

## Case Study for Data-Intensive Traffic from Vera Rubin Observatory Supported by Path Aware PolKA Network



*Magno  
Martinello*<sup>1</sup>



*Rafael  
Guimarães*<sup>2</sup>



*Everson  
Borges*<sup>2</sup>



*Daniel  
Ventrone*<sup>12</sup>



*Cristina  
Dominicini*<sup>2</sup>



*Moises R. N.  
Ribeiro*<sup>1</sup>



*Harvey B.  
Newman*<sup>3</sup>



*Jeronimo  
Bezerra*<sup>4</sup>

# Agenda

---

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

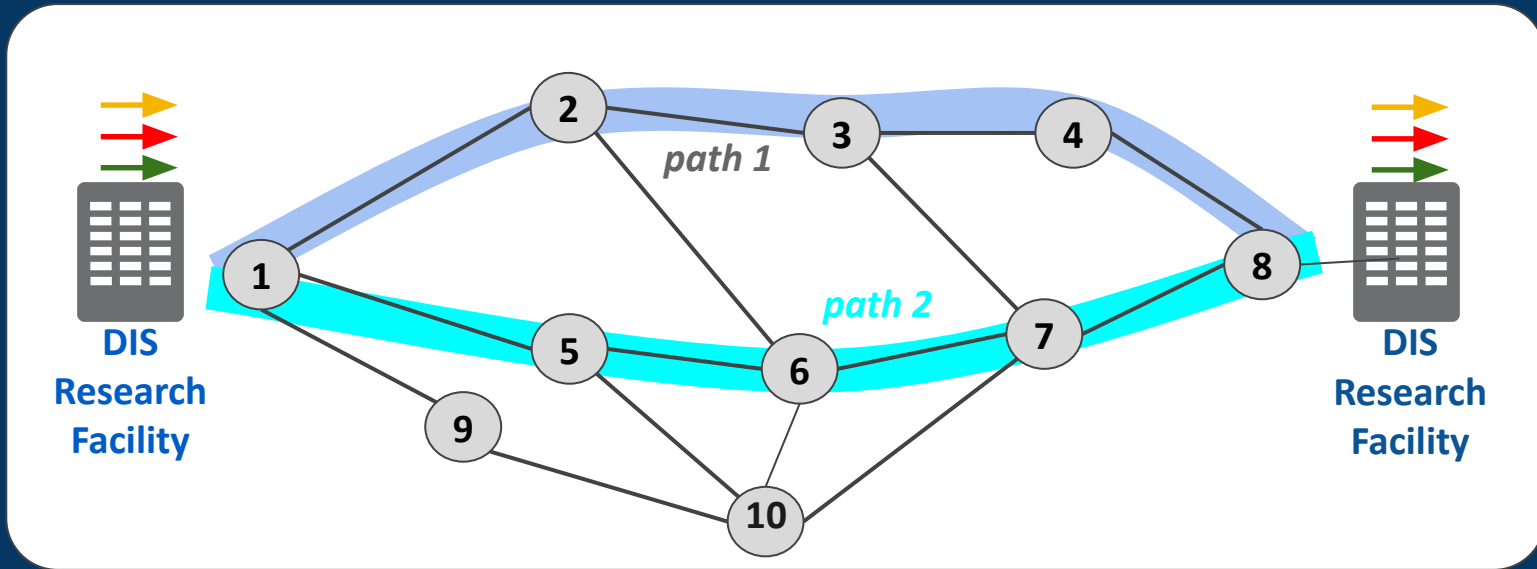
# Motivation

- Global Network Advancement Group (GNA-G) mission:

*to enable ultra-fast, resilient, and adaptive data paths for Data-Intensive Science*

# Data Intensive Science Requirements

- High Performance WANs
- Big Data Flows
  - High throughput over long distances



# Motivation

- Global Network Advancement Group (GNA-G) mission: to enable ultra-fast, resilient, and adaptive data paths for *Data-Intensive Science*
- 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

Large number of  
states

Limited capacity  
of switch tables

Latency for path  
configuration

Network  
scalability

Granularity

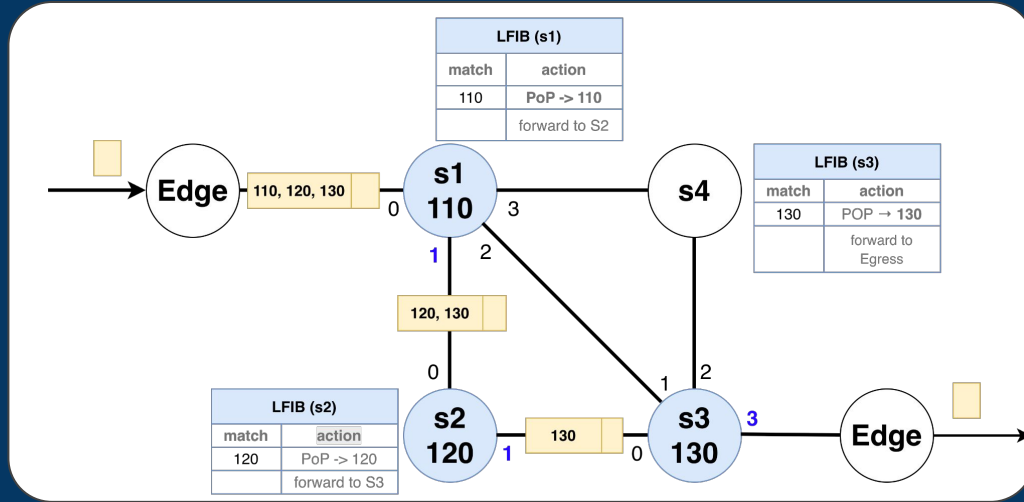
Agility

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

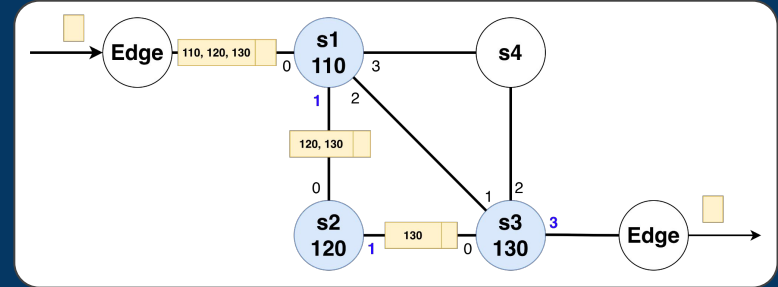
- Traditional way: List-based SR (LSR)
  - Path: a list of ports or a stack of labels.
    - Edge pushes a stack [110,120,130]
    - Each node matches on the top label performs a pop





# 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

- 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 its next hop with its own key.

# 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

# Agenda



- Motivation
- Proposal
- Design
- Deployment
- Demonstration
- Conclusions

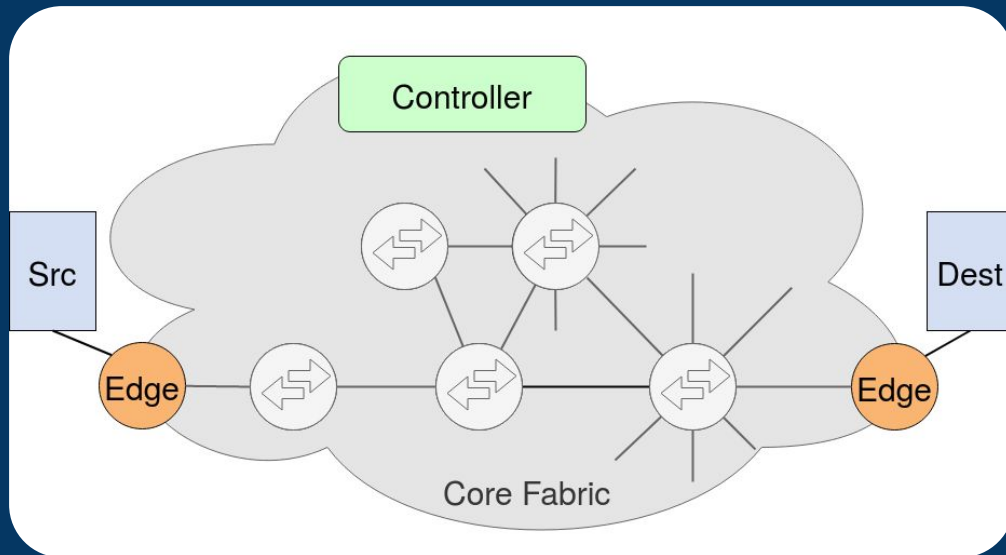
# How Does PolKA Work?

- 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):

$$\text{portID} = \langle \text{routeID} \rangle_{\text{nodeID}}$$

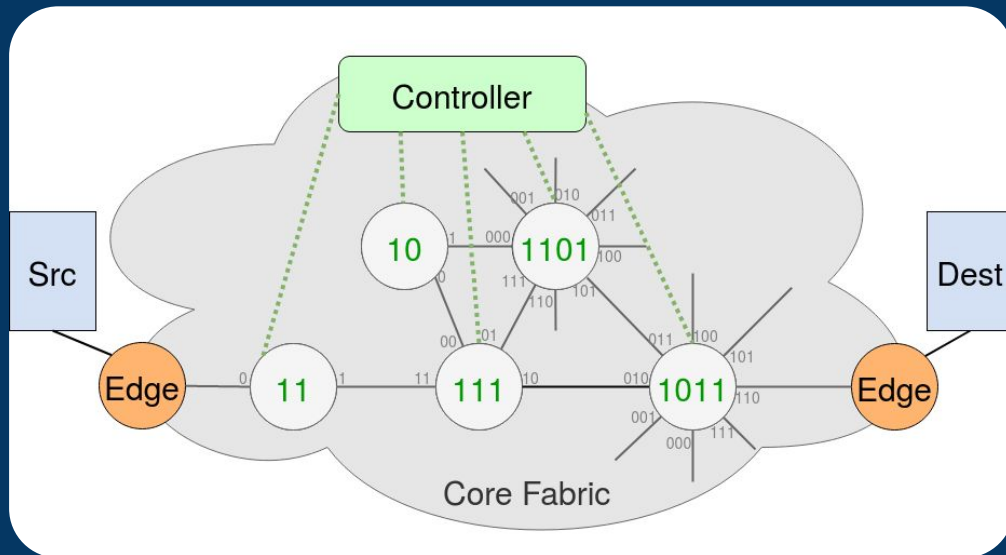
# How Does PolKA Work?

