# Bit Index Explicit Replication (BIER) Multicasting in Transport Networks

**A. Giorgetti[(1)], A. Sgambelluri[(1)], F. Paolucci[(1)], N. Sambo[(1)], P. Castoldi[(1)], F. Cugini[(2)]**

(1) Scuola Superiore Sant'Anna, Pisa, Italy

(2) CNIT, Pisa, Italy,

**ONDM 2017**

# Outline

- **Segment Routing (SR)**
  - technology
  - Path encoding
  - TE performance (ECMP vs strict routing)
  - Use cases

- **Bit Index Explicit Replication (BIER)**
  - technology
  - Experimental validation

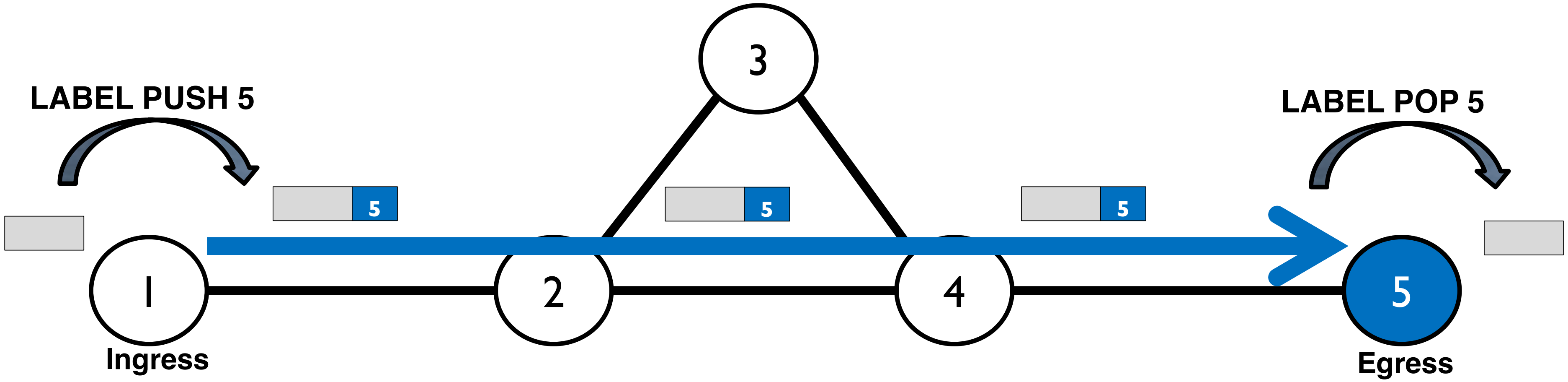
# Segment Routing

- **Segment Routing (SR)** is a traffic engineering (TE) technique compatible with traditional **MPLS data plane**.
- Using SR, a **signaling protocol is not required** and path state is not maintained in intermediate nodes → simplified control plane operation.

- Each packet is forwarded according to an header composed of **segment identifiers** (SIDs), e.g., representing a specific network node.
- SIDs are advertised by properly extended IGP (e.g., OSPF-TE).

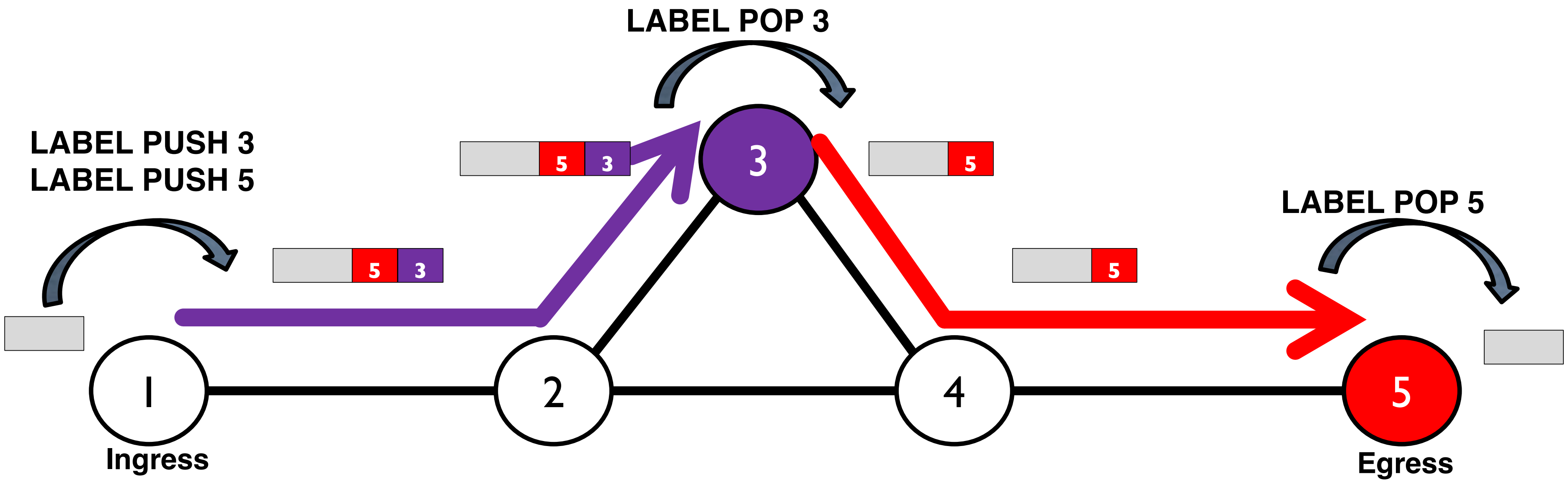
- Intermediate nodes forward the received packet along the **shortest path** toward the node indicated in the top SID.

