TNC 2021 P4 and Data Plane Programming BoF
June 18, 2021

PolKA: Polynomial Key-based Architecture for Source Routing

Cristina Dominicini[1]*, Rafael Guimarães*[1]*, Diego Mafioletti*[1]*, Magnos Martinello*[2]*, Moises R. N. Ribeiro*[2]*, Rodolfo Villaça*[2]*, Frédéric Loui*[3]*, Jordi Ortiz*[4]*, Frank Slyne*[5]*, Marco Ruffini*[5]*, and Eoin Kenny*[6]

[1]*Federal Institute of Espírito Santo,* [2]*Federal University of Espírito Santo,*
[3]*RENATER,* [4]*University of Murcia,* [5]*Trinity College Dublin, and* [6]*HEANET*

Contact: cristina.dominicini@ifes.edu.br

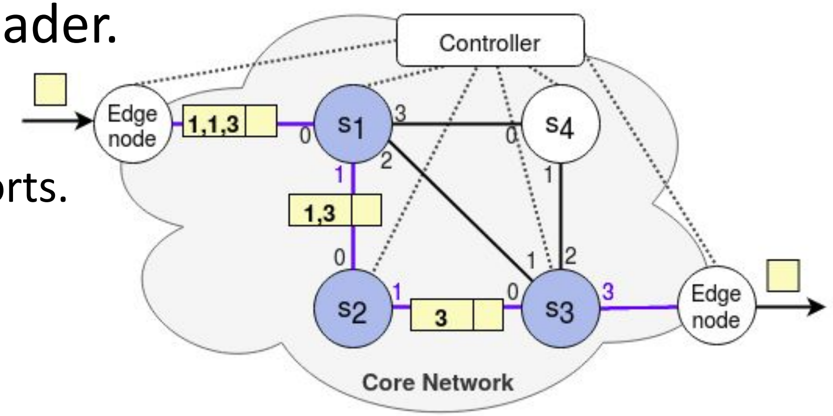
INSTITUTO FEDERAL Espírito Santo RENATER CONNECTEUR DE SAVOIRS UNIVERSIDAD DE MURCIA Trinity College Dublin The University of Dublin HEAnet Ireland's National Education & Research Network UFES

Motivation

- **SDN and Programmable Network Devices**:
  - Innovation and custom protocols.
- **Challenge**: How to select paths and load-balance between them to adapt to variable workloads?
  - **Common solution**: encode multiple paths in core nodes as forwarding **table entries**, and allow the edge to select among them.
- **Problems**:
  - Large number of states ⟶ Management burden
  - Restricted capacity of switch tables ⟶ Traffic engineering cannot exploit all paths
  - Latency for path setup

Source Routing (SR)

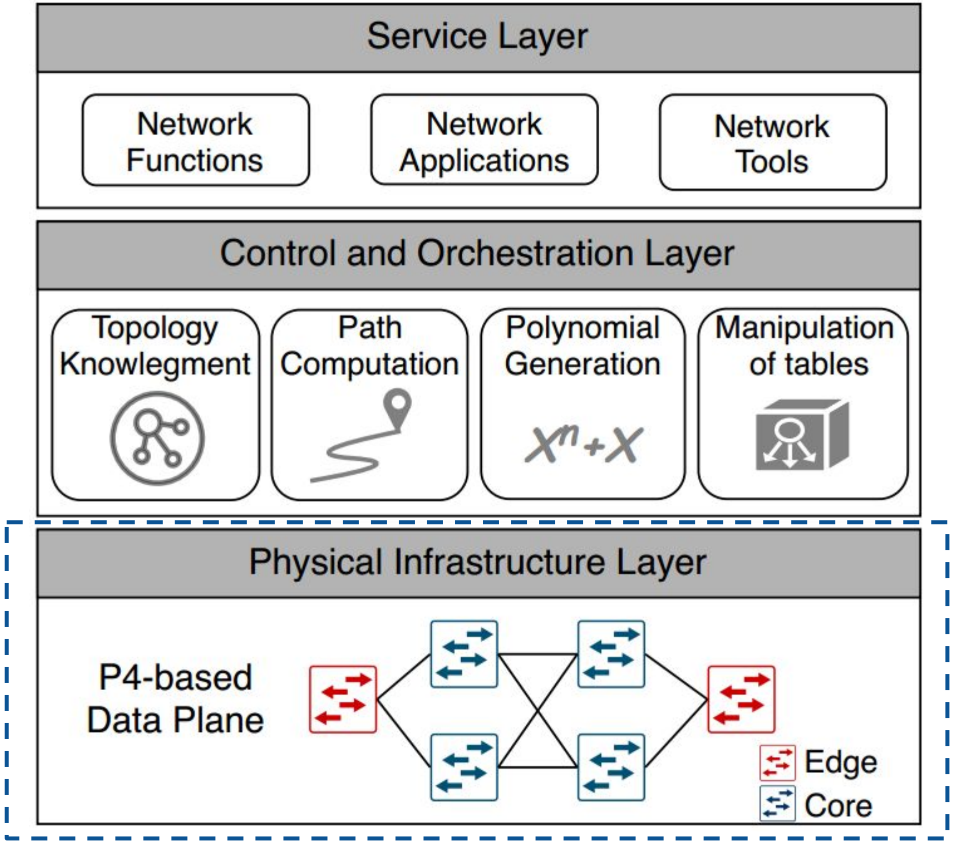
- A source specifies all forwarding nodes in the path.
- A route label is added to the packet header.
- **Traditional way: List-based SR (LSR)**
  - The path is defined as a stack of output ports.
- **Limitations**:
  - State in the packet:
    - Each node performs a pop on the stack.
    - Rewrite operation.
  - No implicit way of representing multiple paths.