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



# How Does PolKA Work?

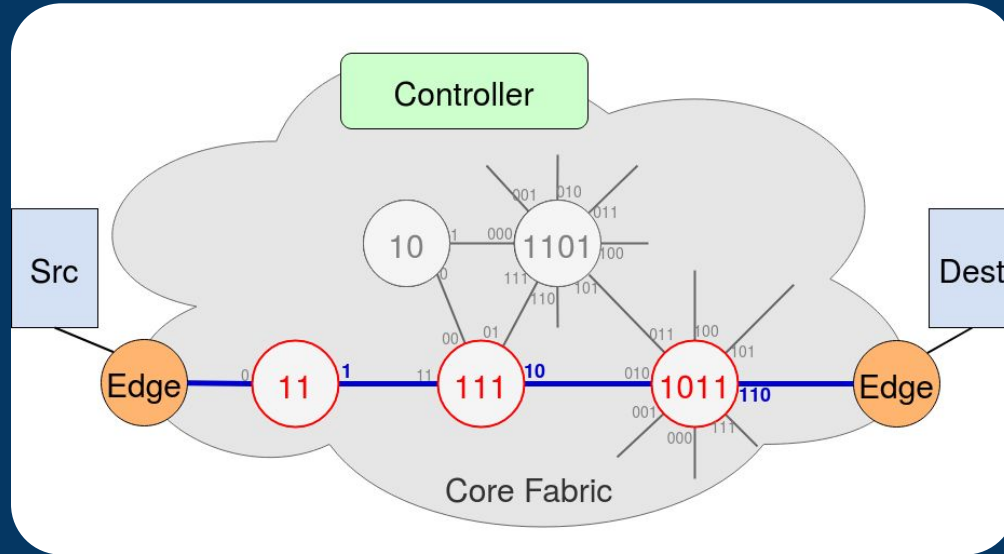
- In a network configuration phase, the **Controller** assigns irreducible polynomials to core switches (**nodeIDs**).
- Port labels are represented as binary polynomials (**portIDs**).





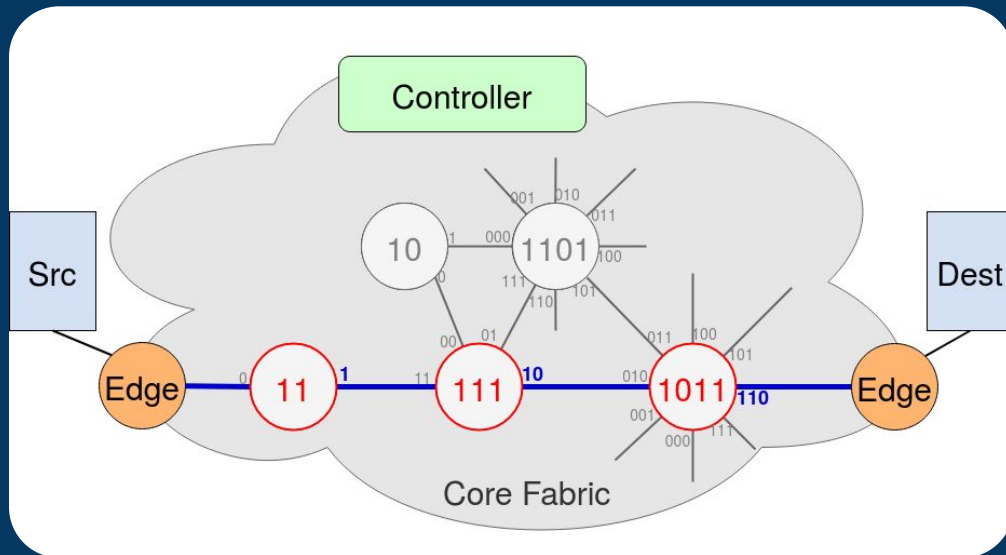
# How Does PolKA Work?

- 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:  $O(\text{len}(M)^2)$ , where  $M(t) = \prod_{i=1}^N s_i(t)$

$N$  = number of hops

$M$  = product of nodeIDs

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

$$001 = X \bmod 0011$$

$$010 = X \bmod 0111$$

$$110 = X \bmod 1011$$

**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*

$$t^4 \equiv 1 \bmod (t + 1)$$

$$t^4 \equiv t \bmod (t^2 + t + 1)$$

$$t^4 \equiv (t^2 + t) \bmod (t^3 + t + 1)$$

$$t^4 = 10000$$

# How Does PolKA Work?

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

**R = 10000**

*routeID*

- Forwarding operations along the path :

**portID = <routeID><sub>nodeID</sub>**

$$\begin{aligned} 001 &= \langle 10000 \rangle_{0011} \rightarrow 10000 \bmod 0011 \\ 010 &= \langle 10000 \rangle_{0111} \rightarrow 10000 \bmod 0111 \\ 110 &= \langle 10000 \rangle_{1011} \rightarrow 10000 \bmod 1011 \end{aligned}$$

*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*

$$t^4 \equiv 1 \bmod (t + 1)$$

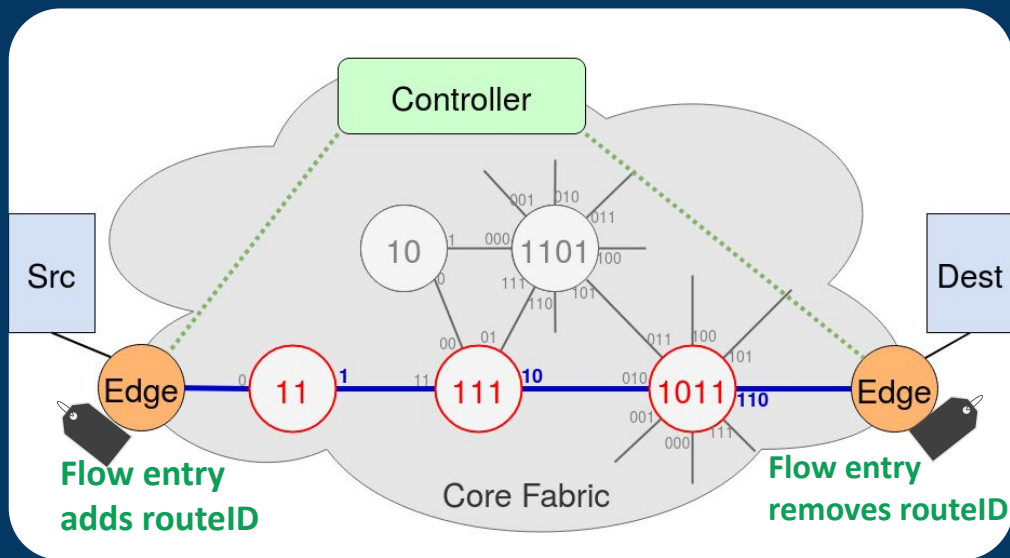
$$t^4 \equiv t \bmod (t^2 + t + 1)$$

$$t^4 \equiv (t^2 + t) \bmod (t^3 + t + 1)$$

$$t^4 = 10000$$

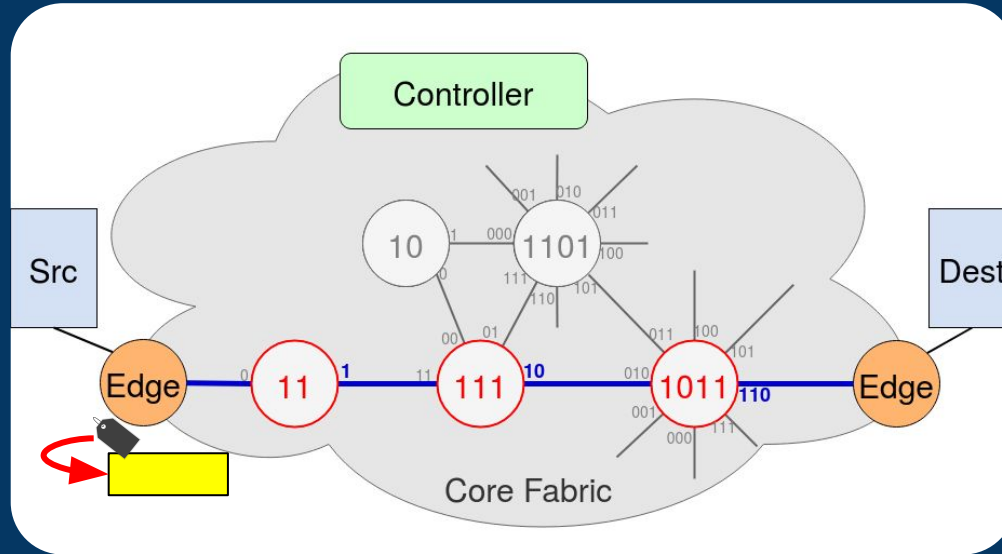
# How Does PolKA Work?

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



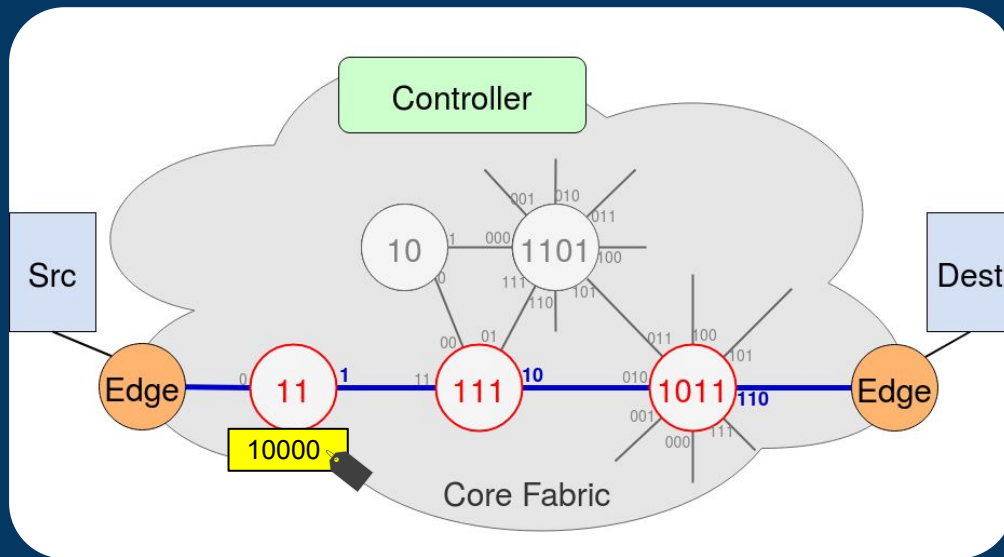
# How Does PolKA Work?

