nodes, 354 links, 940 out of 2460 flows
  - testing the overall implementation of the solution
    - both control and data planes



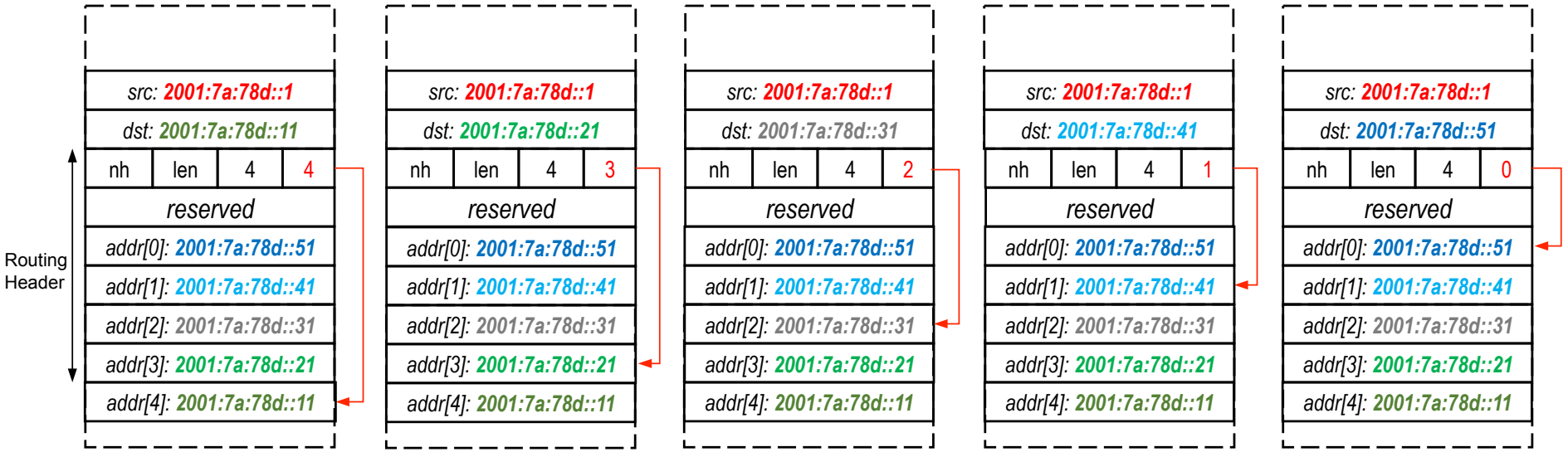


IPv6 SR

- IPv6 SR support requires the changing of IPv6 packet processing
- IPv6 Segment Routing Header (SRH)