PolKA

- **Problem: Is it possible to define a fully stateless SR approach?**
  - No packet rewrite, No tables
  - … and offer support for complex use cases...
- **NetSoft 2020: “PolKA: Polynomial Key-based Architecture for Source Routing in Network Fabrics”**
  - Source Routing based on a arithmetic operation
  - Residue Number System (**RNS**) and Chinese Remainder Theorem (**CRT**)
  - **Emulated proof-of-concept in Mininet**
- **ONDM 2021: “Deploying PolKA Source Routing in P4 Switches”**
  - **Deployment in the GEANT P4 Lab testbed with Tofino switches**
  - PoC of PolKA in real-world environment

PolKA: Architecture



PolKA: Data Plane

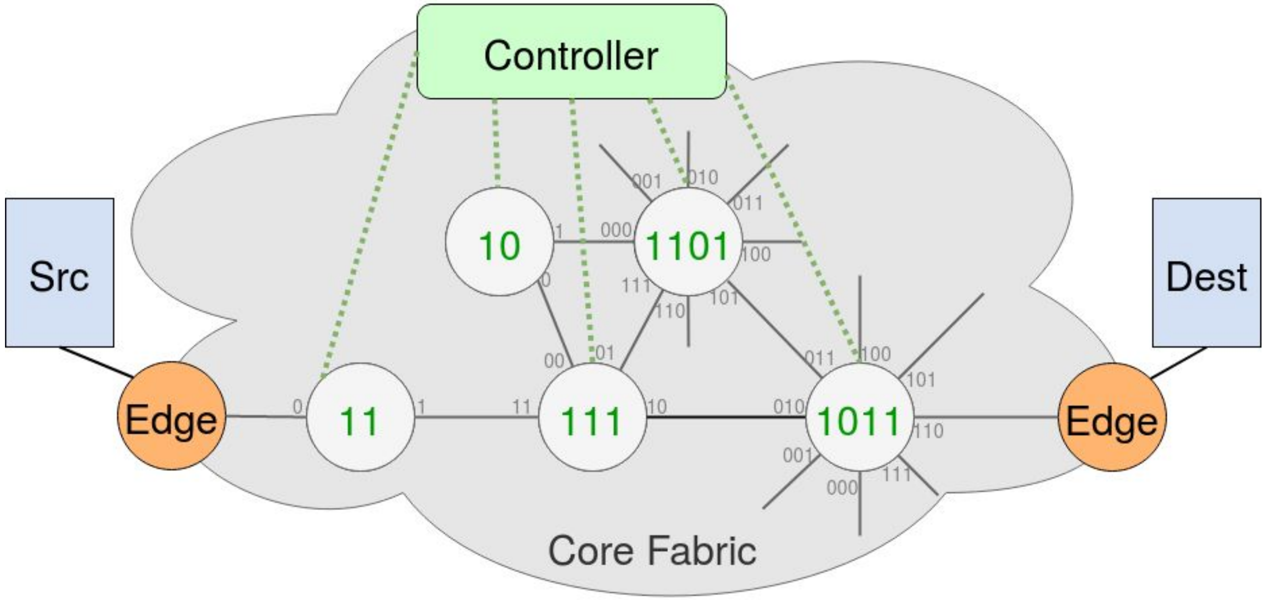
- The forwarding uses a **mod** operation (remainder of division):

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

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

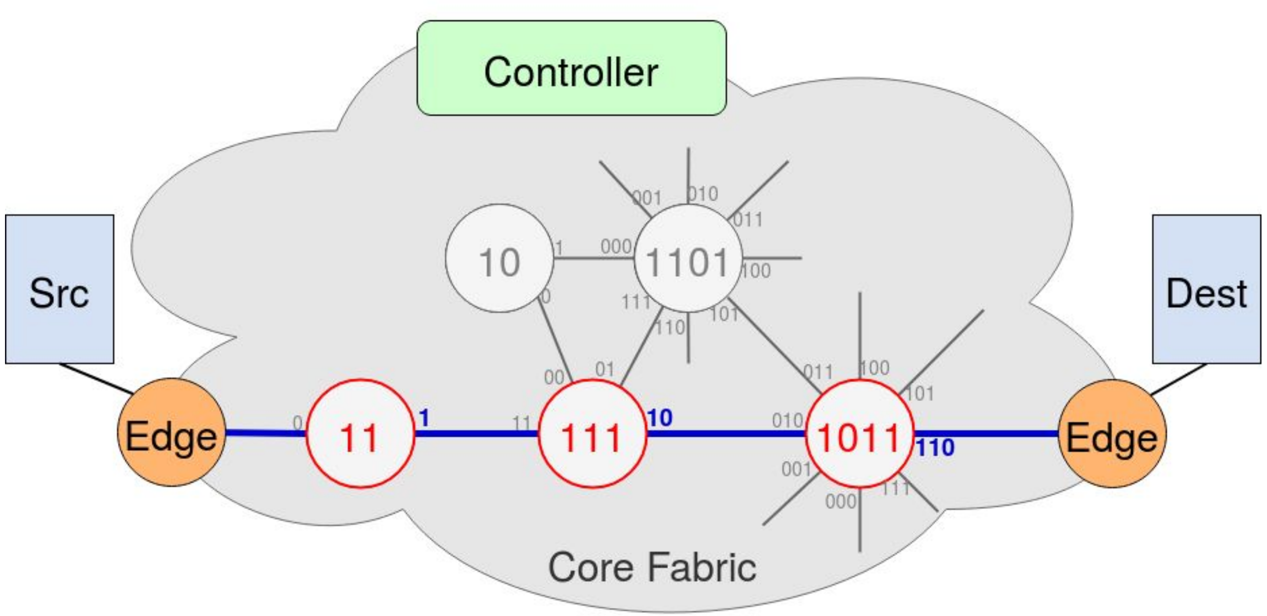
How does PolKA work?

