LHC-ONE R&D meeting
October 25th, 2022 // Virtual Presentation

PolKA and SR to support global distributed data-intensive science workflows

Magnos Martinello[2], Cristina Klippel Dominicini[1], Rafael S. Guimarães[1], Everson Borges[2], Moises R. N. Ribeiro[2], Rodolfo Villaça[2], Diego Mafioletti[1], Ana Locateli[2], Edgard Cunha[2], and Isis Oliveira[1]

[1]Federal Institute of Espírito Santo, [2]Federal University of Espírito Santo
Contact: magnos.martinello@ufes.br

INSTITUTO FEDERAL
Espírito Santo

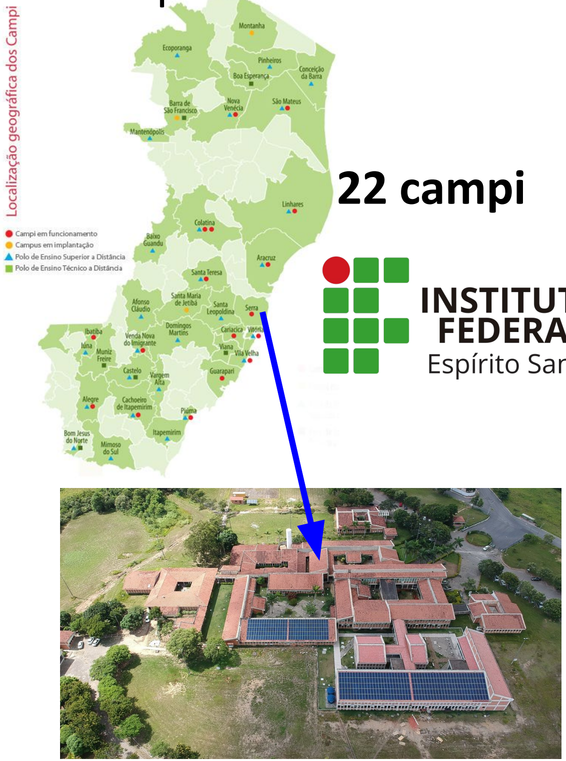
UFES

Location of the Institutions

- **Espírito Santo, Brazil**

Brasil
Espírito Santo

- **IFES:** Federal Institute of Espírito Santo



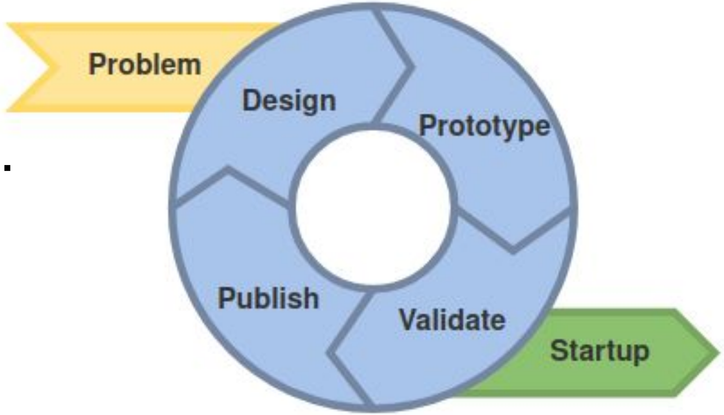
22 campi

- **UFES:** Federal University of Espírito Santo

LabNERDS: Software Defined Networks Research Group

- **Mission:** Innovate in networking systems
- **Areas**: SDN, NFV, autonomous networks, …

Nerds http://nerds.inf.ufes.br

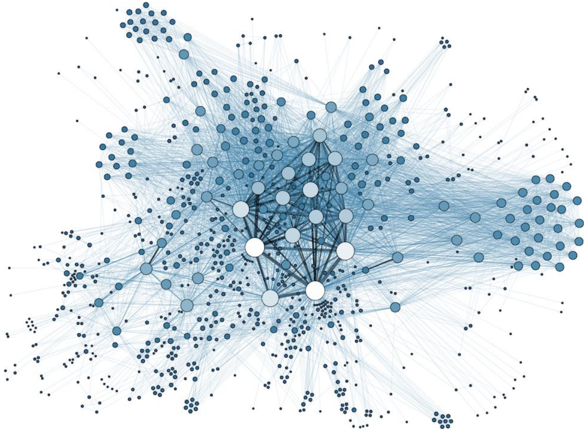


Agenda

- **Motivation**
- Proposal
- Design
- Deployment and experiments in P4 testbeds
- Conclusions and future works

Motivation

- **Scientific Applications**:
  - High-speed WAN networks
  - Transfer massive data & Large number of flows
- **Requirements**:
  - E2E reliability and performance
  - Multiple domains



Motivation

- **Scientific Applications**:
  - High-speed WAN networks
  - Massive data transfer & Large number of flows
  - E2E reliability
  - Multiple domains

- **Table-based forwarding bottlenecks:**
  - Set of shortest paths ⟶ Traffic Engineering 🙁
  - Large number of states ⟶ Scalability 🙁
  - Latency for path configuration ⟶ Agility 🙁

Motivation

- **Scientific Applications**:
  - High-speed WAN networks
  - Massive data transfer & Large number of flows
  - E2E reliability
  - Multiple domains
- **Table-based forwarding bottlenecks:**
  - Set of shortest paths → Traffic Engineering 🙁
  - Large number of states → Scalability 🙁
  - Latency for path configuration → Agility 🙁

Subutilization

Ossification

Endpoints with no control over paths

Bad Congestion Detection/Avoidance

Motivation

- **Scientific Applications**:
  - ○ High-speed WAN networks
  - ○ Massive data transfer & Large number of flows
  - ○ E2E reliability
  - ○ Multiple domains

- **Table-based forwarding bottlenecks:**
  - ○ Set of shortest paths → Traffic Engineering 🙁
  - ○ Large number of states → Scalability 🙁
  - ○ Latency for path configuration → Agility 🙁

- Alternative: **Source Routing (SR)**
  - ○ A source specifies a path and adds a route label to the packet header.

Subutilization

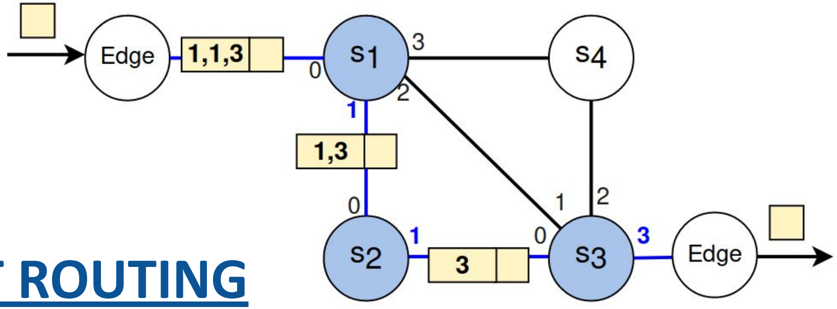
Ossification

No endpoint
control over paths

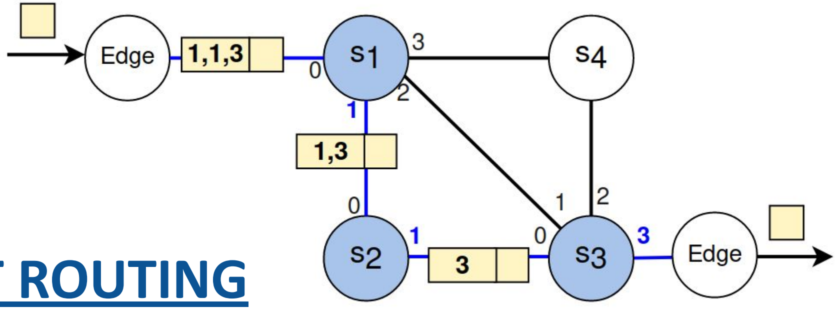
Bad Congestion
Detection/Avoidance

Source Routing (SR)

- Traditional way: **List-based SR (LSR)**
  - Path: a list of ports or addresses.
  - Each node performs a pop.
- Most remarkable protocol: **SEGMENT ROUTING**



Source Routing (SR)



- Traditional way: **List-based SR (LSR)**
  - Path: a list of ports or addresses.
  - Each node performs a pop.
- Most remarkable protocol: SEGMENT ROUTING
- **Limitations** :
  - Expensive equipment & proprietary implementations
  - Still depends on tables in the core nodes (MPLS)
  - Variable-length of headers (and big headers for both SRV4 and SRv6)

    - Limited maximum hops
  - No multicast**https://www.ciscolive.com/c/dam/r/ciscolive/emea/docs/2019/pdf/BRKIPM-2249.pdf*

Agenda

- Motivation
- **Proposal**
- Design
- Deployment and experiments in P4 testbeds
- Conclusions and future works

PolKA Proposal

- A Source Routing approach that meets all the requirements:

open source/ interoperable

no tables in the core

fixed length header

implementable in prog. switches

Ethernet encapsulation

policy-based tunneling

topology agnostic multipath routing

PolKA Proposal

- A Source Routing approach that meets all the requirements:

open source/ interoperable

no tables in the core

fixed length header

implementable in prog. switches

Ethernet encapsulation

policy-based tunneling

topology agnostic multipath routing

- PolKA: Polynomial Key-based Architecture for Source Routing
  - Polynomial Residue Number System (**RNS**) (*Shoup, 2008*)
  - Chinese Remainder Theorem (**CRT**)
  - Packet forwarding based on an arithmetic operation: **remainder of division**

Ok… Why should I use PolKA?

- One good reason…

  … It is easy to setup paths/tunnels!

- Open source implementation in open white-box hardware
  - RARE/FreeRtr

- It also has some interesting properties (not in this presentation)
  - Ex: multicast, failure protection, telemetry…

Agenda

- Motivation
- Proposal
- **Design**
- Deployment and experiments in P4 testbeds
- Conclusions and future works

How does PolKA work?

- PolKA SR relies on three polynomials:
  - **routeID**: a route identifier calculated using the CRT.
  - **nodeID**: to identify each core node.

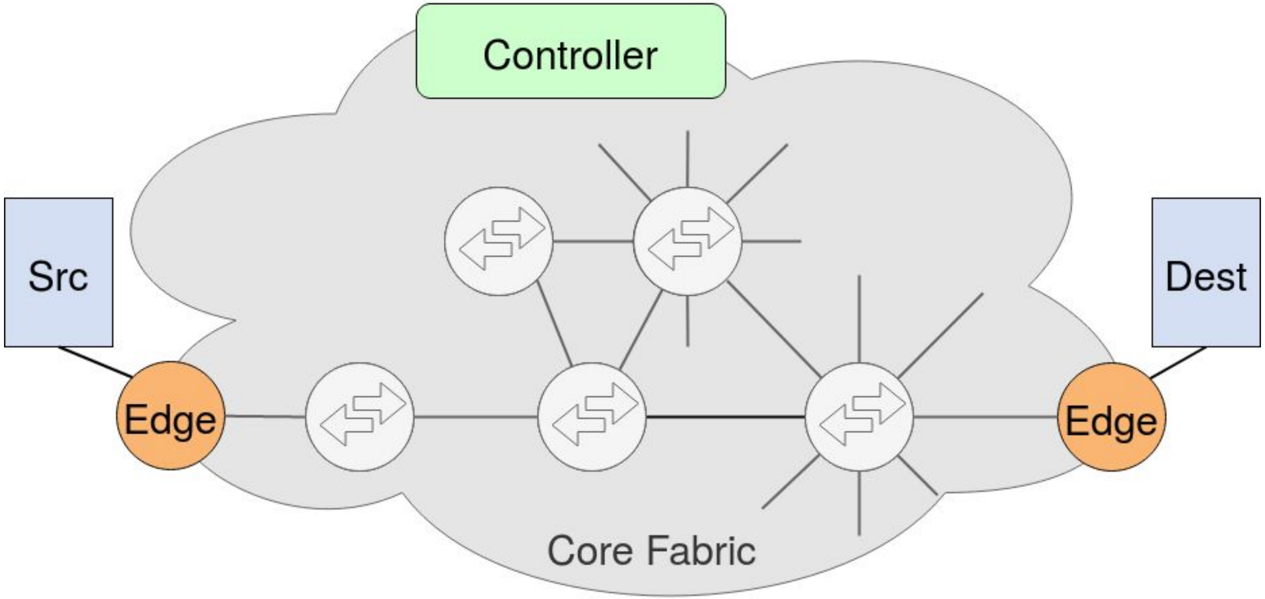
    - Irreducible polynomial
  - **portID**: to identify the ports of each core node.

- The forwarding uses a **mod** operation (remainder of division):

$$\boxed{\textbf{portID} = \langle \textbf{routeID} \rangle_{\textbf{nodeID}}}$$

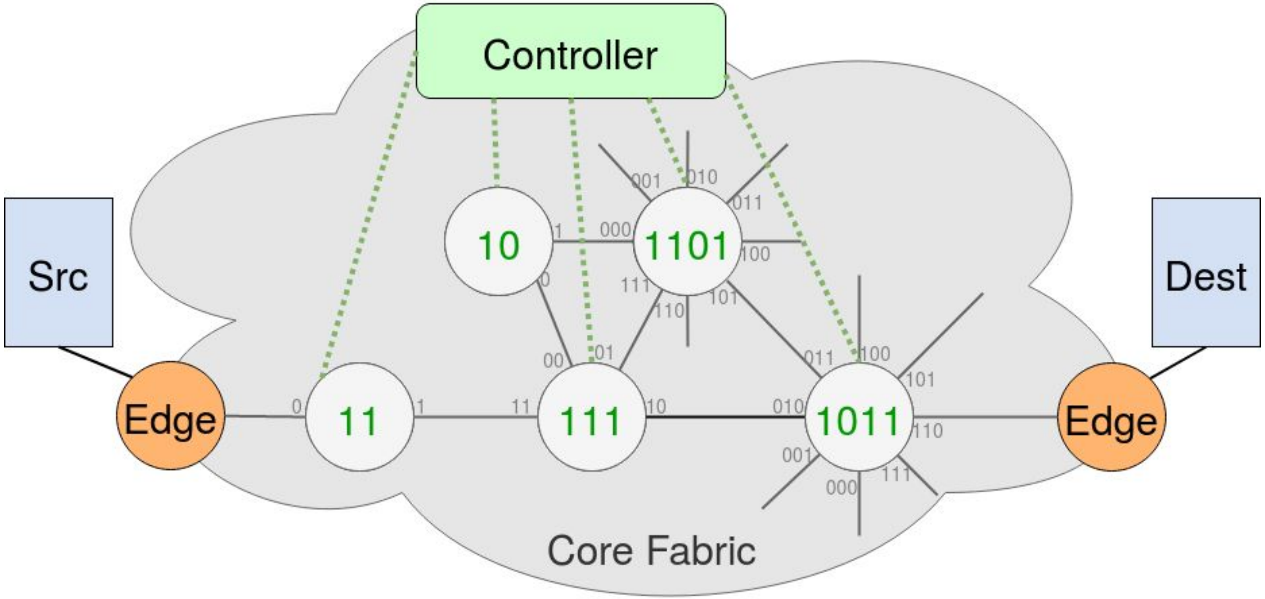
How does PolKA work?

- Hosts are connected to **edge switches.**
- Edges are connected to a fabric of **core switches.**



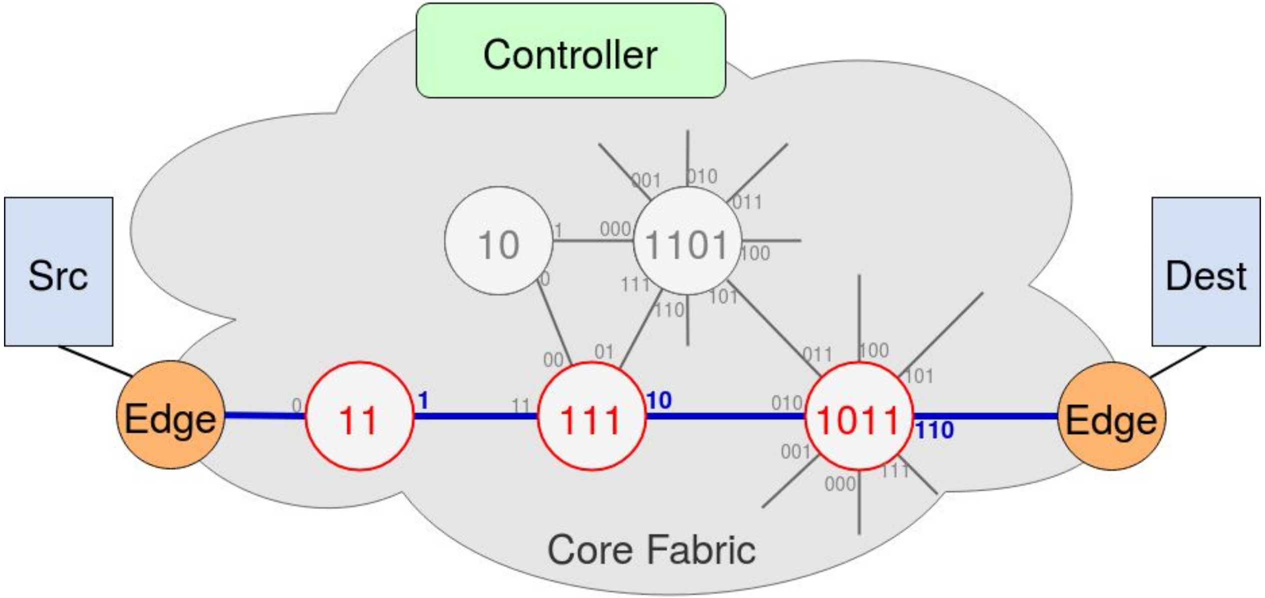
How does PolKA work?