# Segment Routing (SR) basic behavior (1/2)



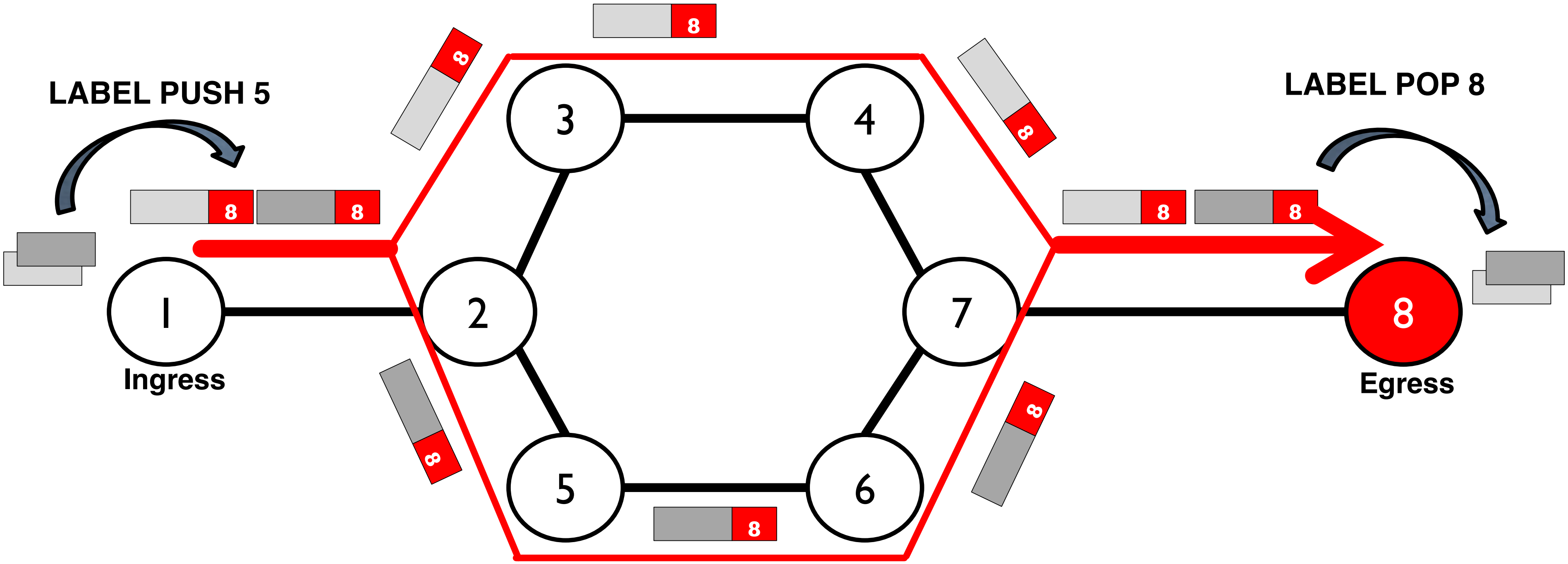

Target path: 1,2,4,5 → segment list: 5

# Segment Routing (SR) basic behavior (2/2)




Target path: 1,2,3,4,5 → segment list: 3, 5

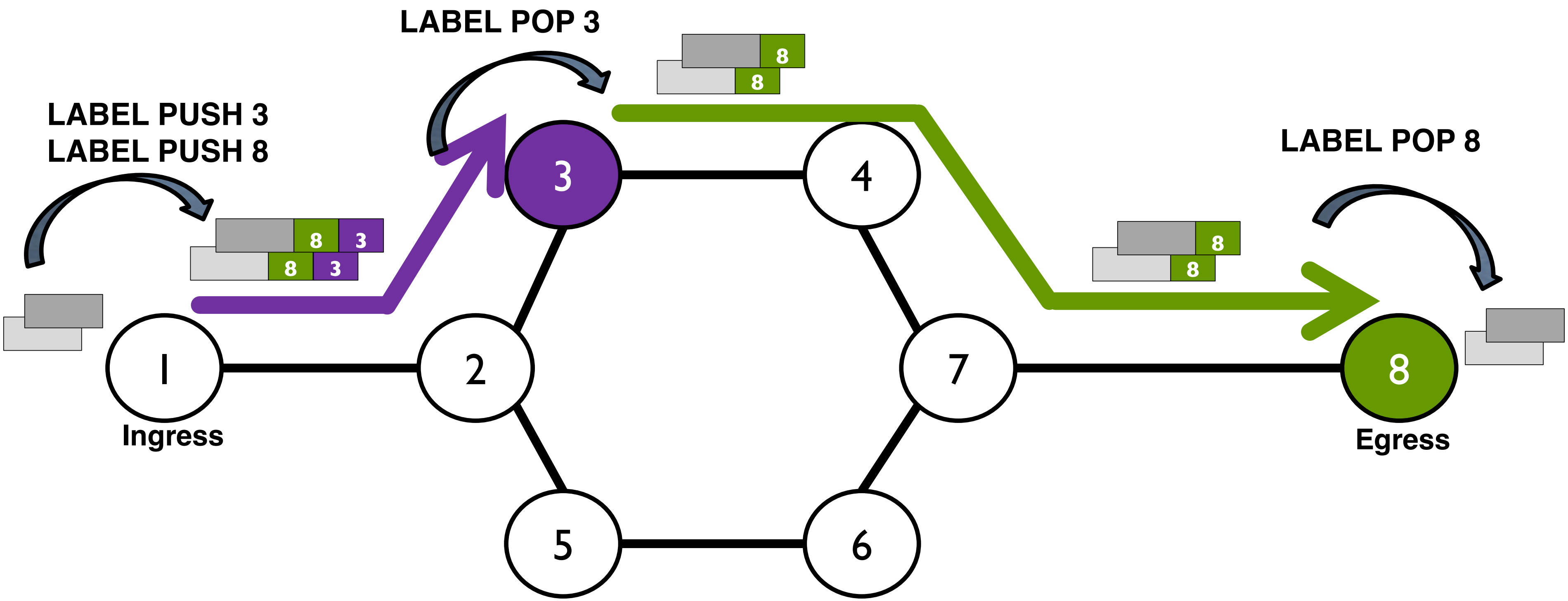
# Equal Cost Multi Paths (ECMP)