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



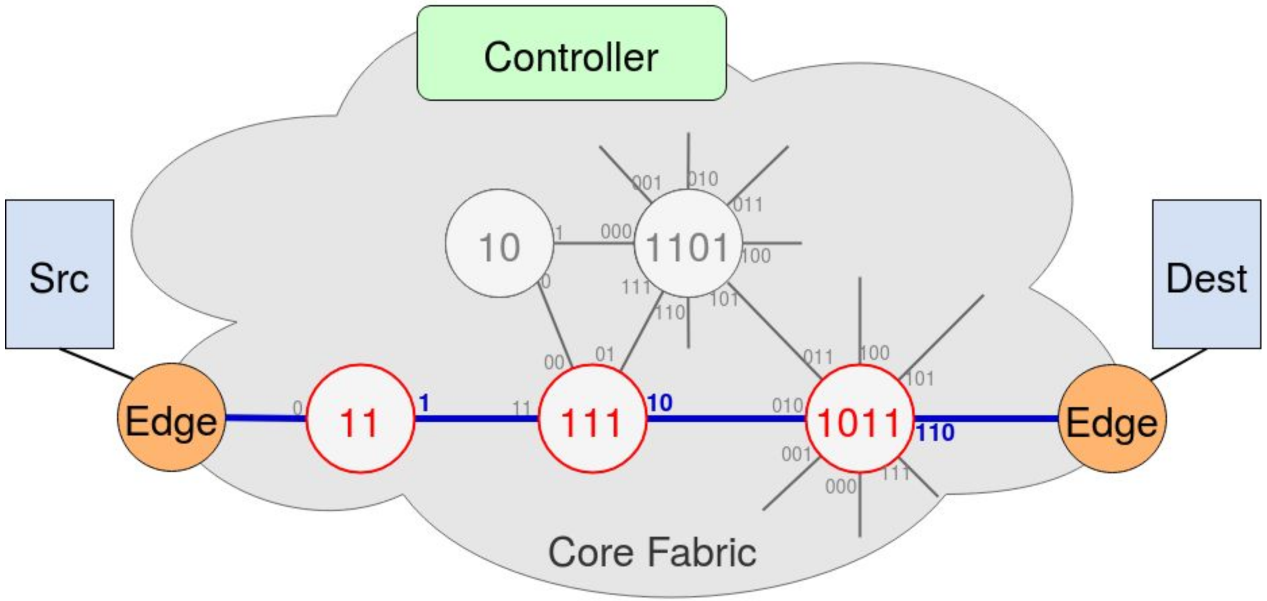
How does PolKA work?

- 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** calculates the ***routeIDs*** using the polynomial **Chinese Remainder Theorem.**



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

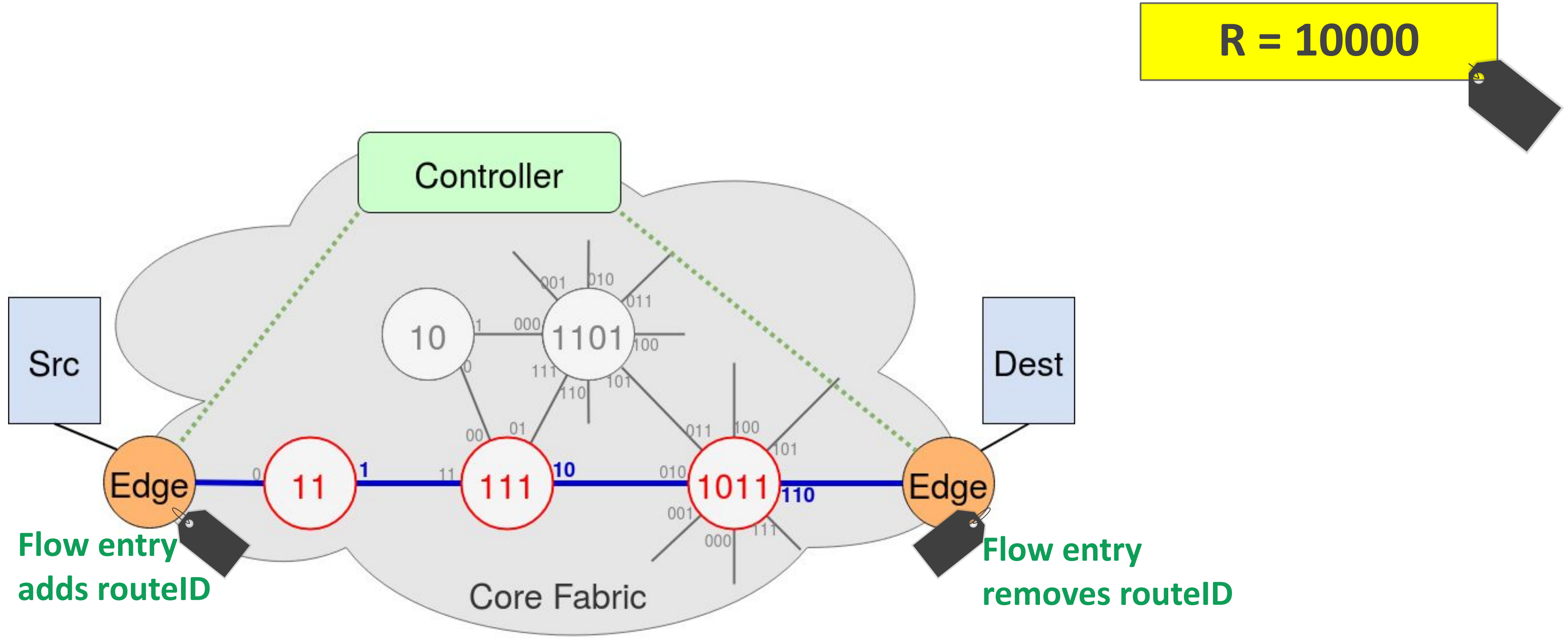
$$o_1(t) = 1$$
$$o_2(t) = t = 10$$
$$o_3(t) = t^2 + t = 110$$