  https://tools.ietf.org/html/draft-ietf-6man-segment-routing-header

IPv6 SR

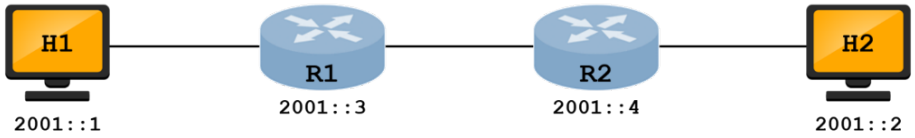

Routing Header
src: 2001:7a:78d::1
dst: 2001:7a:78d::11
nh | len | 4 | 4
reserved
addr[0]: 2001:7a:78d::51
addr[1]: 2001:7a:78d::41
addr[2]: 2001:7a:78d::31
addr[3]: 2001:7a:78d::21
addr[4]: 2001:7a:78d::11
src: 2001:7a:78d::1
dst: 2001:7a:78d::21
nh | len | 4 | 3
reserved
addr[0]: 2001:7a:78d::51
addr[1]: 2001:7a:78d::41
addr[2]: 2001:7a:78d::31
addr[3]: 2001:7a:78d::21
addr[4]: 2001:7a:78d::11
src: 2001:7a:78d::1
dst: 2001:7a:78d::31
nh | len | 4 | 2
reserved
addr[0]: 2001:7a:78d::51
addr[1]: 2001:7a:78d::41
addr[2]: 2001:7a:78d::31
addr[3]: 2001:7a:78d::21
addr[4]: 2001:7a:78d::11
src: 2001:7a:78d::1
dst: 2001:7a:78d::41
nh | len | 4 | 1
reserved
addr[0]: 2001:7a:78d::51
addr[1]: 2001:7a:78d::41
addr[2]: 2001:7a:78d::31
addr[3]: 2001:7a:78d::21
addr[4]: 2001:7a:78d::11
src: 2001:7a:78d::1
dst: 2001:7a:78d::51
nh | len | 4 | 0
reserved
addr[0]: 2001:7a:78d::51
addr[1]: 2001:7a:78d::41
addr[2]: 2001:7a:78d::31
addr[3]: 2001:7a:78d::21
addr[4]: 2001:7a:78d::11
2001:7a:78d::1
2001:7a:78d::11
2001:7a:78d::21
2001:7a:78d::31
2001:7a:78d::41
2001:7a:78d::51
packet source
specified router
non-specified router
packet final destination

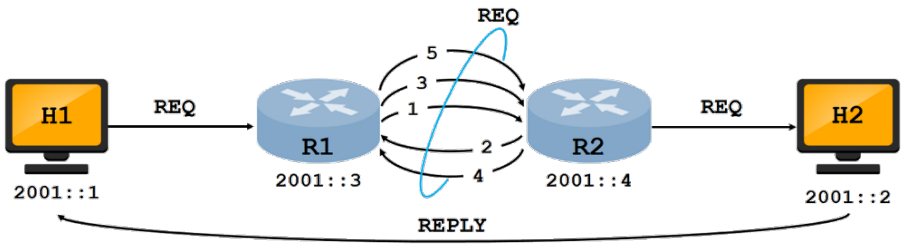
IPv6 SR

- Alternatives for supporting IPv6 SR on Linux
  - Modification of the Linux kernel
    - Kernel-level dev → Highest performance
  - Netfilter hooks + kernel modules
    - Kernel module dev
  - Iptables NFQUEUE + userspace dev
    - User level dev
      - Compatible with JNI-based userspace dev
      - Neither Kernel modification nor kernel module required
  - Rawsocket
    - User level dev

IPv6 SR on Linux: kernel patch

- Kernel modification
  - Available patch from UCLovail, Belgium
- Different experimental tests

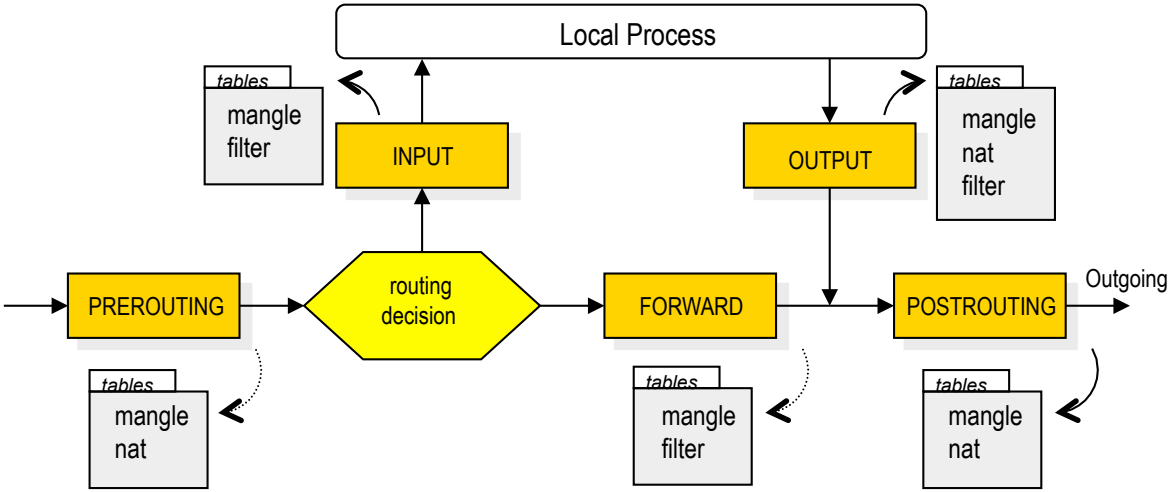




- Different kernels tested
- Limitations
  - Require a patched Linux kernel → Lower flexibility
  - Difficulty to combine SRH processing with possible network functions
  - Issues with large-scale network emulation
    - Problems with Mininet