- In a network configuration phase, the **Controller** assigns irreducible polynomials to core switches (***nodeIDs***).
- Port labels are represented as binary polynomials (***portIDs***).



How does PolKA work?

- The **Controller** chooses a **path** for a specific flow (proactively or reactively):
  - A set of switches: {0011,0111,1011}
  - and their output ports: {1 , 10, 110}



How does PolKA work?

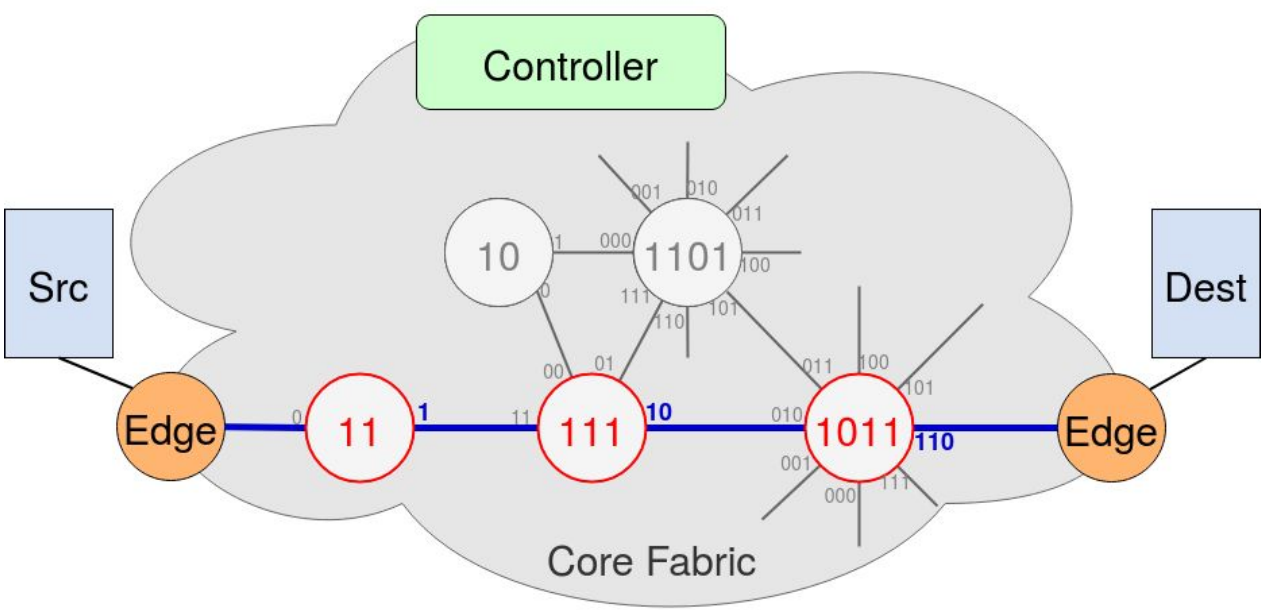
- The **Controller** chooses a **path** for a specific flow:
  - A set of switches: {0011,0111,1011}
  - and their output ports: {1 , 10, 110}

nodeID polynomials

$$s_1(t) = t + 1 = 11$$
$$s_2(t) = t^2 + t + 1 = 111$$
$$s_3(t) = t^3 + t + 1 = 1011$$

portID polynomials

$$o_1(t) = 1$$
$$o_2(t) = t = 10$$
$$o_3(t) = t^2 + t = 110$$



How does PolKA work?

- The **Controller** calculates the ***routeID*** using CRT:
  - Complexity: $\mathcal{O}(len(M)^2)$, where $M(t) = \prod_{i=1}^{N} s_i(t)$

R = 10000

routeID

nodeID polynomials

$$s_1(t) = t + 1 = 11$$
$$s_2(t) = t^2 + t + 1 = 111$$
$$s_3(t) = t^3 + t + 1 = 1011$$

portID polynomials

$$o_1(t) = 1$$
$$o_2(t) = t = 10$$
$$o_3(t) = t^2 + t = 110$$

Calculate routeID with CRT

$$t^4 \equiv 1 \mod (t+1)$$
$$t^4 \equiv t \mod (t^2+t+1)$$
$$t^4 \equiv (t^2+t) \mod (t^3+t+1)$$
$$t^4 = 10000$$

How does PolKA work?

- The **Controller** calculates the ***routeID*** using CRT:
  - Complexity: $\mathcal{O}(len(M)^2)$, where $M(t) = \prod_{i=1}^{N} s_i(t)$

R = 10000

routeID

- Forwarding:

portID = < routeID >$_{\text{nodeID}}$

1 = $<10000>_{0011}$

10 = $<10000>_{0111}$

110 = $<10000>_{1011}$

nodeID polynomials

$$s_1(t) = t + 1 = 11$$
$$s_2(t) = t^2 + t + 1 = 111$$
$$s_3(t) = t^3 + t + 1 = 1011$$

portID polynomials

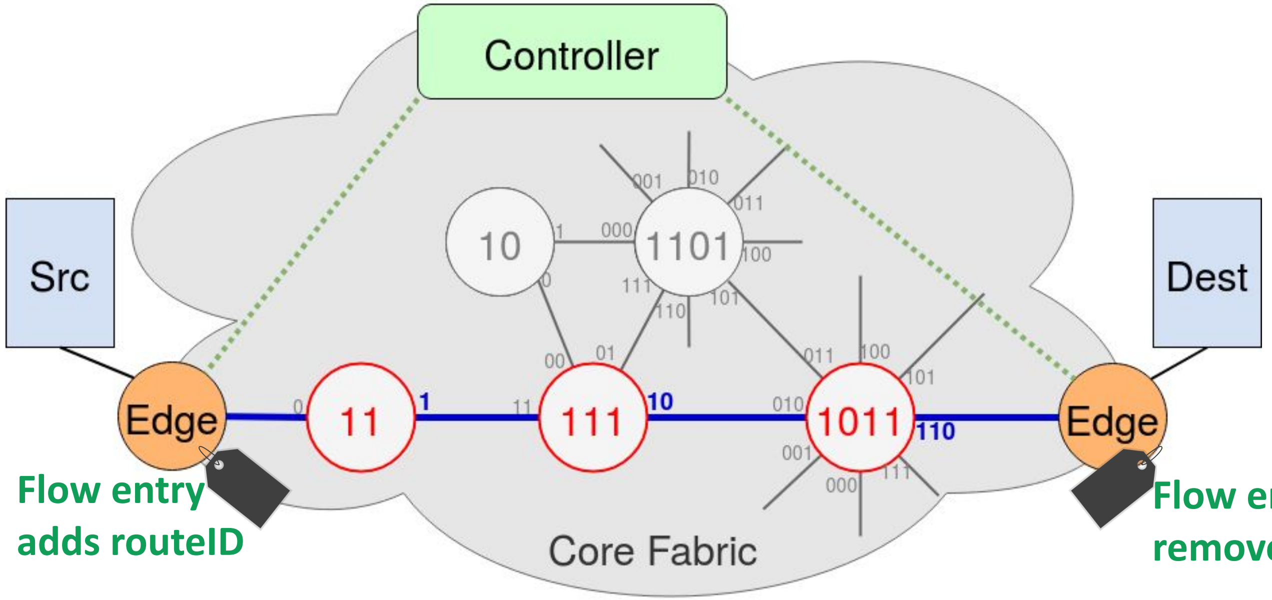
$$o_1(t) = 1$$
$$o_2(t) = t = 10$$
$$o_3(t) = t^2 + t = 110$$

Calculate routeID with CRT

$$t^4 \equiv 1 \mod (t+1)$$
$$t^4 \equiv t \mod (t^2+t+1)$$
$$t^4 \equiv (t^2+t) \mod (t^3+t+1)$$
$$t^4 = 10000$$

How does PolKA work?

- The **Controller** installs **flow entries** at the edges to add/remove ***routeIDs***.

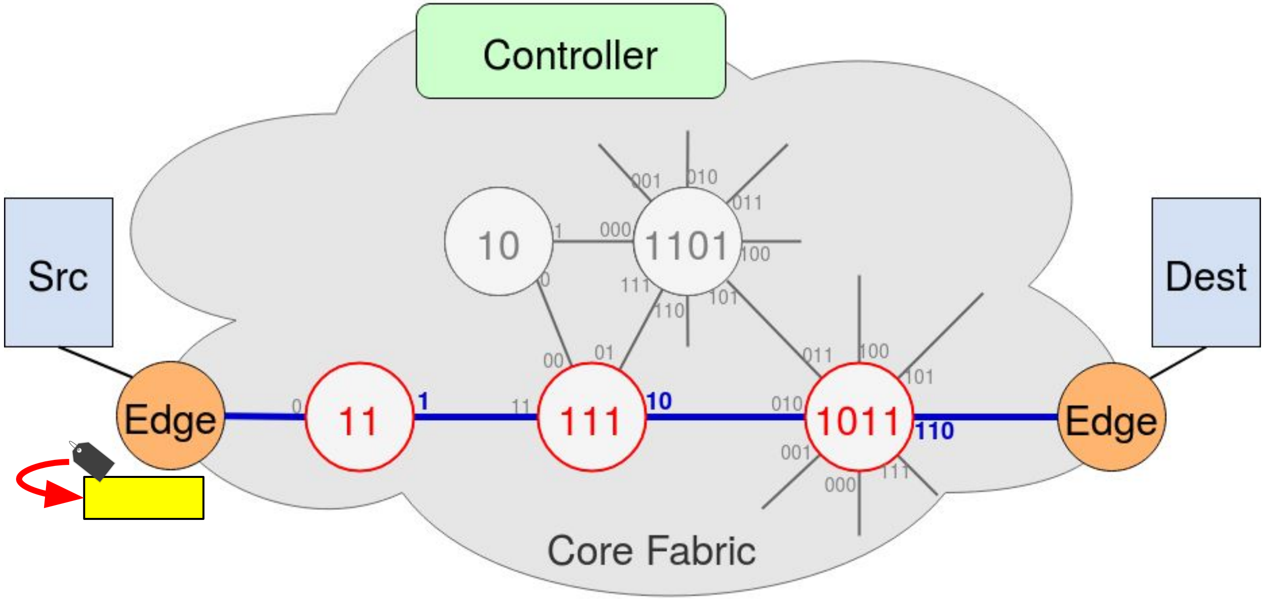


Flow entry adds routeID

Flow entry removes routeID

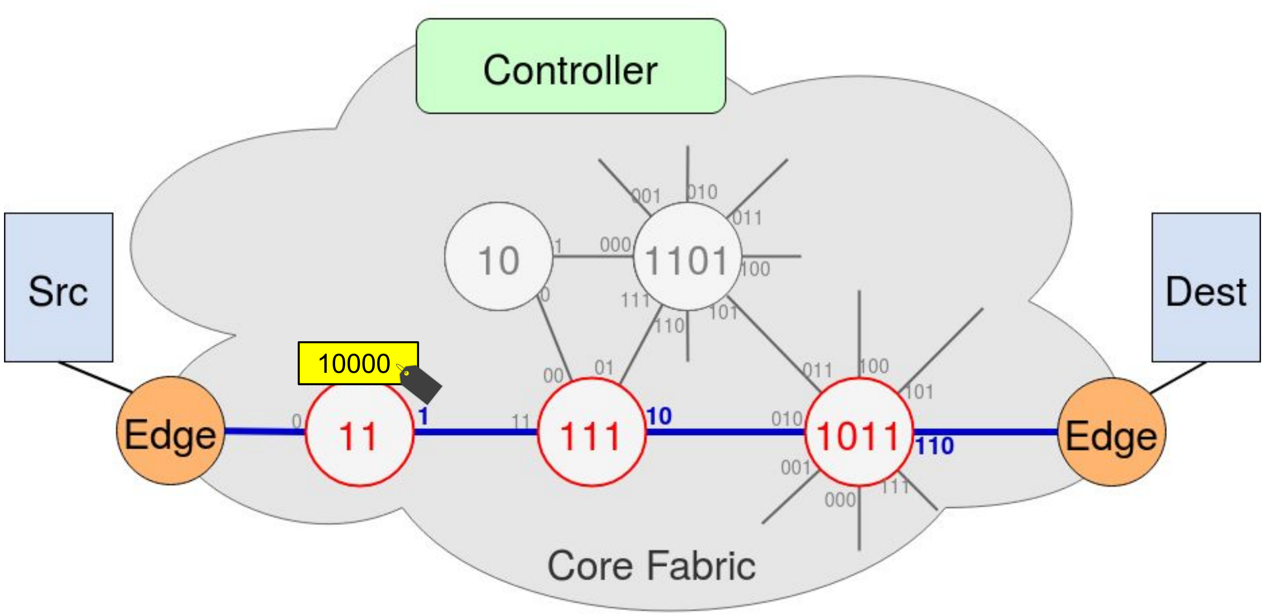
How does PolKA work?

- When packets arrive, an action at ingress embeds *routeID* into the packets.



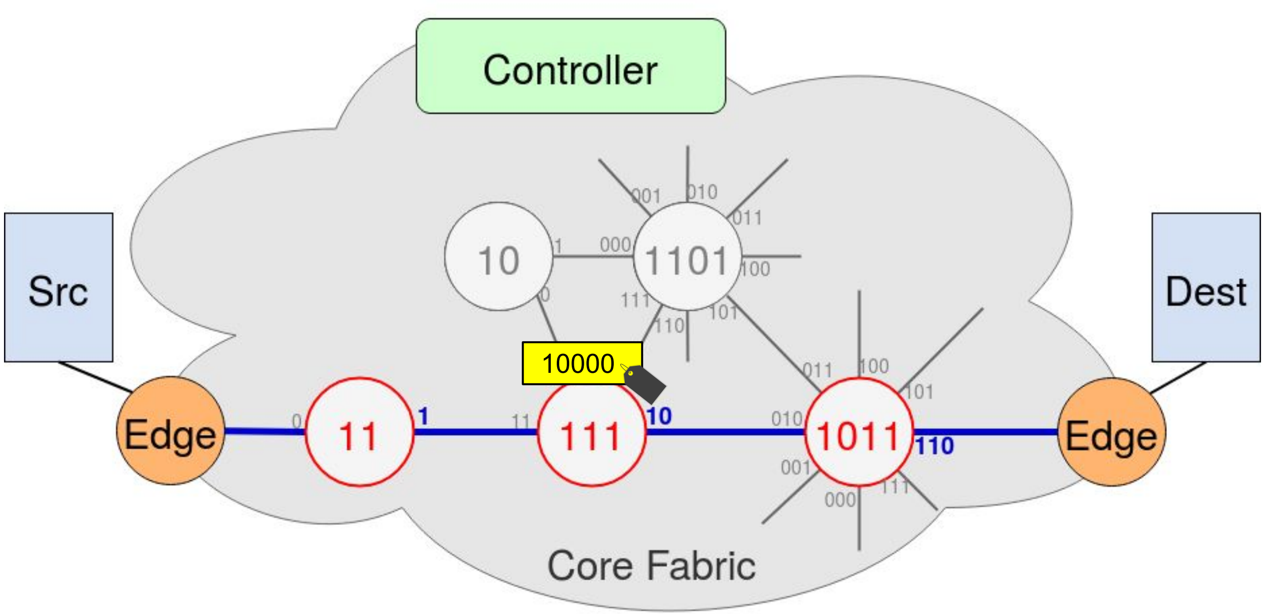
How does PolKA work?