Default behavior → load balancing on ECMPs

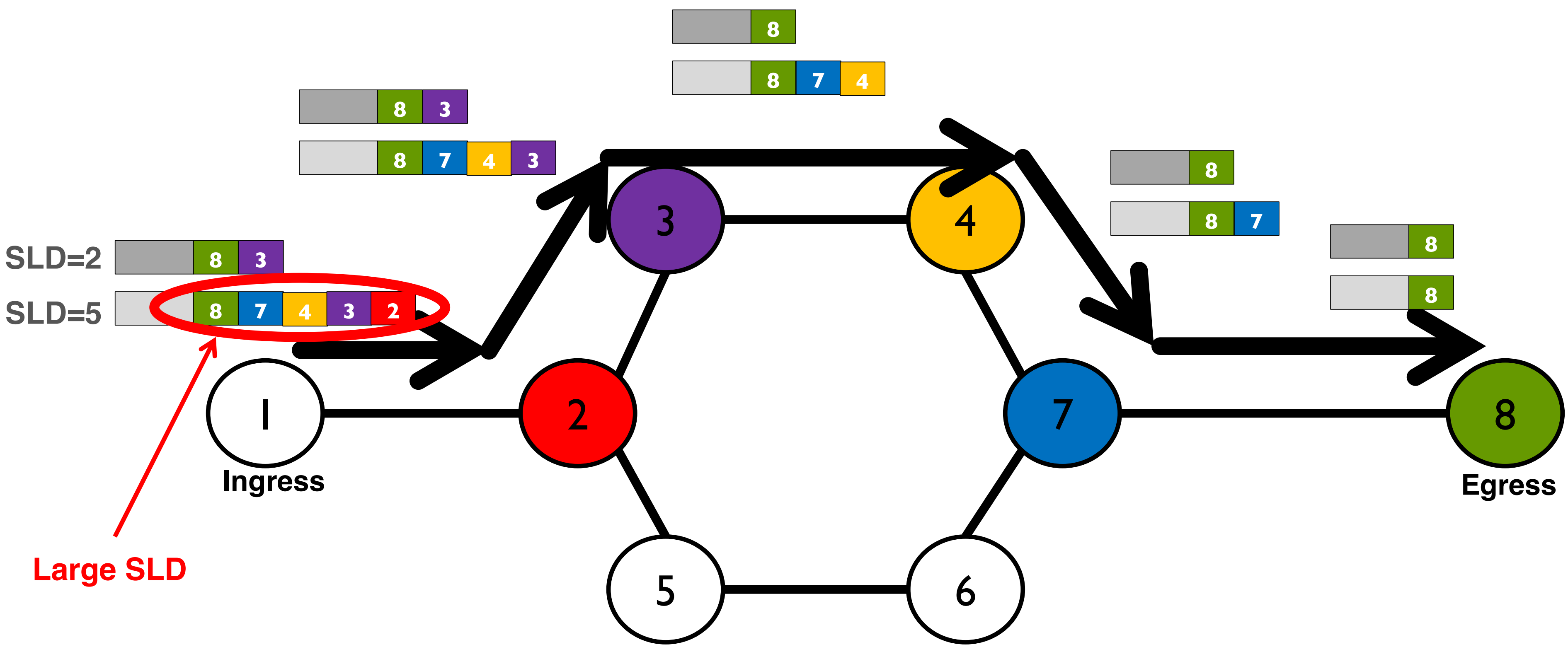
Target paths: 1,2,3,4,5 AND 1,2,6,4,5 → segment list: 8