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



# How Does PolKA Work?

- Forwarding using **mod** operation:  $\langle 10000 \rangle_{0011} = 1 \rightarrow \text{output port}$
- No *routeID* rewrite! No tables in the core nodes!

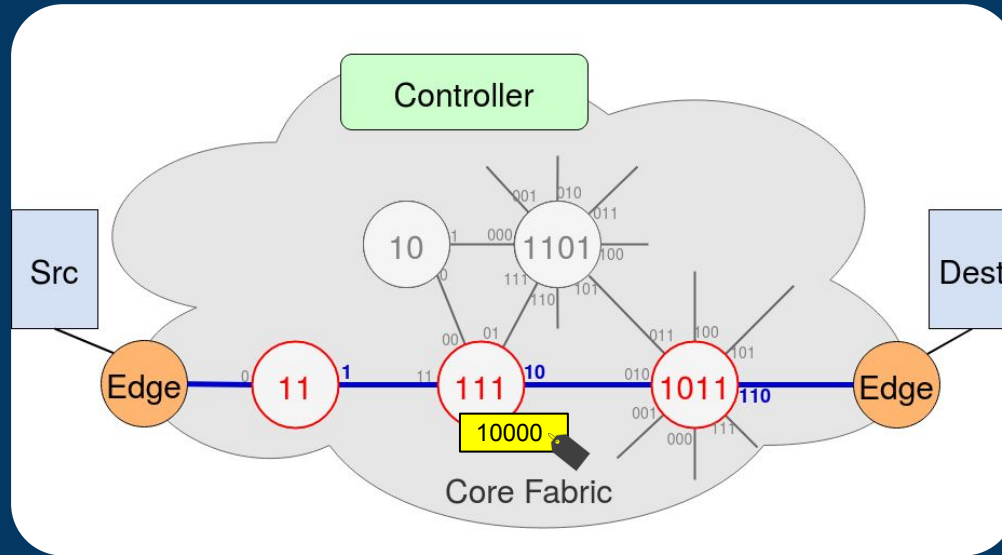


10000 mod 11 GF(2)

```
-----  
      10000  
⊕  11000  
-----  
      01000  
⊕   1100  
-----  
      0100  
⊕    110  
-----  
       10  
⊕     11  
-----  
portID:  1
```

# How Does PolKA Work?

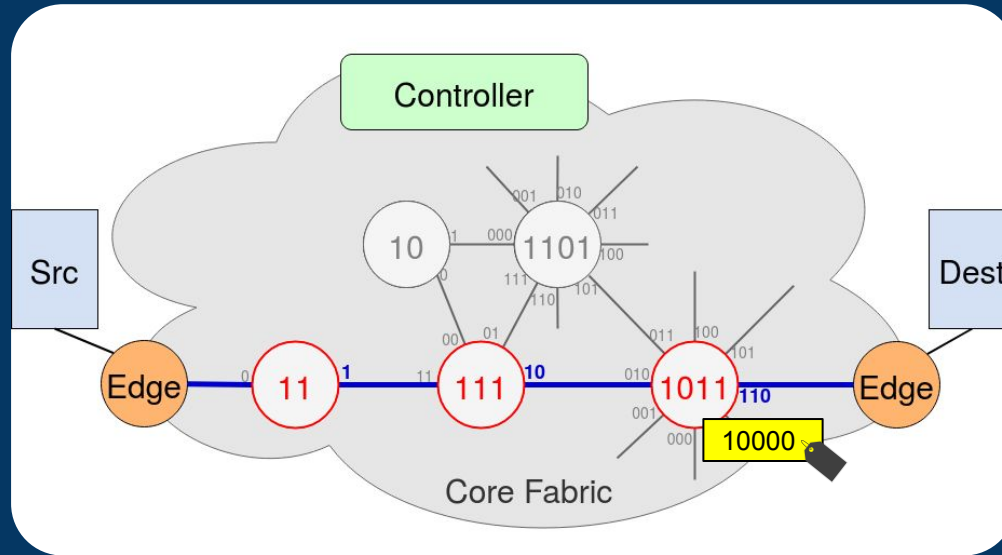
- Forwarding using **mod** operation:  $\langle 10000 \rangle_{0111} = 10 \rightarrow \text{output port}$
- <No *routeID* rewrite! No tables!





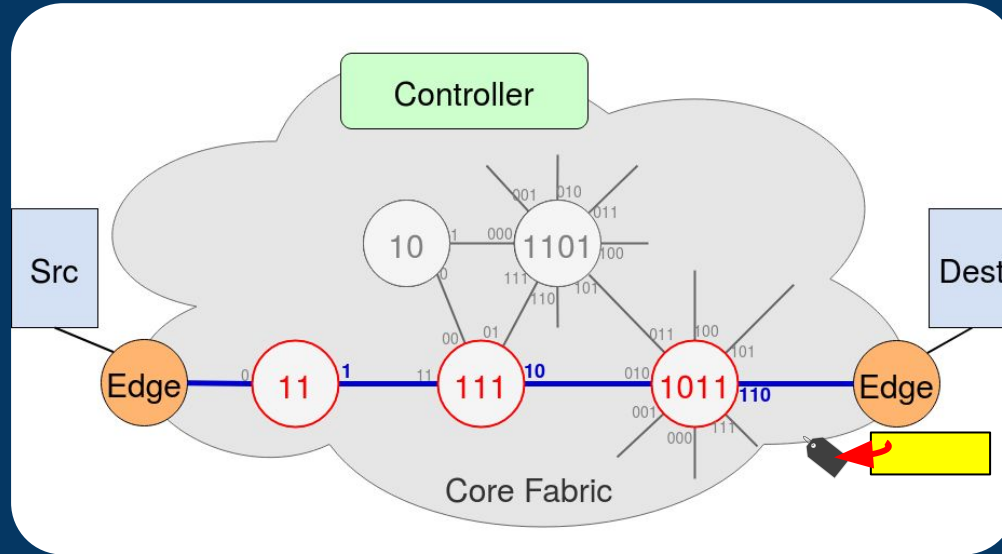
# How Does PolKA Work?

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



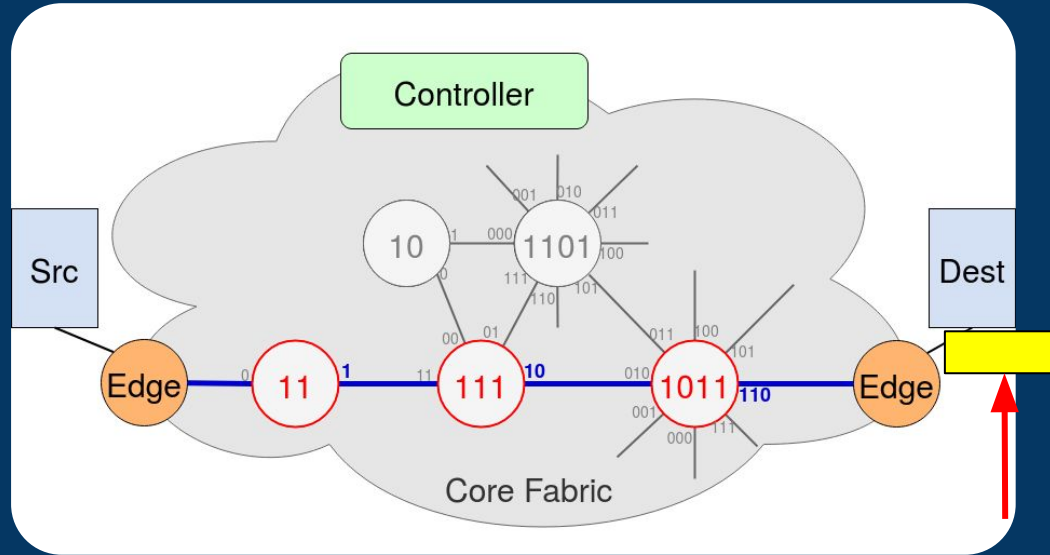
# How Does PolKA Work?

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



# How Does PoKA 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.  $G = \text{nodeID} = \mathbf{01011}$ , portanto  $r = \deg(G) = \mathbf{3}$

2.  $D = \text{routeID} \div 2^r = 100101\mathbf{444} \gg \mathbf{3} = 100101$

(SHIFT RIGHT)

3.  $\text{dif} = \text{routeID} - D \cdot 2^r = 100101111 \oplus (100101 \ll \mathbf{3})$   
 $= 100101111 \oplus 100101\mathbf{000} = 111$

(SHIFT LEFT, XOR)

4.  $R = \langle D \cdot 2^r \rangle_G = \langle 100101\mathbf{000} \rangle_{(01011)} = 110$

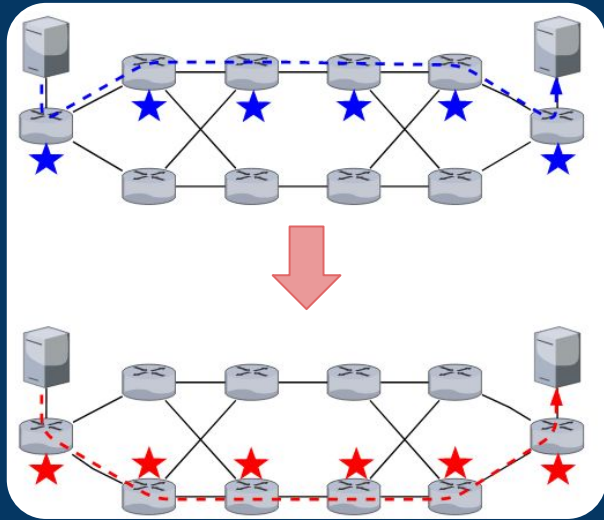
(CRC)

