2025 IETF High Performance -WAN Research Group
November 04th, 2025 // Virtual Presentation

PolKA: Polynomial Key-based Architecture for High-Performance WANs

Magnos Martinello[1], Cristina Klippel Dominicini[2], Rafael Guimarães[2], Moises R. N. Ribeiro[1], Rodolfo Villaça[1], Everson Borges[2], Daniel Ventorim[12], Adailton Saraiva[1], and Pedro Picolli Filho[12]

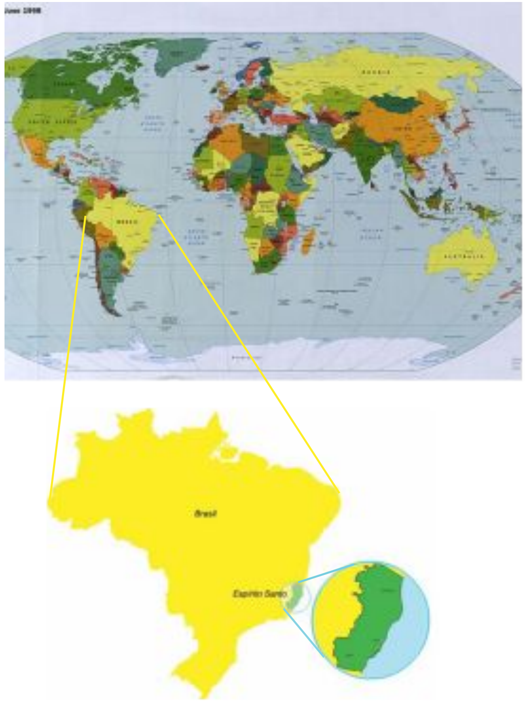
[1]Federal University of Espírito Santo, [2]Federal Institute of Espírito Santo,
Contact: magnos.martinello@ufes.br

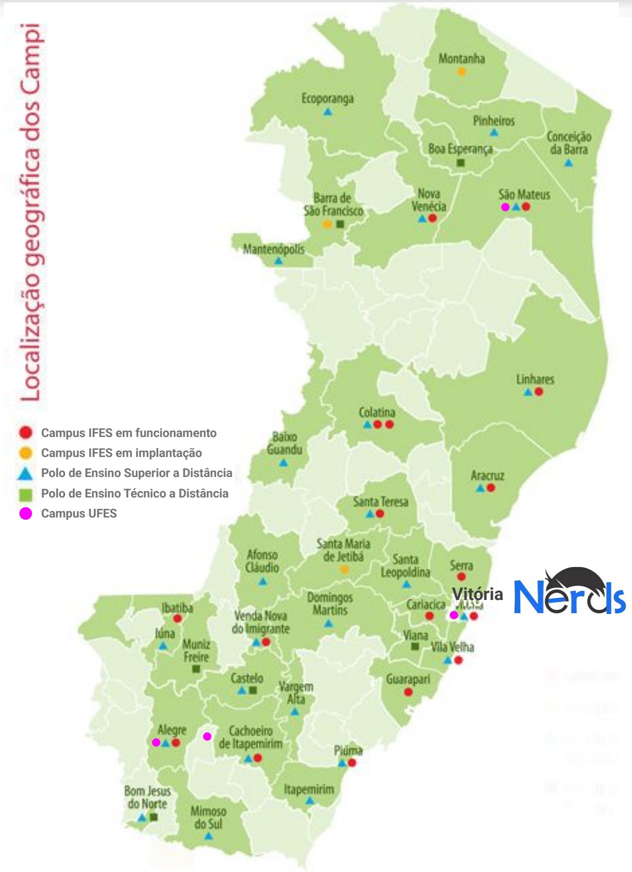
UFES

INSTITUTO FEDERAL
Espírito Santo

Location and Institutions

- **Espírito Santo, Brazil**





UFES

● *5 Campi*

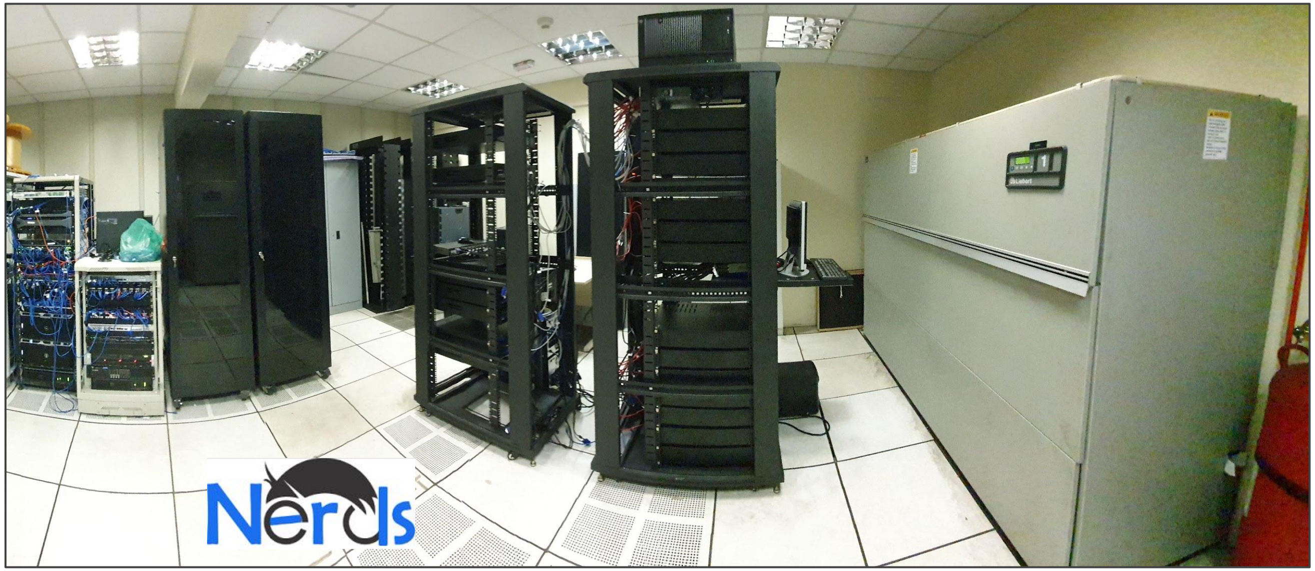
INSTITUTO FEDERAL Espírito Santo

● *22 Campi*

LabNERDS: Software Defined Networks Research Group

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

https://nerds-ufes.github.io/



Nerds Datacenter - UFES

Agenda

- **Motivation**
- Proposal
- Design
- Prototype
- Applications
- Conclusions and future works

Motivation

- **HP-WAN mission:** to enable ultra-fast, resilient, and adaptive data paths for Data-Intensive Science and AI-driven workloads
- **SDN and Network Programmability**
    - Innovation of protocols

Motivation

- **SDN and Programmable Network Devices**:
  - Innovation and custom protocols.
- The packet forwarding based on table lookups has some **bottlenecks:**

Large number of states	**Limited capacity of switch tables**	**Latency for path configuration**
⇩	⇩	⇩
Network scalability	**Granularity**	**Agility**

Motivation

- **SDN and Programmable Network Devices**:
  - Innovation and custom protocols.
- **Bottleneck**: forwarding based on **table entries**

$$\begin{tikzcd}
\textbf{Large number of states} \arrow[d] & \textbf{Limited capacity of switch tables} \arrow[d] & \textbf{Latency for path configuration} \arrow[d] \\
\textbf{Network scalability} & \textbf{Granularity} & \textbf{Agility}
\end{tikzcd}$$

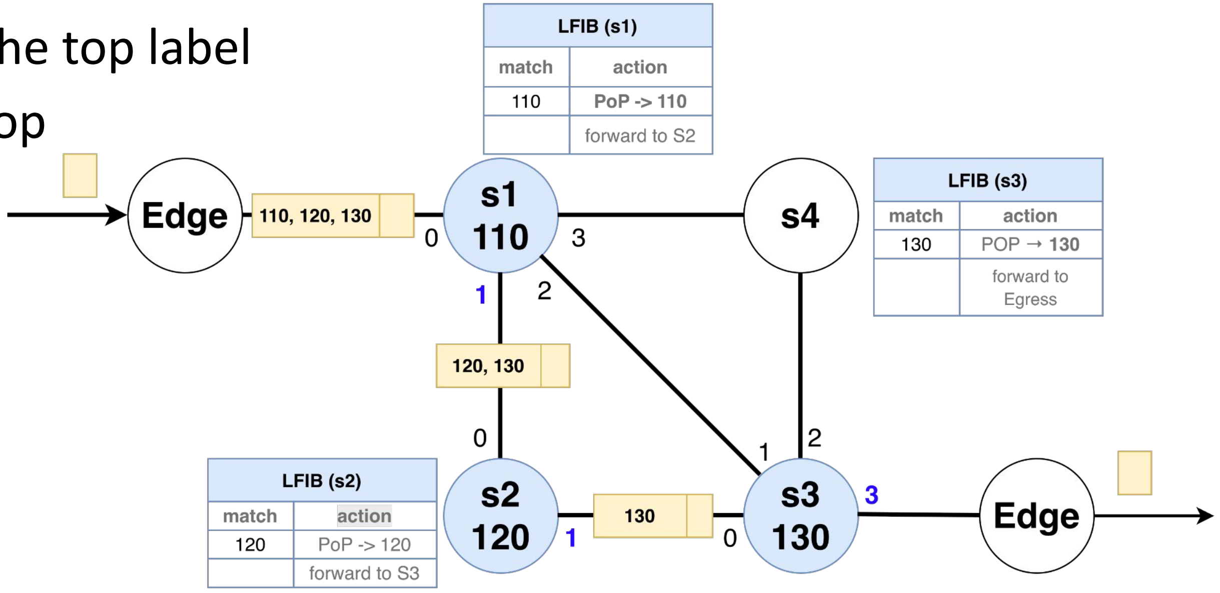
What are the alternatives to tackle these problems?

- **Source Routing (SR):**

  - A source specifies a path and adds a route label to the packet header.