# Strict route selection avoiding ECMPs



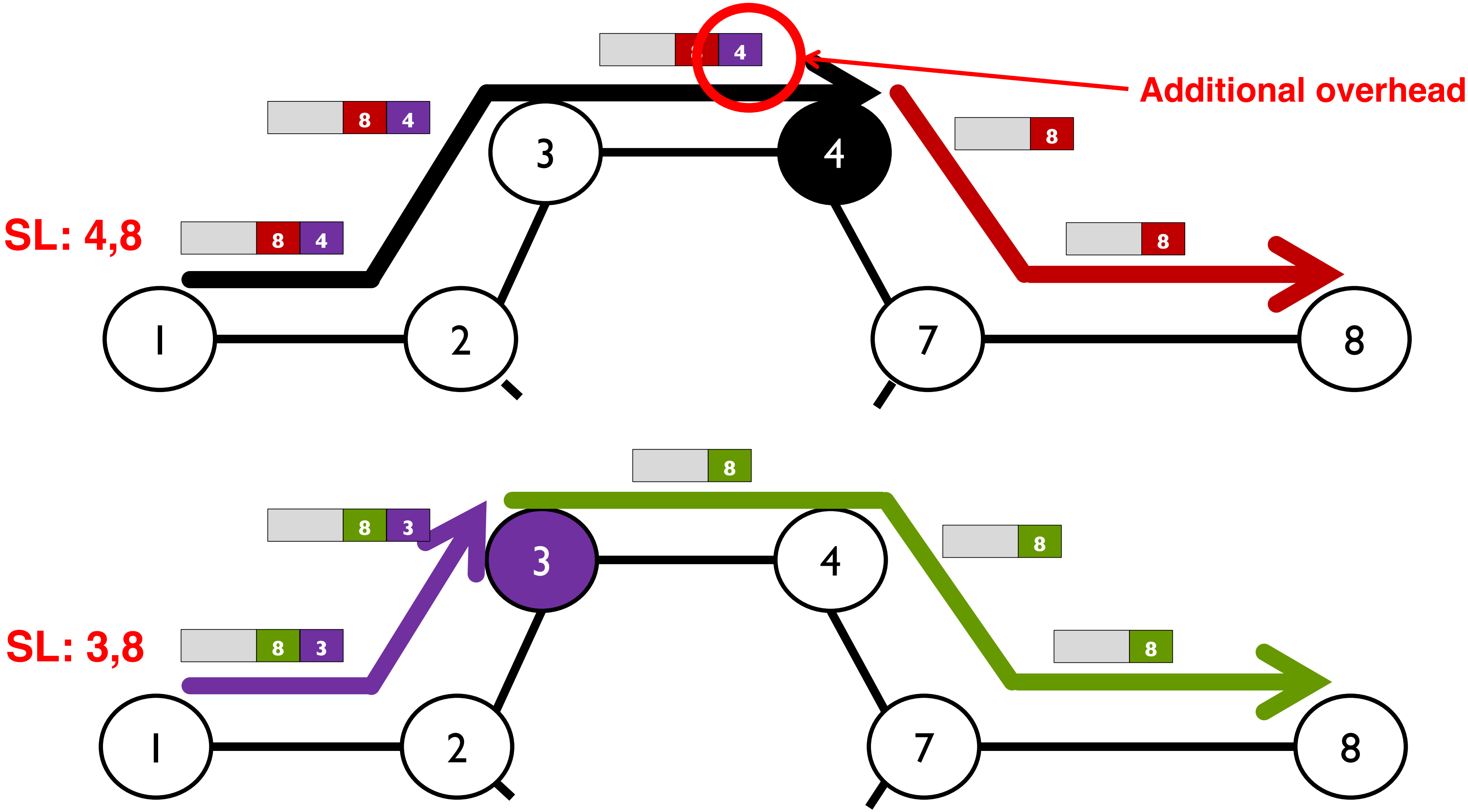

Target path: 1,2,3,4,5,7,8 → segment list: 3, 8

# Path encoding: Segment List with minimum SLD



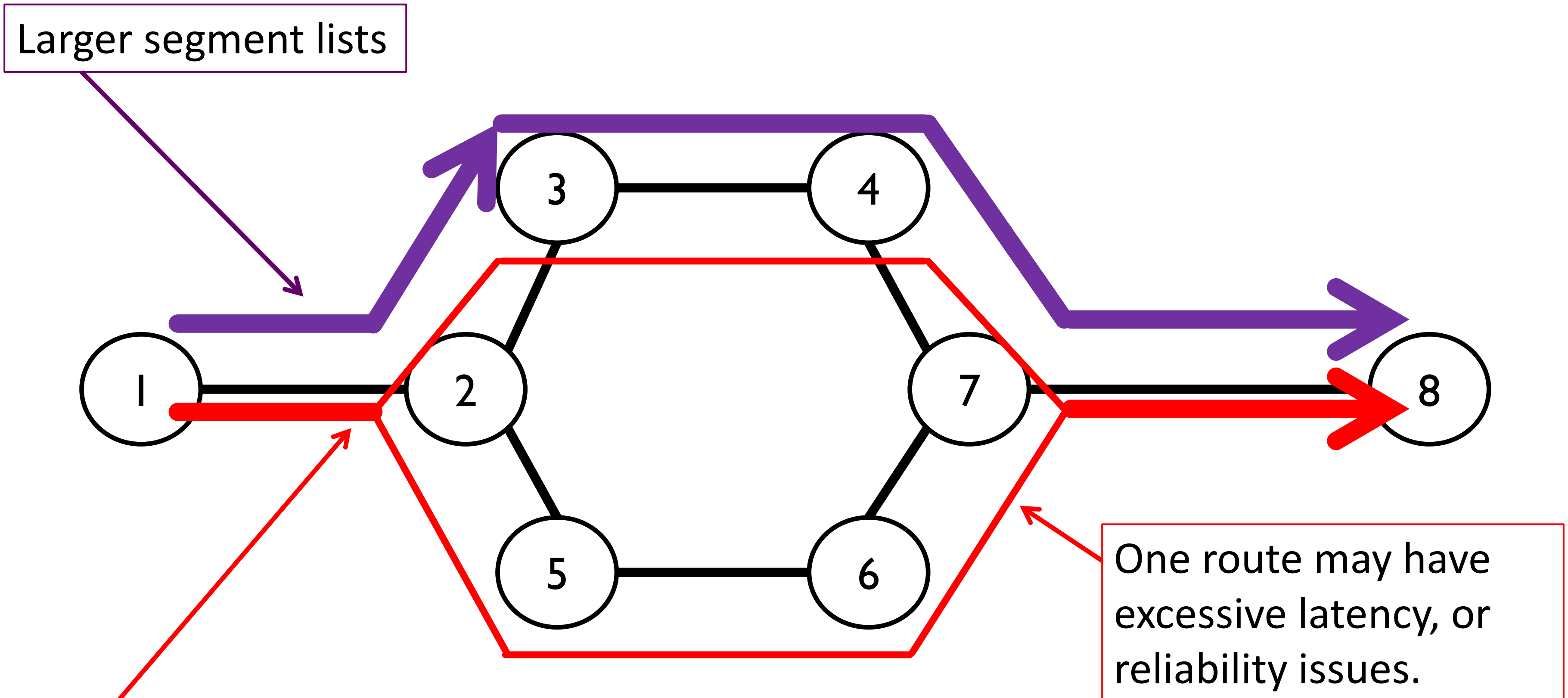

- **Problem**: given a strict route, identify the segment list having the minimum Segment List Depth (SLD).
- Routers may **not** support large SLD

# Path encoding: Segment List with minimum overhead




A Giorgetti, P Castoldi, F Cugini, J Nijhof, F Lazzeri, G Bruno, “Path encoding in segment routing”, GLOBECOM 2015

# ECMP vs. Strict route




In ECMP, Deep Packet Inspection (DPI) is needed to perform load balancing on a per-flow basis, avoiding packet misordering
→ additional HW requirements, not needed in case of strict routes