5.  $\text{portID} = \text{dif} \oplus R = 111 \oplus 110 = 001$

(XOR)

# Is PolKA Scalable ?

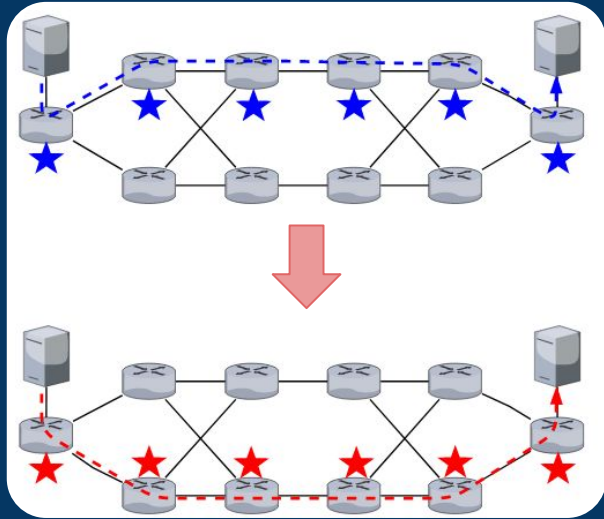
Table-based SDN



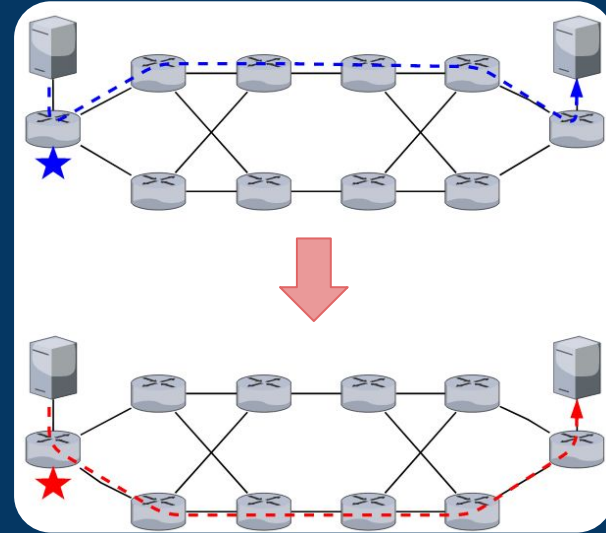
# Is PolKA Scalable ?

- **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)
<i>Two-tier S16 L16*</i>	24	3	32	<b>21</b>
<i>Fat-tree 16 pods</i>	16	5	320	<b>55</b>
<i>ARPANET</i>	4	7	20	<b>42</b>
<i>GEANT2</i>	8	7	30	<b>49</b>

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



# Is PolKA Scalable With the Number of Nodes?

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

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

# Comparison to existing protocols

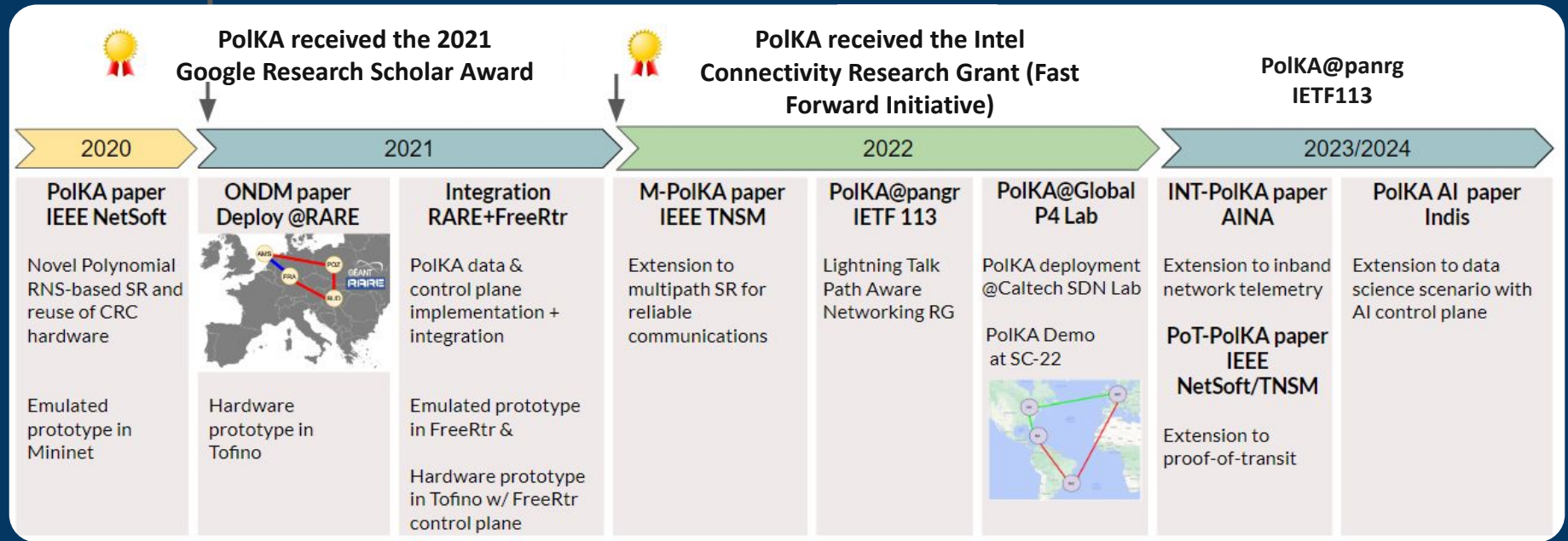
$$\text{PolKA (RouteID length)} = \text{CRC}_{\text{degree}} * \text{hops}$$

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
<i>PolKA (CRC16)</i>	<i>2 B</i>	<i>10 hops = 20 Bytes</i>
<i>PolKA (CRC32)</i>	<i>4 B</i>	<i>10 hops = 40 Bytes</i>

# Agenda

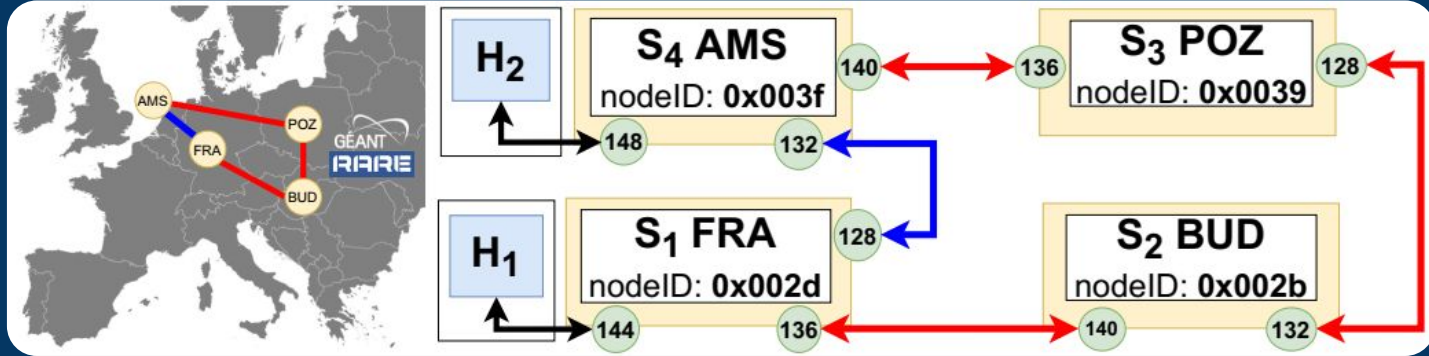
- Motivation
- Proposal
- Design
- **Deployment**
- Demonstration
- Conclusions

# Timeline

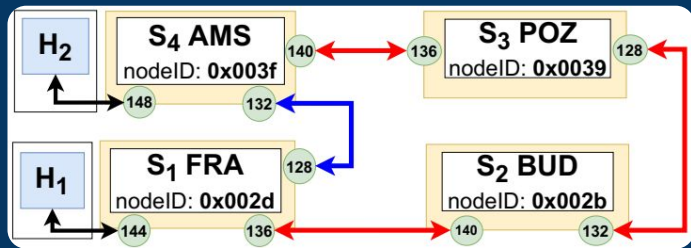


# PolKA: Data Plane Prototype

- 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](#)

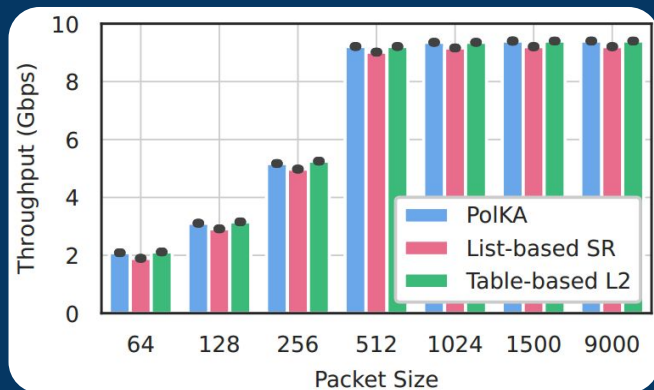


# Experiments

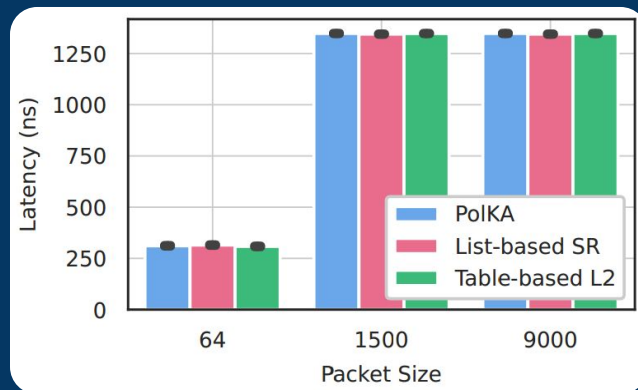


PolKA's performance matches traditional approaches.