- Forwarding using **mod** operation: $\langle 10000 \rangle_{0011}$ = 1 → output port
- No *routeID* rewrite! No tables!



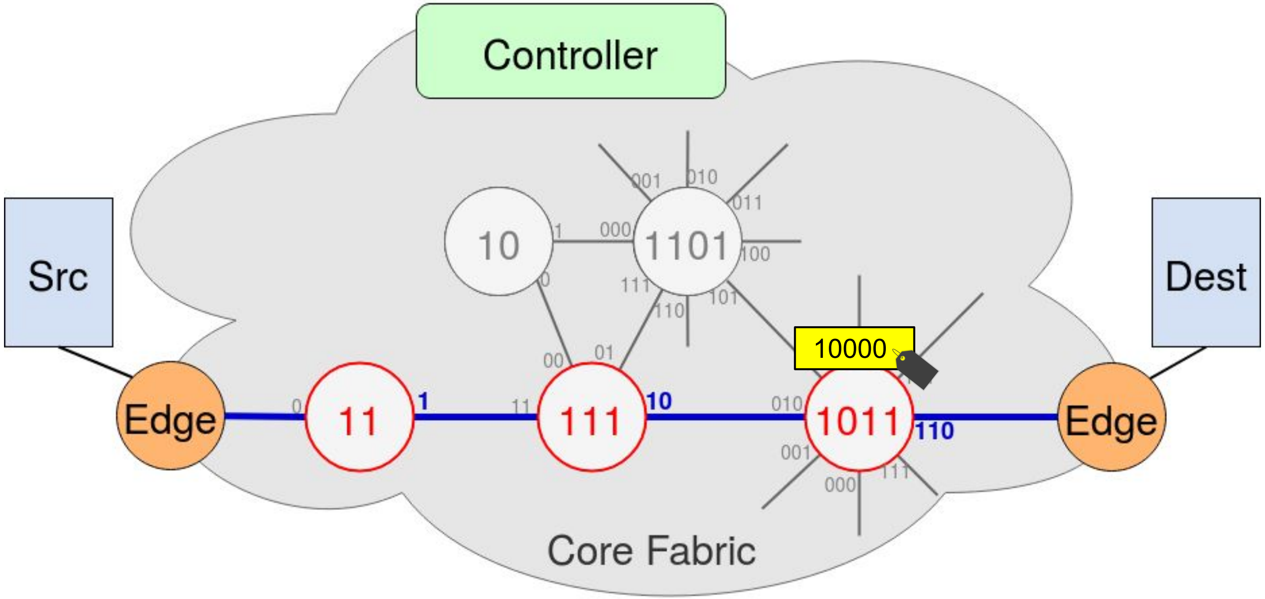
How does PolKA work?

- Forwarding using **mod** operation: $<10000>_{0111}$ = 10 → output port
- No *routeID* rewrite! No tables!



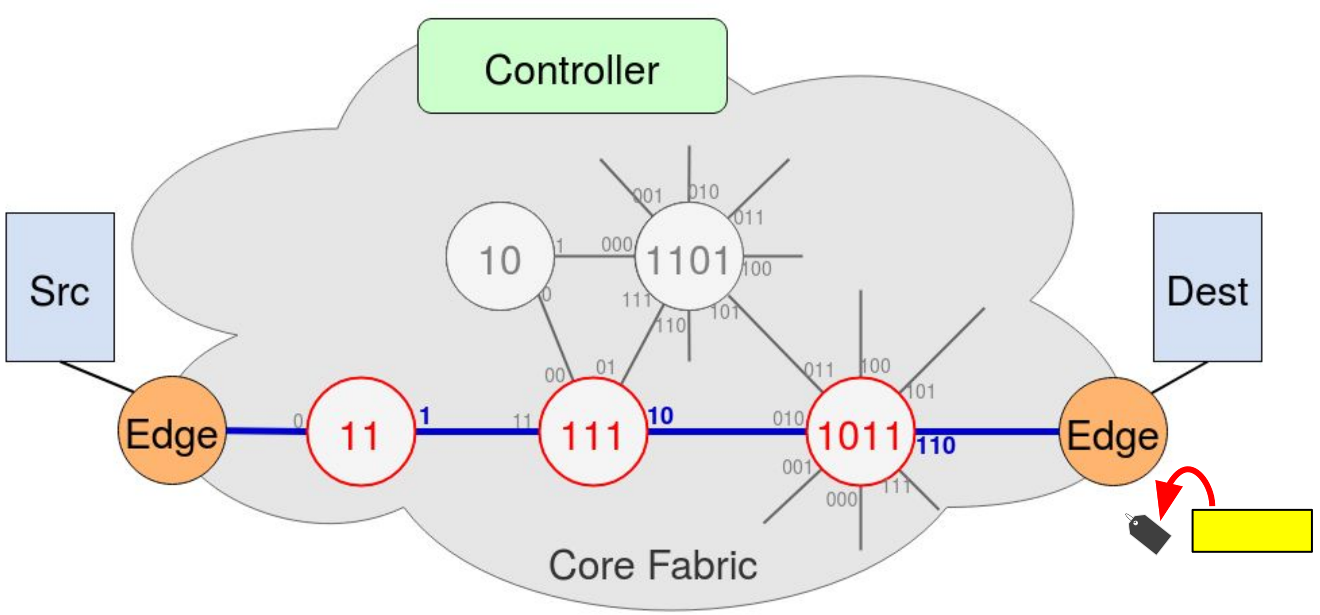
How does PolKA work?

- Forwarding using **mod** operation: $\langle 10000 \rangle_{1011}$ = 110 → output port
- No *routeID* rewrite! No tables!



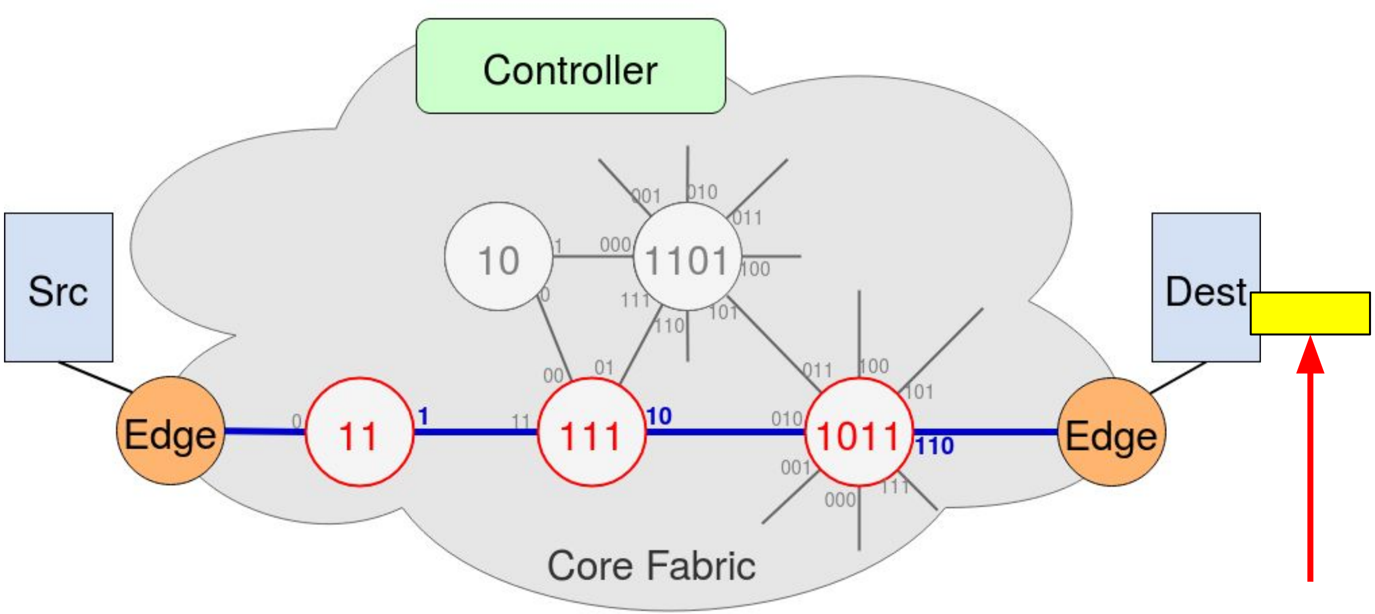
How does PolKA work?

- Finally, an action at edge egress node removes *routeID*.



How does PolKA work?

- Packet is delivered to the application in a transparent manner.



How to implement PolKA’s data plane?

- **P4 language does not natively support the mod operation.**
- **CRC hardware** (Cyclic Redundancy Check) offers polynomial mod.
  - The Tofino Native Architecture (**TNA**) supports **custom** CRC polynomials.

Agenda

- Motivation
- Proposal
- Design
- **Deployment and experiments in P4 testbeds**
- Conclusions and future works

Timeline

PolKA received the 2021 Google Research Scholar Award

M-PolKA received the Intel Connectivity Research Grant (Fast Forward Initiative)

2020

PolKA paper IEEE NetSoft

Novel Polynomial RNS-based SR and reuse of CRC hardware

Emulated prototype in Mininet

2021/1

ONDM paper Deploy @RARE

AMS
POZ
FRA
BUD
GÉANT
RARE

Hardware prototype in Tofino

2021/2

Integration with RARE+FreeRtr

PolKA data & control plane implementation + integration

Emulated prototype in FreeRtr &

Hardware prototype in Tofino w/ FreeRtr control plane

2022

M-PolKA paper IEEE TNSM

Extension to multipath SR for reliable communications

Innovative apps: inband network telemetry, and load balance

PolKA@pangr IETF 113

Lightning Talk Path Aware Networking RG

PolKA@Global P4 Lab

PolKA deployment @Caltech SDN Lab

PolKA Talk at LHC-ONE

PolKA Demo at SC-22

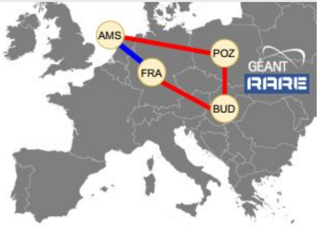
PolKA Deployment in RARE P4 testbed

2020

PolKA paper
IEEE NetSoft

2021/1

ONDM paper
Deploy @RARE



Hardware prototype in Tofino

2021/2

Integration with
RARE+FreeRtr

2022

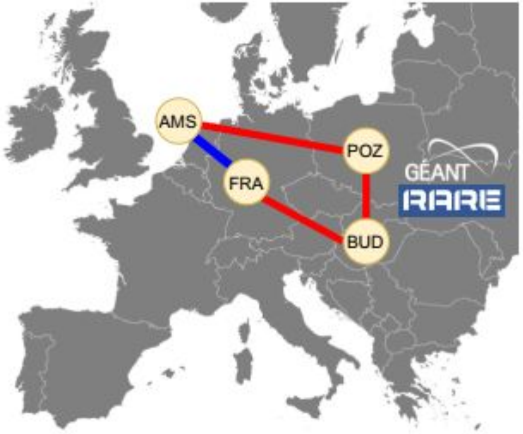
M-PolKA paper
IEEE TNSM

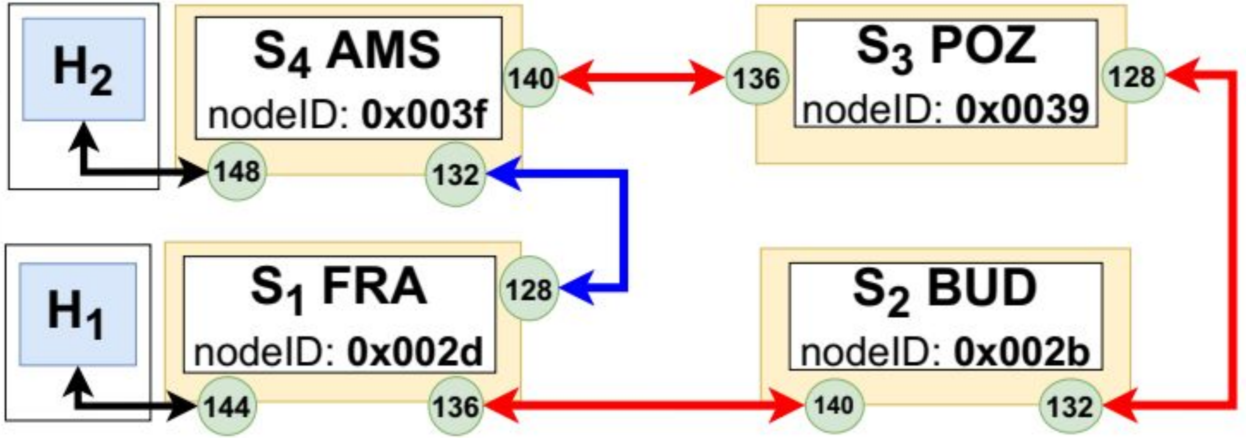
PolKA@pangr
IETF 113

PolKA@Global
P4 Lab

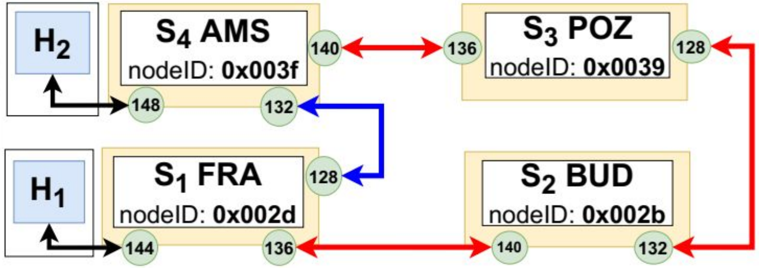
PolKA: Data Plane Prototype

- Deployment: GEANT P4 Lab testbed
  - P4 language and high-performance Tofino switch
- Comparison of PolKA with list-based and table-based approaches
- Results: ONDM 2021 conference paper



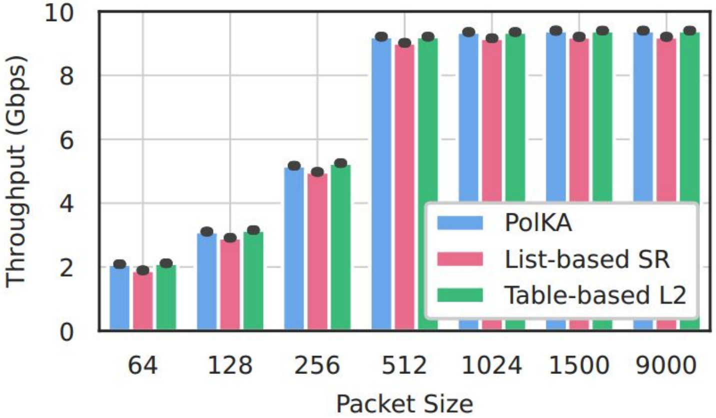


Experiments and Performance results

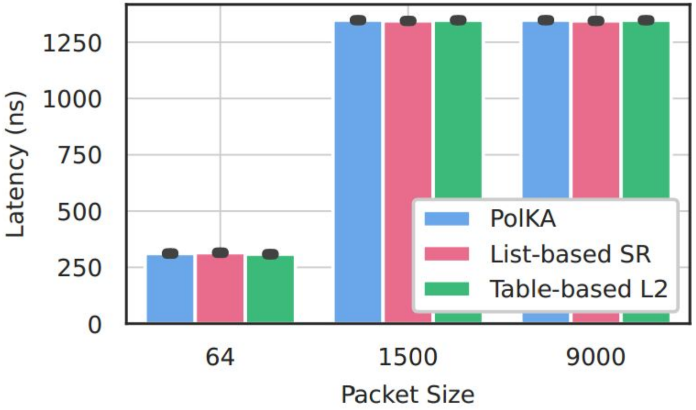


PolKA's performance matches traditional approaches.