Calculate routeID with CRT

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

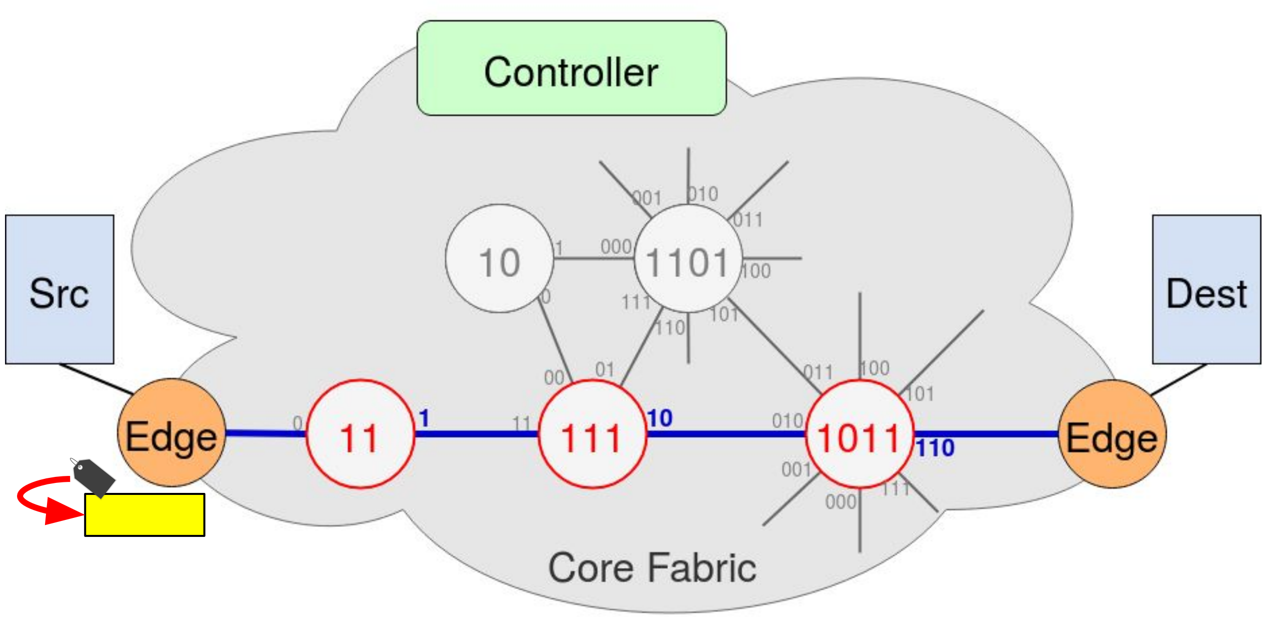
How does PolKA work?

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



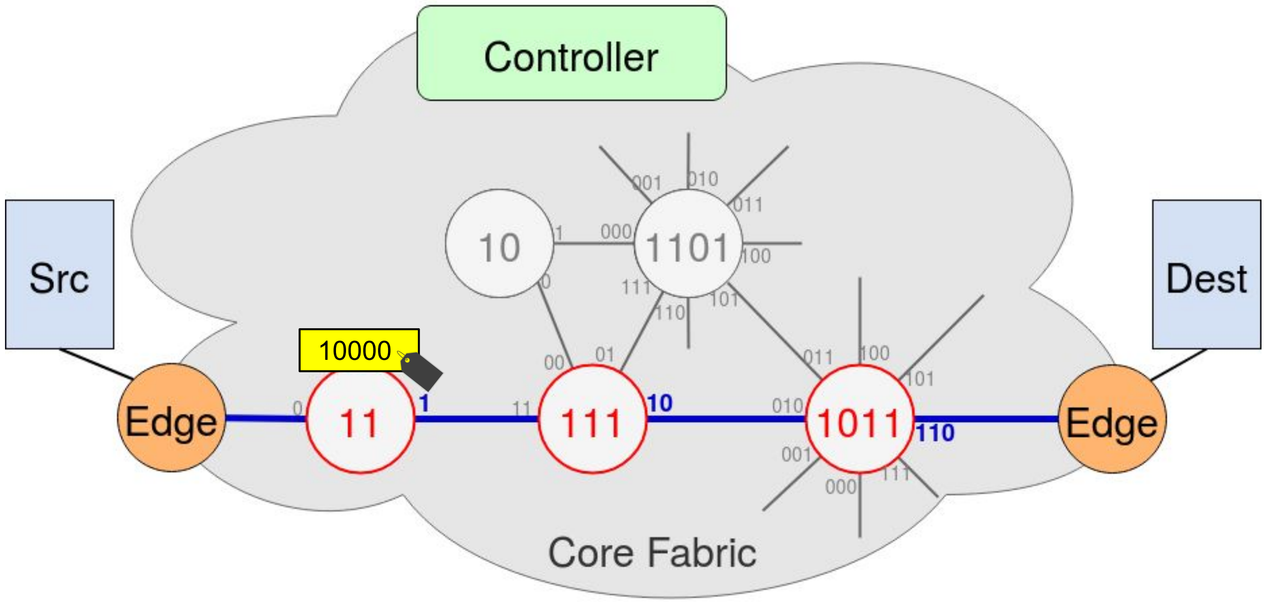
How does PolKA work?