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



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

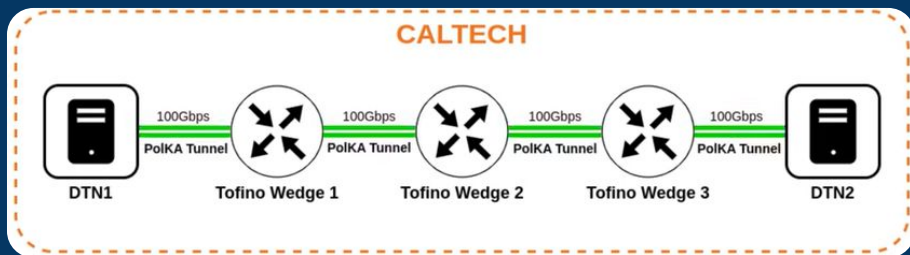


# PolKA Integrated in *freerTr* and Deployed at Global P4 Testbed



# PolKA Demo@SC2023

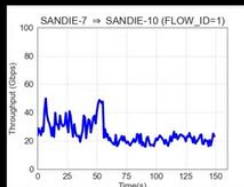
- High Throughput Transfers achieving 100 Gbps line speed
  - Caltech P4 lab testbed
  - Multiple TCP aggregate over PolKA tunnels



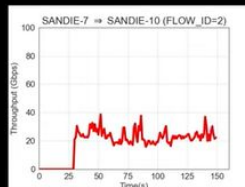
bwm-ng v0.6.2 (probing every 0.500s), press 'h' for help

input: /proc/net/dev type: rate

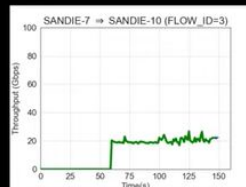
\	iface	Rx	Tx	Total
	enp130s0f0np0:	50.75 Mb/s	99.05 Gb/s	99.10 Gb/s
	total:	50.75 Mb/s	99.05 Gb/s	99.10 Gb/s



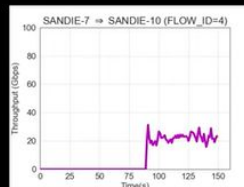
TCP Flow 1



TCP Flow 2



TCP Flow 3



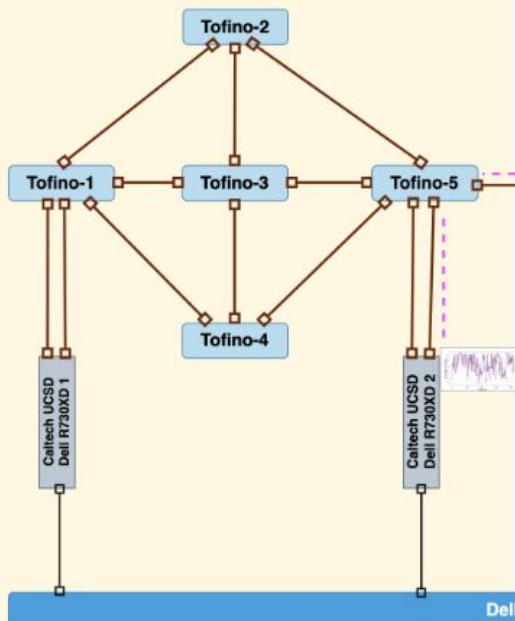
TCP Flow 4



# PolKA@Supercomputing 2024

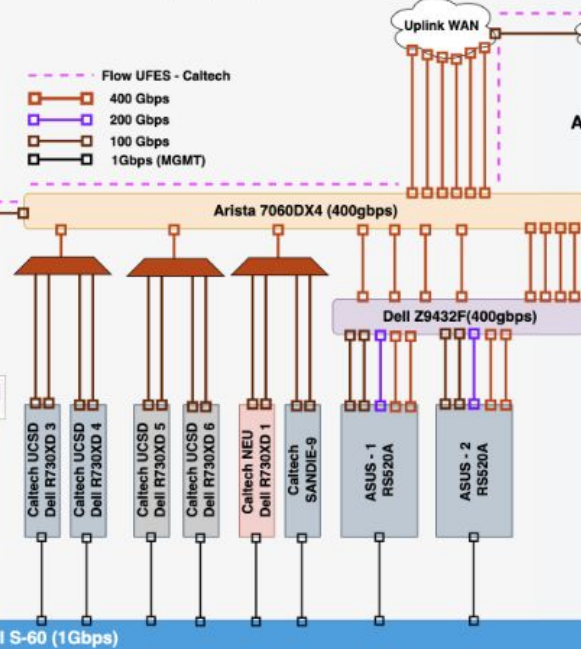
(a) Caltech/UFES/IFES at P4 testbed at Supercomputing 2024

(Atlanta Datacenter)

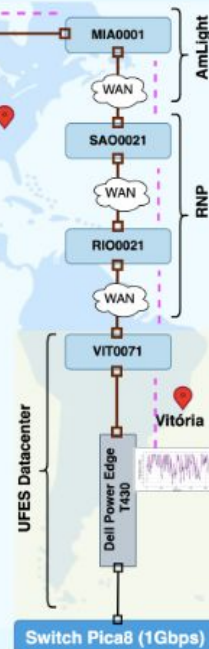


(b) Caltech - infrastructure booth at Supercomputing 2024

High capacity conventional network



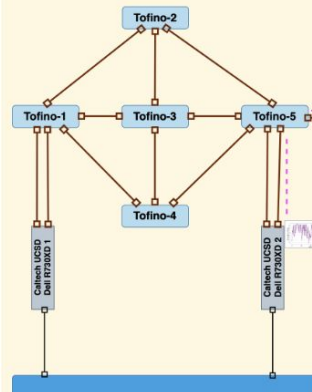
(c) GP4L and UFES infrastructure



# PolKA@Supercomputing 2024

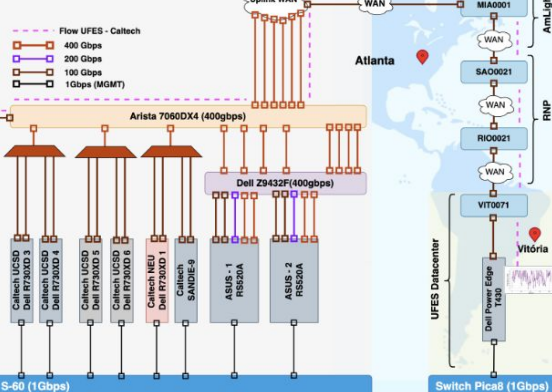
(a) Caltech/UFES/IFES at P4 tested at Supercomputing 2024

(Atlanta Datacenter)