# SR Traffic engineering performance

- Network usage for different topologies
- Default SR behavior exploiting ECMP typically leads to inefficient network usage
- Segment Routing with large Segment List Depth (e.g., 8) guarantees optimal TE solutions
- Surprisingly, Segment Routing with just a Segment List Depth of 3 is able to guarantee optimal TE solutions

| | ECMP | SHP | SEGMR $\kappa = 8$ | SEGMR $\kappa = 3$ |
|---|---|---|---|---|
| Grid 2 × 2 | 2.00 | 2.00 | 2.00 | 2.00 |
| Grid 3 × 3 | 7.00 | 6.00 | 6.00 | 6.00 |
| Grid 4 × 4 | 18.63 | 16.00 | 16.00 | 16.00 |
| Grid 5 × 5 | 36.75 | 30.00 | 30.00 | 30.00 |
| Eurocore | 4.96 | 4.00 | 4.00 | 4.00 |
| NFSNET | 15.33 | 13.00 | 13.00 | 13.00 |
| EON | 25.00 | 18.00 | 18.00 | 18.00 |
| UKNET | 27.63 | 21.00 | 19.00 | 19.00 |
| ITALNET | 48.38 | 33.00 | 28.00 | 28.00 |
| Arpanet | 35.88 | 33.00 | 33.00 | 33.00 |
| Eurolarge | 131.60 | 88.04 | 66.00 | 66.00 |

Number of flows in the most utilized link

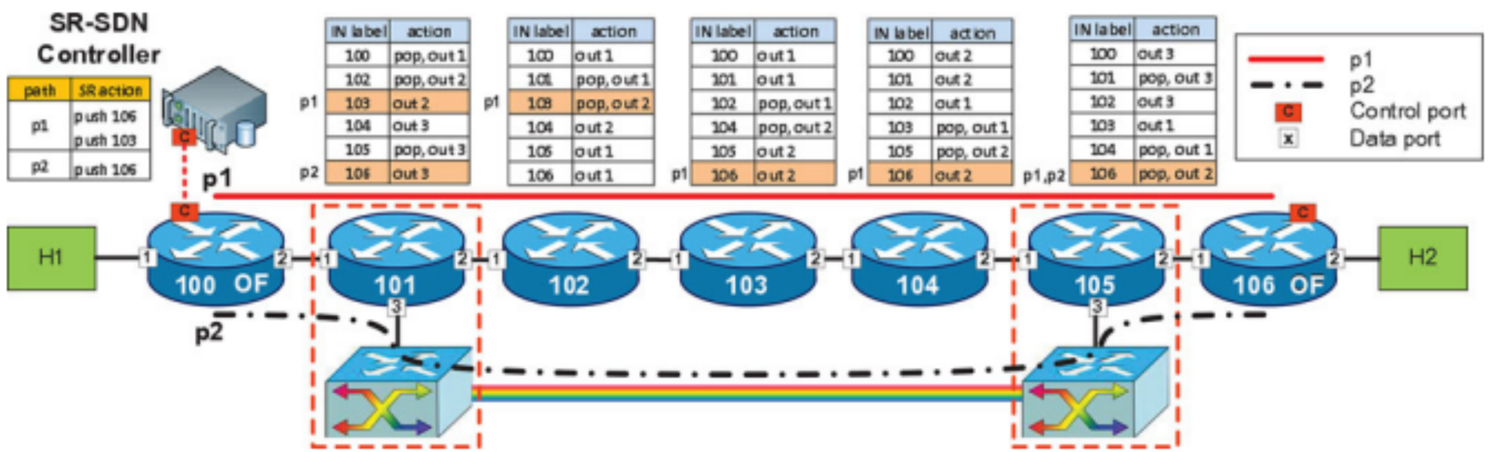
- ECMP: default SR behavior (*k*=1)
- SHP: shortest path, no ECMP
- SEGMR: SR with *k*=8 and *k*=3

*k: segment list depth*

Uniform traffic matrix is assumed. Results are confirmed fr non-uniform matrices

E Moreno, A Beghelli, F Cugini, “Traffic Engineering in Segment Routing Networks”, J. Computer Networks 2017

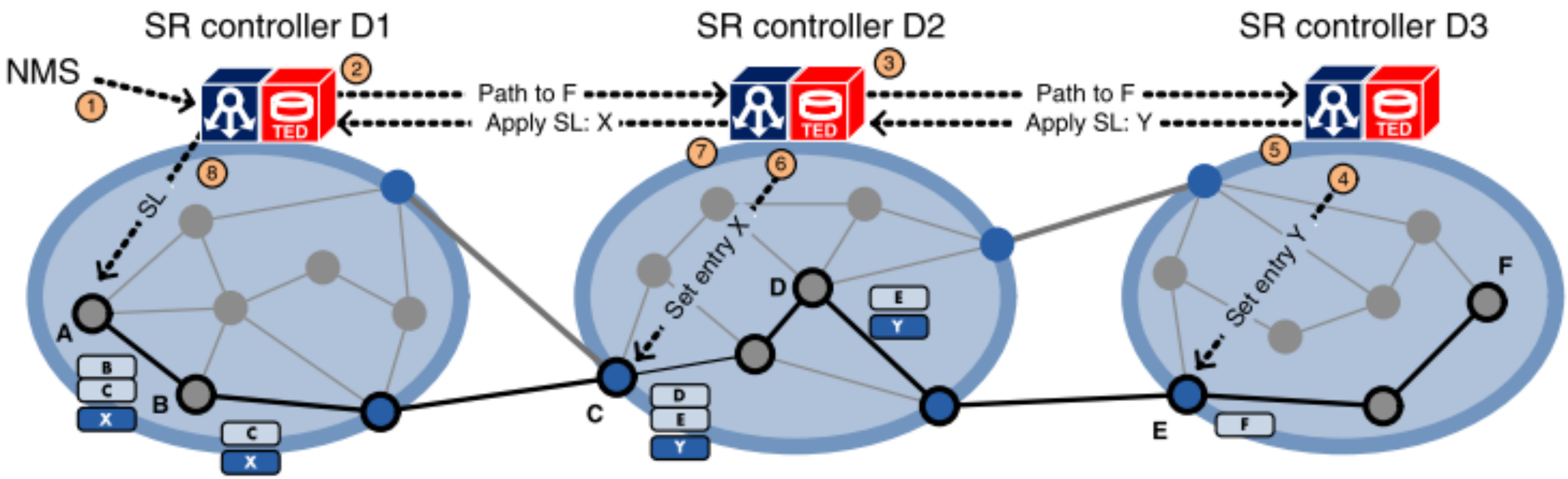
# Use cases for SR: Optical bypass




- Dynamic selection of pre-established optical bypass, with no signaling

A Sgambelluri, F Paolucci, A Giorgetti, F Cugini, P Castoldi
“Experimental demonstration of segment routing”, JLT 2016