Source Controlled Routing

- **Traditional way: List-based SR (LSR)**
  - Path: a list of ports or a stack of labels.
    - Edge pushes a stack [110,120,130]
    - Each node
      - matches on the top label
      - performs a pop



LFIB (s1)

match	action
110	PoP -> 110
	forward to S2

LFIB (s3)

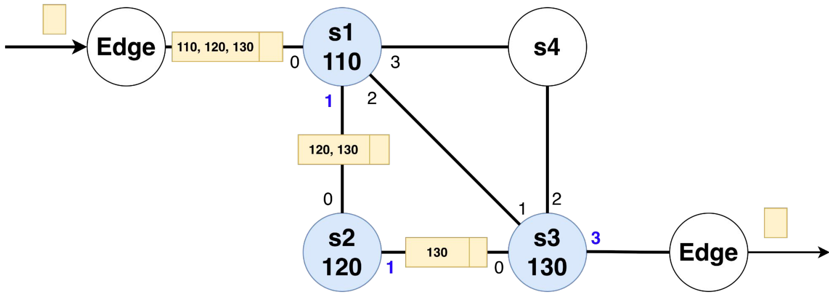
match	action
130	POP → 130
	forward to Egress

LFIB (s2)

match	action
120	PoP -> 120
	forward to S3

Source Routing (SR)

- **Traditional way: List-based SR (LSR)**
  - Path: a list of ports or a stack of labels.
  - Each node performs a pop.



- **Limitations**:
  - **Forwarding state** in the packet & rewrite operation (push/pop/swap)
  - **Variable size of headers** that affects the number of encoded hops
  - **Not fully stateless**, still rely on per-node forwarding tables in the core
    - MPLS LFIB lookup (match on top label), SR SIDs, BIER forwarding tables.

Agenda

- Motivation
- **Proposal**
- Design
- Prototype
- Applications
- Conclusions and future works

PolKA Proposal

Aims to design an SR approach to meet the requirements

↓

topology agnostic | **fixed header** | **encoded path** | **no tables in the core**

↓

Deployable in programmable switches

PolKA Proposal

- PolKA: Polynomial Key-based Architecture for High-Performance WAN
  - *NetSoft 2020 conference paper*
  - Polynomial Residue Number System (**RNS**) (*Shoup, 2008*)
  - Chinese Remainder Theorem (**CRT**)
  - Forwarding based on an arithmetic operation: **remainder of division**

    - Path is encoded in a route label.

    - Each node decodes only its next hop with its own key.

Agenda

- Motivation
- Proposal
- **Design**
- Prototype
- Applications
- Conclusions and future works

How Does PolKA Work?

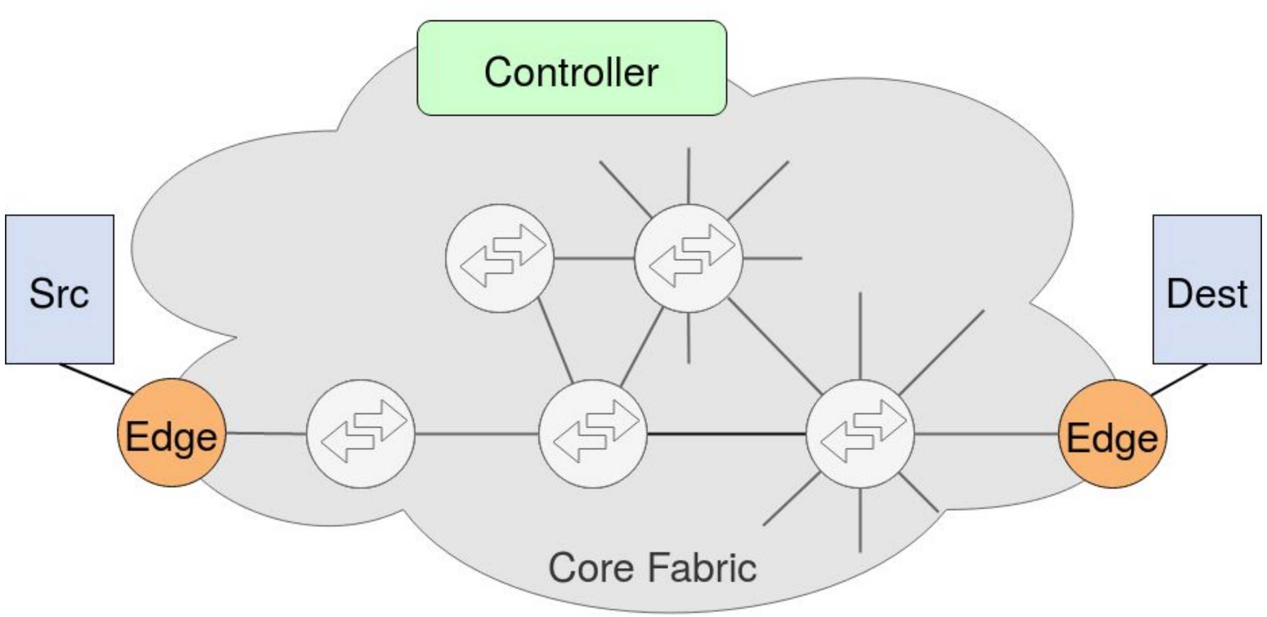
- Three polynomials:
  - **routeID**: a route identifier calculated using the CRT.
  - **nodeID**: to identify each core node.

    - Irreducible polynomial
  - **portID**: to identify the ports of each core node.
- The forwarding uses a **mod** operation (remainder of division):

$$\boxed{\mathbf{portID} = \langle\, \mathbf{routeID} \,\rangle_{\mathbf{nodeID}}}$$

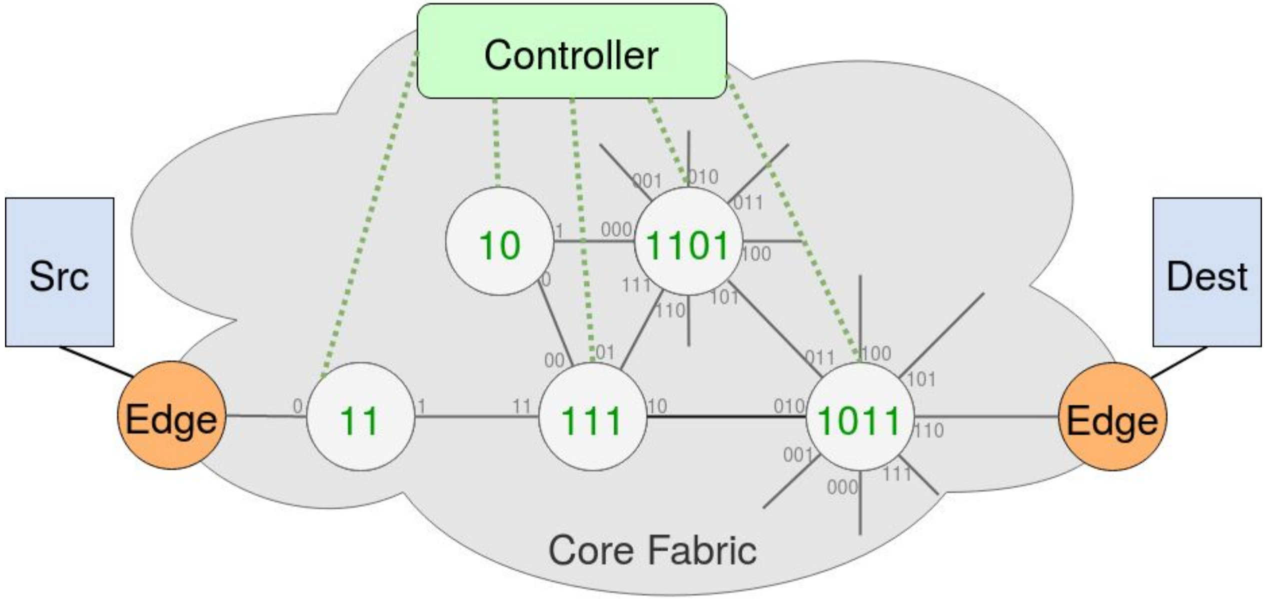
How Does PolKA Work?

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



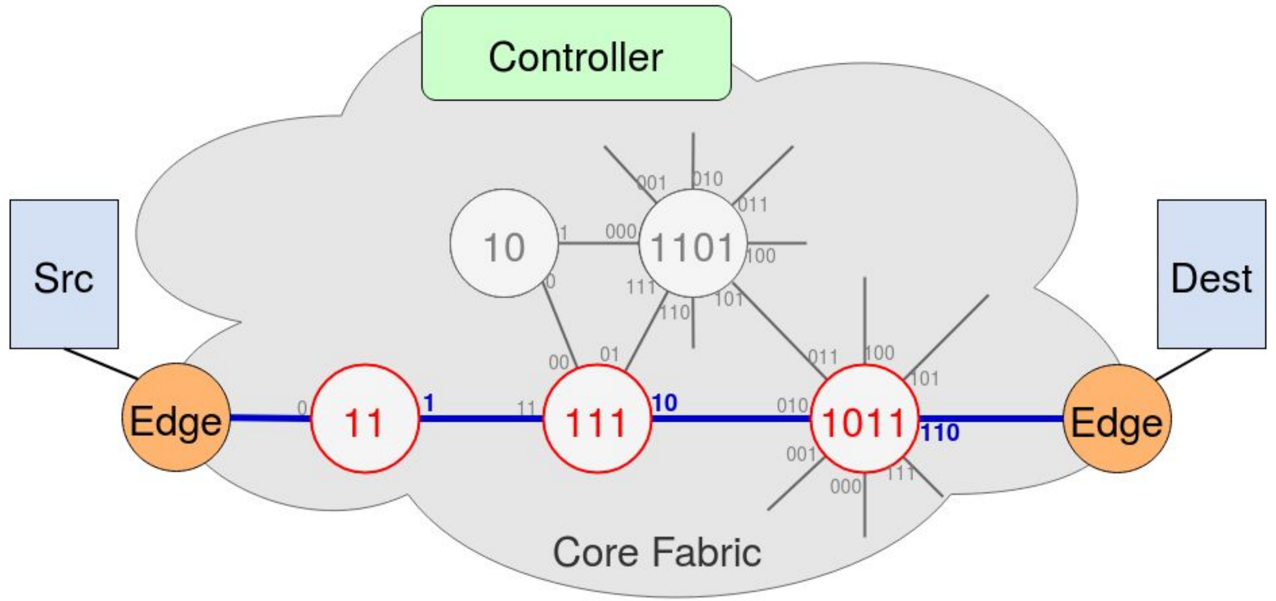
How Does PolKA Work?