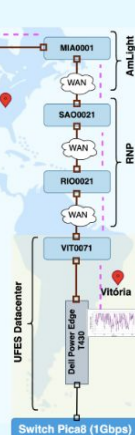


(b) Caltech - infrastructure booth at Supercomputing 2024

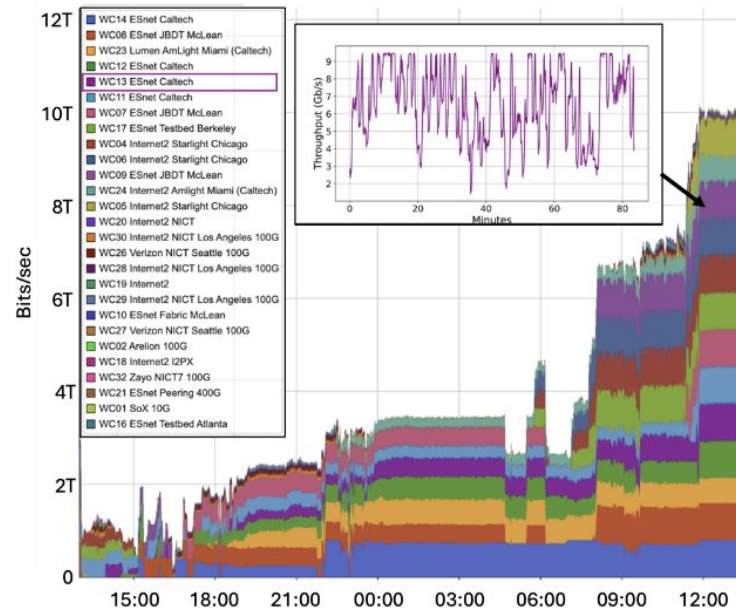
High capacity conventional network



(c) GP4L and UFES infrastructure



SC Aggregate WAN IN+OUT



# Agenda

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

# 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

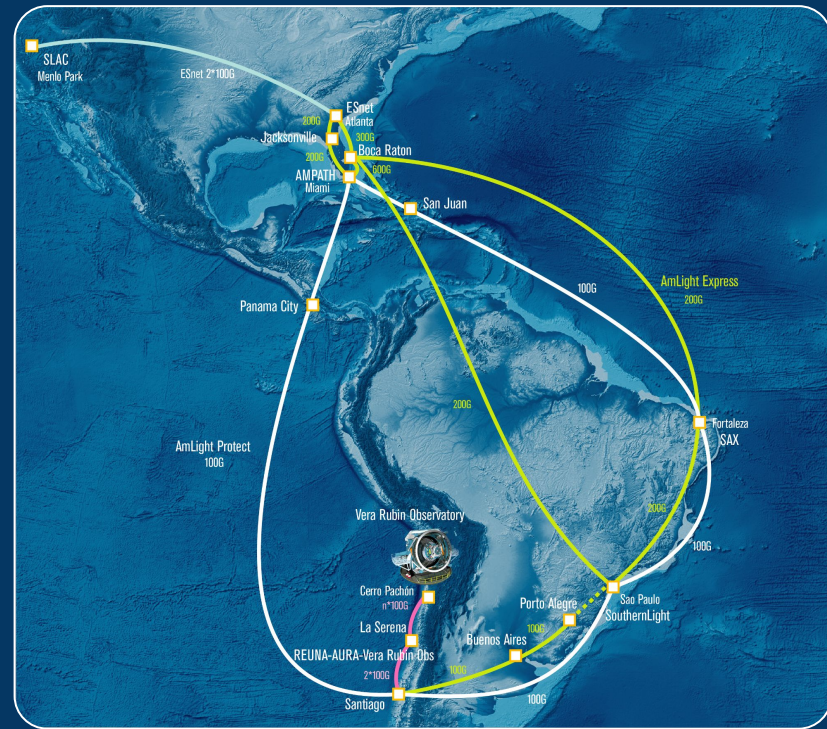
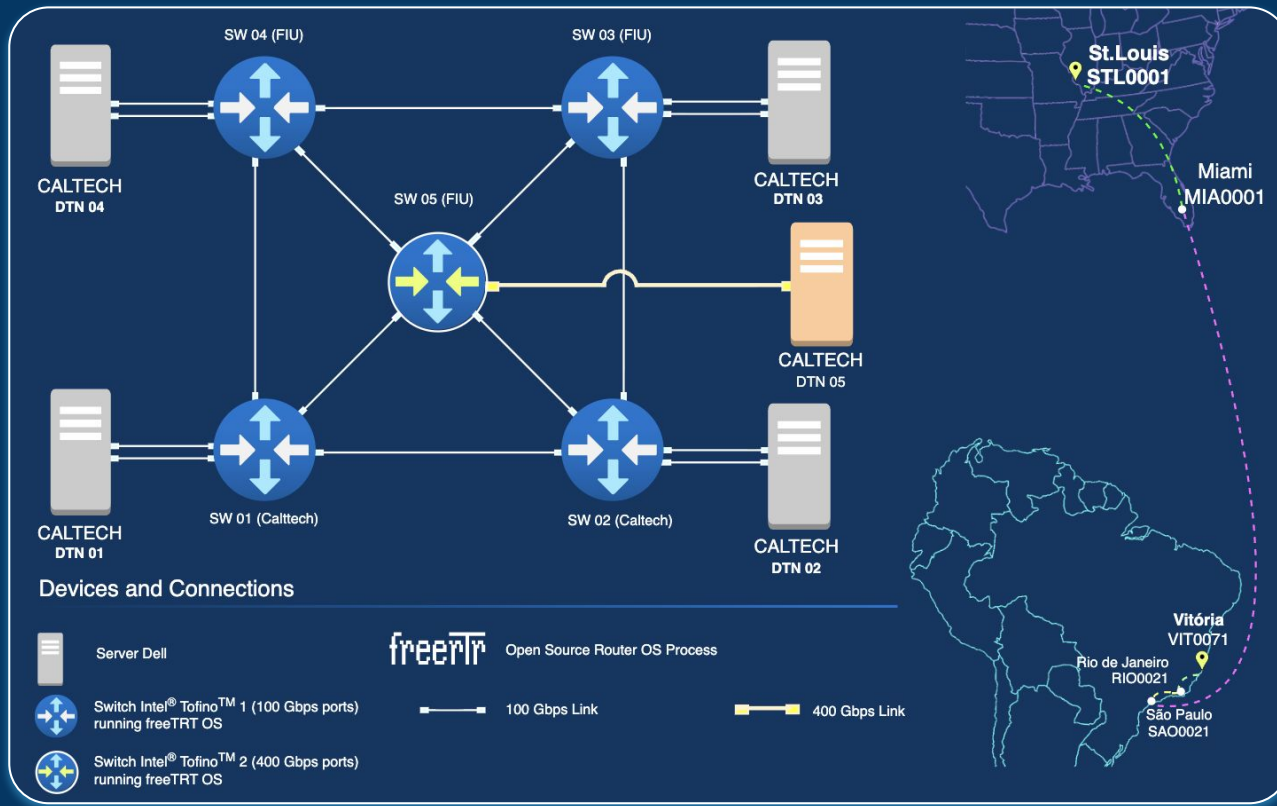


Image from Amlight

# PolKA@SuperComputing 2025





# PolKA@SuperComputing 2025

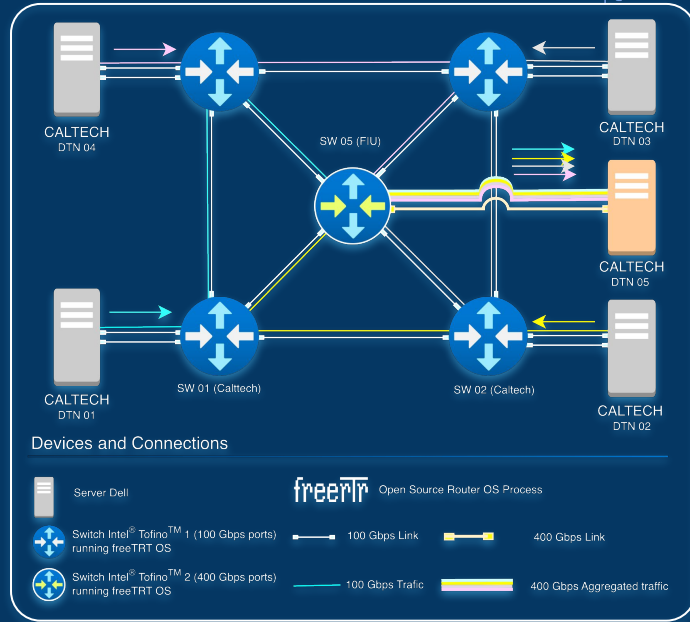
bwm-ng v0.6.3 (probing every 0.500s), press 'h' for help

input: /proc/net/dev type: rate

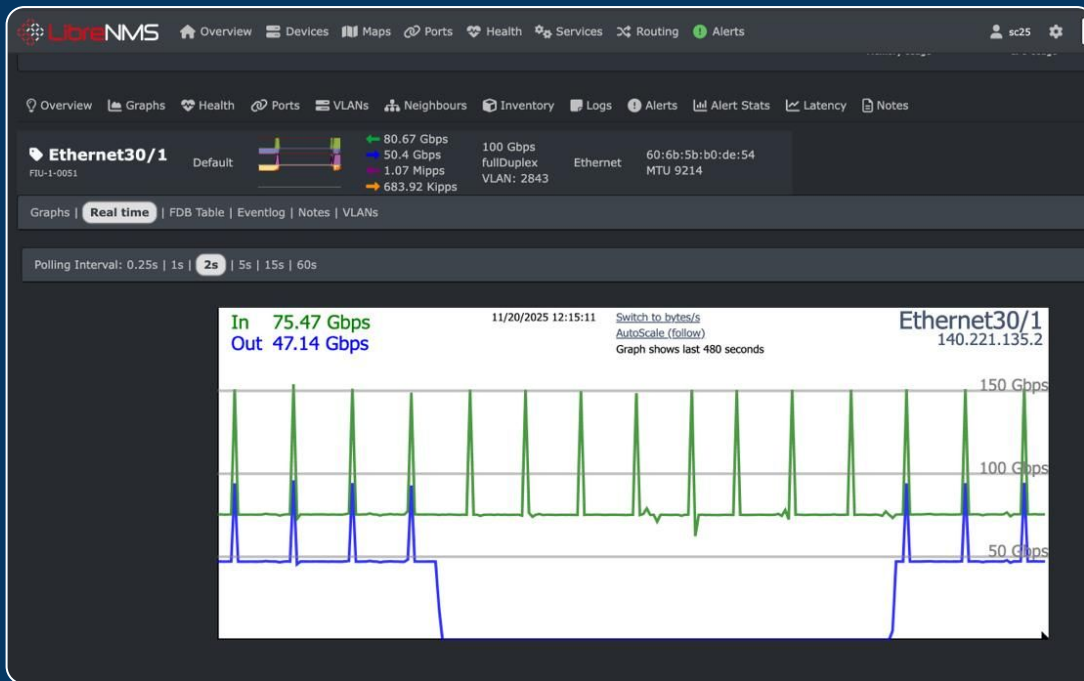
iface	Rx		Tx		Total	
lo:	0.00	b/s	0.00	b/s	0.00	b/s
enp33s0f0:	0.00	b/s	0.00	b/s	0.00	b/s
enp193s0np0:	142.64	Gb/s	97.09	Gb/s	239.73	Gb/s
enp161s0np0:	0.00	b/s	0.00	b/s	0.00	b/s
vlan1321:	0.00	b/s	0.00	b/s	0.00	b/s
docker0:	0.00	b/s	0.00	b/s	0.00	b/s
enp161s0np0.100:	0.00	b/s	0.00	b/s	0.00	b/s
enp161s0np.2830:	0.00	b/s	0.00	b/s	0.00	b/s
total:	142.64	Gb/s	97.09	Gb/s	239.73	Gb/s

*New deployment in Tofino 2 switches*

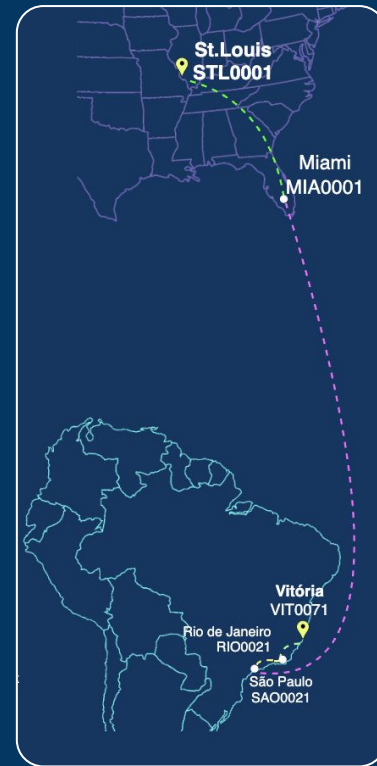
*PolKA delivers a line rate throughput at the Caltech Booth@SC25*