# Use cases for SR: multi-domain

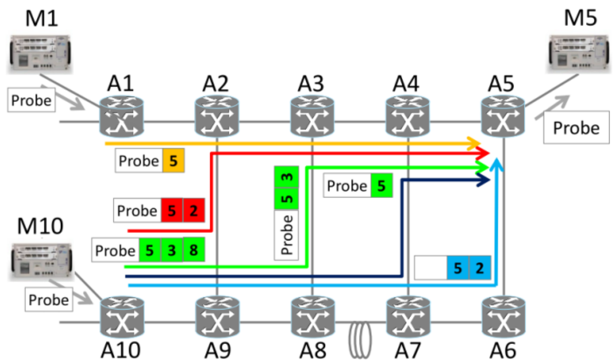



- No e2e signaling (critical in multi-domain/vendor scenarios)
- Compressed Segment Lists to limit label stacking
- Confidentiality

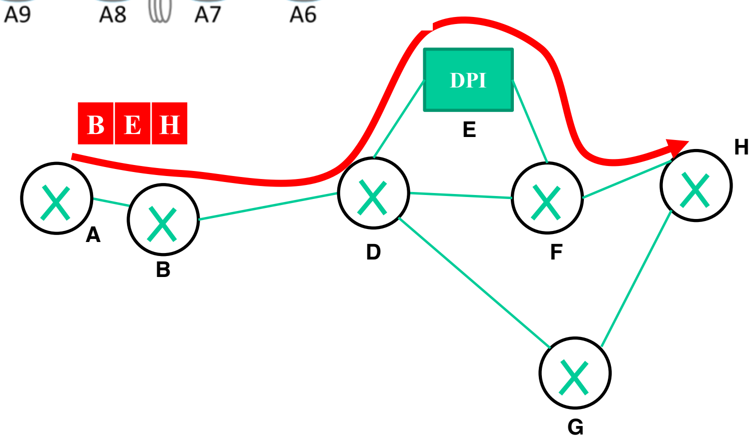
A Giorgetti, A Sgambelluri, F Paolucci, F Cugini, P Castoldi, “Segment routing for effective recovery and multi-domain traffic engineering”, JOCN 2017

# Other use cases for SR

- Recovery:
  - SR-FAILOVER: rerouting to the destination
  - SR-DETOUR: rerouting to the next(-next) hop
- OAM




- Service chaining



# Bit Index Explicit Replication (BIER)

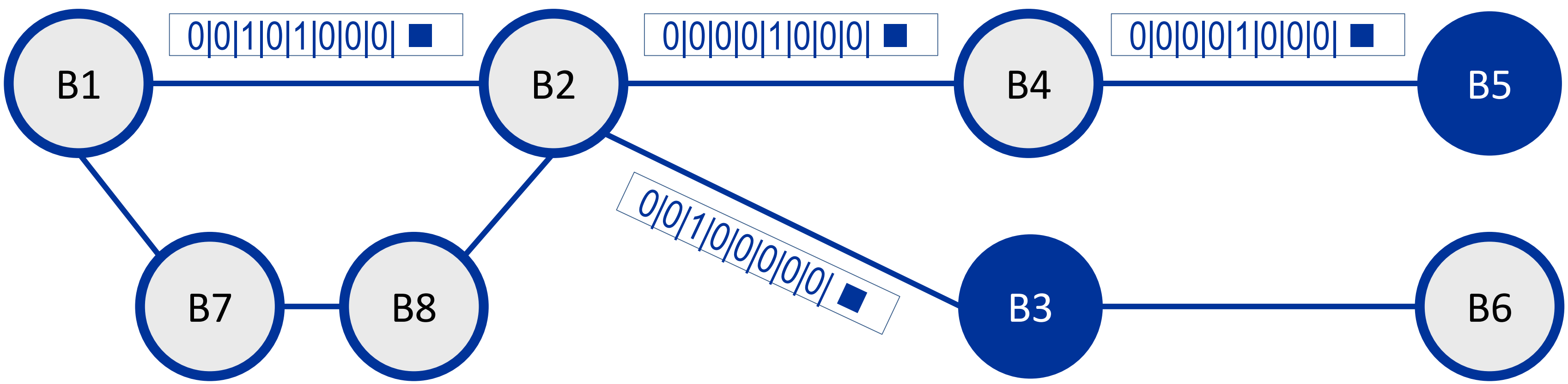
- BIER has been recently proposed for P2MP
- As in SR,
  - No signaling protocol
  - No forwarding state at intermediated nodes
  - the ingress router applies a specifically designed label (here called BitString) which defines the forwarding actions
- In the BIER BitString, each bit represents exactly one egress router in the domain.
- Forwarding is then performed by each intermediate node by just processing and updating the BitString
- In large networks, a hierarchical structure of the BIER header is used.