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



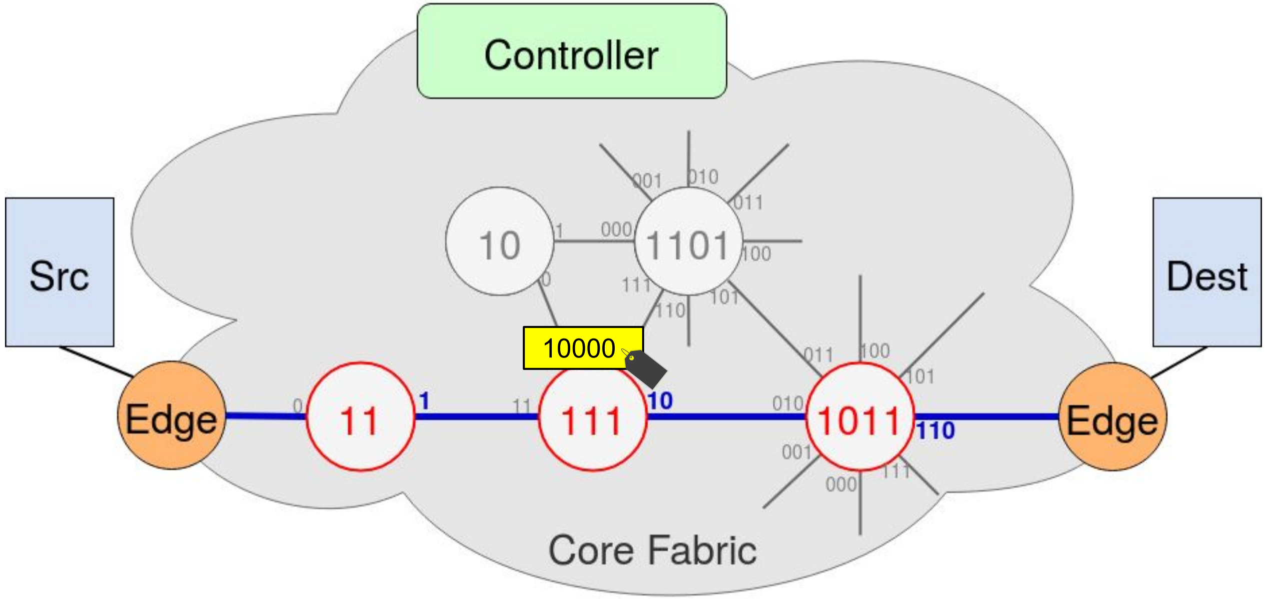
How does PolKA work?

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



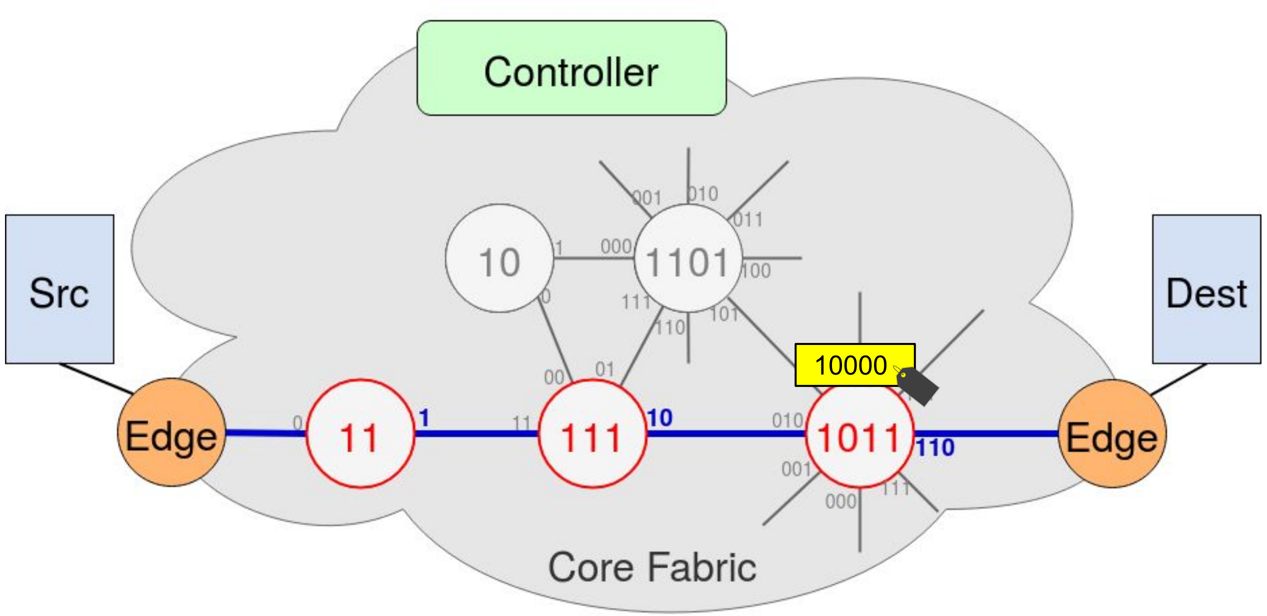
How does PolKA work?