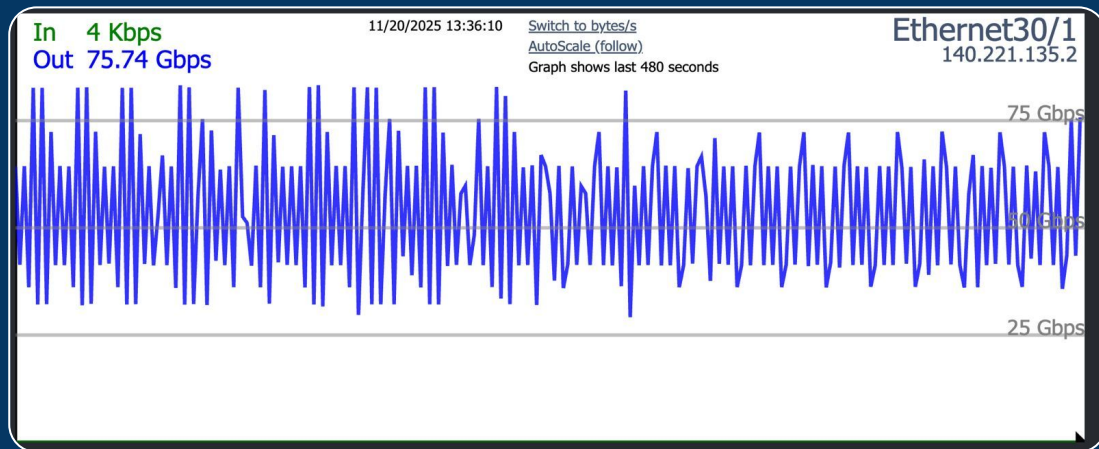
# PolKA@SuperComputing 2025



*PolKA delivered 5 times more throughput than SC24 achieving high-performance at long distance from Caltech Booth@SC25 to UFES via Amlight and RNP*



# PolKA@SuperComputing 2025

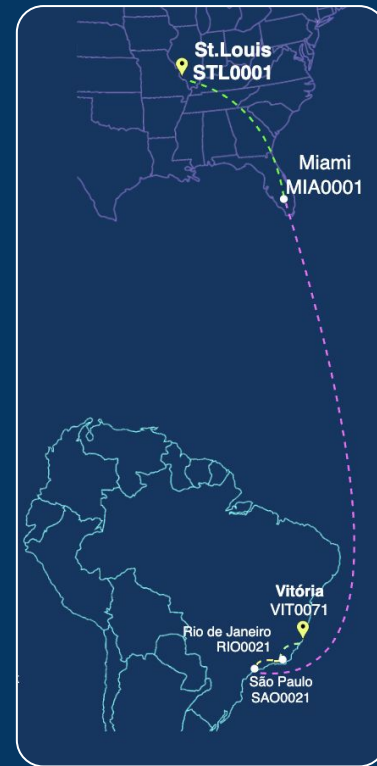


bwm-ng v0.6.2 (probing every 0.500s), press 'h' for help

input: /proc/net/dev type: rate

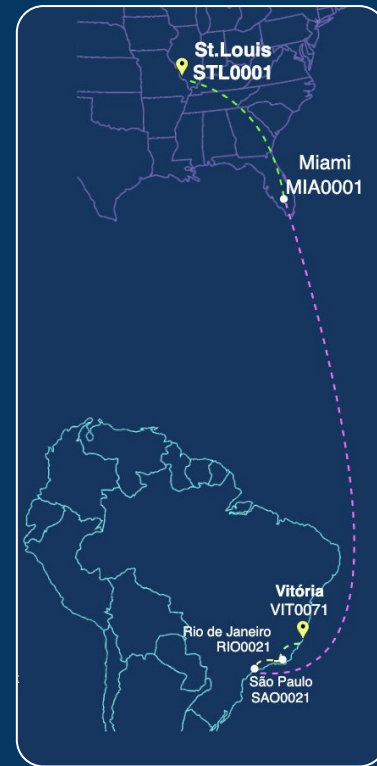
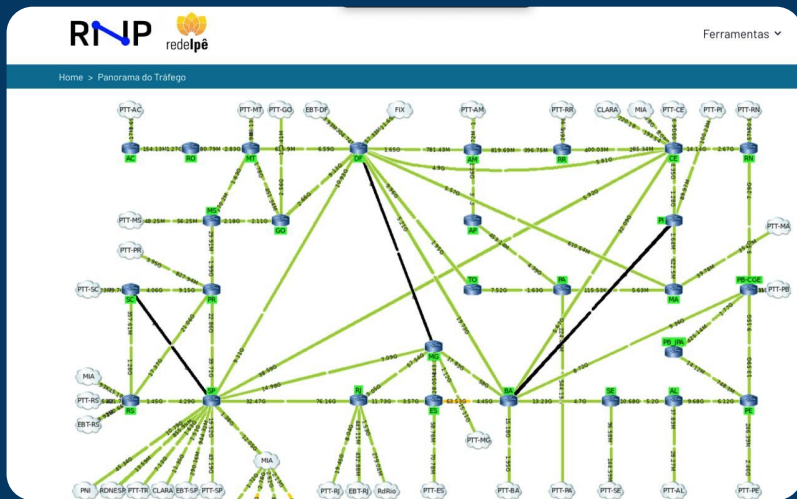
iface	Rx	Tx	Total
enp3s0f1:	54.18 Gb/s	9.44 kb/s	54.18 Gb/s
total:	54.18 Gb/s	9.44 kb/s	54.18 Gb/s

*PolKA delivered 5 times more throughput than SC24 achieving high-performance at long distance from Caltech Booth@SC25 to UFES via Amlight and RNP*





# PolKA@SuperComputing 2025



*PolKA delivered 5 times more throughput than SC24 achieving high-performance at long distance from Caltech Booth@SC25 to UFES via Amlight and RNP*

# 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, SC2024 and SC2025

# Recap of demonstrations Caltech@SuperComputing



- **Big data streams at line rate 100 Gbps**
  - PolKA@ Caltech P4 lab testbed
  - Multiple aggregated traffic flows transported by PolKA tunnels



- **Multiple big data streams achieving 200 Gbps**
  - PolKA@ Caltech P4 lab testbed at the SC 23



- **Novelty to be demonstrated:**
  - ***Path aware networking*** for data intensive traffic flows with highly adaptable, dynamic traffic steering and agile path reconfiguration
  - Traffic engineering with ***optimal traffic flow*** allocation



- **Vera Rubin Observatory (VRO) Use Case**
- **Intradomain traffic at ~250 Gbps in Programmable Dataplanes**

# Takeaway Message of PolKA Approach 2/2

Network  
scalability

Granularity

Agility

Only at the  
Edges

Fine granularity

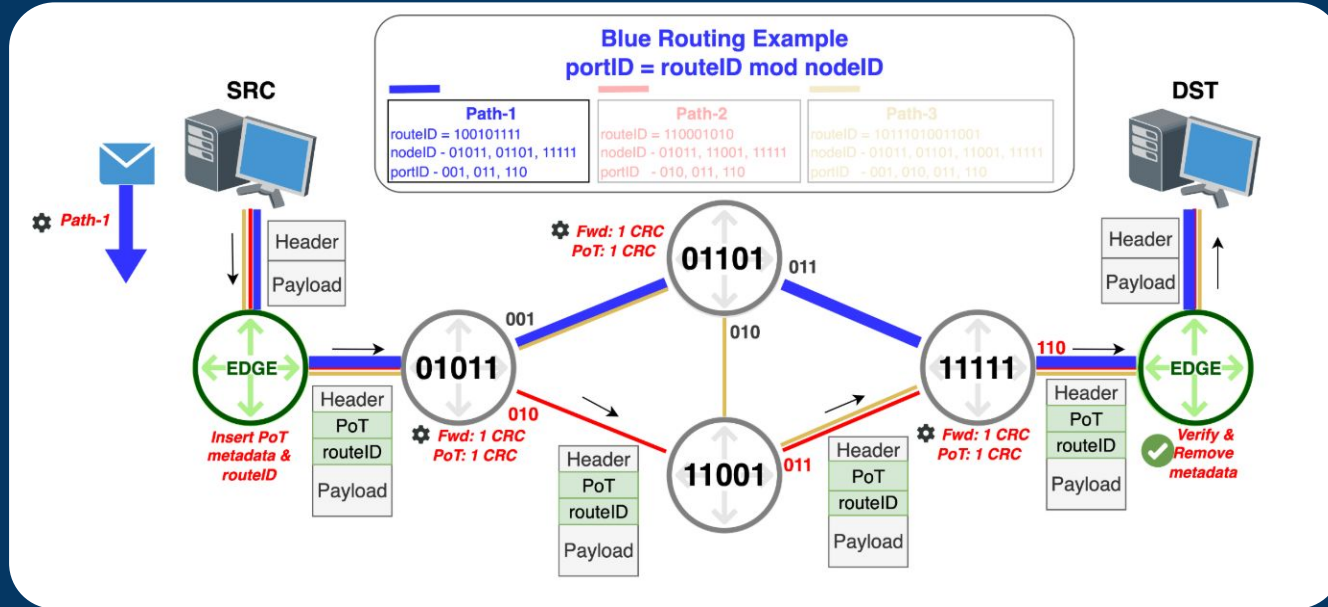
Low latency

- **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

# Applications in different domains

- Security: Sovereign paths
  - Paths with a proof-of-transit signature
  - Telefonica is interested in path anonymization properties (e.g ToR)

## CRC4EVER: Cyclic Redundancy Check for Enhanced Verification and Efficient Routing



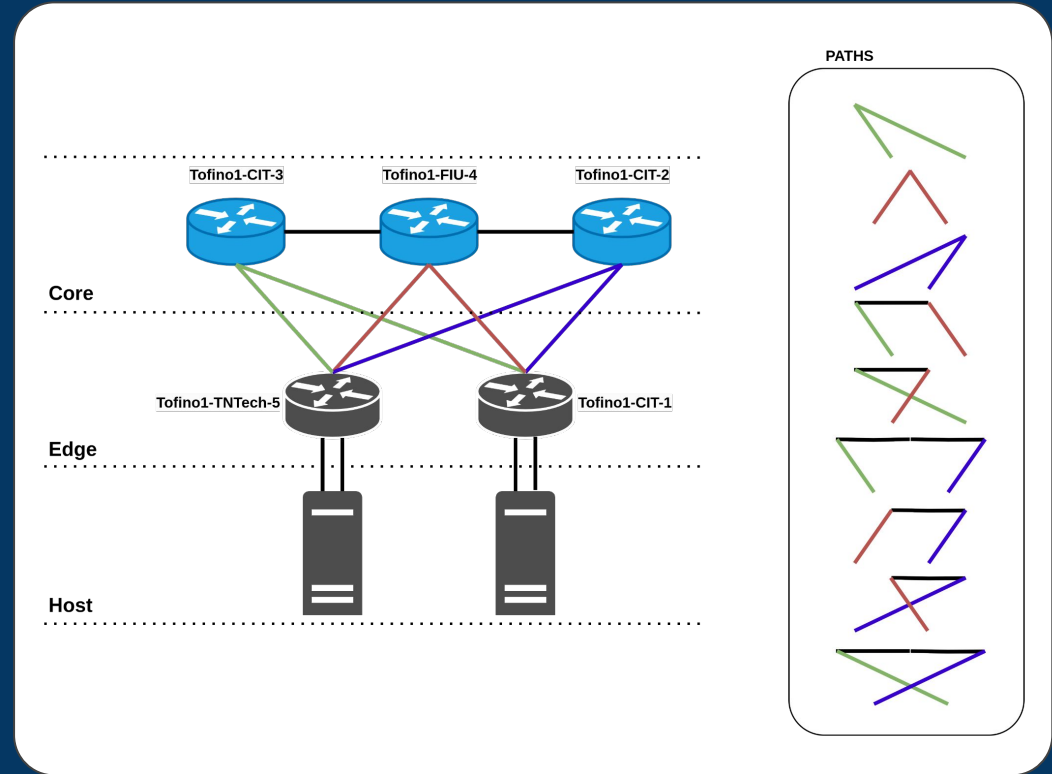
Authors: M. Martinello and Everson Borges et al. IEEE SDN NFV 2024