# BIER
# Basic behavior

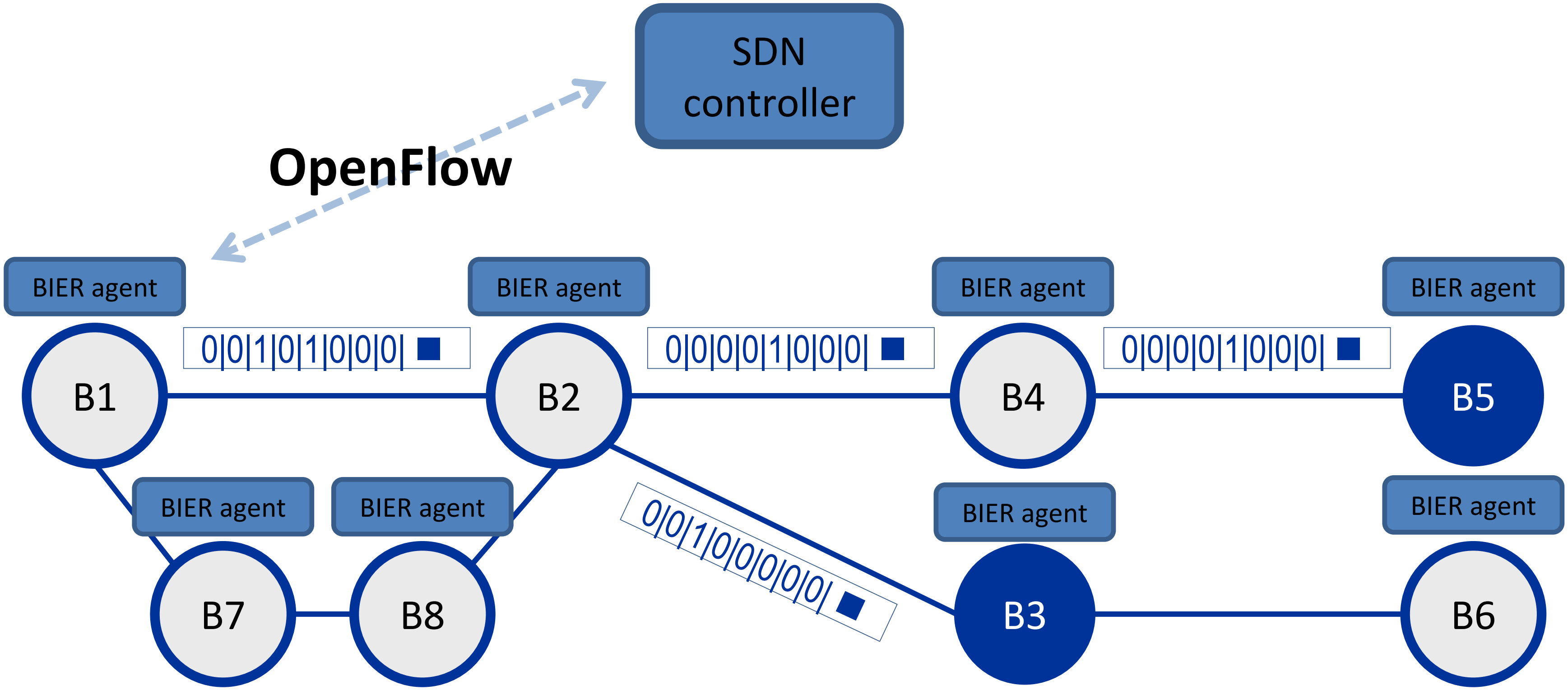
Bit Indexed Forwarding Tables

| BIFT at $B2$ | | | BIFT at $B3$ | | |
|---|---|---|---|---|---|
| BitMask | Neighbour | Port | BitMask | Neighbour | Port |
| 00000001 | $B1$ | 1 | 00000100 | - | local |
| 00000010 | - | local | 11011011 | $B2$ | 1 |
| 00011000 | $B4$ | 2 | 00100000 | $B6$ | 2 |
| 00100100 | $B3$ | 3 | - | - | - |
| 11000000 | $B8$ | 4 | - | - | - |

BIFT table scales with the number of outgoing links and not with the number of flows traversing the node.



# BIER
# Implementation with OF controller




Open vSwitch

# BIER implementation

- SDN network controller implemented in Ryu, with OpenFlow 1.3
- Packet nodes: Open vSwitch + a BIER agent in each node implemented on a local instance of the Ryu controller
- The BIER agent stores the list of nodes reachable with a shortest path using each of the outgoing links (BIFT table - provided and updated by the network SDN controller.
- When a multicast request arrives, the SDN controller only has to enforce the proper BIER header at the ingress (no signaling protocol).
- A group is created with the *id* equal to the numerical value of the BitString including all recipients joining the multicast address.
- If packets need to be replicated at the specific node, the group will include a number of buckets.

