IETF 113 - Vienna, Austria
March 24, 2022 // Onsite Presentation

PolKA: Polynomial Key-based Architecture for Source Routing

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

[1]Federal Institute of Espírito Santo, [2]Federal University of Espírito Santo,
Contact: rafaelg@ifes.edu.br

INSTITUTO FEDERAL
Espírito Santo

UFES

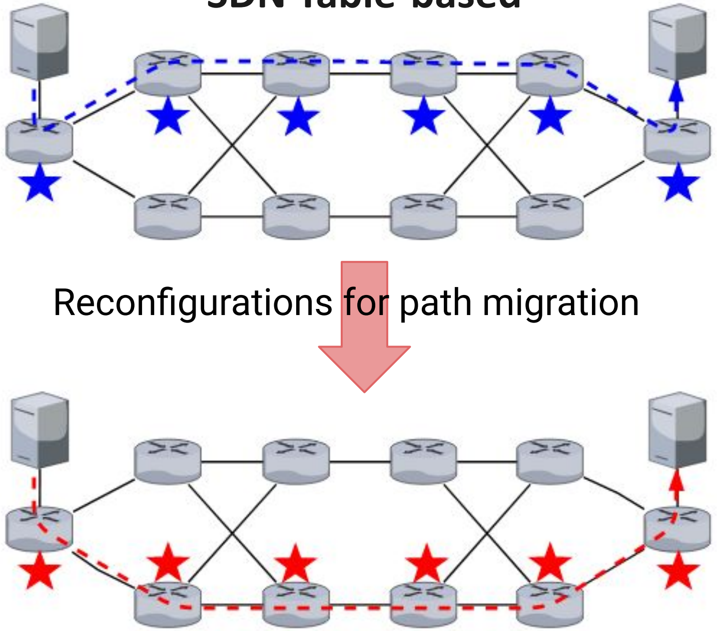
Which PANRG's problem does PolKA solve?

- “(...) Endpoints have very little information about the paths over which their traffic is carried, <u>**and no control at all beyond the destination address**</u>. (...)”

PANRG: The Control Problem

- “(...) Endpoints have very little information about the paths over which their traffic is carried, **<u>and no control at all beyond the destination address</u>**. (...)”

SDN Table-based



Can SDN table-based solutions offer path-aware control?

- **Problems:**
  - Large number of states ⟶ **Scalability** 🙁
  - Limited capacity of tables ⟶ **Granularity** 🙁
  - Latency for path configuration ⟶ **Agility** 🙁

PANRG: The Control Problem

- "(...) Endpoints have very little information about the paths over which their traffic is carried, **and no control at all beyond the destination address**. (...)"