IPv6 SR on Linux: netfilter

- Netfilter
  - Framework in Linux kernel for packet mangling
  - Series of hooks in various points of the kernel network stack
  - The hooks can be exploited to define custom functions for manipulating IP packets

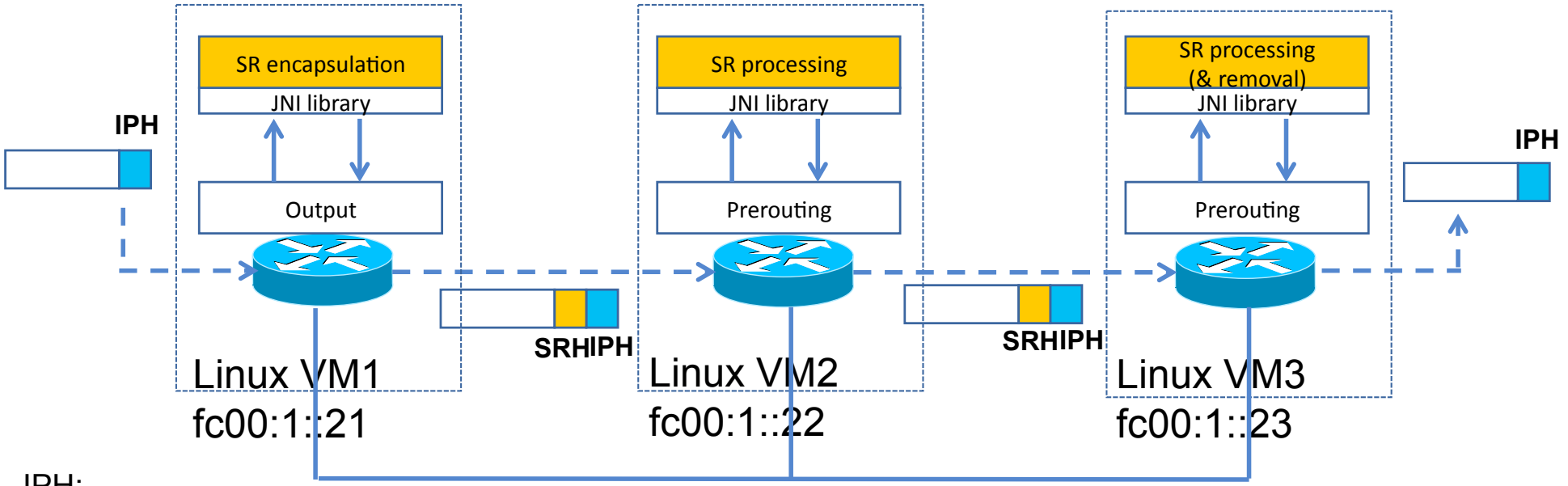


IPv6 SR on Linux: Userspace

- SR implementation
  - Packets are captured in prerouting, enqueued (NFQUEUE), processed at userspace, returned with SRH updated
  - C, JNI, Java development
    - C JNI library for receiving packets form the netfilter queue
      - providing a Java interface to netfilter queues
    - Java-based SR processing
- ***Note***: no kernel modification required

```
ip6tables -t mangle -A PREROUTING -p ipv6 -j NFQUEUE --queue-num 0
```