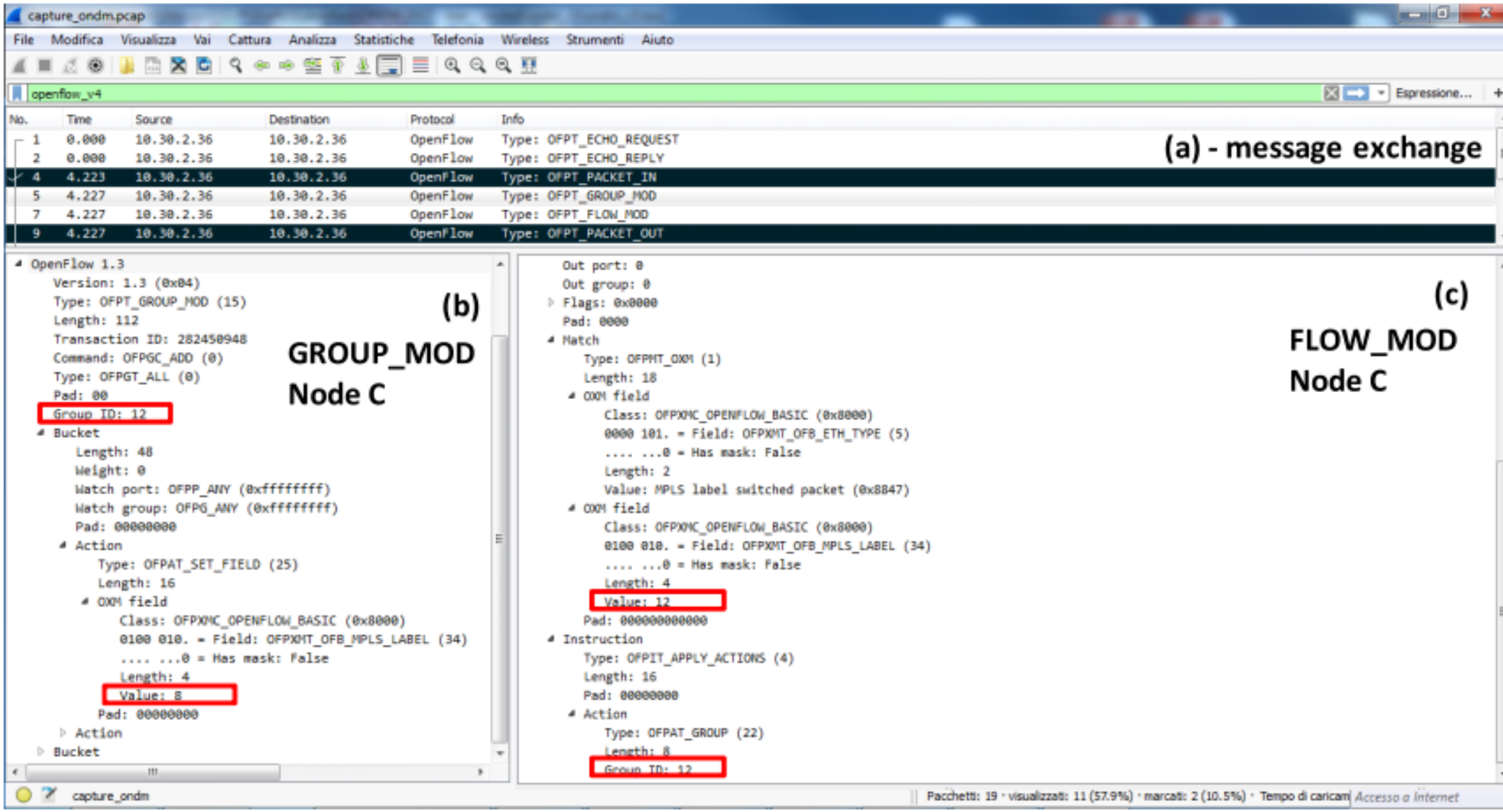
**Node 2: Flow Table**

| Flow match | Action |
|---|---|
| MPLS label 10100 | group 10100 |
| MPLS label 10010 | group 10010 |

**Node 2: Group Table**

| Group # | Action |
|---|---|
| 10100 | Bucket 1: set label 00100, output port 3 |
| | Bucket 2: set label 10000, output port 2 |
| 10010 | Bucket 1: pop label, port local |
| | Bucket 2: set label 10000, output port 2 |

# Capture


capture_ondm.pcap
File Modifica Visualizza Vai Cattura Analizza Statistiche Telefonia Wireless Strumenti Aiuto
openflow_v4
Espressione...
No. Time Source Destination Protocol Info
1 0.000 10.30.2.36 10.30.2.36 OpenFlow Type: OFPT_ECHO_REQUEST
2 0.000 10.30.2.36 10.30.2.36 OpenFlow Type: OFPT_ECHO_REPLY
4 4.223 10.30.2.36 10.30.2.36 OpenFlow Type: OFPT_PACKET_IN
5 4.227 10.30.2.36 10.30.2.36 OpenFlow Type: OFPT_GROUP_MOD
7 4.227 10.30.2.36 10.30.2.36 OpenFlow Type: OFPT_FLOW_MOD
9 4.227 10.30.2.36 10.30.2.36 OpenFlow Type: OFPT_PACKET_OUT
(a) - message exchange
OpenFlow 1.3
Version: 1.3 (0x04)
Type: OFPT_GROUP_MOD (15)
Length: 112
Transaction ID: 282450948
Command: OFPGC_ADD (0)
Type: OFPGT_ALL (0)
Pad: 00
Group ID: 12
Bucket
Length: 48
Weight: 0
Watch port: OFPP_ANY (0xffffffff)
Watch group: OFPG_ANY (0xffffffff)
Pad: 00000000
Action
Type: OFPAT_SET_FIELD (25)
Length: 16
OXM field
Class: OFPXMC_OPENFLOW_BASIC (0x8000)
0100 010. = Field: OFPXMT_OFB_MPLS_LABEL (34)
.... ...0 = Has mask: False
Length: 4
Value: 8
Pad: 00000000
Action
Bucket
(b)
GROUP_MOD
Node C
Out port: 0
Out group: 0
Flags: 0x0000
Pad: 0000
Match
Type: OFPMT_OXM (1)
Length: 18
OXM field
Class: OFPXMC_OPENFLOW_BASIC (0x8000)
0000 101. = Field: OFPXMT_OFB_ETH_TYPE (5)
.... ...0 = Has mask: False
Length: 2
Value: MPLS label switched packet (0x8847)
OXM field
Class: OFPXMC_OPENFLOW_BASIC (0x8000)
0100 010. = Field: OFPXMT_OFB_MPLS_LABEL (34)
.... ...0 = Has mask: False
Length: 4
Value: 12
Pad: 000000000000
Instruction
Type: OFPIT_APPLY_ACTIONS (4)
Length: 16
Pad: 00000000
Action
Type: OFPAT_GROUP (22)
Length: 8
Group ID: 12
(c)
FLOW_MOD
Node C
capture_ondm
Pacchetti: 19 · visualizzati: 11 (57.9%) · marcati: 2 (10.5%) · Tempo di caricam
Accesso a Internet

# Conclusions

- Overview of Segment Routing technology and use cases
  - Path encoding
  - traffic engineering (limited SLD are typically adequate)
  - Multi-domain (no e2e signaling)
  - Recovery (straightforward)
  - OAM (easy mechanism to probe the network)
  - Service chaining (services can be described with Segment ID)
- BIER
  - Experimental demonstration

# thank you!

email:
filippo.cugini@cnit.it