SDN Table-based

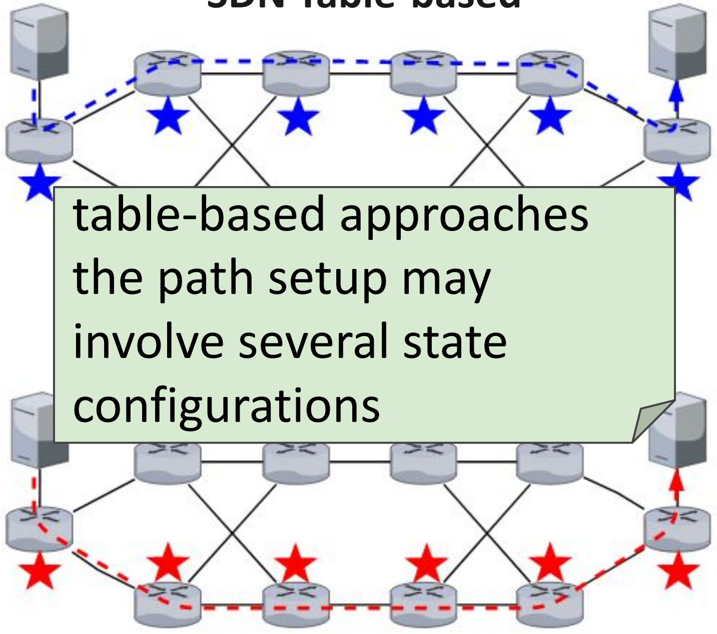


table-based approaches the path setup may involve several state configurations

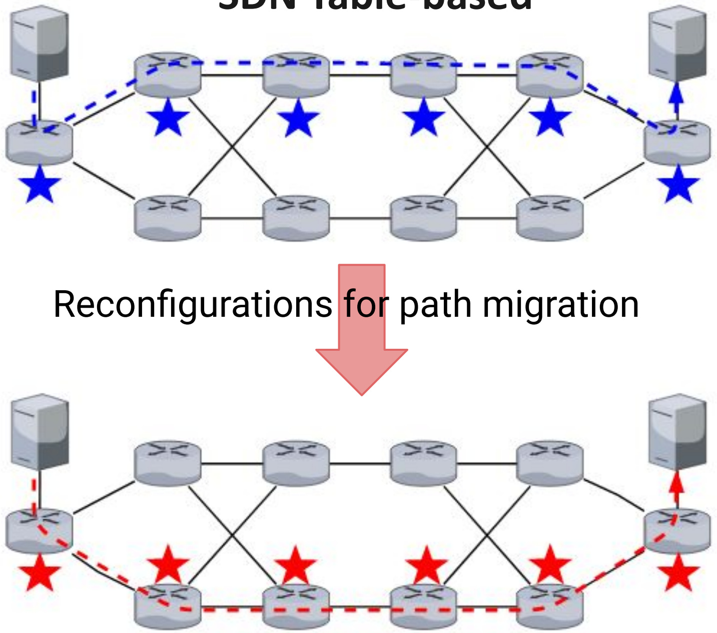
Can SDN table-based solutions offer path-aware control?

- **Problems:**
  - Large number of states ⟶ **Scalability** 🙁
  - Limited capacity of tables ⟶ **Granularity** 🙁
  - Latency for path configuration ⟶ **Agility** 🙁

Source Routing: a key mechanism for endpoints

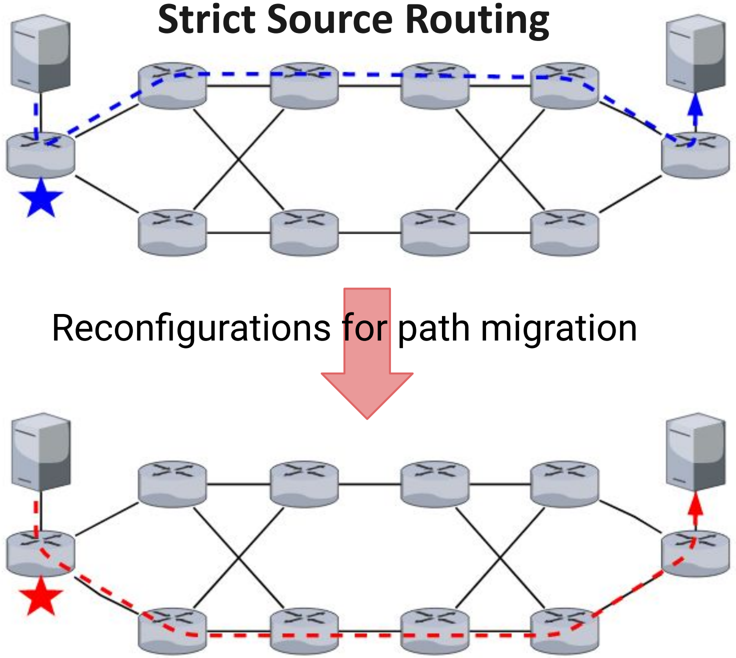
- **Endpoints controlling paths**: setup *routeID* at the edges