- 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

$s_1(t) = t + 1 = 11$

$s_2(t) = t^2 + t + 1 = 111$

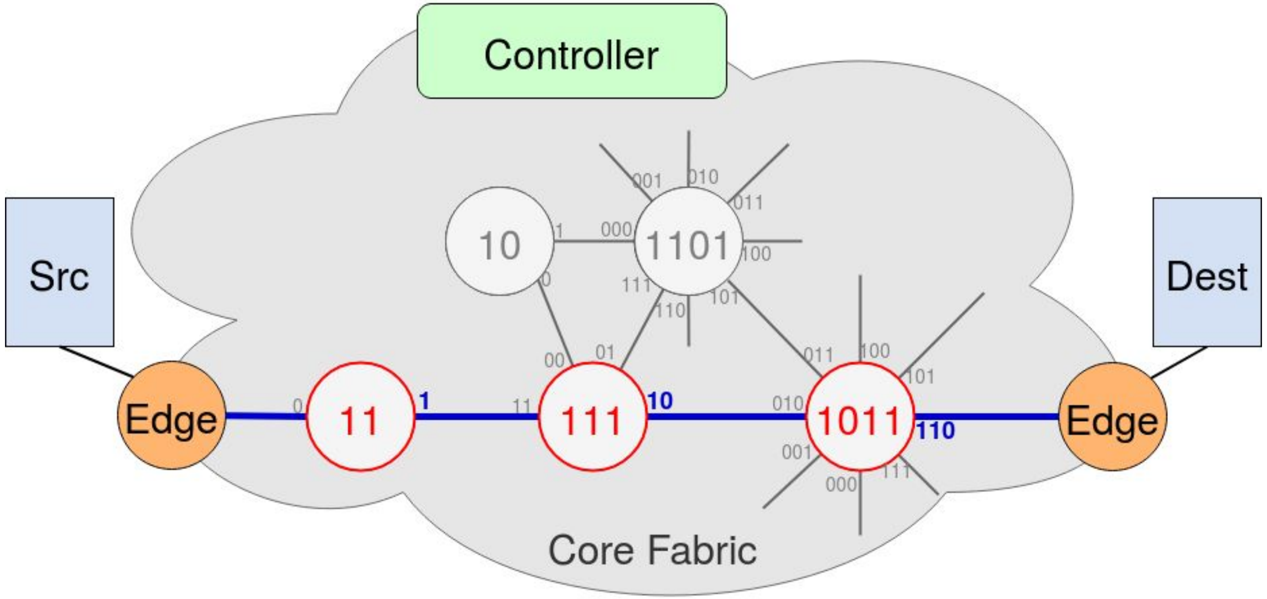
$s_3(t) = t^3 + t + 1 = 1011$

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 $M(t) = \prod_{i=1}^{N} s_i(t)$

N = number of hops

M = product of nodeIDs

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

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

The routeID (**X**) is the result of congruent system in GF(2)

001 = ***X*** mod *0011*

010 = ***X*** mod *0111*

110 = ***X*** mod *1011*

R = 10000

routeID

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 operations along the path :

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

001 = <10000>$_{0011,}$ $\rightarrow$ 10000 mod *0011*

010 = <10000>$_{0111}$ $\rightarrow$ 10000 mod *0111*

110 = <10000>$_{1011}$ $\rightarrow$ 10000 mod *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

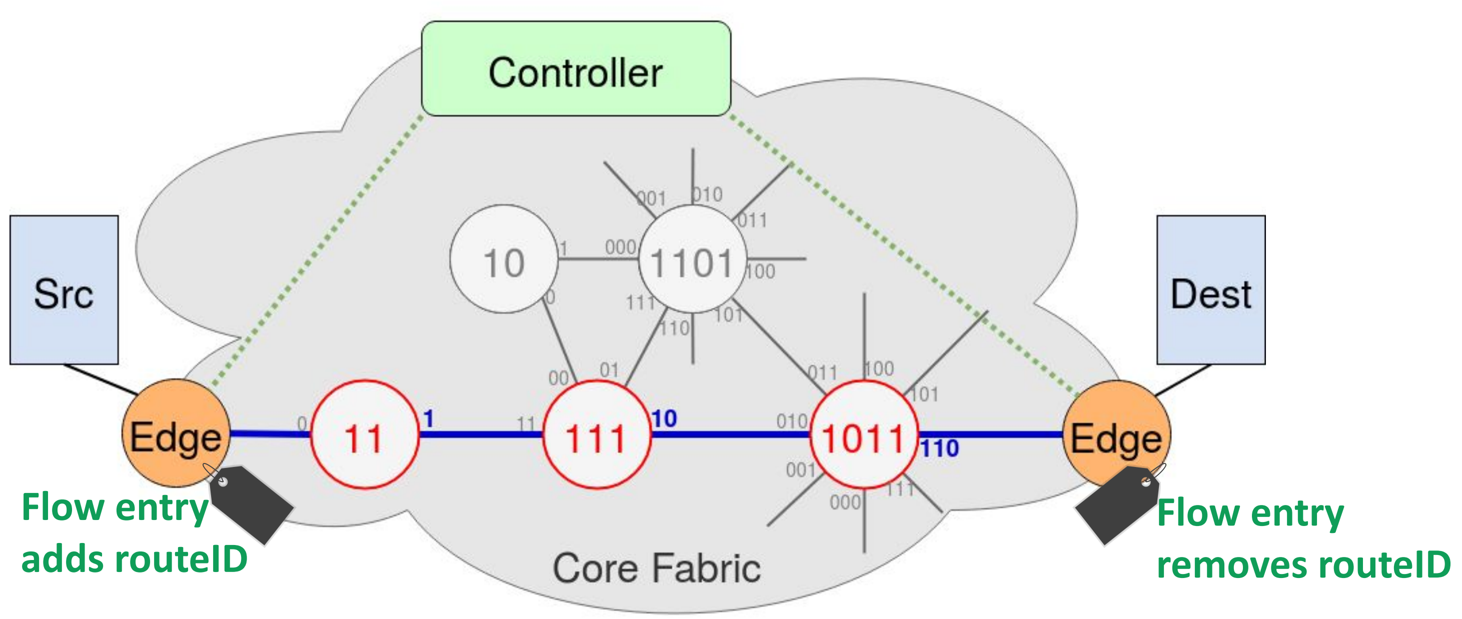
$$o_1(t) = 1$$
$$o_2(t) = t = 10$$
$$o_3(t) = t^2 + t = 110$$

Calculate routeID with CRT

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

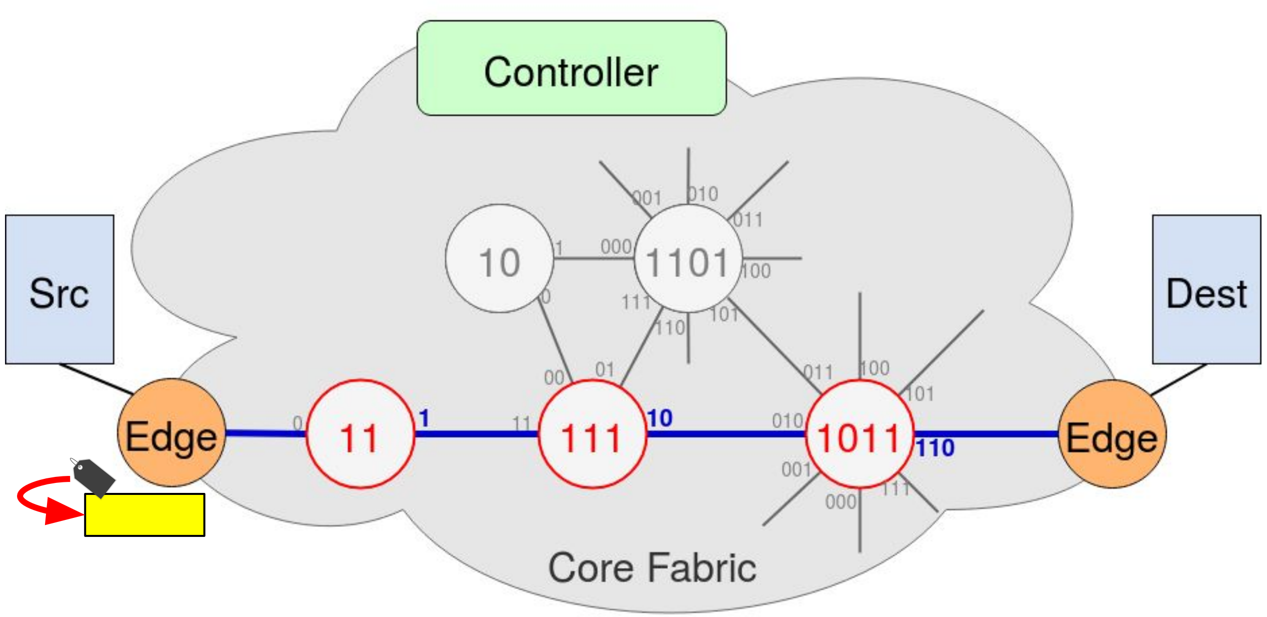
How Does PolKA Work?