- Forwarding using **mod** operation: $\langle 10000 \rangle_{0111}$ = 10 → output port
- No packet rewrite! No tables!



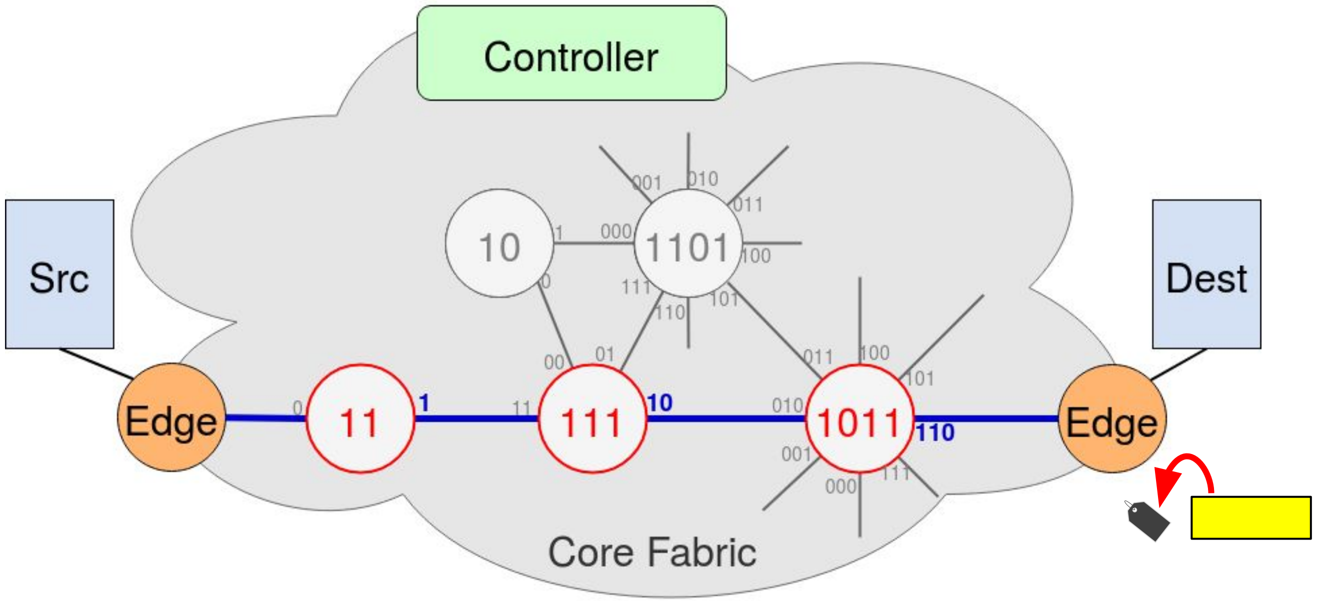
How does PolKA work?

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



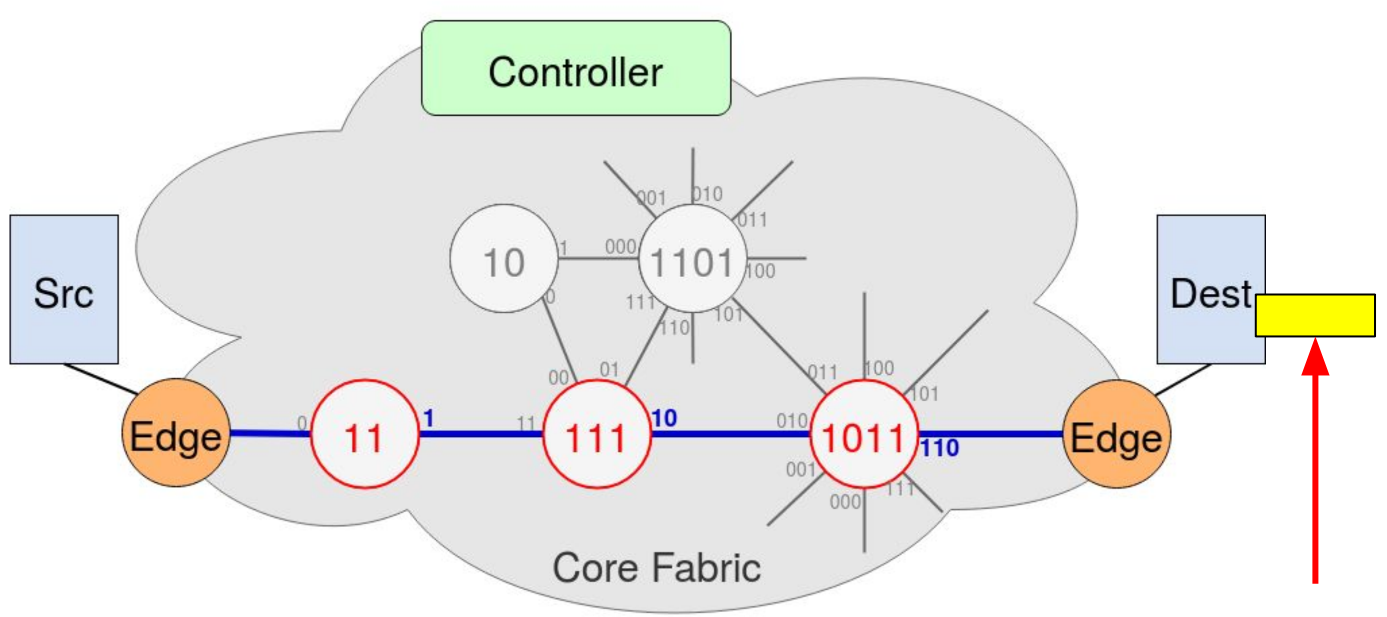
How does PolKA work?