# Applications in different domains

- **Security : Sovereign paths**
  - Paths with proof-of-transit signatures integrated in a blockchain
  - Telefonica is interested in path anonymization properties (e.g ToR)
- **5G transport network : slices with stringent performance guarantees**
  - Ongoing initiative with Open RAN Brazilian Program
  - Brazilian telecom agency regulator (Anatel) is giving us support
- **HPC : PolKA Architecture design based on fabric stateless core with edges**
  - High performance ethernet alternative to infiniband

# HPC use case : PolKA design based on stateless core fabric with edge intelligence

- PolKA provides a path controlled at the edge to:
- Any nic, any GPU
- Investigate the deployment in broadcom chips
  - Trident4/X has a NPL/P4 pipeline
  - Jericho 2





# Standardization and Commercialization : Looking for partners

- Technologies are formally registered at INPI
- Looking for partners to support the standardization, integration, and commercialization.
- We invite institutions and companies to co-develop, scale and create new solutions with us.

## Dados do Programa

Data de Publicação: 24/01/2023

Data de Criação: 24/01/2023

- § 2º do art. 2º da Lei 9.609/98: "Fica assegurada a tutela dos direitos relativos a programa de computador pelo prazo de cinquenta anos contados a partir de 1º de janeiro do ano subsequente ao da sua publicação ou, na ausência desta, da sua criação"

**Título:** POLKA: ARQUITETURA BASEADA EM CHAVES POLINOMIAIS PARA ROTEAMENTO DE ORIGEM EM REDES PROGRAMÁVEIS

**Algoritmo hash:** SHA-512 - Secure Hash Algorithm

**Resumo digital hash:** 8bce9214b6caf23d8b7fdb103c8d2b3dddf58c76de859a602c595d1ea2f0ab838e463ca4fd8da9b330154f63e26ec0cf7a92b8b0aa0668a074eebae700b70ef

## Dados do Programa

Data de Publicação: 24/01/2023

Data de Criação: 24/01/2023

- § 2º do art. 2º da Lei 9.609/98: "Fica assegurada a tutela dos direitos relativos a programa de computador pelo prazo de cinquenta anos contados a partir de 1º de janeiro do ano subsequente ao da sua publicação ou, na ausência desta, da sua criação"

**Título:** INNETMOD: HABILITAÇÃO DA OPERAÇÃO DA DIVISÃO POLINOMIAL DA ARITMÉTICA MODULAR POR MEIO DA VERIFICAÇÃO CÍCLICA DE REDUNDÂNCIA (CRC) EM PROCESSADORES DE PACOTES PROGRAMÁVEIS

**Algoritmo hash:** SHA-512 - Secure Hash Algorithm

**Resumo digital hash:** 7ef721f9f831d686334e269e62d89d31e6ee2f72e571a8350eca60b13ce38a460aea646521d84ba20ba84bae01b8763d401b7ea27a5826d07d31fc43dcdcf3d

Data de Publicação: 24/01/2023

Data de Criação: 24/01/2023

- § 2º do art. 2º da Lei 9.609/98: "Fica assegurada a tutela dos direitos relativos a programa de computador pelo prazo de cinquenta anos contados a partir de 1º de janeiro do ano subsequente ao da sua publicação ou, na ausência desta, da sua criação"

**Título:** M-POLKA: ROTEAMENTO DE ORIGEM POR MÚLTIPLOS CAMINHOS EM REDES PROGRAMÁVEIS

**Algoritmo hash:** SHA-512 - Secure Hash Algorithm

**Resumo digital hash:** 633315f2484b83bac5373c534cae6c61257241aa2411dc49d08b11624c032f5a2af9331b8e6a46000b42b29cfd49267e1b3790451690347a9bda63945640cf08

**INPI** INSTITUTO  
NACIONAL  
DA PROPRIEDADE  
INDUSTRIAL

27/01/2023 870230007557  
13:09



2940919158911560

# Acknowledgments

- The framework **provided by the GNA-G** and the whole ecosystem of **NRENs (Global P4 Lab)** enabled the PolKA routing to be tested thanks to:
  - GNA-G Data Intensive Science WG
  - GNA-G AutoGOLE / SENSE WG
  - GEANT RARE Project
  - ... And all it's collaborating institutions and teams
- Collaboration **with Caltech is crucial** hosting us at the booth
- Provides all the resources (e.g. Tofinos + servers +...) to deploy it in a near production allowing us to demonstrate PolKA in the best conditions



# PolKA Community, Partners and Collaborations

- **Caltech** : Professor Harvey Newman and Raimondas Širvinskas
- **RARE GÉANT**: Frédéric Loui, Csaba Mate, Eoin Kenny
- **RNP**: Marcos Schwarz
- **Qualcomm**: Jordi Ros Giralt
- **Trinity College Dublin (Connect)**: Marco Ruffini
- **CNPq, FAPES and FAPESP-(Brazilian research funding agencies )**
  - **PorVIR 5G and Smartness projet**
- **2021 Google Research Scholar Award**
- **2022 Intel Connectivity Research Grant (Fast Forward Initiative)**

# Selection of Our Recent Publications

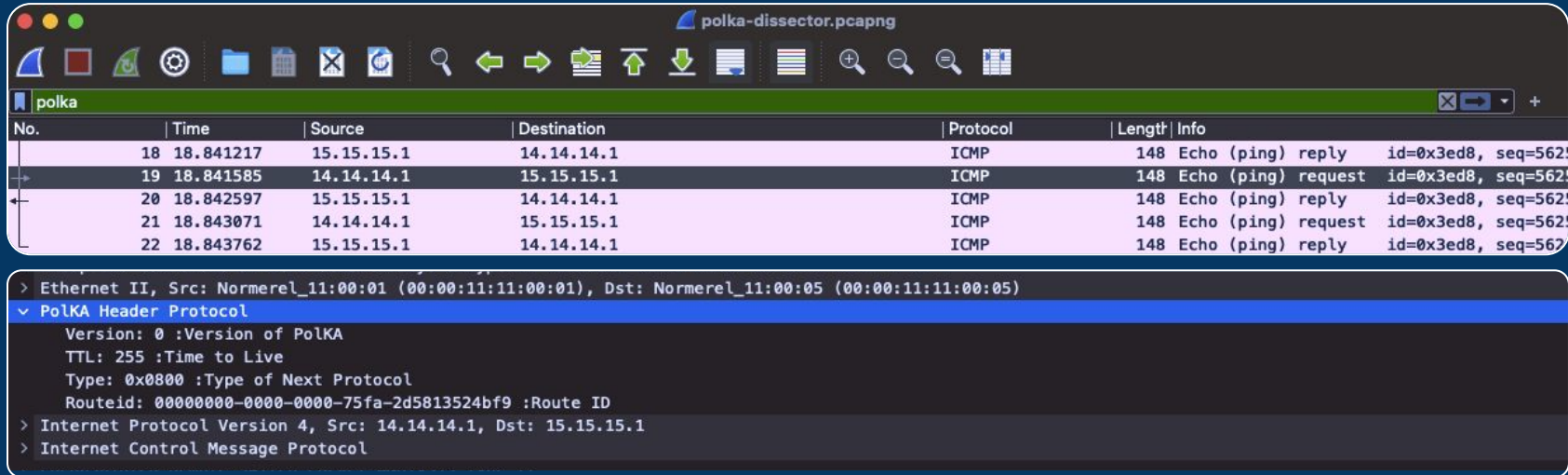
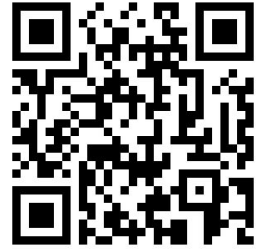
- [CRC4EVER: Cyclic Redundancy Check for Enhanced Verification and Efficient Routing](#) (Demo at Sigcomm 2025)
- [A Path-Aware Routing for Data Intensive Science: Proposal, Deployment and Evaluation in High-Performance Testbed](#) (WGRS 2025)
- [Transport efficiency for data-intensive science: deployment 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)

# Additional references

1. [Global Network Advancement Group: Towards a Next Generation System for Data Intensive Sciences](#)
2. [Documentation: Let's enable PolKA in freeRtr](#)
3. [PolKA presentation at Google Research Scholar Award](#)
4. [Multipath PolKA presentation at ONF 2022](#)
5. [PolKA github](#)
6. [RARE website](#)
7. [FreeRouter website](#)
8. [LabNERDS Videos](#)
9. [PolKA NetSoft 2020 conference paper](#)
10. [V. Shoup, A computational introduction to number theory and algebra, 2008.](#)

# PolKA: Github

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



# Thank you for attention !

Everson Scherrer Borges

*everson@ifes.edu.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.*



# Ok... Why should I use PolKA?

- One good reason...  
... It is easy to setup paths/tunnels!
- It has some interesting properties that enable innovative applications.
  - Ex: multicast communication model support, multipath routing, failure protection, Proof of Transit, telemetry...
- Open source implementation in software and in hardware
  - RARE/FreeRtr