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



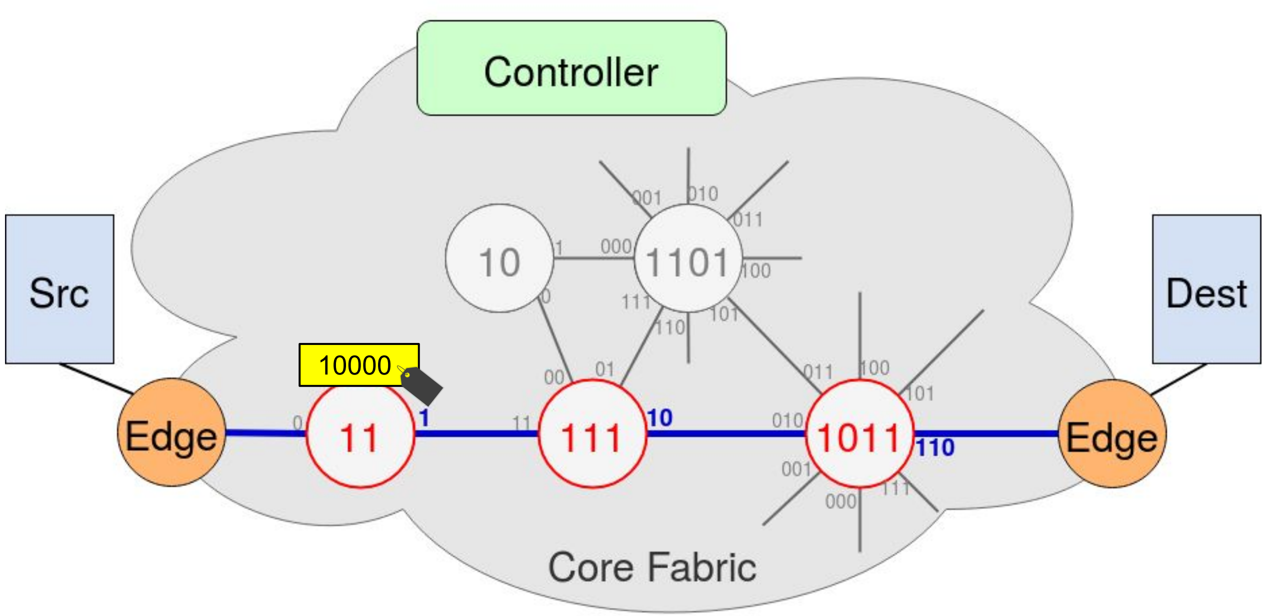
How Does PolKA Work?

- 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 in the core nodes !

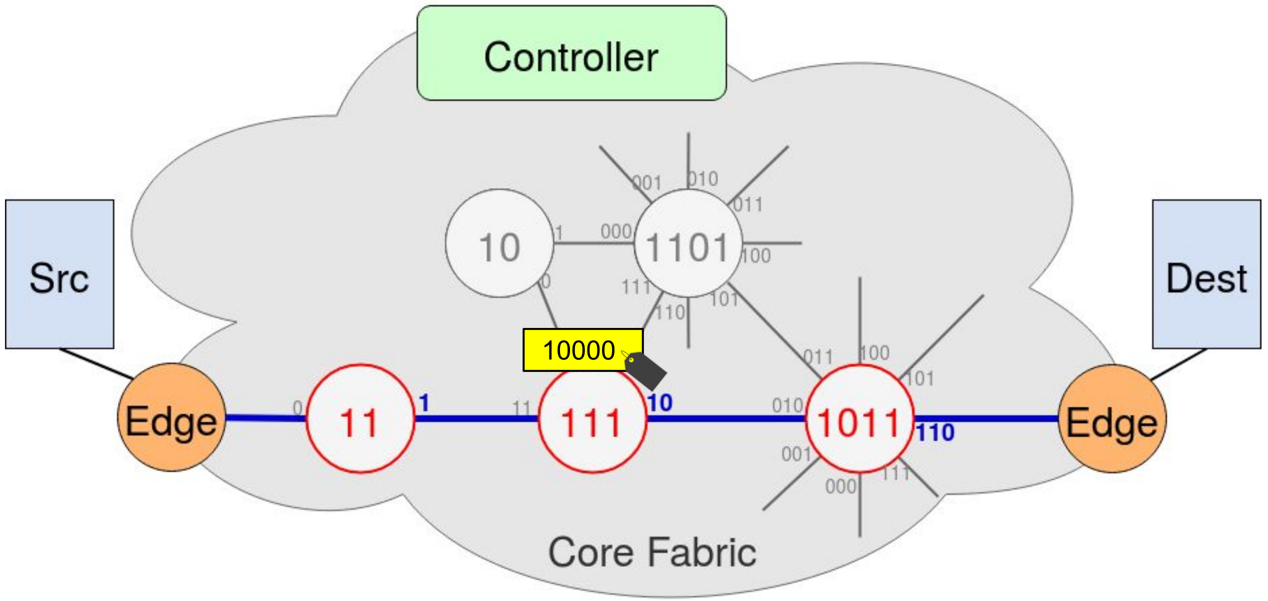


10000 mod 11 GF(2)

```
---------
    10000
⊕   11000
----------
    01000
⊕    1100
----------
     0100
⊕     110
----------
       10
⊕      11
-----------
mod/port: 1
```

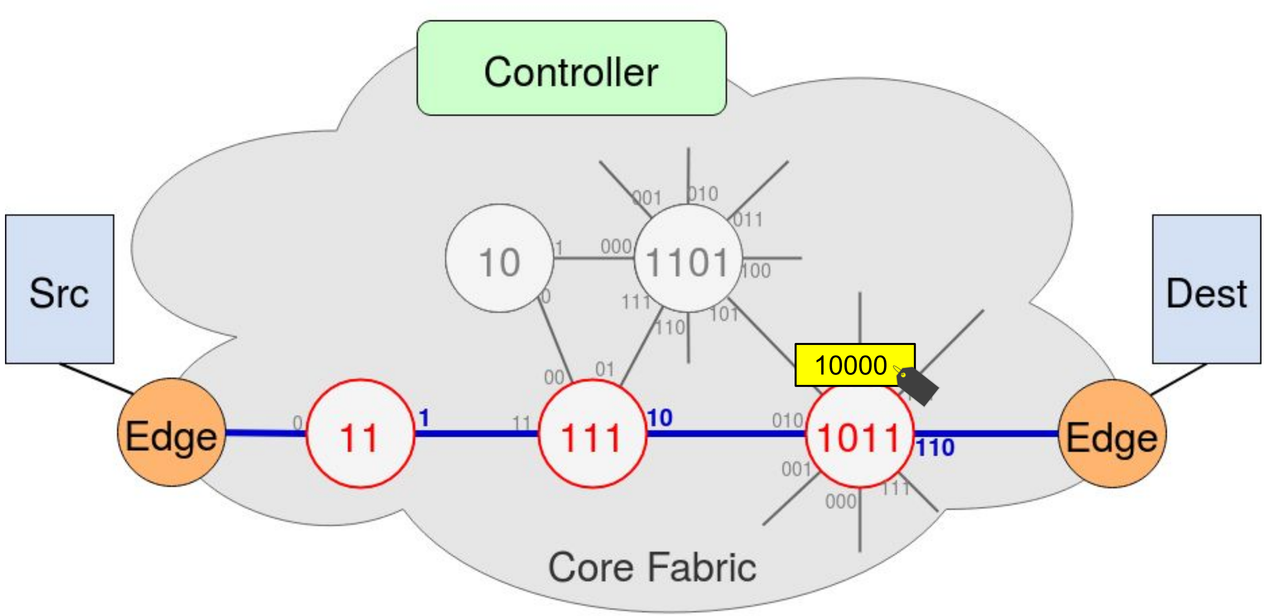
How Does PolKA Work?

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



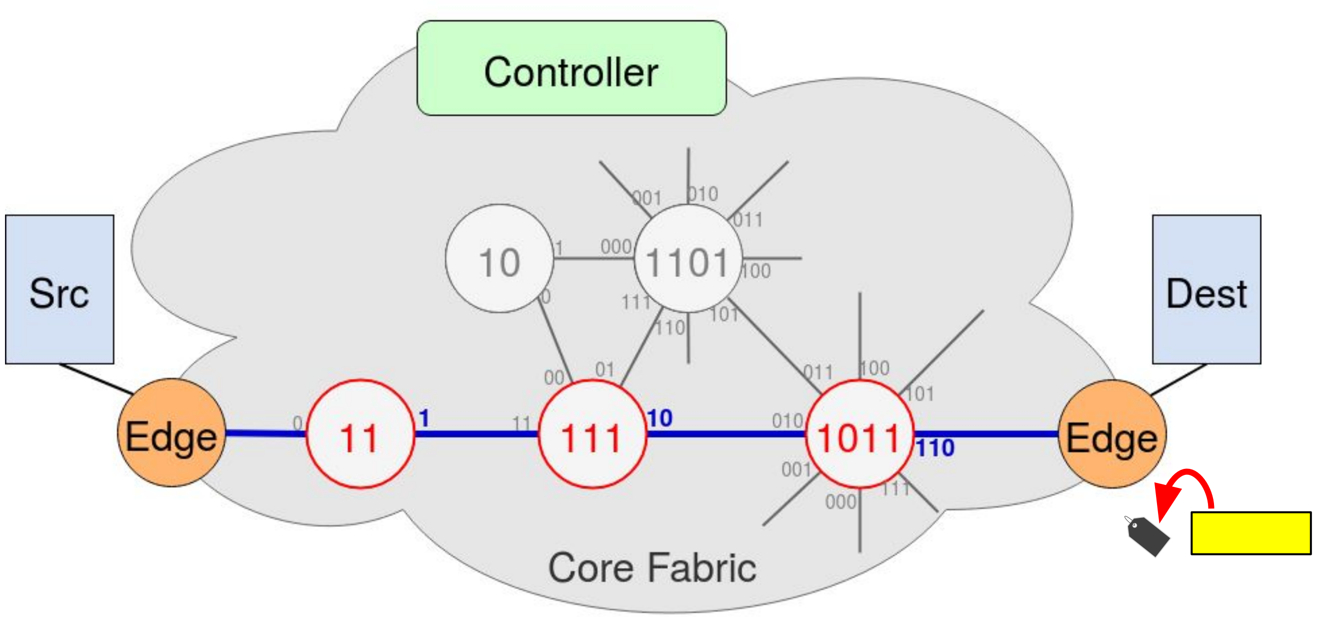
How Does PolKA Work?