- **Throughput** (S1-S2-S3-S4):
  - High throughput and pps rates



- **Forwarding Latency**:
  - Use of hardware timestamps



PolKA Integration with RARE + freeRtr

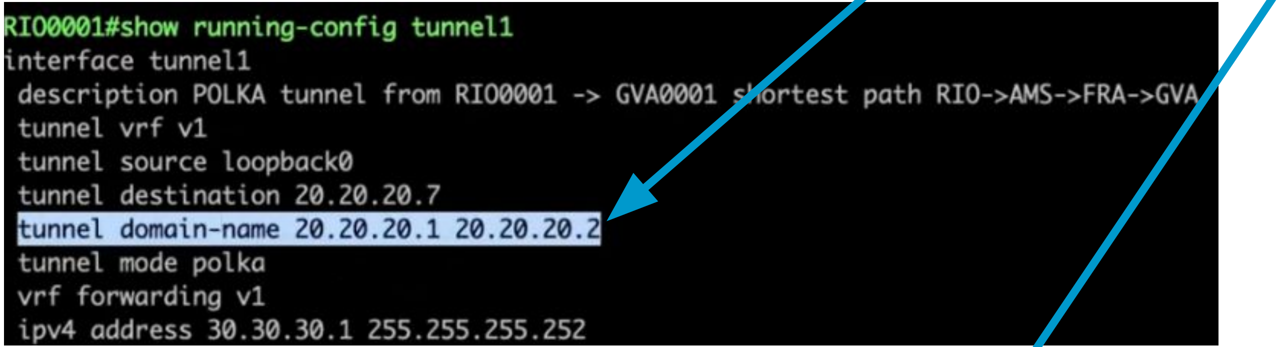
2020	2021/1	2021/2	2022		
PolKA paper IEEE NetSoft	**ONDM paper Deploy @RARE**	**Integration with RARE+FreeRtr** PolKA data & control plane implementation + integration Emulated prototype in FreeRtr & Hardware prototype in Tofino w/ FreeRtr control plane	**M-PolKA paper IEEE TNSM**	**PolKA@pangr IETF 113**	**PolKA@Global P4 Lab**

RARE FreeRtr: Example of PolKA tunnel creation

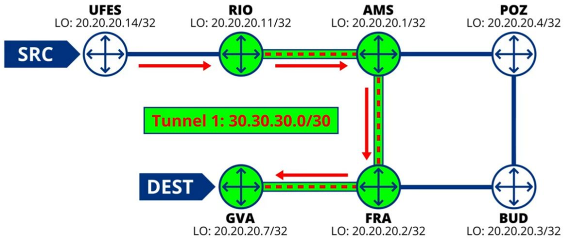
- PolKA tunnel creation

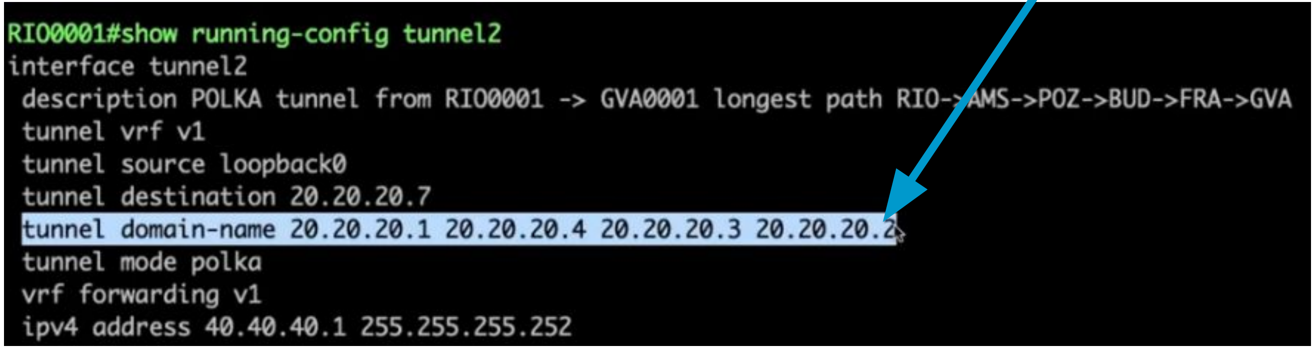
shortest path

longest path

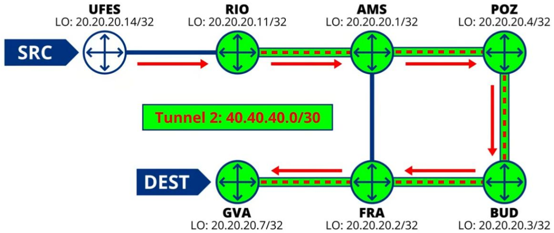


```
RIO0001#show running-config tunnel1
interface tunnel1
 description POLKA tunnel from RIO0001 -> GVA0001 shortest path RIO->AMS->FRA->GVA
 tunnel vrf v1
 tunnel source loopback0
 tunnel destination 20.20.20.7
 tunnel domain-name 20.20.20.1 20.20.20.2
 tunnel mode polka
 vrf forwarding v1
 ipv4 address 30.30.30.1 255.255.255.252
```



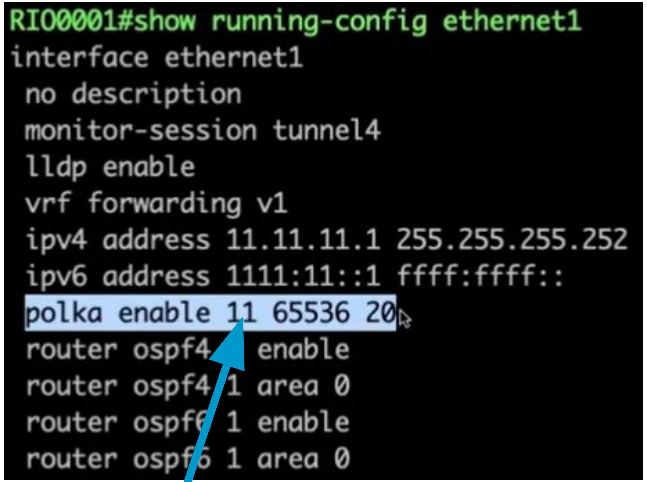


```
RIO0001#show running-config tunnel2
interface tunnel2
 description POLKA tunnel from RIO0001 -> GVA0001 longest path RIO->AMS->POZ->BUD->FRA->GVA
 tunnel vrf v1
 tunnel source loopback0
 tunnel destination 20.20.20.7
 tunnel domain-name 20.20.20.1 20.20.20.4 20.20.20.3 20.20.20.2
 tunnel mode polka
 vrf forwarding v1
 ipv4 address 40.40.40.1 255.255.255.252
```



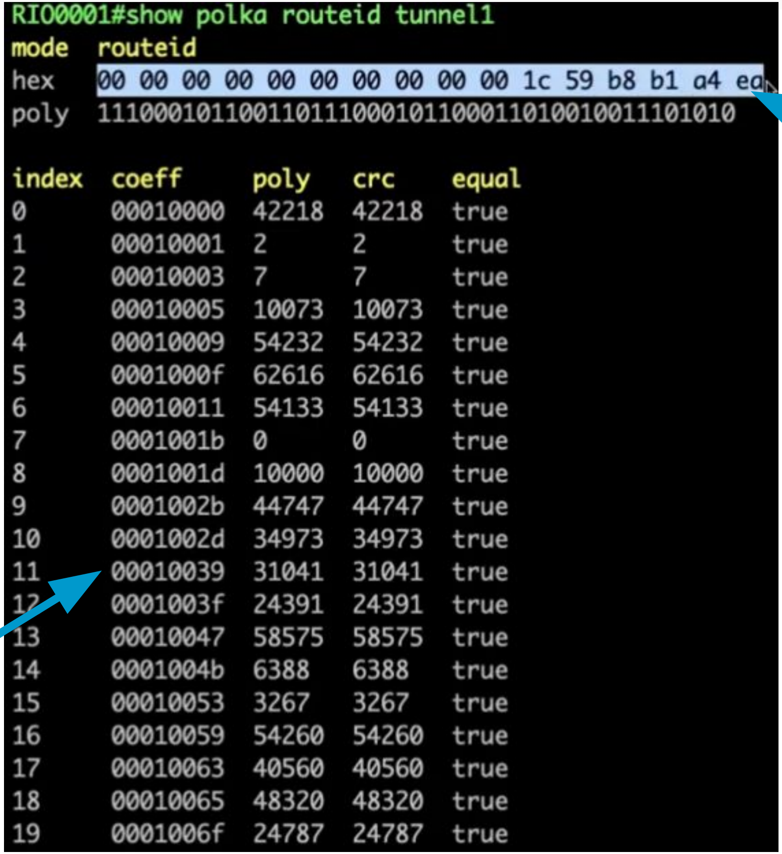
RARE FreeRtr: Example of PolKA configuration

- PolKA configuration



```
RIO0001#show running-config ethernet1
interface ethernet1
 no description
 monitor-session tunnel4
 lldp enable
 vrf forwarding v1
 ipv4 address 11.11.11.1 255.255.255.252
 ipv6 address 1111:11::1 ffff:ffff::
 polka enable 11 65536 20
 router ospf4 1 enable
 router ospf4 1 area 0
 router ospf6 1 enable
 router ospf6 1 area 0
```

node index maps to nodeID polynomial



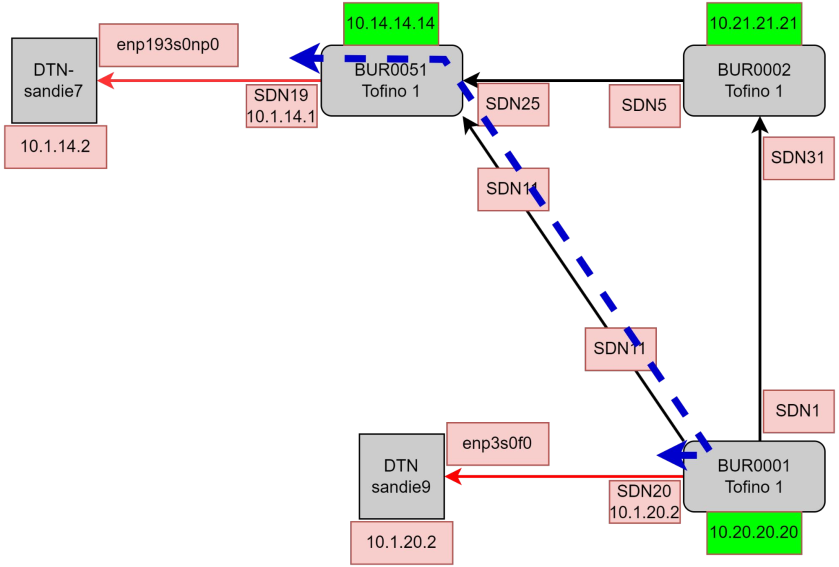
```
RIO0001#show polka routeid tunnel1
mode  routeid
hex   00 00 00 00 00 00 00 00 00 00 1c 59 b8 b1 a4 ea
poly  111000101100110111000101100011010010011101010

index  coeff     poly   crc    equal
0      00010000  42218  42218  true
1      00010001  2      2      true
2      00010003  7      7      true
3      00010005  10073  10073  true
4      00010009  54232  54232  true
5      0001000f  62616  62616  true
6      00010011  54133  54133  true
7      0001001b  0      0      true
8      0001001d  10000  10000  true
9      0001002b  44747  44747  true
10     0001002d  34973  34973  true
11     00010039  31041  31041  true
12     0001003f  24391  24391  true
13     00010047  58575  58575  true
14     0001004b  6388   6388   true
15     00010053  3267   3267   true
16     00010059  54260  54260  true
17     00010063  40560  40560  true
18     00010065  48320  48320  true
19     0001006f  24787  24787  true
```

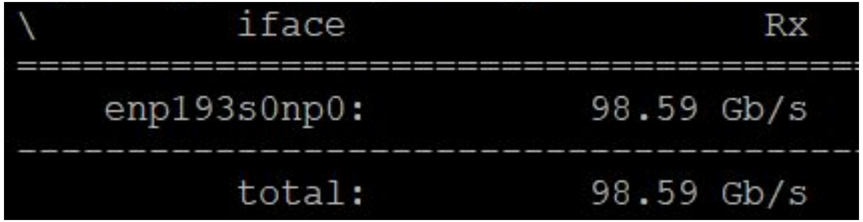
routeID of tunnel1

Timeline

2020	2021/1	2021/2	2022		
PolKA paper IEEE NetSoft	**ONDM paper Deploy @RARE**	**Integration with RARE+FreeRtr**	**M-PolKA paper IEEE TNSM**	**PolKA@pangr IETF 113**	**PolKA@Global P4 Lab** PolKA deployment @Caltech SDN Lab PolKA Talk at LHC-ONE PolKA Demo at SC-22

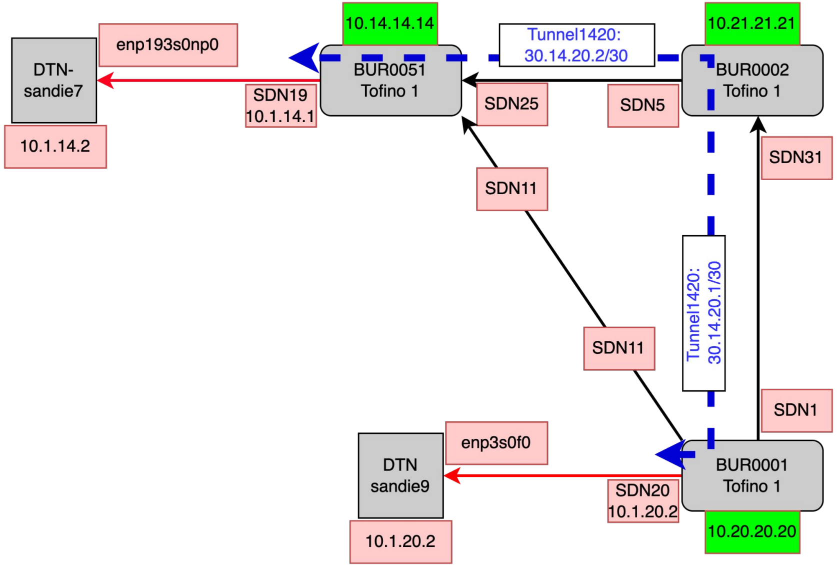
Caltech 100Gbps: Baseline Shortest Path via OSPF

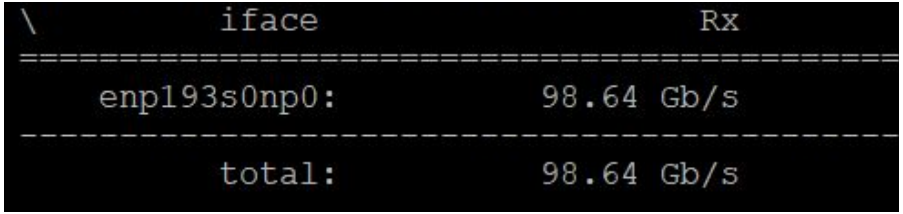




```
\           iface                          Rx
=================================================
     enp193s0np0:                 98.59 Gb/s
-------------------------------------------------
           total:                 98.59 Gb/s
```

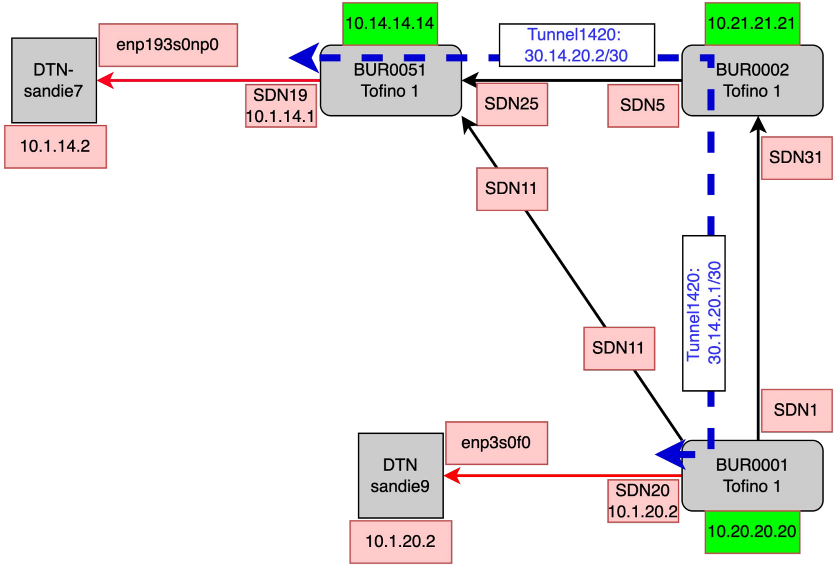
PolKA: 100 Gbps in Caltech Testbed

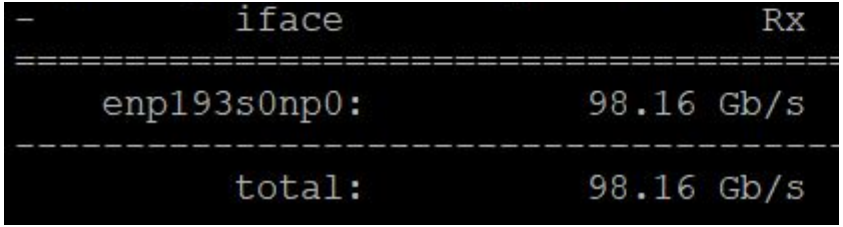




```
\           iface                              Rx
===============================================
     enp193s0np0:               98.64  Gb/s
-----------------------------------------------
           total:               98.64  Gb/s
```