IPv6 SR on Linux: Testbed

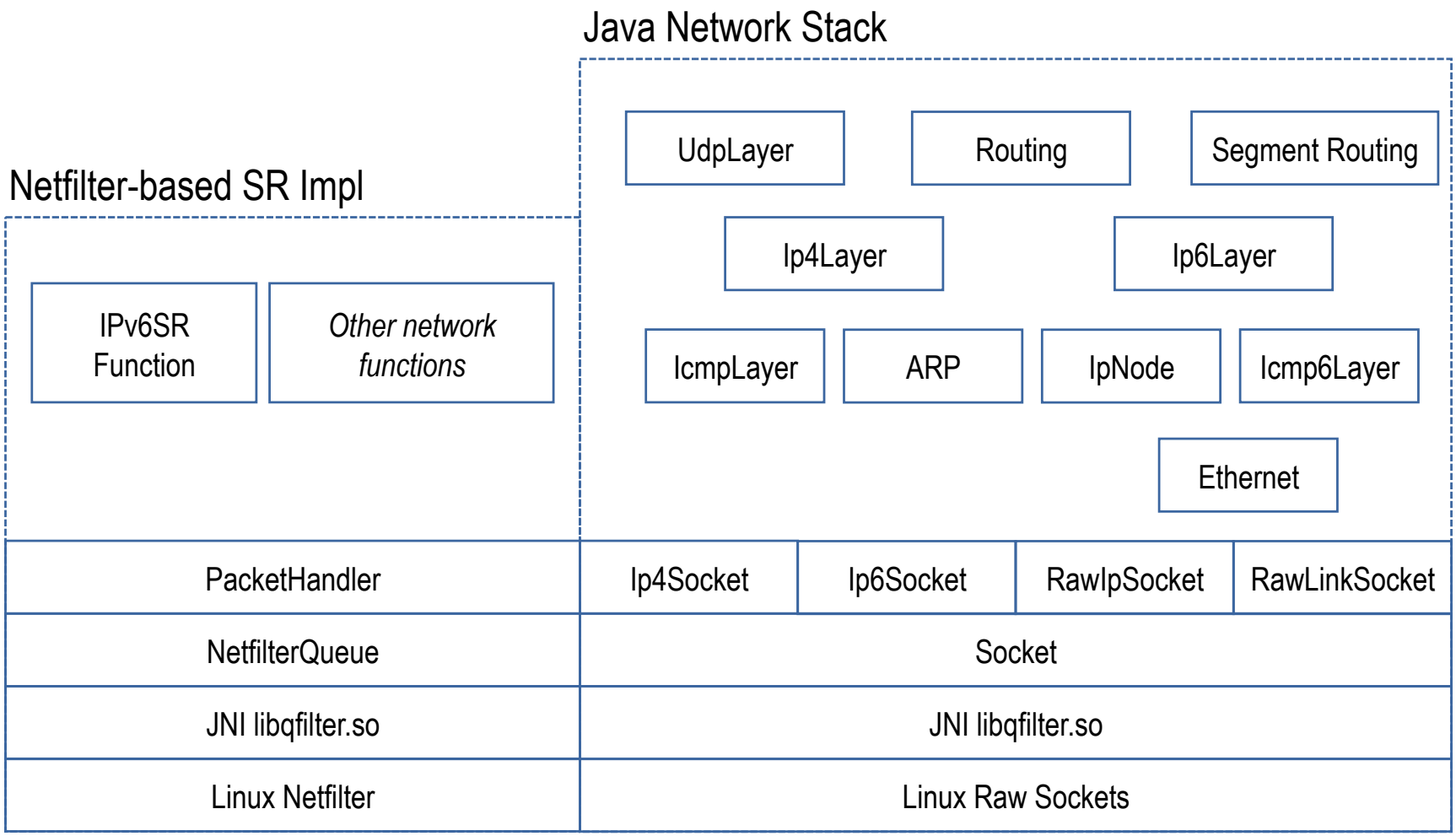


IPH:
SA=fc00:1::21
DA=fc00:1::23

DA → SRH:
SID1=fc00:1::22
SID2=fc00:1::23
SID3=fc00:1::23

DA → SRH:
SID1=fc00:1::22
SID2=fc00:1::23
SID3=fc00:1::23

SR and Network Implementation



Network Emulator (NEMO

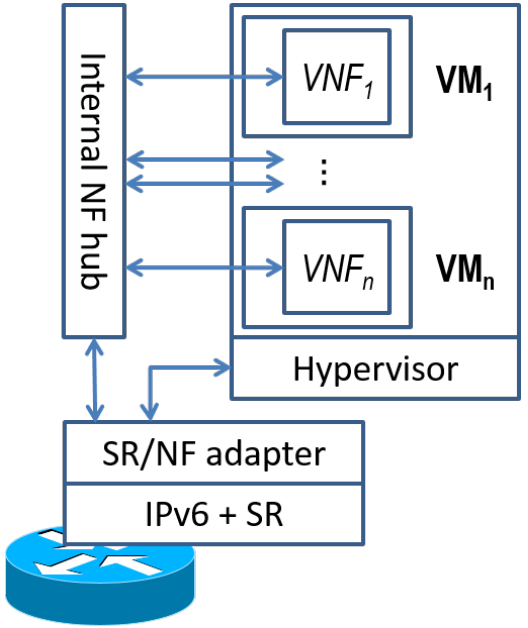
- Java-based framework able to emulate an IP-based network (based on the Java Network Stack) → large amount of virtual routers
- Possibility to interconnect a real network card with multiple NICs
  → route traffic coming from/going to an external IP-based network
- Complete IP stack
  - Ethernet / ARP / IPv4 / IPv6 / ICMP / ICMPv6 / UDP
- Routing Function → Shortest Path for dynamic RT configuration
- Unix-based platforms: possibility to intercept incoming packets with *Rawsocket* mechanism (at Ethernet layer)
- Integration successfully tested and verified
  → Possibility to deploy virtual networks whose routing is handled via SRH