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



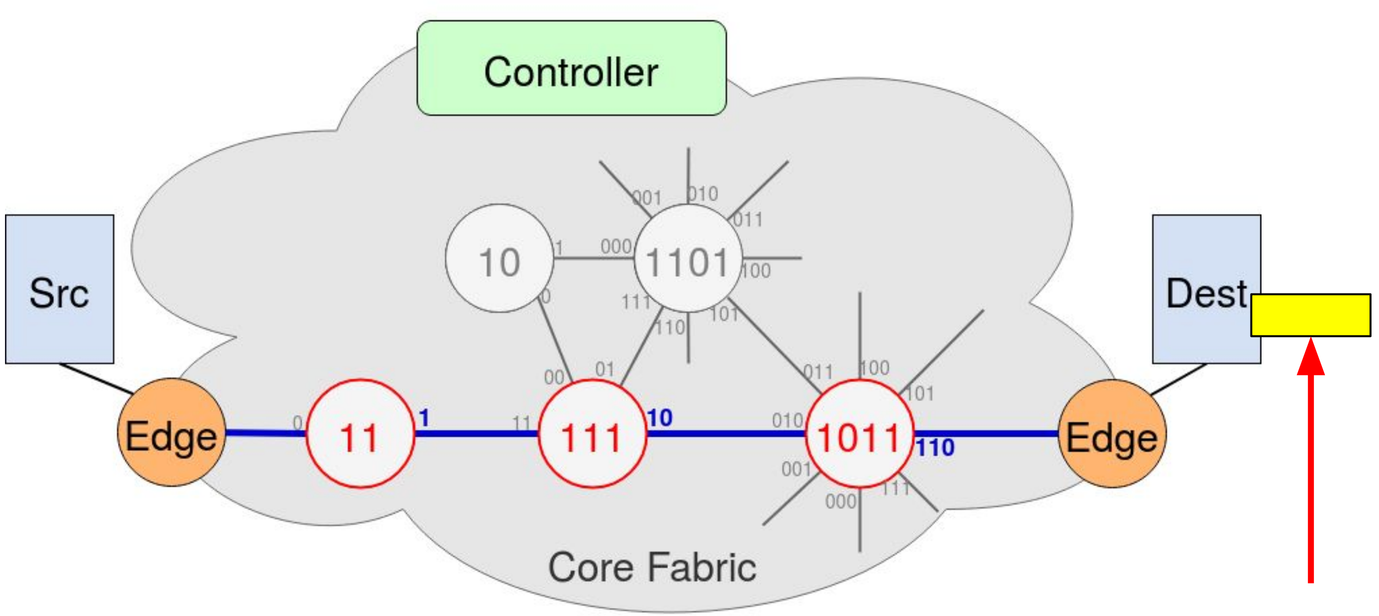
How Does PolKA Work?

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



How Does PolKA Work?

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



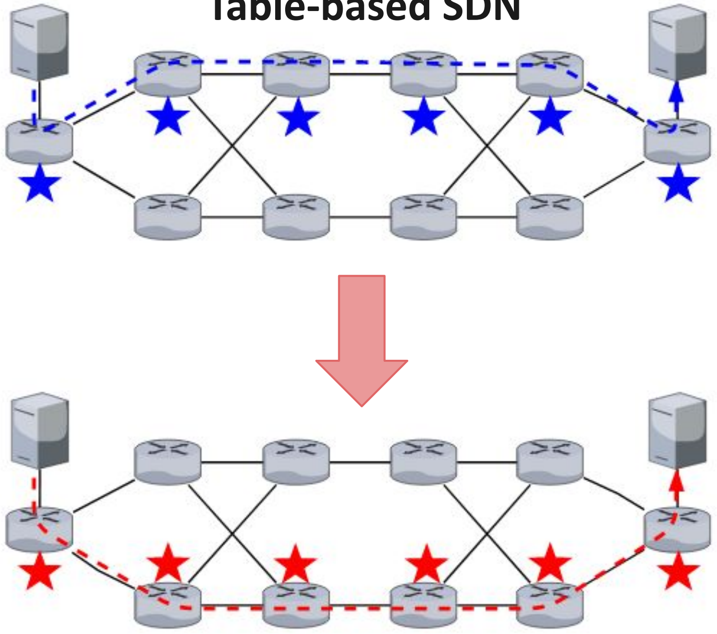
How to Implement a *mod* Efficiently in Data Plane?

- **P4 language does not natively support the mod operation.**
- **By using CRC** (Cyclic Redundancy Check), we can calculate the mod.
  - The Tofino Native Architecture (**TNA**) supports **custom** CRC polynomials.
  - MOD = 2 SHIFTs + 1 **CRC** + 2 XORs

1. G = nodeID = **01011**, portanto r = deg(G) = **3**
2. D = routeID $\div 2^r$ = 100101~~111~~ >> 3 = 100101 **(SHIFT RIGHT)**
3. dif = routeID – D $\cdot 2^r$ = 100101111 $\oplus$ (100101 << **3**)
   = 100101111 $\oplus$ 100101**000** = 111 **(SHIFT LEFT, XOR)**
4. R = $\langle D \cdot 2^r \rangle_G$ = $\langle 100101\mathbf{000} \rangle_{(\mathbf{01011})}$ = 110 **(CRC)**
5. portID = dif $\oplus$ R = 111 $\oplus$ 110 = 001 **(XOR)**

Is PolKA Scalable ?

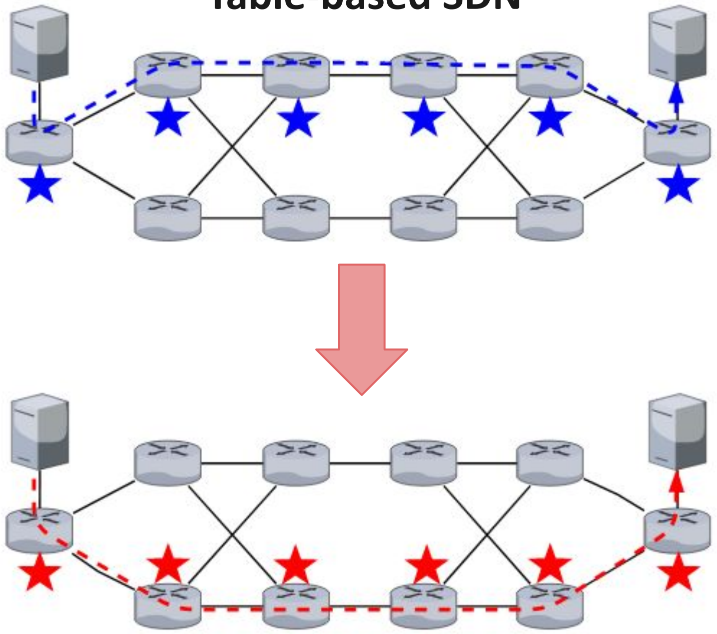
Table-based SDN



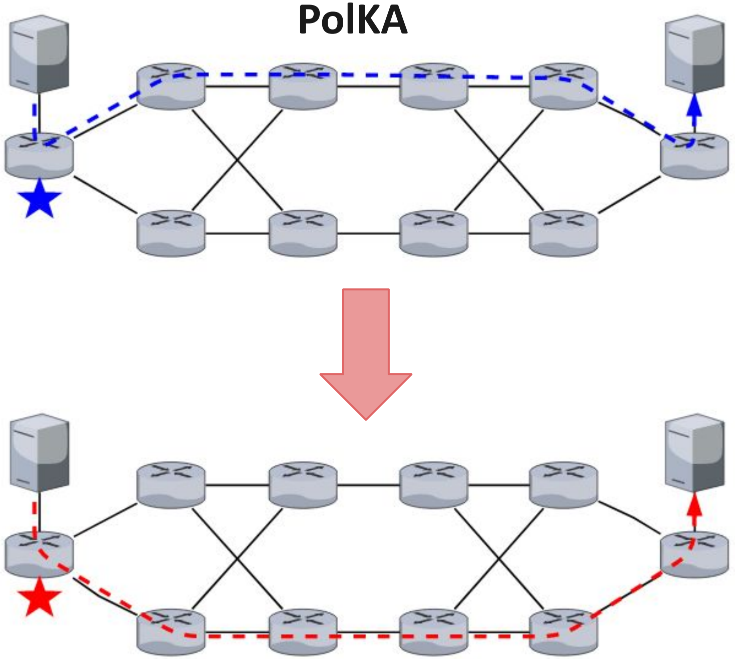
Is PolKA Scalable ?

- **Number of flow entries:**
  - Communication states stored only at the **edges** for encapsulation
  - No need to update all the tables along the path

Table-based SDN



PolKA



Is PolKA Scalable With the Number of Nodes?

- **Length of the *routeID*: *len(R)***

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

Where:

$R = Route$

$N = Number\ of\ hops$

$s_i = nodeID\ polynomial$

- We select *nodeIDs* with the lowest possible degree
- Worst case for data center and WAN topologies (NetSoft 2020)

Is PolKA Scalable With the Number of Nodes?