SR MPLS : 100 Gbps in Caltech Testbed

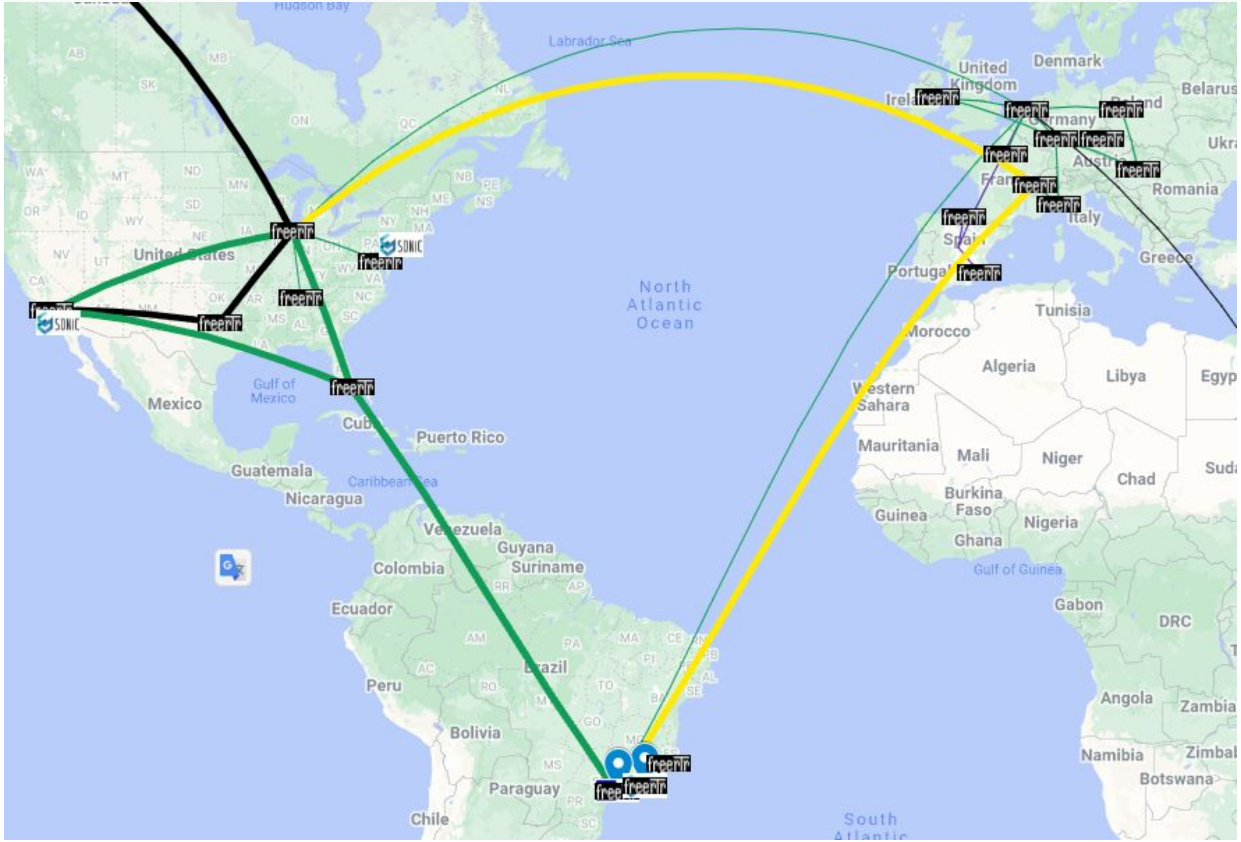




```
-          iface                                  Rx
=============================================================
     enp193s0np0:                    98.16 Gb/s
-------------------------------------------------------------
           total:                    98.16 Gb/s
```

Plan of Experiments for SC 22 in the Global P4 Lab

- **Experiments:**
- Flow Steering exploring PolKA properties
  - Explicit path and TE both at the source

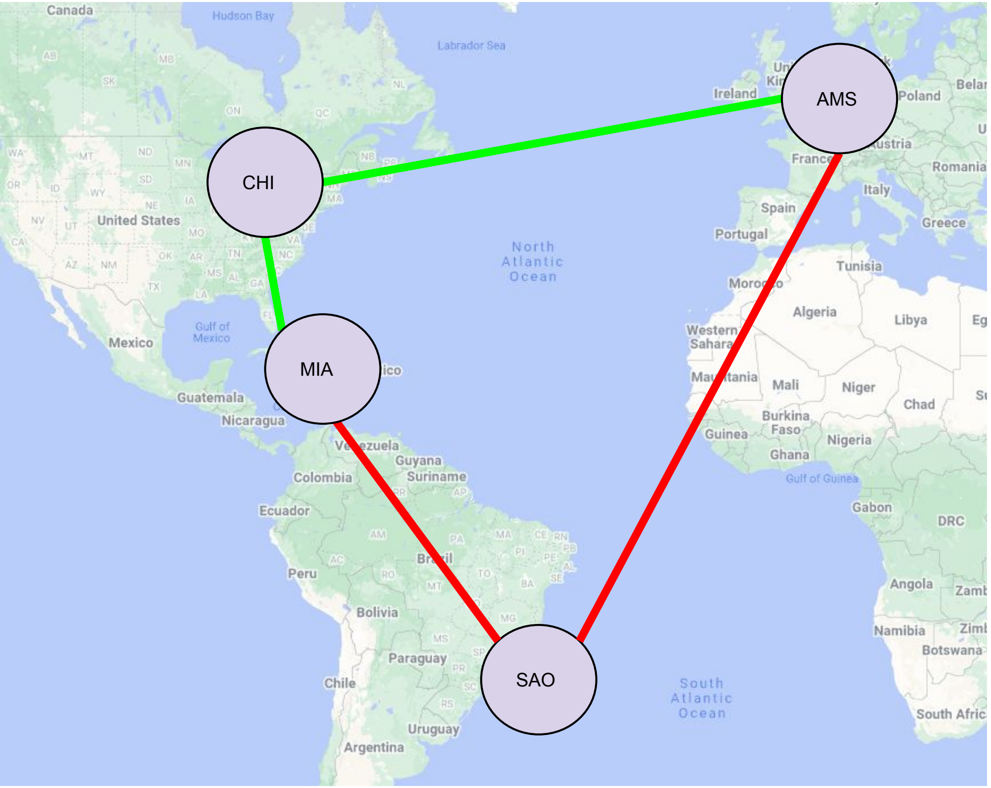


Plan of Experiments for SC 22: Global P4 Lab

- Global P4 Lab testbed :
  - P4 nodes connected via an underlay network.
  - Create an **overlay network with PolKA/SR tunnels**

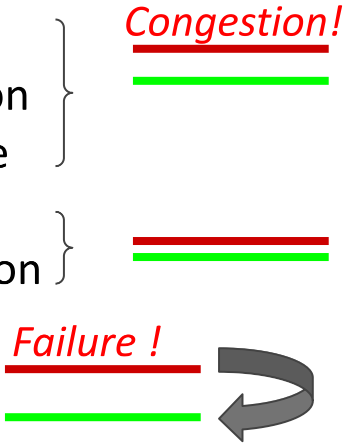
🟥

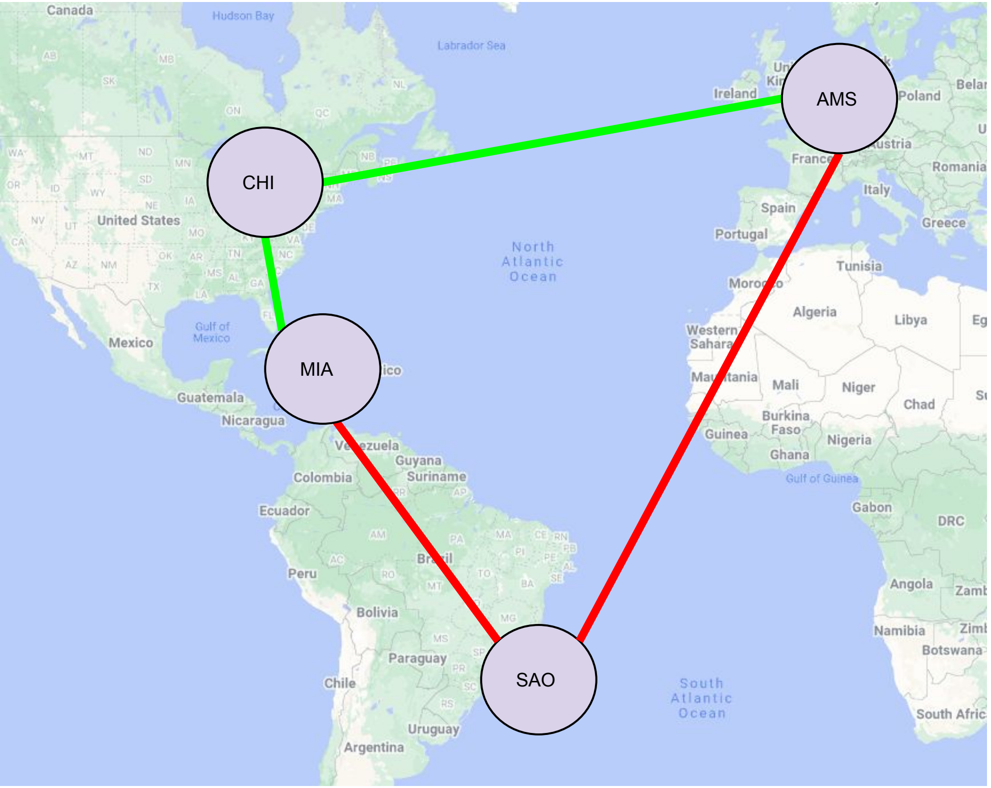
🟩



Plan of Experiments for SC 22: Global P4 Lab

- Experiments:
  - Flow Steering -> explicit path and TE, both at the source
    - Use case: Congestion avoidance
    - Traffic aggregation
  - Agile path reconfiguration





Agenda

- Motivation
- Proposal
- Design
- Deployment and Experiments
- **Conclusions and future works**

Conclusions

- It is **feasible to deploy PolKA in high-performance programmable network equipment** by reusing CRC hardware.
    - PolKA deployment in Caltech P4 lab testbed demonstrated its performance achieving transfer rate of 100 Gbps to multiple aggregated flows (TCP)

- **Easy to configure PolKA tunnels** with a common standard (CLI)

- Potential to support
    - Massive data transfer with aggregation of a large number of flows
    - Big pipes/tunnels configured in a underlay network

Future Perspectives

- Experimentation of SRv6 on Sonic OS after SC22
- In-Network Telemetry (INT) to provide new functionalities, e.g. PoT

PolKA Community, Partners and Collaborations

- **GNA + Caltech:** Harvey Newman, …
    - Internship of Everson in Caltech
- **RARE GÉANT:** Eoin Kenny, Frédéric Loui, Csaba Mate, Simon Leinen, and Jordi Ortiz, …
- **RNP:** Marcos Schwarz
- **Trinity College Dublin (TCD)**: Frank Slyne and Marco Ruffini
- **2021 Google Research Scholar Award (Cristina's PhD thesis)**
- **2022 Intel Connectivity Research Grant (Fast Forward Initiative)**

Selection of Our Recent Publications

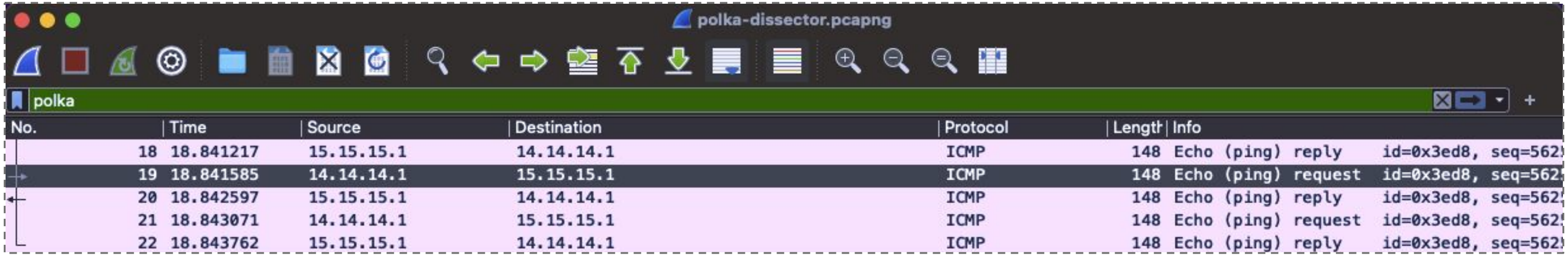
- M-PolKA: Multipath Polynomial Key-based Source Routing for Reliable Communications (IEEE TNSM, 2022)
- Chaining-Box: A Transparent Service Function Chaining Architecture Leveraging BPF (IEEE TNSM, 2021)
- Programmable Switches for in-Networking Classification (IEEE INFOCOM, 2021)
- Deploying PolKA Source Routing in P4 Switches (ONDM, 2021)
- PolKA: Polynomial Key-based Architecture for Source Routing in Network Fabrics (IEEE NetSoft, 2020)
- PlaFFE: A Place-as-you-go In-network Framework for Flexible Embedding of VNFs (IEEE ICC, 2020)
- ProgLab: Programmable labels for QoS provisioning on software defined networks (COMPUTER COMMUNICATION, 2020)
- KeySFC: Traffic steering using strict source routing for dynamic and efficient network orchestration (Computer Networks, 2020)
- FUTEBOL Control Framework: Enabling Experimentation in Convergent Optical, Wireless, and Cloud Infrastructures (IEEE COMMUNICATIONS MAGAZINE, 2019)
- RDNA: Residue-defined networking architecture enabling ultra-reliable low-latency datacenters (IEEE TNSM, 2018)
- VirtPhy: Fully Programmable NFV Orchestration Architecture for Edge Data Centers (IEEE TNSM, 2017)

References

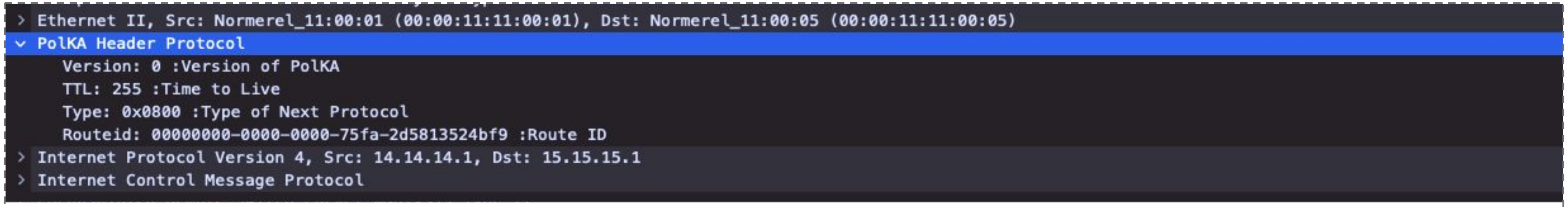
1. Segment Routing RFC
2. PolKA NetSoft 2020 conference paper
3. V. Shoup, A computational introduction to number theory and algebra, 2008.
4. PolKA ONDM 2021 conference paper
5. RARE website
6. FreeRouter website

PolKA: Github

- https://nerds-ufes.github.io/polka/
  - References
  - Tutorials (Mininet and FreeRouter)
  - Wireshark dissector
  - More to come…



```
polka
No.      Time        Source       Destination    Protocol  Length Info
     18  18.841217   15.15.15.1   14.14.14.1     ICMP      148  Echo (ping) reply    id=0x3ed8, seq=562
     19  18.841585   14.14.14.1   15.15.15.1     ICMP      148  Echo (ping) request  id=0x3ed8, seq=562
     20  18.842597   15.15.15.1   14.14.14.1     ICMP      148  Echo (ping) reply    id=0x3ed8, seq=562
     21  18.843071   14.14.14.1   15.15.15.1     ICMP      148  Echo (ping) request  id=0x3ed8, seq=562
     22  18.843762   15.15.15.1   14.14.14.1     ICMP      148  Echo (ping) reply    id=0x3ed8, seq=562
```



```
> Ethernet II, Src: Normerel_11:00:01 (00:00:11:11:00:01), Dst: Normerel_11:00:05 (00:00:11:11:00:05)
v PolKA Header Protocol
     Version: 0 :Version of PolKA
     TTL: 255 :Time to Live
     Type: 0x0800 :Type of Next Protocol
     Routeid: 00000000-0000-0000-75fa-2d5813524bf9 :Route ID
> Internet Protocol Version 4, Src: 14.14.14.1, Dst: 15.15.15.1
> Internet Control Message Protocol
```

Thank you for attention !