Service Function Chaining & Network Function Virtualization

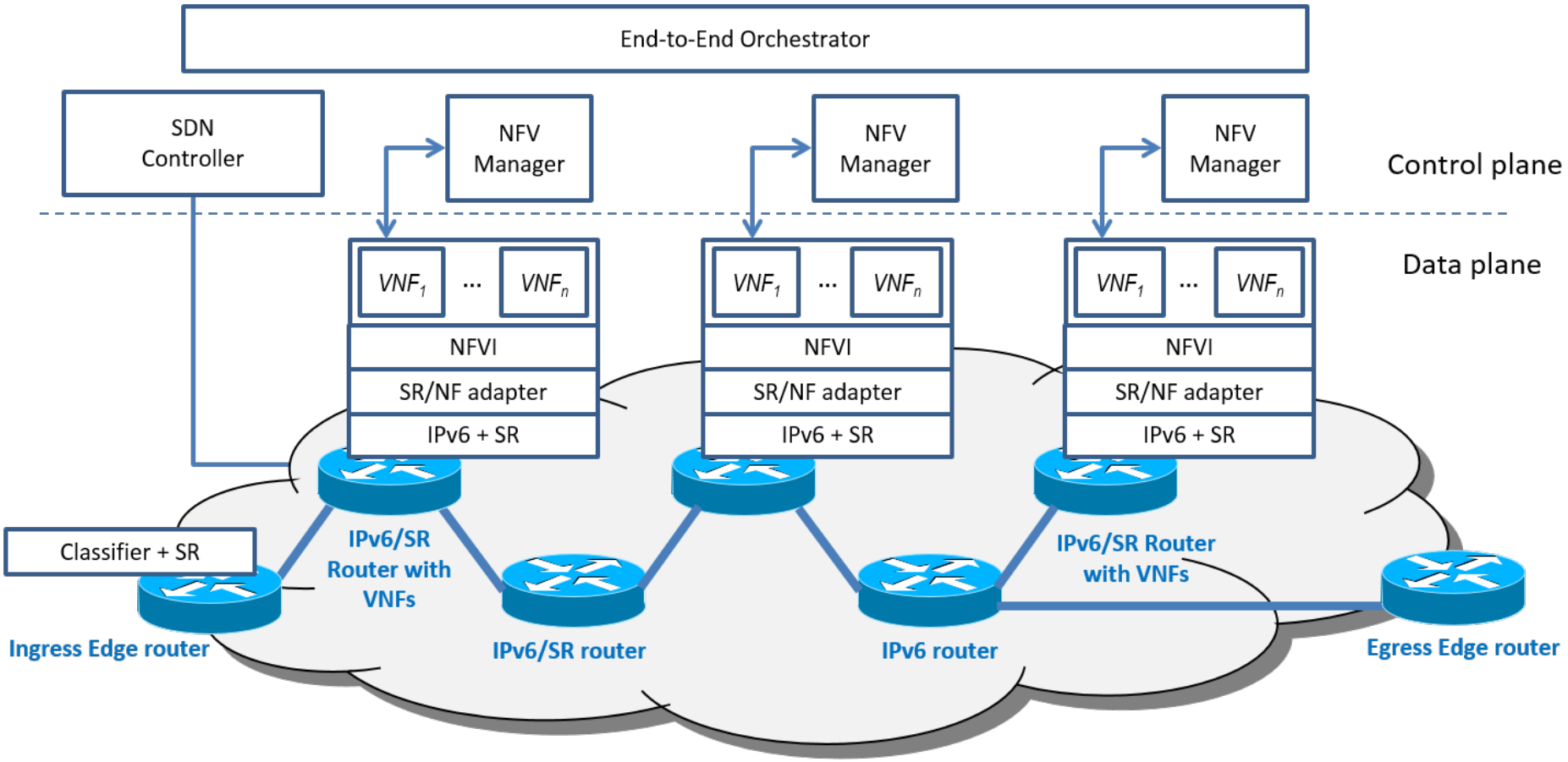
- SR as enabler technology for SFC

- Set of network services interconnected through the network to support an application
- SFs or NFs can be physically deployed or virtualized (NFV)
- NFV moves network functions out of dedicated HW and into SW

- Network functions typically execute as VMs under control of an hypervisor

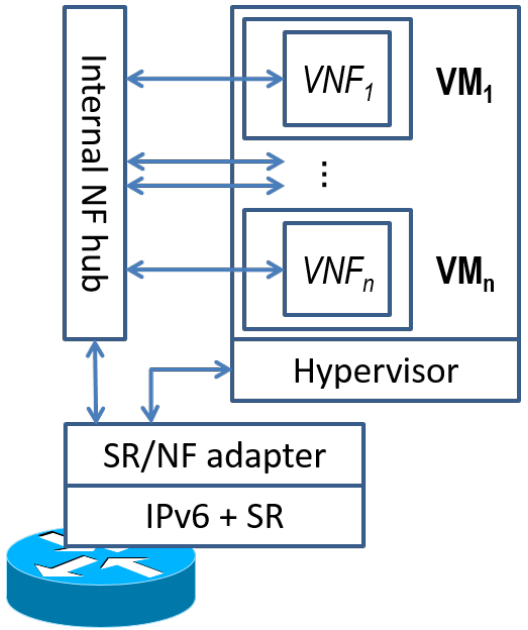


SR/VNF Chaining Architecture



SR and Network Functions

- SR-aware Network Functions
  - SR packets are passed to the NF
  - NF may enforce other NFs modifying the segment list in the SRH
    - inserted between this node and the next segment
    - inserted in any position along the SR path
      - the NF is aware of the NFs corresponding to the segment IDs already present in the segment list
    - adds, removes, changes the order of the next segments
- SR-unaware Network Functions
  - packets are first processed before the NF
    - original packet is extracted (without the SRH)
    - the SR list is in some way stored in order to be re-attached to the packet when returned by the NF
      - Stateful/Stateless