- **Length of the *routeID*: *len(R)***

  - We select *nodeIDs* with the lowest possible degree
  - Worst case for data center and WAN topologies (NetSoft 2020)

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 Length in Practice

PolKA (RouteID length) = $\text{CRC}_{\text{degree}}$ * hops

Polynomial degree	Number of irreducible polynomials
8	30
16	4080
32	134,215,680

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

RouteID Length in Practice

PolKA (RouteID length) = $\text{CRC}_{\text{degree}}$ * hops

Scheme	**Unit size (per hop)**	**Example of overhead (10 hops)**
MPLS	4 B	10 labels = 40 Bytes
SR-MPLS	4 B	10 segments = 40 Bytes
SRv6	16 B	10 segments = 160 Bytes
PolKA (CRC16)	***2 B***	***10 hops = 20 Bytes***
PolKA (CRC32)	***4 B***	***10 hops = 40 Bytes***

Agenda

- Motivation
- Proposal
- Design
- **Prototype**
- Applications
- Conclusions and future works

Timeline

PolKA received the 2021 Google Research Scholar Award

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

ONDM paper Deploy @RARE

AMS
POZ
FRA
BUD
GÉANT
RARE

Hardware prototype in Tofino

Integration 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

PolKA@panrg IETF113

Lightning Talk Path Aware Networking RG

PolKA@Global P4 Lab

PolKA deployment @Caltech SDN Lab

PolKA Demo at SC-22

2023/2024

INT-PolKA paper AINA

Extension to inband network telemetry

PoT-PolKA paper IEEE NetSoft/TNSM

Extension to proof-of-transit

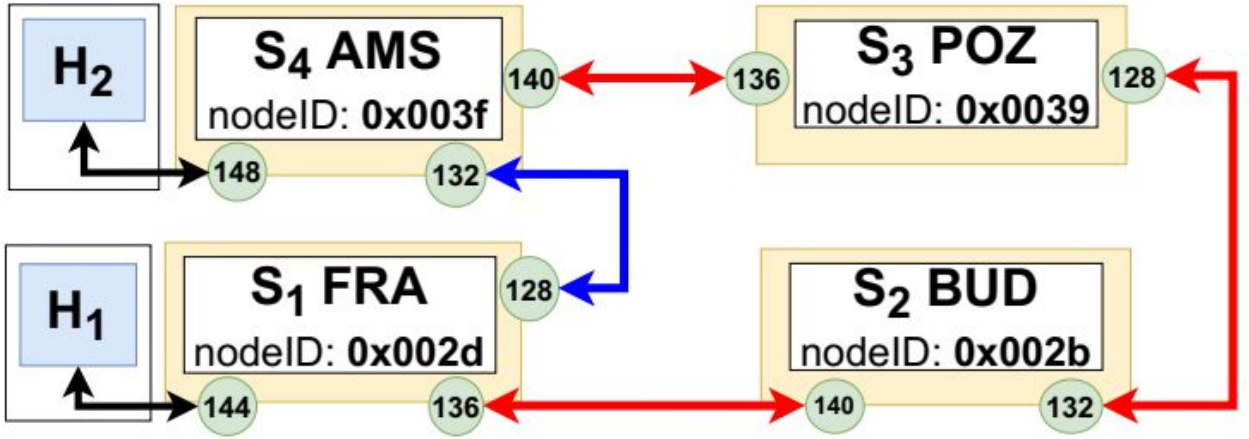
PolKA AI paper Indis

Extension to data science scenario with AI control plane

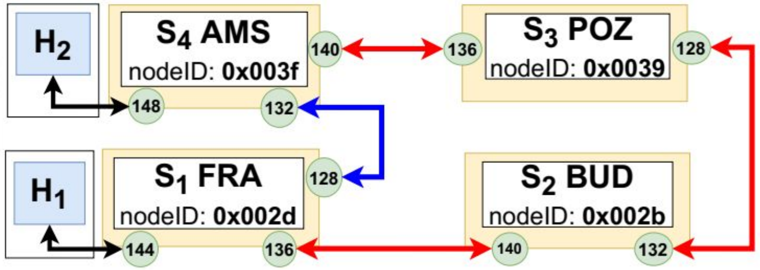
PolKA: Data Plane Prototype

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

AMS
POZ
FRA
BUD
GÉANT
RARE

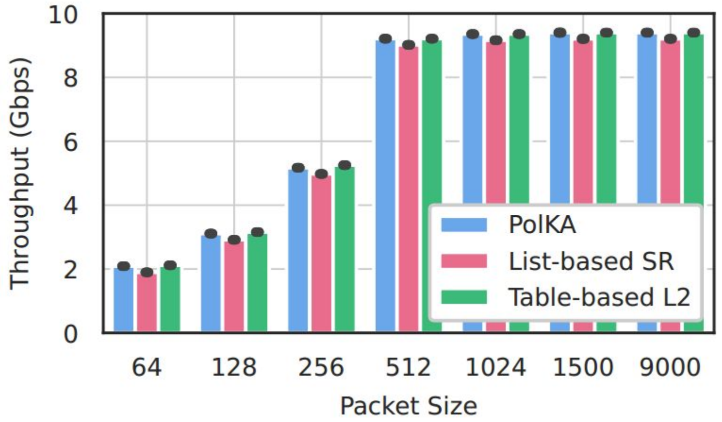


Experiments

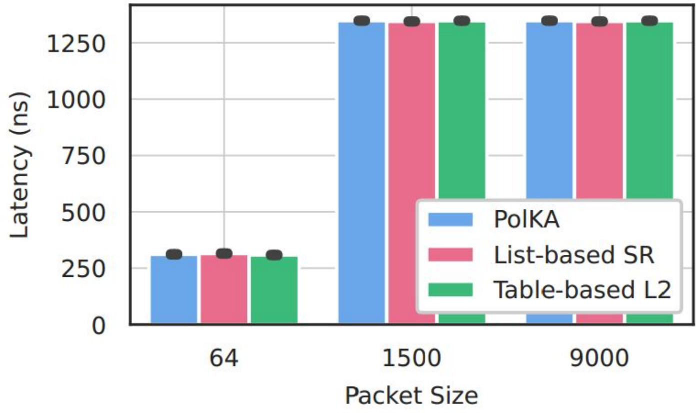


PolKA's performance matches traditional approaches.

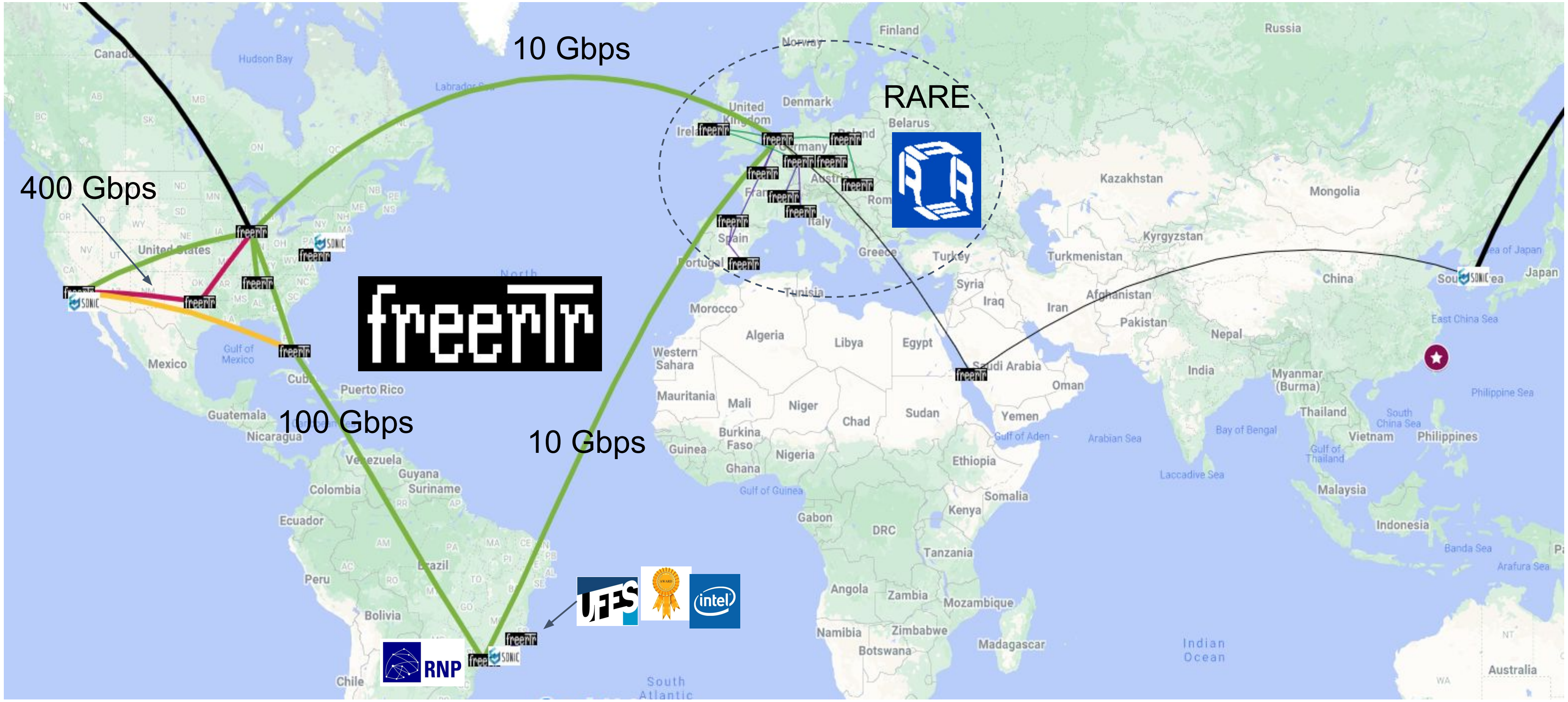
- **Throughput** (S1-S2-S3-S4):
  - High throughput and pps rates