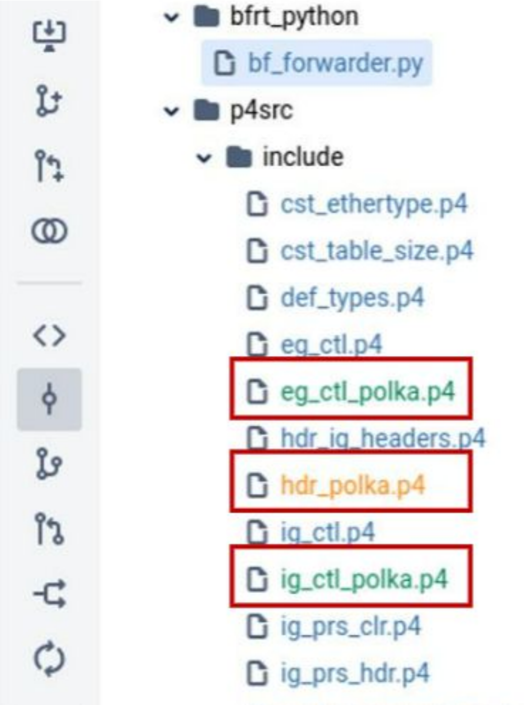
Magnos Martinello

magnos.martinello@ufes.br

** This work was a recipient of the 2021 Google Research Scholar and the 2022 Intel Connectivity Research Grant (Fast Forward Initiative) Awards, and Received funds from CAPES (Finance Code 001), CNPq, FAPESP, FAPES, CTIC, and RNP.*

Integration of PolKA into RARE project

- **PolKA is the first non-standard protocol**
  - Thanks to Csaba Mate, and Frédéric Loui
    https://bitbucket.software.geant.org/projects/RARE/repos/rare/browse/p4src/include
    https://docs.freertr.net/guides/reference/



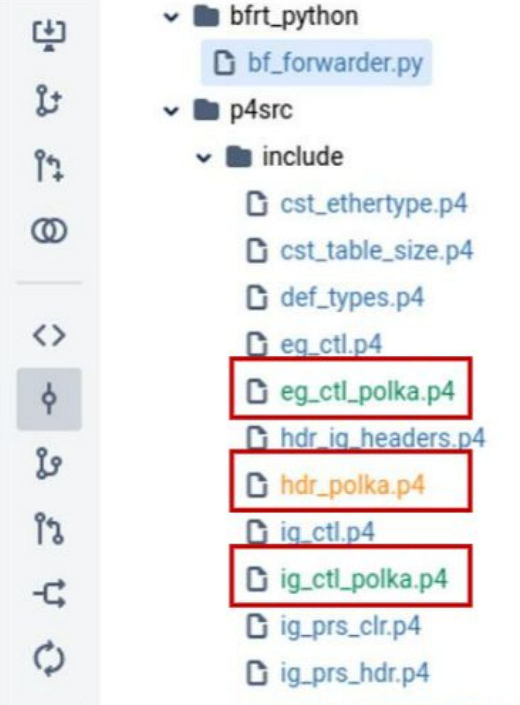
bfrt_python
bf_forwarder.py
p4src
include
cst_ethertype.p4
cst_table_size.p4
def_types.p4
eg_ctl.p4
eg_ctl_polka.p4
hdr_ig_headers.p4
hdr_polka.p4
ig_ctl.p4
ig_ctl_polka.p4
ig_prs_clr.p4
ig_prs_hdr.p4

Integration of PolKA into RARE project

- **PolKA is the first non-standard protocol**
  - Thanks to Csaba Mate, and Frédéric Loui
    https://bitbucket.software.geant.org/projects/RARE/repos/rare/browse/p4src/include
    https://docs.freertr.net/guides/reference/

- How to integrate **PolKA's control plane**?
  - Centralized Controller
  - **Reuse of standard distributed protocols**
    - Topology from link-state routing protocols

- **Fixed-length** PolKA header (various encaps)

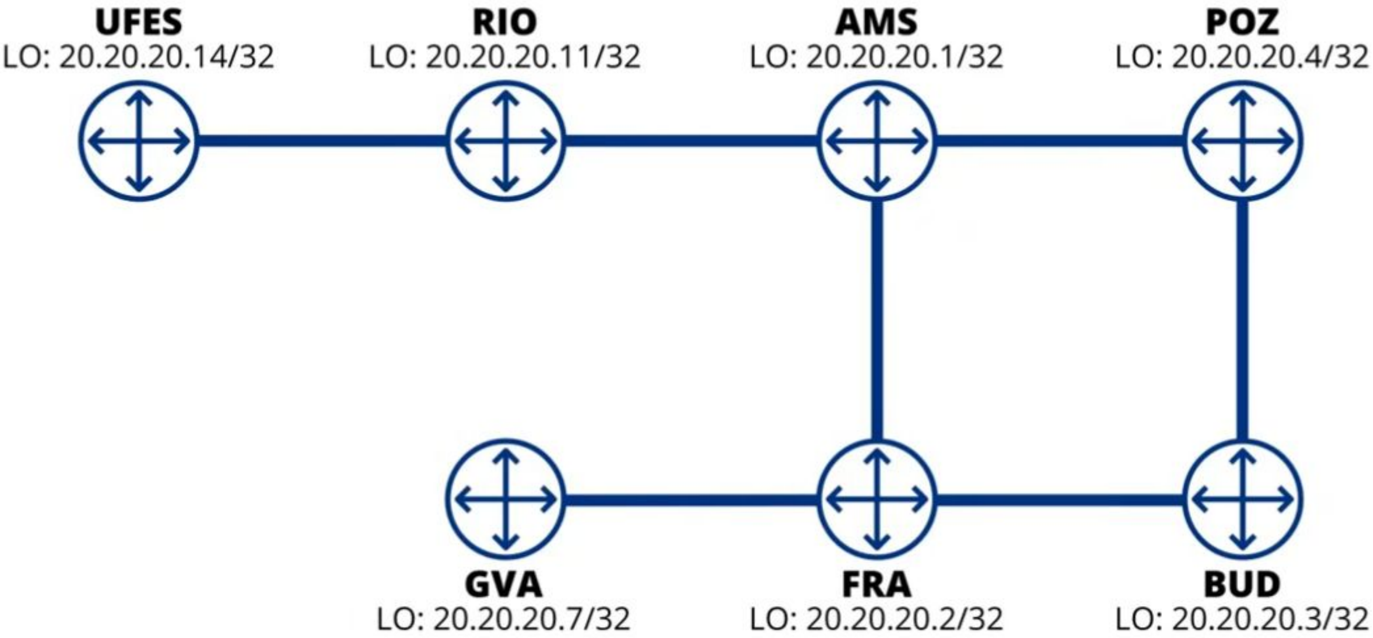


- bfrt_python
  - bf_forwarder.py
- p4src
  - include
    - cst_ethertype.p4
    - cst_table_size.p4
    - def_types.p4
    - eg_ctl.p4
    - eg_ctl_polka.p4
    - hdr_ig_headers.p4
    - hdr_polka.p4
    - ig_ctl.p4
    - ig_ctl_polka.p4
    - ig_prs_clr.p4
    - ig_prs_hdr.p4

Ethernet	version	ttl	proto	routeID	IP	data
	8 bits	8 bits	16 bits	128 bits		

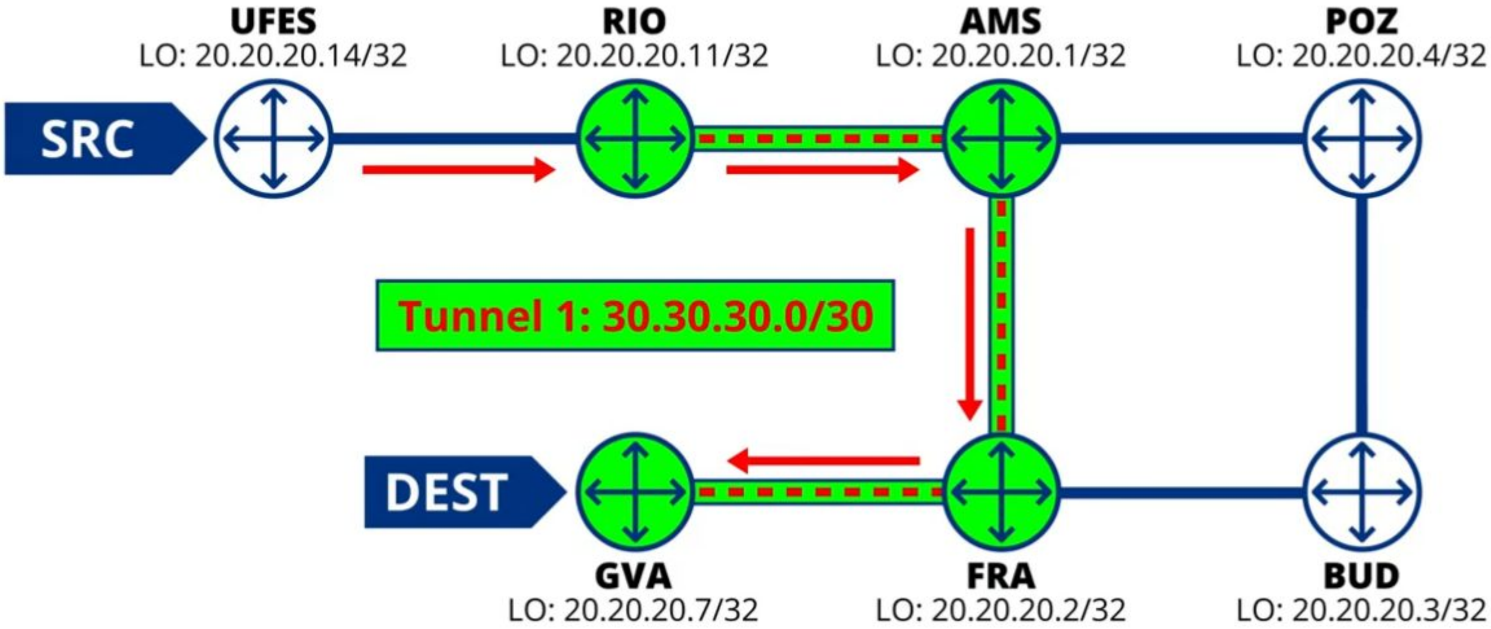
Demonstration: Policy-based routing

- Policy-based routing: https://www.youtube.com/watch?v=YAvajCAvF8Q
  - Emulation in FreeRtr: Agile path reconfiguration in the RARE topology



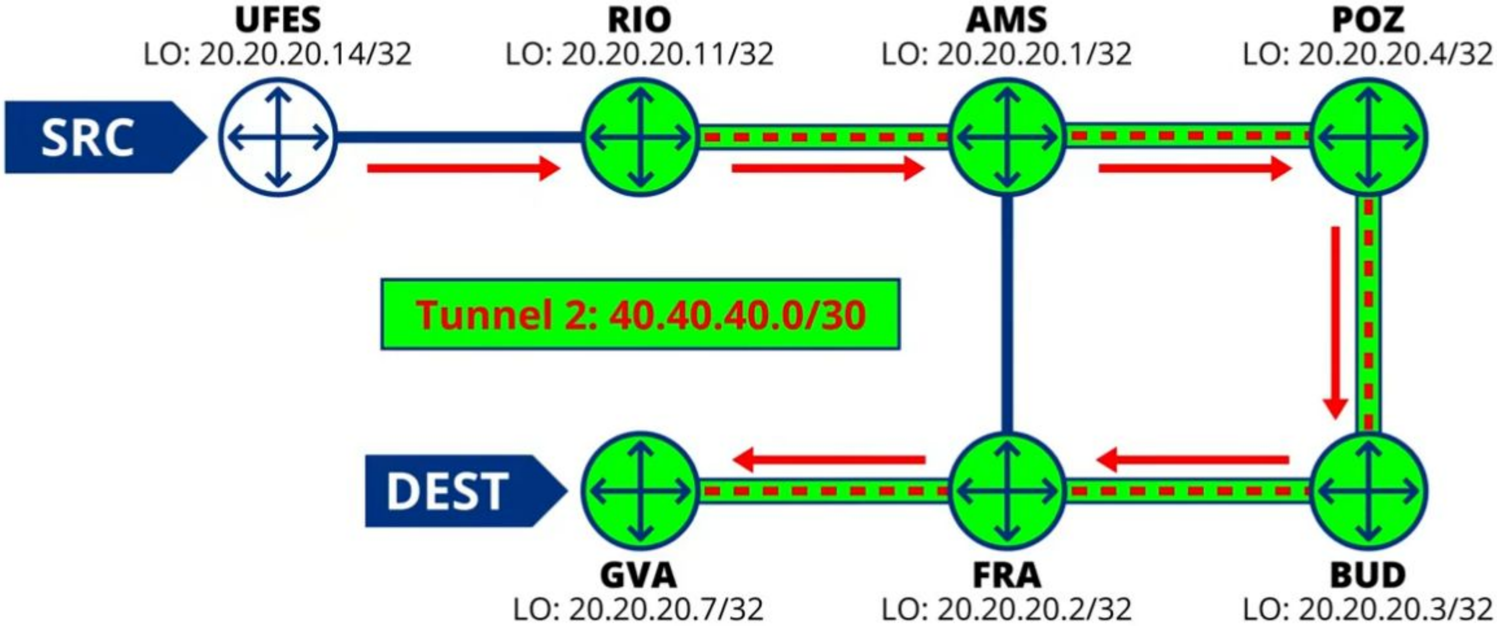
Demonstration: Policy-based routing

- Policy-based routing: https://www.youtube.com/watch?v=YAvajCAvF8Q
  - Emulation in FreeRtr: Agile path reconfiguration in the RARE topology



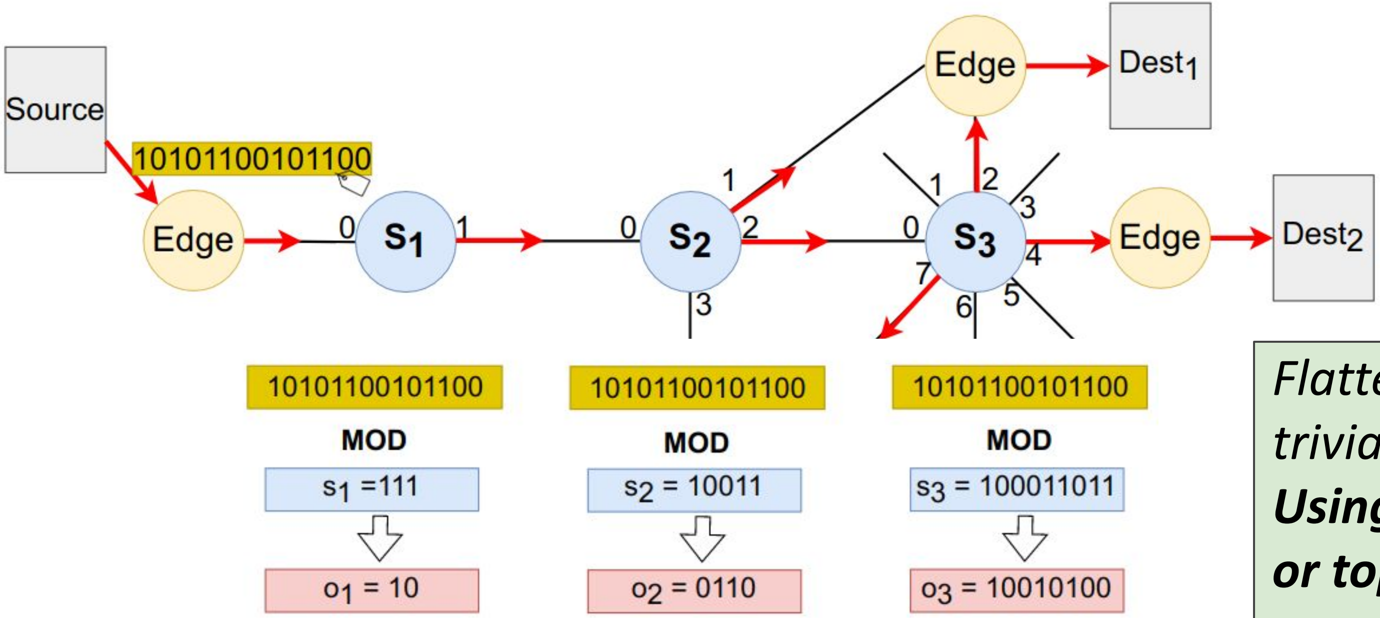
Demonstration: Policy-based routing

- Policy-based routing: https://www.youtube.com/watch?v=YAvajCAvF8Q
  - Emulation in FreeRtr: Agile path reconfiguration in the RARE topology



How to enable Multipath Routing in PolKA (M-PolKA)?