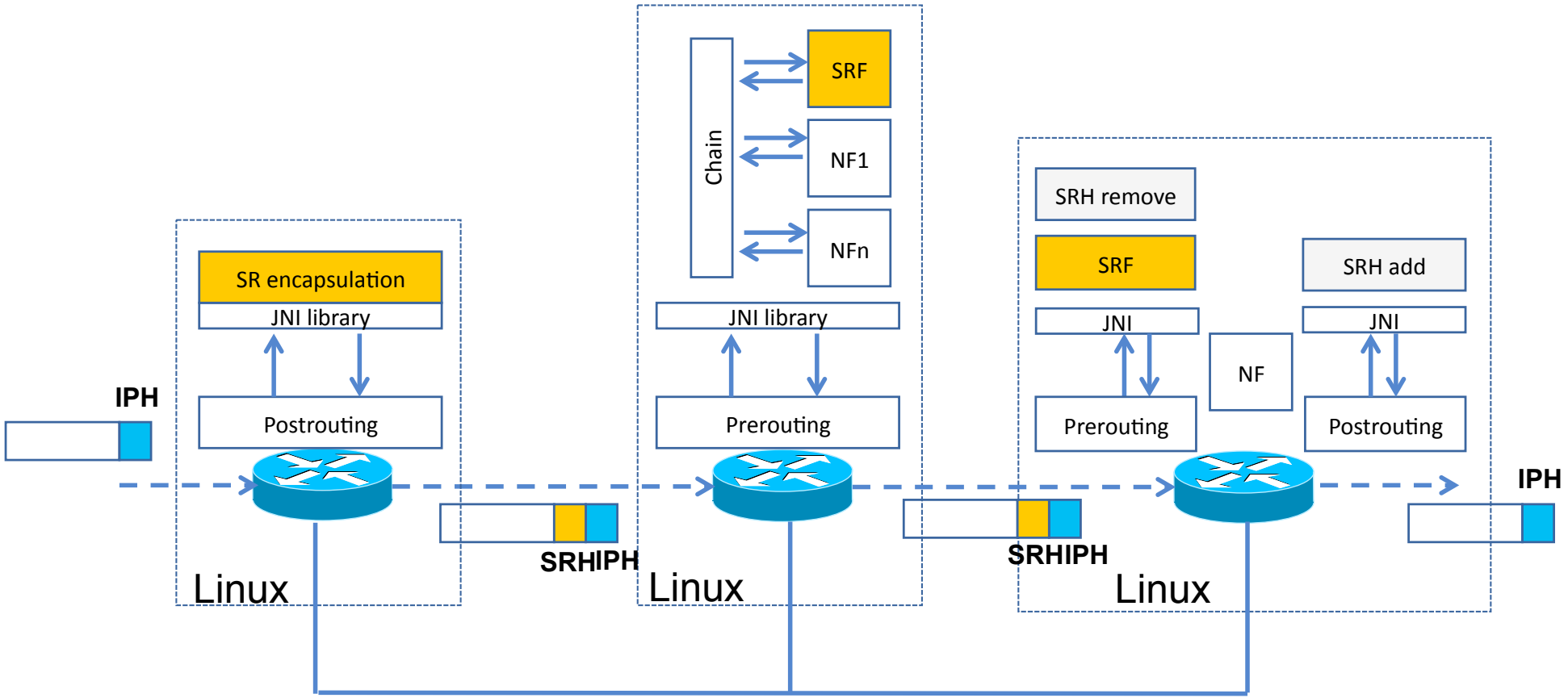


SR/VNF Chaining Implementation

- Based on the userspace SR development
- Packets are captured in prerouting and passed to a set of NFs
- ***Note***: SR can be seen as special case of NF

- Both SR-aware and SR-unaware NFs are supported
  - In case of SR-unaware NFs, SRH removed and inserted in IPv6 Destination Options Extension Header
    - Stateless mechanism
    - SRH reinserted in postrouting
  - Example of SR-unaware NF: netfilter firewall

SR/VNF Chaining Testbed



Consortium GARR

8° BORSISTI DAY
"ORIO CARLINI"
MERCOLEDÌ 14 DICEMBRE 2016 - ROMA

MONSTER

Managing an **O**perator's **N**etwork with **S**oftware Defined Ne**t**working and S**e**gment **R**outing

Thanks for your attention!