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

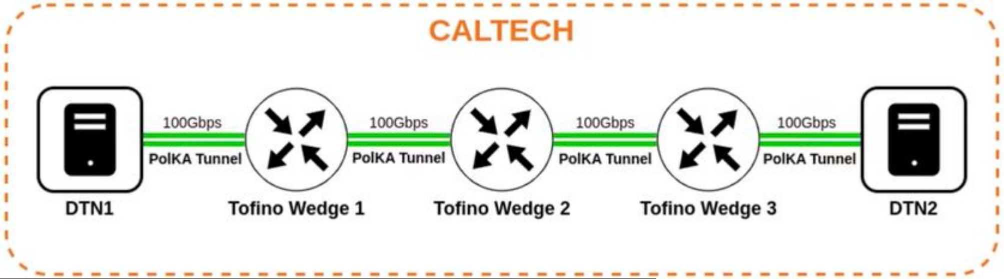


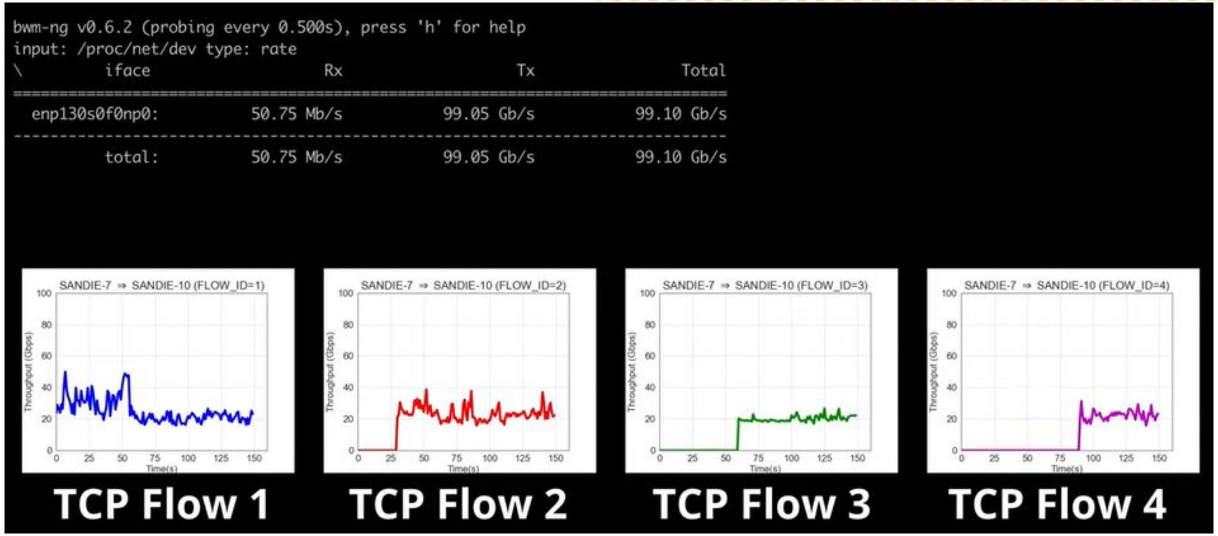
PolKA Integrated in *freerTr* and Deployed at Global P4 Testbed



PolKA Demo@SC2023

- **High Throughput Transfers achieving 100 Gbps line speed**
  - **Caltech P4 lab testbed**
  - Multiple TCP aggregate over PolKA tunnels

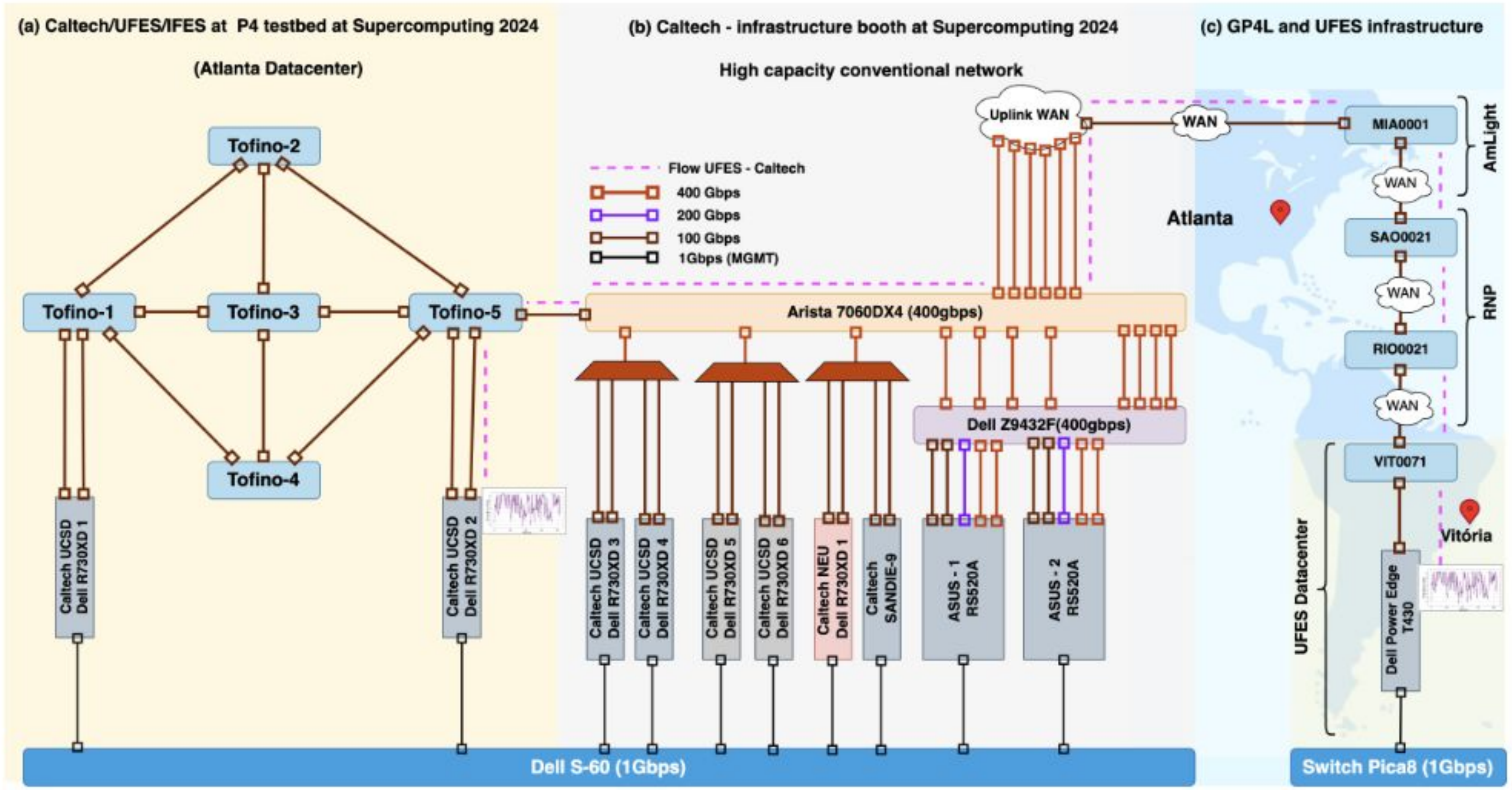




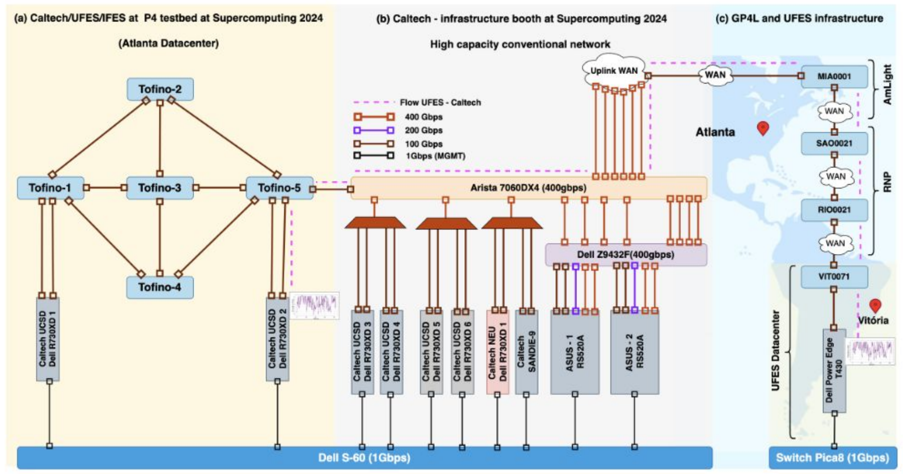
```
bwm-ng v0.6.2 (probing every 0.500s), press 'h' for help
input: /proc/net/dev type: rate
\         iface                   Rx                   Tx                Total
==============================================================================
  enp130s0f0np0:          50.75 Mb/s           99.05 Gb/s           99.10 Gb/s
------------------------------------------------------------------------------
          total:          50.75 Mb/s           99.05 Gb/s           99.10 Gb/s
```

SANDIE-7 → SANDIE-10 (FLOW_ID=1)
SANDIE-7 → SANDIE-10 (FLOW_ID=2)
SANDIE-7 → SANDIE-10 (FLOW_ID=3)
SANDIE-7 → SANDIE-10 (FLOW_ID=4)
Throughput (Gbps)
Time(s)
TCP Flow 1
TCP Flow 2
TCP Flow 3
TCP Flow 4

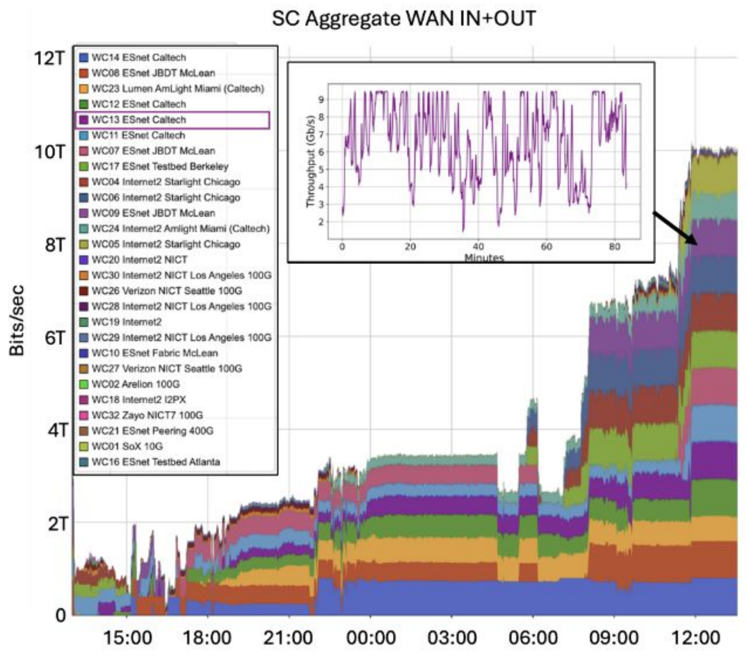
PolKA@Supercomputing 2024


(a) Caltech/UFES/IFES at P4 testbed at Supercomputing 2024
(Atlanta Datacenter)
(b) Caltech - infrastructure booth at Supercomputing 2024
High capacity conventional network
(c) GP4L and UFES infrastructure
Tofino-1
Tofino-2
Tofino-3
Tofino-4
Tofino-5
Flow UFES - Caltech
400 Gbps
200 Gbps
100 Gbps
1Gbps (MGMT)
Uplink WAN
WAN
MIA0001
SAO0021
RIO0021
VIT0071
AmLight
RNP
Atlanta
Vitória
Arista 7060DX4 (400gbps)
Dell Z9432F(400gbps)
Caltech UCSD Dell R730XD 1
Caltech UCSD Dell R730XD 2
Caltech UCSD Dell R730XD 3
Caltech UCSD Dell R730XD 4
Caltech UCSD Dell R730XD 5
Caltech UCSD Dell R730XD 6
Caltech NEU Dell R730XD 1
Caltech SANDIE-9
ASUS - 1 RS520A
ASUS - 2 RS520A
UFES Datacenter
Dell Power Edge T430
Dell S-60 (1Gbps)
Switch Pica8 (1Gbps)