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



How does PolKA work?

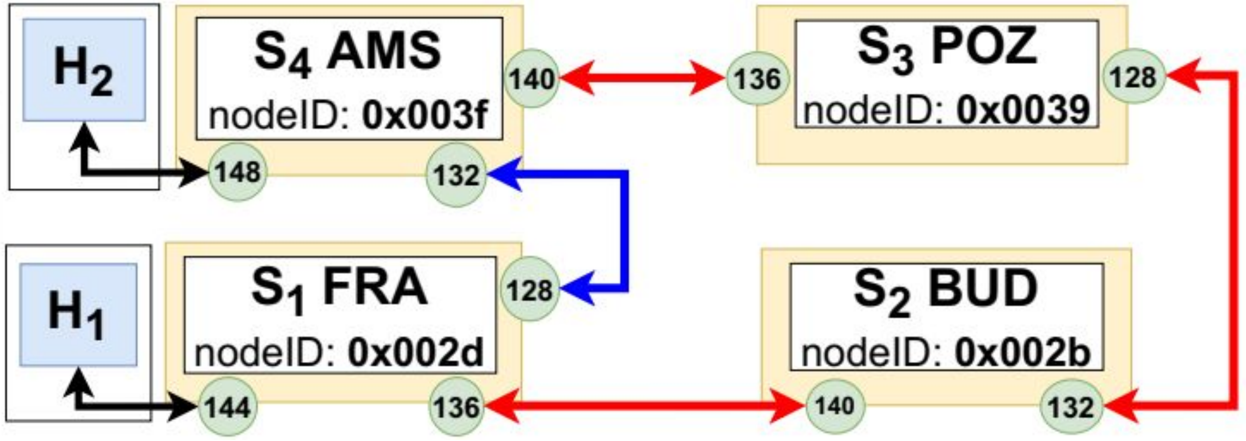
- Packet is delivered to the application in a transparent manner.



GÉANT P4 Lab Testbed

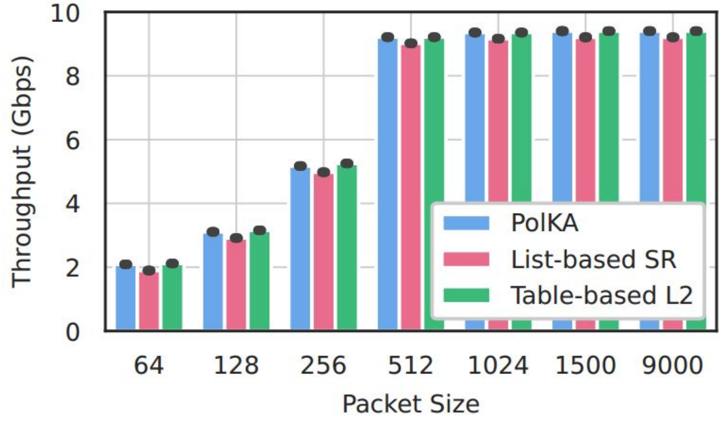
- RARE project: https://wiki.geant.org/display/RARE
- Testbed with Intel/Tofino Barefoot P4 Switches

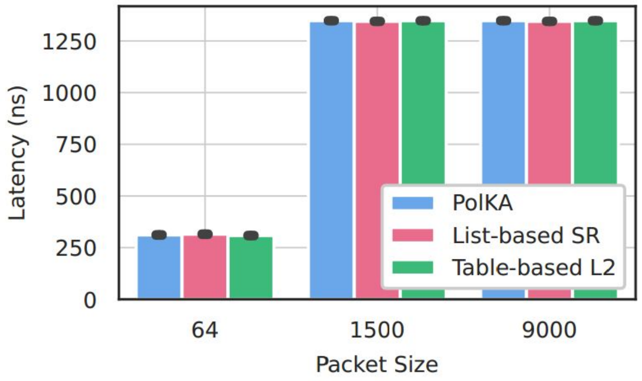
AMS
POZ
FRA
BUD
GÉANT
RARE



Preliminary Results

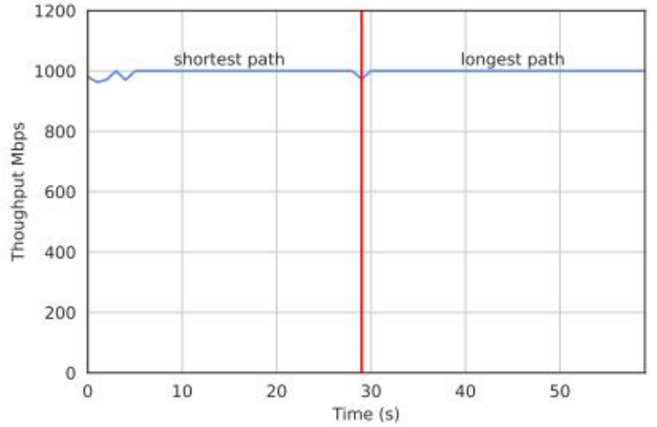
- Throughput & Forwarding Latency:
  - PolKA matches the performance of traditional L2 table-based forwarding and LSR approaches.