- The *portid* represents the transmitting state of the port (instead of port label).
- The pipeline duplicates packets according to the *portid*.



10101100101100

MOD

$s_1 = 111$

$o_1 = 10$

10101100101100

MOD

$s_2 = 10011$

$o_2 = 0110$

10101100101100

MOD

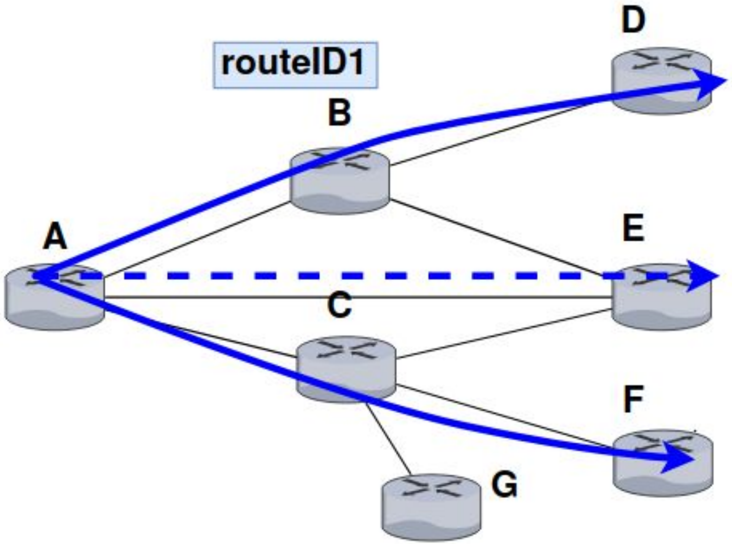
$s_3 = 100011011$

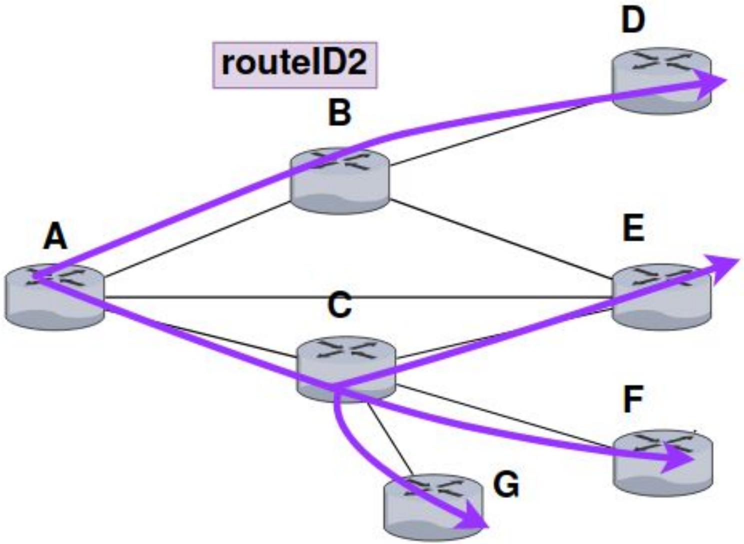
$o_3 = 10010100$

Flattening trees is not trivial…
Using lists: not possible or topology-specific

Applications: Agile Multicast Reconfiguration

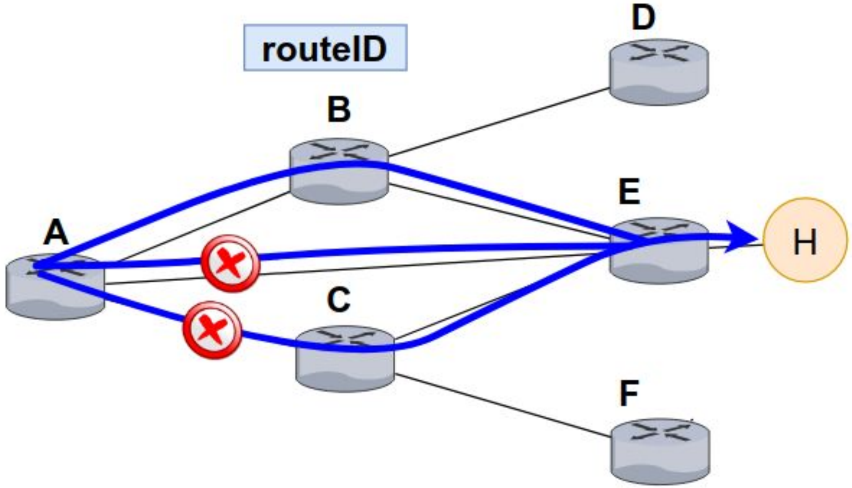
- Agile modification of branches in Multicast tree:
  - Add G + Change path to E (via C)
- The Controller modifies a single entry at the edge:
  - Packets are tagged with the new *routeID*.

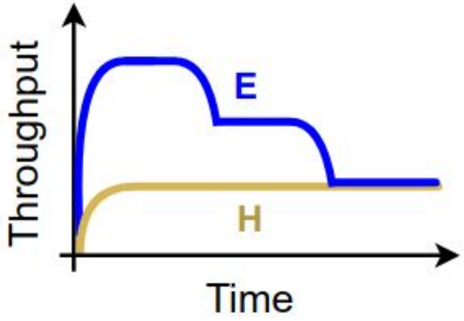




Applications: Failure Protection

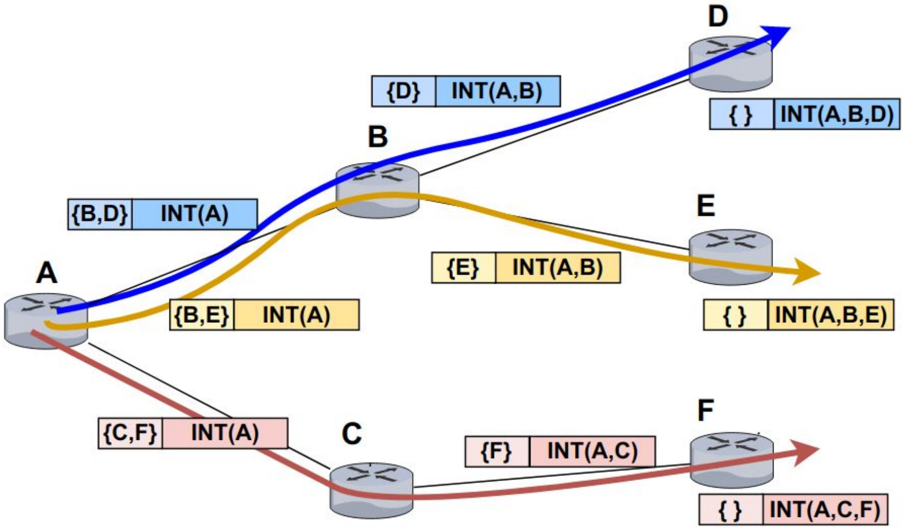
- Packet duplication can also be explored for failure protection.
- A single *routeID* expresses a multipath to steer the traffic.





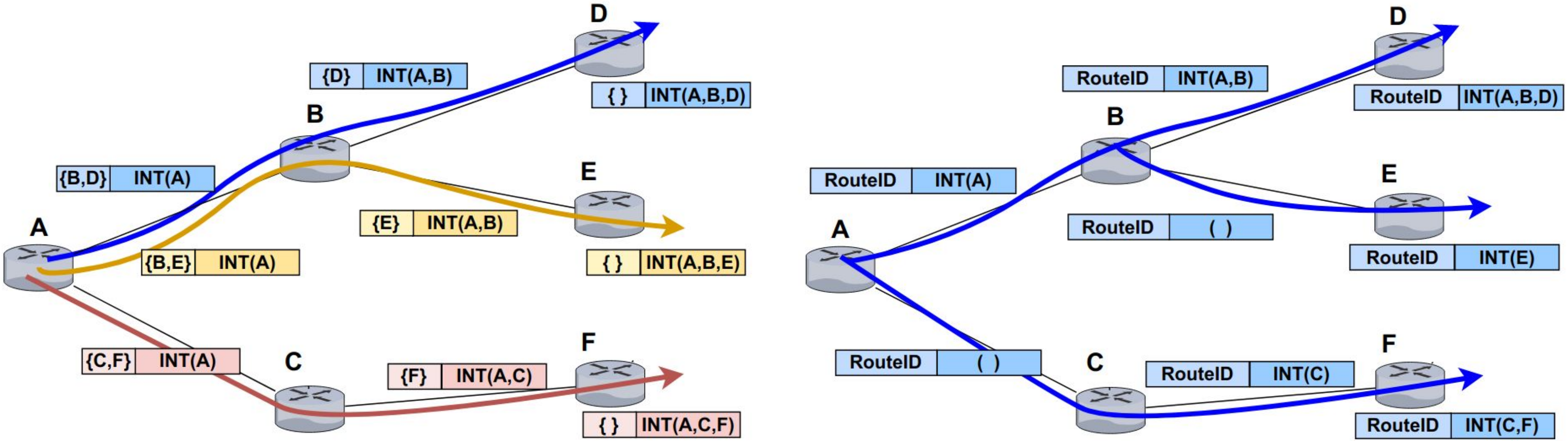
Applications: Inband Network Telemetry

- **Single path** solutions: duplicate probes and INT info. Overhead!



Applications: Inband Network Telemetry

- **Single path** solutions: duplicate probes and INT info. Overhead!
- **M-PolKA**: minimal overhead, no tables, and agile path setup.
  - For any topology!



Is PolKA scalable with the number of nodes?

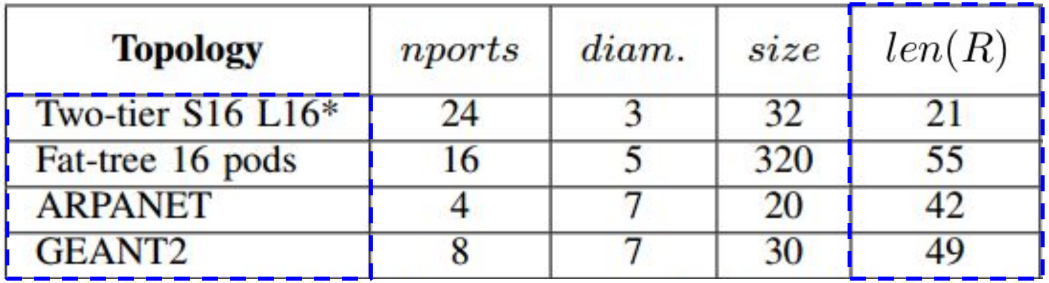
- **Number of flow entries:**
  - Simple and efficient tableless core (only *nodeID* configuration)
  - Edges: table entries for flow classification (depends on granularity)

- **Bit length of the *routeID*: len(R)**
  - Depends on the degrees of the *nodeID* polynomials:

    - We select *nodeIDs* with the lowest possible degree.

    - Worst case for data center and WAN topologies (NetSoft 2020)

$$len(R) \leq \sum_{i=1}^{N} deg(s_i)$$



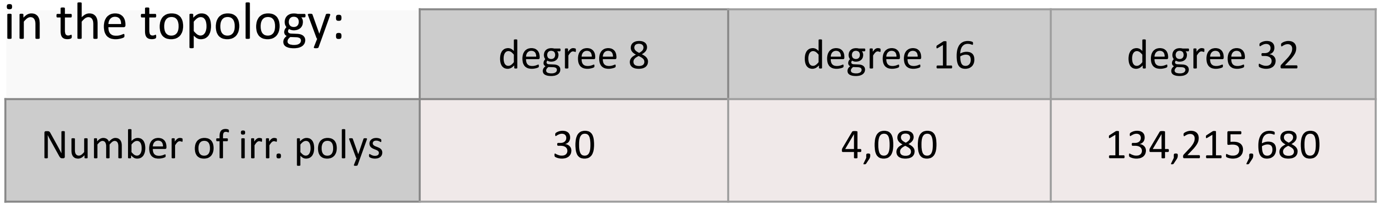
Topology	*nports*	*diam.*	*size*	$len(R)$
Two-tier S16 L16*	24	3	32	21
Fat-tree 16 pods	16	5	320	55
ARPANET	4	7	20	42
GEANT2	8	7	30	49

- In practice, the implementation is linked to CRC 8, 16 or 32.

RouteID bitlength in practice

len(R) = CRC_degree * hops

- CRC degree must provide enough irreducible polynomials to represent all nodes in the topology:



	degree 8	degree 16	degree 32
Number of irr. polys	30	4,080	134,215,680

RouteID bitlength in practice

len(R) = CRC_degree * hops

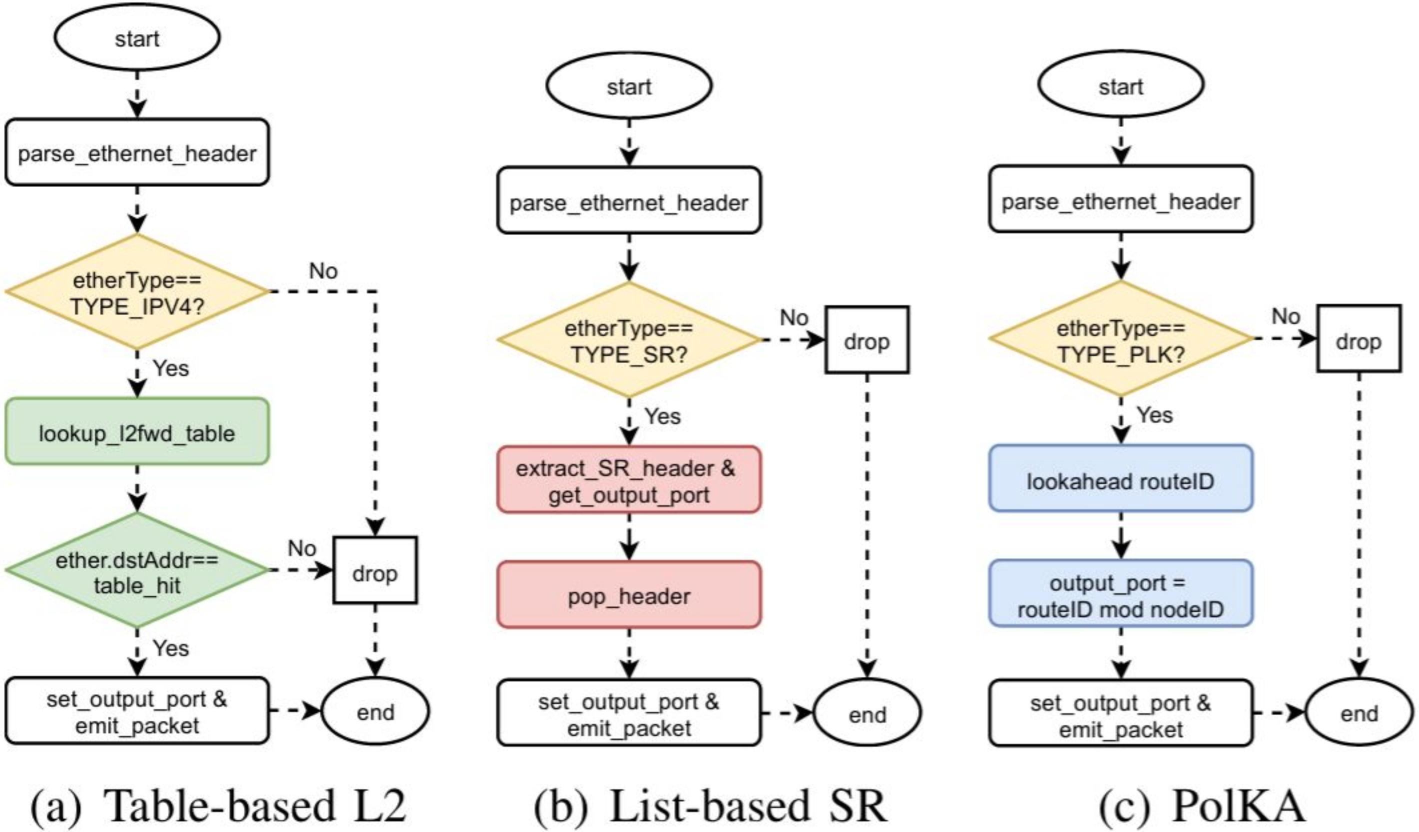
- CRC degree must provide enough irreducible polynomials to represent all nodes in the topology:

	degree 8	degree 16	degree 32
Number of irr. polys	30	4,080	134,215,680

- **Fat tree (CRC16)**:
  - topology size = 320 ⟶ use CRC16
  - diameter = 5 ⟶ len(R) = 16 * 5 = 80 bits
  - Nr. of ports is not a problem: polys of degree 16 can represent 2^{16} ports