PolKA@Supercomputing 2024



(a) Caltech/UFES/IFES at P4 testbed at Supercomputing 2024
(Atlanta Datacenter)
(b) Caltech - infrastructure booth at Supercomputing 2024
High capacity conventional network
(c) GP4L and UFES infrastructure
Tofino-1
Tofino-2
Tofino-3
Tofino-4
Tofino-5
Flow UFES - Caltech
400 Gbps
200 Gbps
100 Gbps
1Gbps (MGMT)
Uplink WAN
WAN
Arista 7060DX4 (400gbps)
Dell Z9432F(400gbps)
Caltech UCSD Dell R730XD 1
Caltech UCSD Dell R730XD 2
Caltech UCSD Dell R730XD 3
Caltech UCSD Dell R730XD 4
Caltech UCSD Dell R730XD 5
Caltech UCSD Dell R730XD 6
Caltech NEU Dell R730XD 1
Caltech SANDIE-9
ASUS - 1 RS520A
ASUS - 2 RS520A
Dell S-60 (1Gbps)
Atlanta
MIA0001
SAO0021
RIO0021
VIT0071
AmLight
RNP
Vitória
UFES Datacenter
Dell Power Edge T430
Switch Pica8 (1Gbps)



Agenda

- Motivation
- Proposal
- Design
- Prototype and Demonstrations
- **Use Case : Vera Rubin Observatory**
- Conclusions and future works

Our Use Case : The Vera Rubin Observatory

- This is a collaborative use case (Amlight+Caltech+UFES+IFES)
- The telescope delivers 13 GB astronomical images every 27 seconds from Chile to the US Data Facility at SLAC

- Challenges:
  - RTT from the Summit to the USDF is approximately 200+ ms
  - 0.0001% of packet loss will compromise the Rubin Observatory application
- PolKA was adopted in the Amlight pipeline

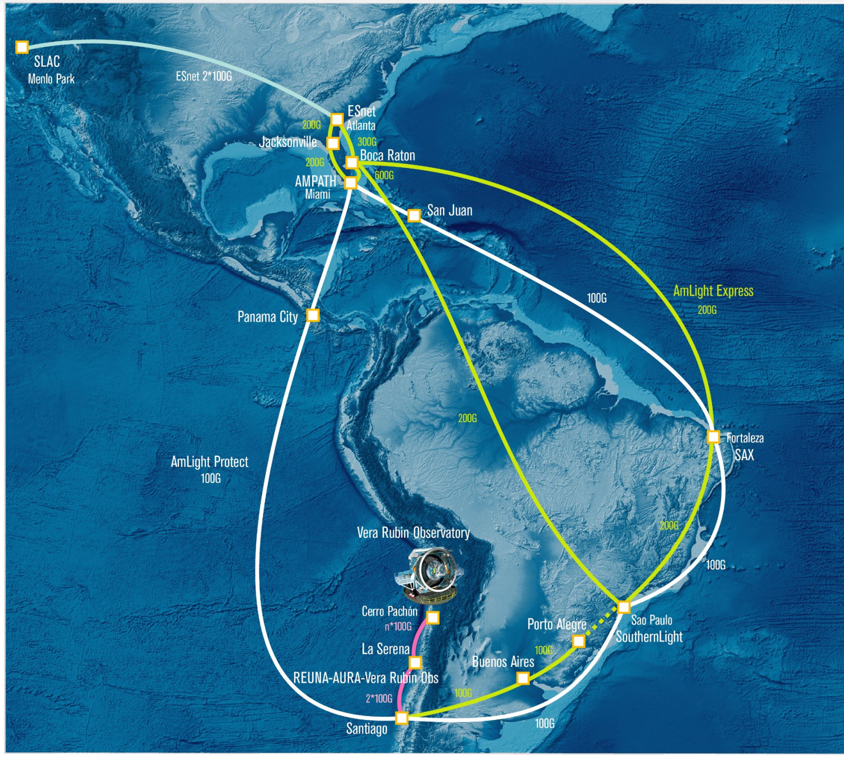
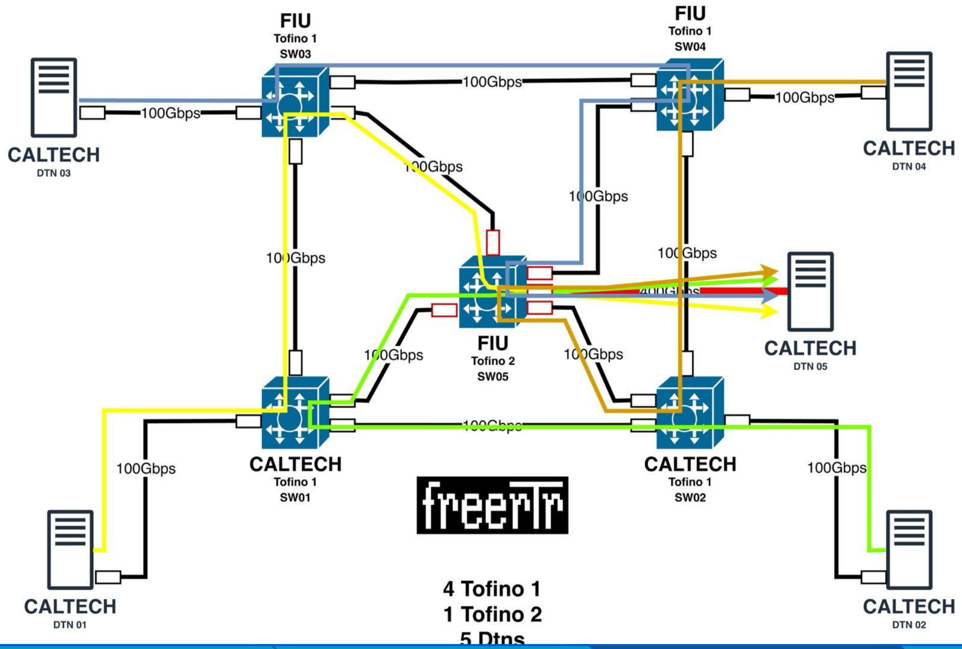


Image from Amlight

Plan for PolKA@ SuperComputing 2025

- Achieve 400 Gbps throughput over PolKA native network
- Achieve 40 Gbps Intercontinental tests for Vera Rubin Use Case
- EBPF implementation to take the edge nodes to the endpoints



Agenda

- Motivation
- Proposal
- Design
- Prototype
- Applications
- **Conclusions and future works**

Takeaway Message of PolKA Approach 1/2

- **PolKA delivers a high-performance network solution** by reusing CRC hardware.
- Supports **line rate performance of packet forwarding** in programmable Switches
  - Validated at multiple testbeds with 10 Gbps, 100 Gbps, and .. 400 Gbps
    - Demonstrations at SC2022, SC2023 and SC2024

Takeaway Message of PolKA Approach 2/2

$$\begin{tikzcd}
\textbf{Network scalability} \arrow[d] & \textbf{Granularity} \arrow[d] & \textbf{Agility} \arrow[d] \\
\textbf{Only at the Edges} & \textbf{Fine granularity} & \textbf{Low latency}
\end{tikzcd}$$

- **Scalable** : Network state is defined **only at the edges**
- Stateless core with tableless nodes enables the selection of any path (**fine granularity** to assign flows and allows TE optimization)
- **Low latency on** path configuration by updating a single table entry at the edge

Future Work

- Potential to help on the HP-WAN mission to support network challenges
  - Supports *high throughput over long distance stable paths*, and fast path reconfiguration to recover from transient failures.
  - Use case of multi-site AI training workloads (in collaboration with Jordi R Giralt from Qualcomm)
    - Distributed AI workloads need synchronized, high-bandwidth connectivity.
    - Working on a framework based on (Quantitative Theory of Bottlenecks Structure) QTBS integrated with PolKA
    - PolKA provides on-demand routing reconfiguration (***any-path***) and policy-based path selection, improving training efficiency.

Selection of Our Recent Publications

- *CRC4EVER: Cyclic Redundancy Check for Enhanced Verification and Efficient Routing (Demo at Sigcomm 2025)*
- *A Path-Aware Routing for Data Intensive Science: Proposal, Deployment and Evaluation in High-Performance Testbed (WGRS 2025)*
- *Transport efficiency for data-intensive science: deployment experiences and bottleneck analysis (An. of Telecomm, 2025)*
- *PINT-BoX: Path-aware networking IN a Tofino BoX (Demo at IEEE NFV/SDN, 2024)*
- *Framework for Integrating Machine Learning Methods for Path-Aware Source Routing (IEEE INDIS@SC, 2024)*
- *PathSec: Path-Aware Secure Routing with Native Path Verification and Auditability (IEEE NFV/SDN, 2024)*
- *PoT-PolKA: Let the Edge Control the Proof-of-Transit in Path-Aware Networks (IEEE TNSM, 2024)*
- *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)
- 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)
- RDNA: Residue-defined networking architecture enabling ultra-reliable low-latency datacenters (IEEE TNSM, 2018)

References

1. Segment Routing RFC
2. BIER RFC
3. PolKA NetSoft 2020 conference paper
4. V. Shoup, A computational introduction to number theory and algebra, 2008.
5. Keyflow 2014 journal paper
6. PolKA ONDM 2021 conference paper
7. RARE website
8. FreeRouter website

Thank you for your 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 CNPq (*Research productivity scholarship), CAPES, FAPES, FAPESP, and RNP.*

INSTITUTO FEDERAL
Espírito Santo

UFES