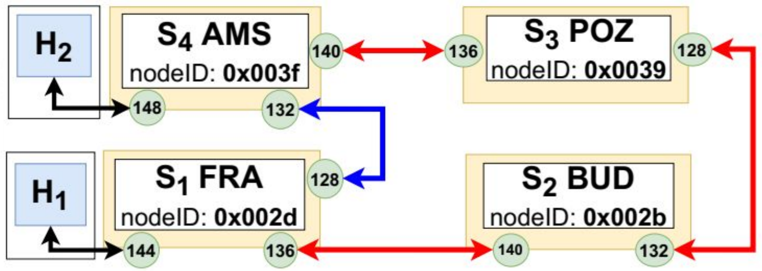




Preliminary Results

- Agile Path Reconfiguration:
  - SDN Controller changes a single flow entry at H1: path is reconfigured from shortest to longest path.



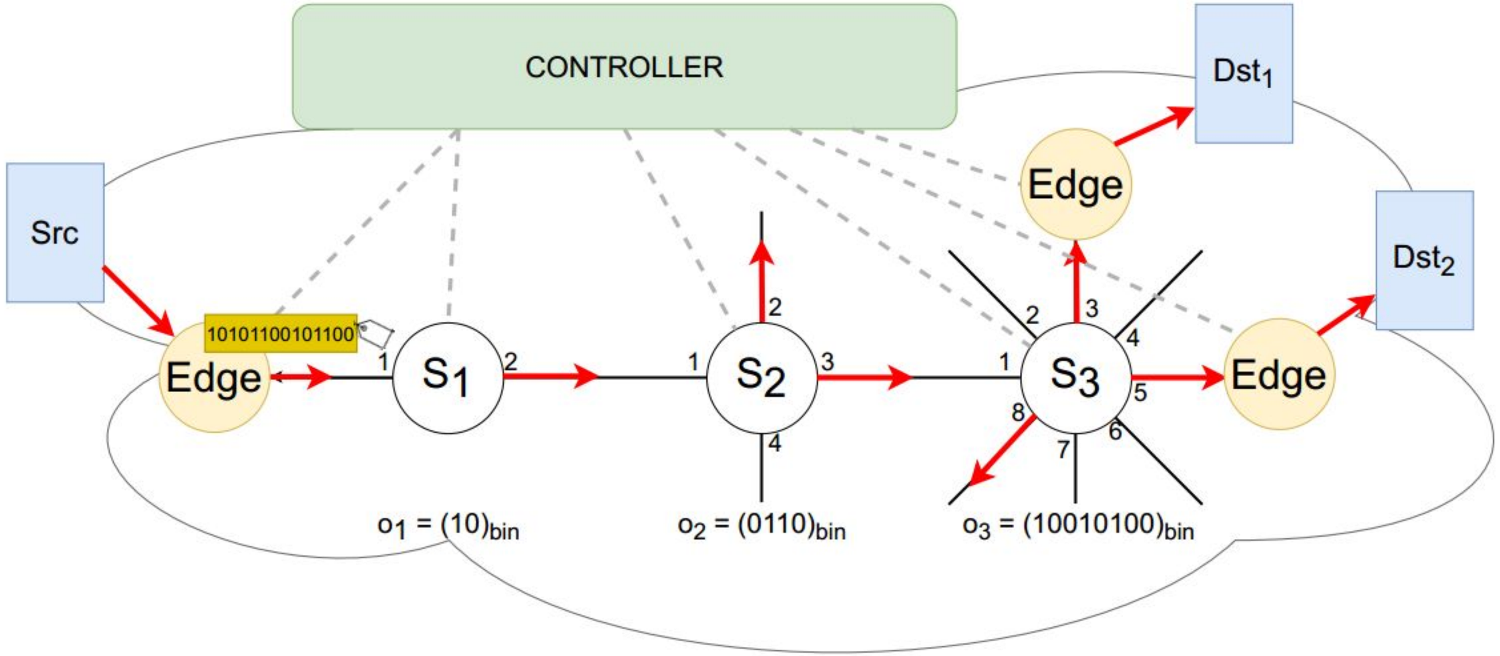


Future Works

- We are integrating PolKA in RARE repository for experimenters.
    - Extension of control planes functionalities.
- We are preparing deployment guidelines for production use cases.
- This proposal was one of the recipients of the 2021 Google Research Scholar Award.
- We are also exploring PolKA properties for innovative applications.
    - Security and Fast Failure Reaction exploring RNS properties.
    - Multipath Routing.
    - ...

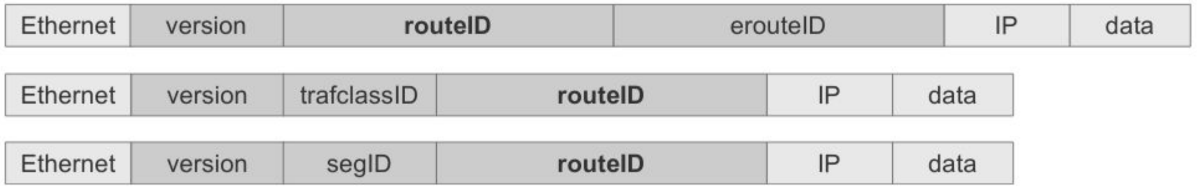
Future Works: Multipath Routing

- Extension: the *portid* coefficients represent the transmitting state of the ports instead of port labels.



Future Works

- Polynomial representation
  - Polynomials of higher orders for **Multi-layer Networks and Slicing**
- Use of multiple keys
  - **Protection paths**
  - **QoS**



- Source Routing
  - **Service Function Chaining**
  - **Save TCAM** for hybrid operation with table-based approaches
  - **Agile Path Reconfiguration**

Thank you!

Cristina Klippel Dominicini

cristina.dominicini@ifes.edu.br