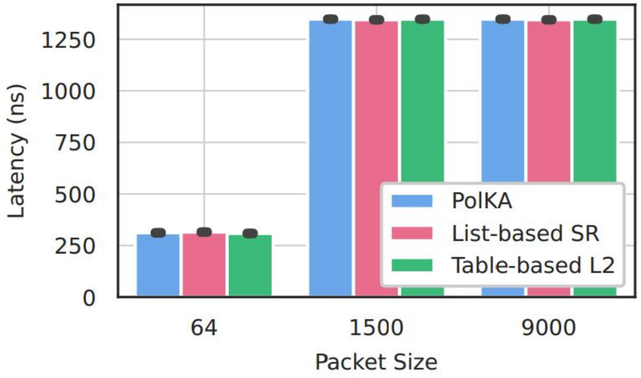
- **Jupiter (CRC32)**:
  - topology size = 14336 fabric switches + nr. of TOR switches ⟶ use CRC32
  - diameter = 5 ⟶ len(R) = 32 * 5 = 160 bits

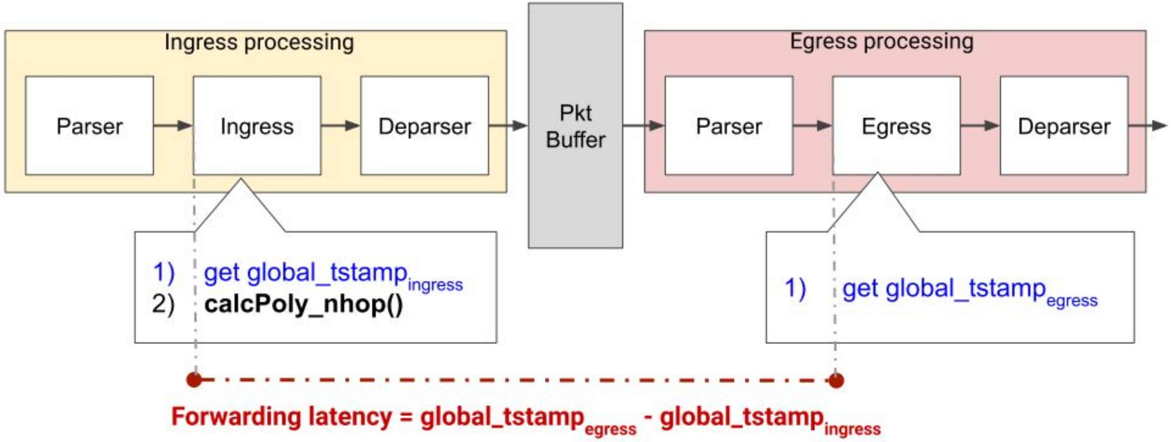
PolKA: P4 Pipelines



Experiments

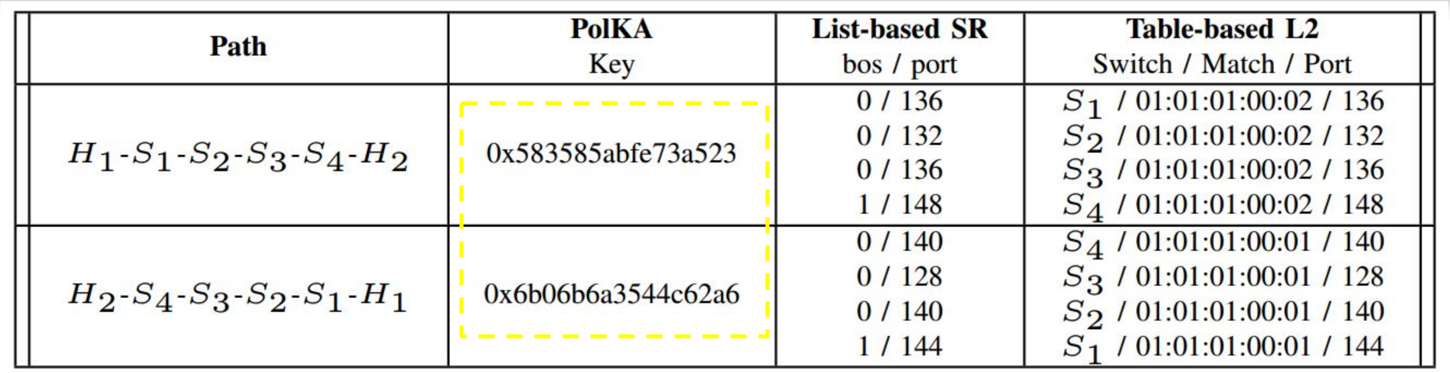
- Forwarding Latency:
    - Hardware timestamps.
    - UDP traffic for different packet sizes with a throughput of 10Gbps.



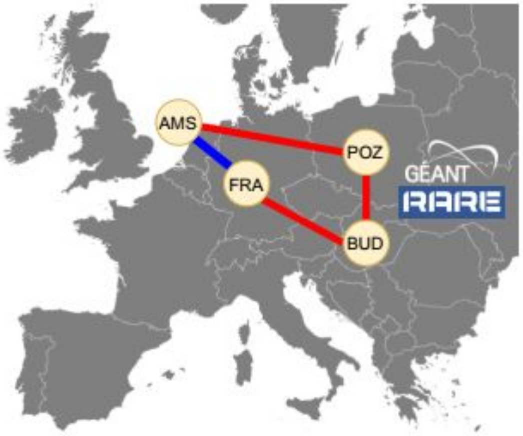


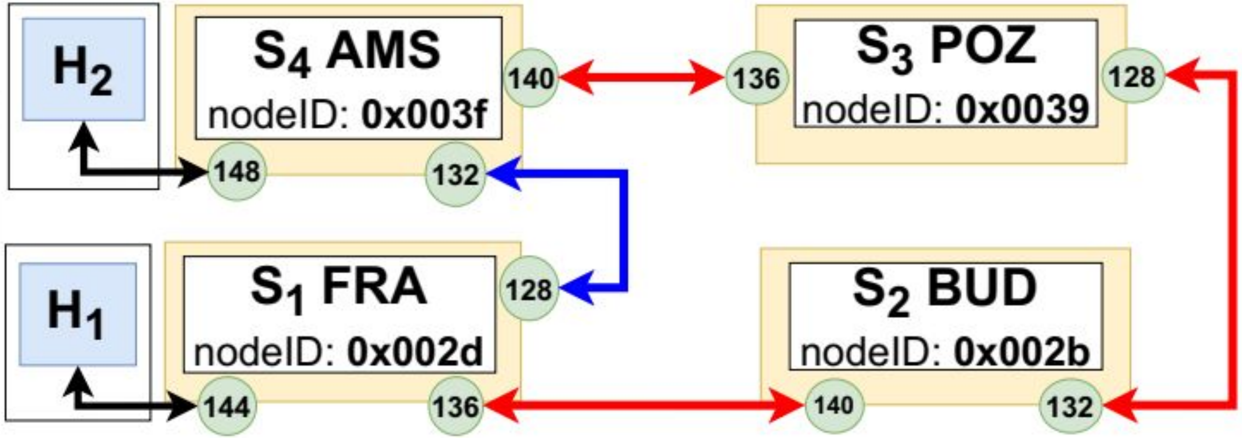
Ingress processing
Parser
Ingress
Deparser
Pkt Buffer
Egress processing
Parser
Egress
Deparser
1) get global_tstamp$_{ingress}$
2) calcPoly_nhop()
1) get global_tstamp$_{egress}$
Forwarding latency = global_tstamp$_{egress}$ - global_tstamp$_{ingress}$

PolKA: Prototype



Path	**PolKA** Key	**List-based SR** bos / port	**Table-based L2** Switch / Match / Port
H_1-S_1-S_2-S_3-S_4-H_2	0x583585abfe73a523	0 / 136	S_1 / 01:01:01:00:02 / 136
		0 / 132	S_2 / 01:01:01:00:02 / 132
		0 / 136	S_3 / 01:01:01:00:02 / 136
		1 / 148	S_4 / 01:01:01:00:02 / 148
H_2-S_4-S_3-S_2-S_1-H_1	0x6b06b6a3544c62a6	0 / 140	S_4 / 01:01:01:00:01 / 140
		0 / 128	S_3 / 01:01:01:00:01 / 128
		0 / 140	S_2 / 01:01:01:00:01 / 140
		1 / 144	S_1 / 01:01:01:00:01 / 144

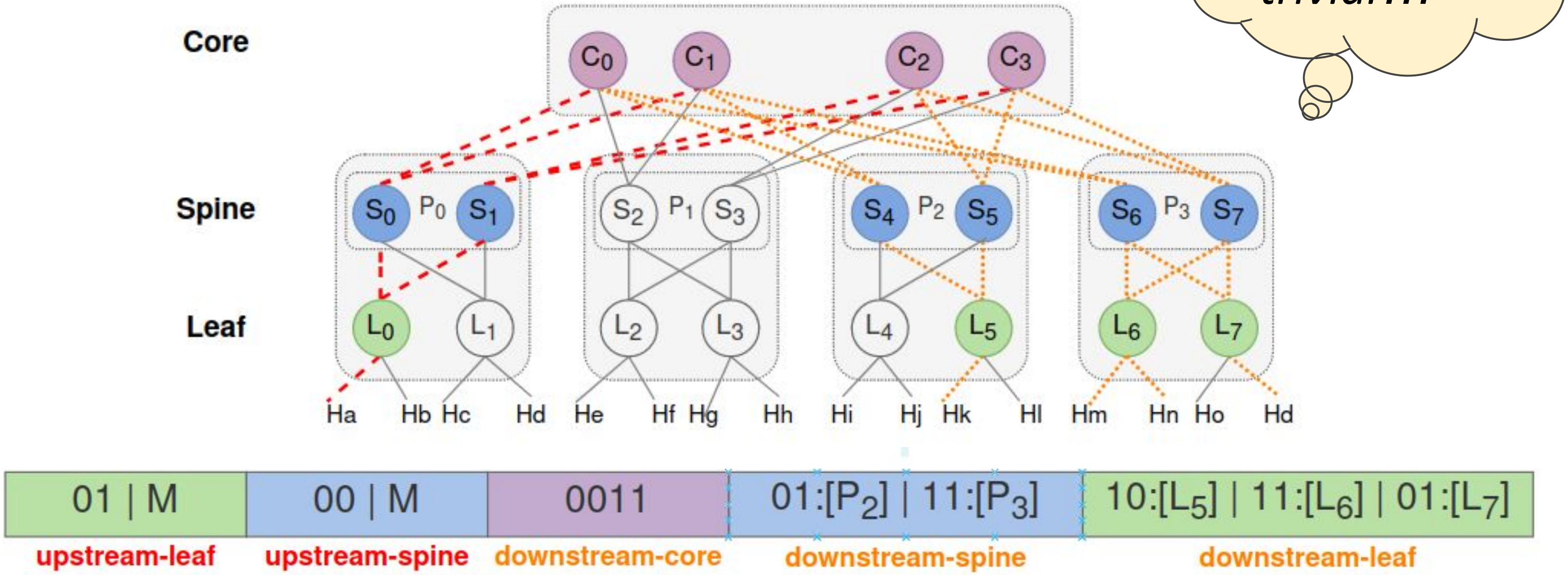




The Multipath Source Routing Problem

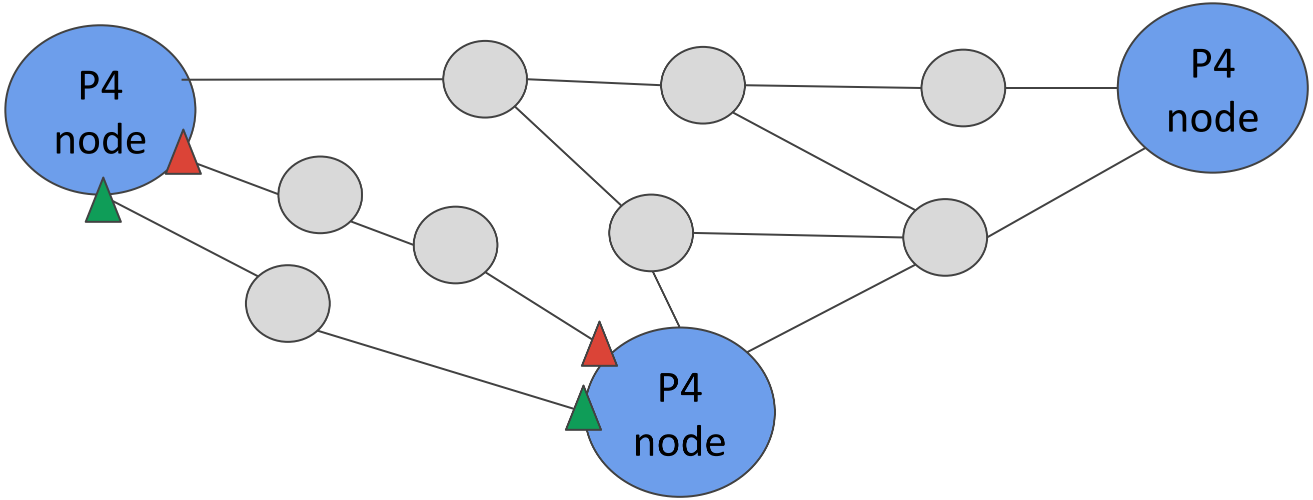
- Related works:
  - Tables + Source Routing. Ex: [BIER]
  - Topology-specific encoding. Ex: [ELMO]

Flattening trees is not trivial…



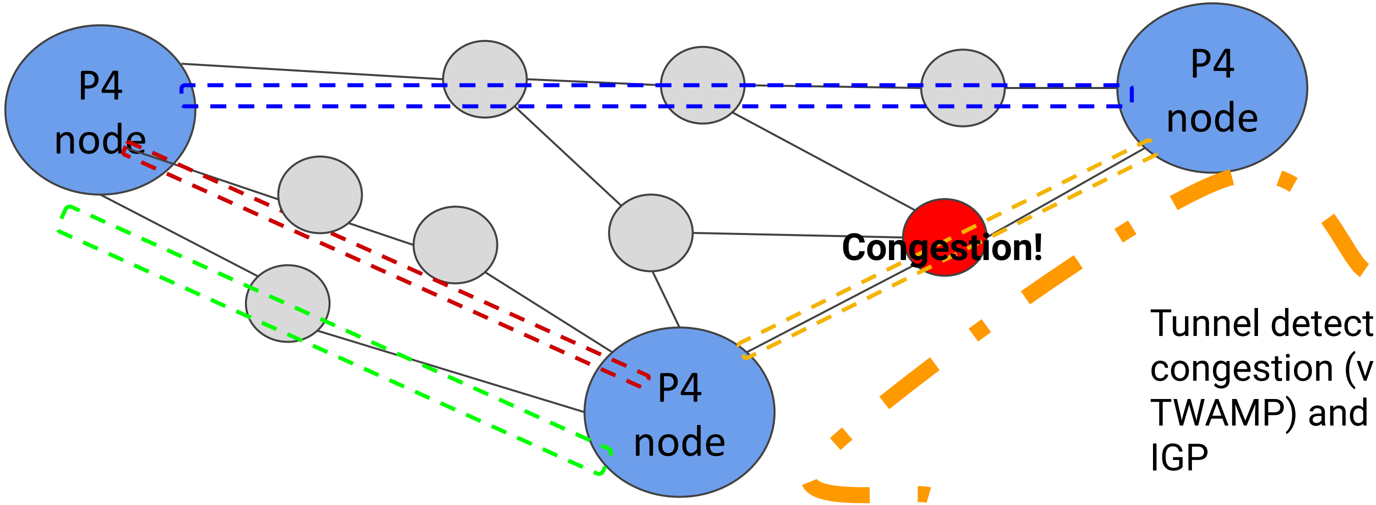
Freerouter supports standard active measurements

- Discussions with Marcos Schwarz (RNP): *Congestion detection and avoidance*
  - P4 nodes are connected via an underlay network.
  - Tunnels ▲ monitored by TWAMP (Two-Way Active Measurement Protocol) *RFC 5357*



Potential Experiments: PolKA vs. Segment Routing

Underlay congestion can be dedicated by tunnel monitoring and fed to the IGP of our overlay network



Tunnel detects congestion (via TWAMP) and feeds IGP