SDN Table-based



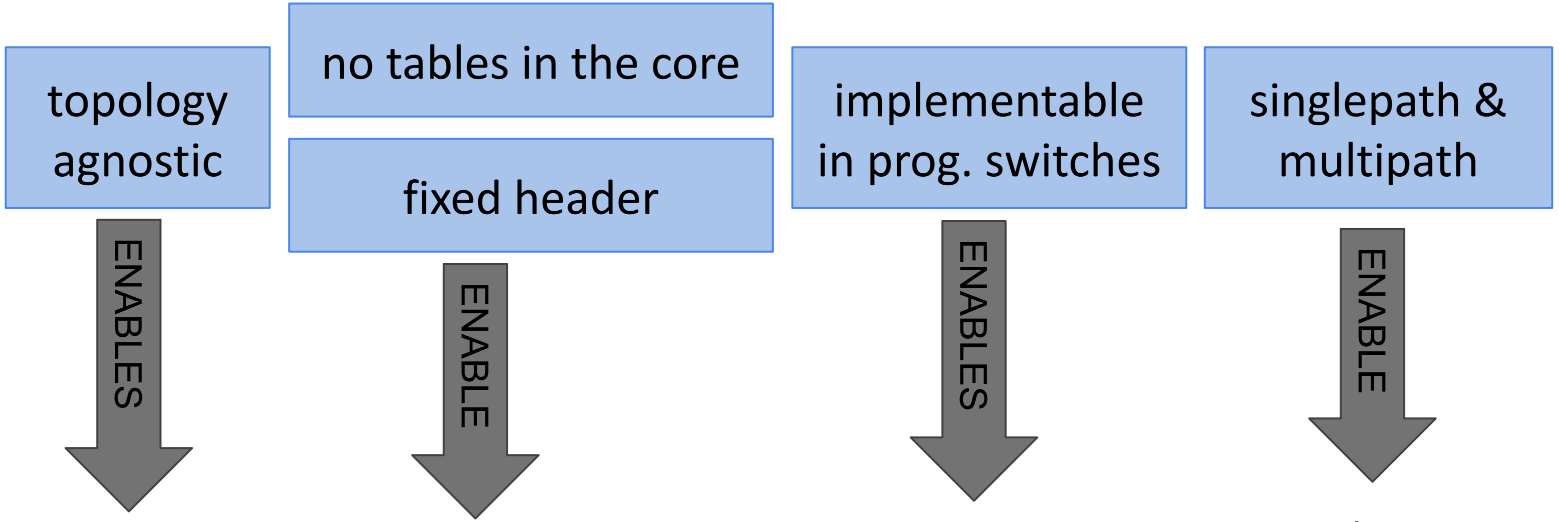
X

Strict Source Routing



Why use PolKA as Strict Source Routing?

- Only PolKA Source Routing simultaneously meets the following requirements:



- general solution for path-aware networking

- fast path reconfiguration
- no var header limitation with respect to hop limit

- exploit wire-speed and low latency dataplanes

- path expressiveness
- reliability by exploring multiple paths
- efficient in-band telemetry

How does PolKA work?

- Three polynomials:
    - **routeID**: a route identifier calculated using the CRT (Chinese Remainder theorem).
    - **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):

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

Timeline

PolKA received the 2021 Google Research Scholar Award

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

2020	2021/1	2021/2	2022
PolKA paper IEEE NetSoft	**IEEE ONDM paper Deployment @ Pan-European GÉANT Testbed RARE**	**Integration with RARE+FreeRouter**	**M-PolKA paper IEEE TNSM** *(accepted for publication)*
Novel **Polynomial RNS-based SR and reuse of CRC hardware**	PolKA **data plane implementation** in intercontinental testbed	PolKA **data & control plane implementation** + **integration**	Extension to **multipath SR** for reliable communications
Emulated prototype in Mininet & Hardware prototype in SmartNICs	Hardware: Intel Tofino	Emulated prototype in FreeRouter & Hardware prototype in Intel Tofino with FreeRouter control plane	Innovative apps: inband network telemetry, and load balance

How to implement PolKA?

- **Reuse CRC hardware** to offer polynomial mod.
  - Externs in P4 language.
  - Support in high-performance Tofino switches.
- RARE: Open source full-featured router on networking hardware for R&E
  - **data plane:** P4 (bmv2 and Tofino) and DPDK
  - **control plane:** FreeRouter
    - Reuse of standard distributed protocols
    - Static table maps Segment Routing indexes to nodeIDs
    - Get available topology info from link-state protocols

AMS
POZ
FRA
BUD
GÉANT
RARE

How does PolKA work?

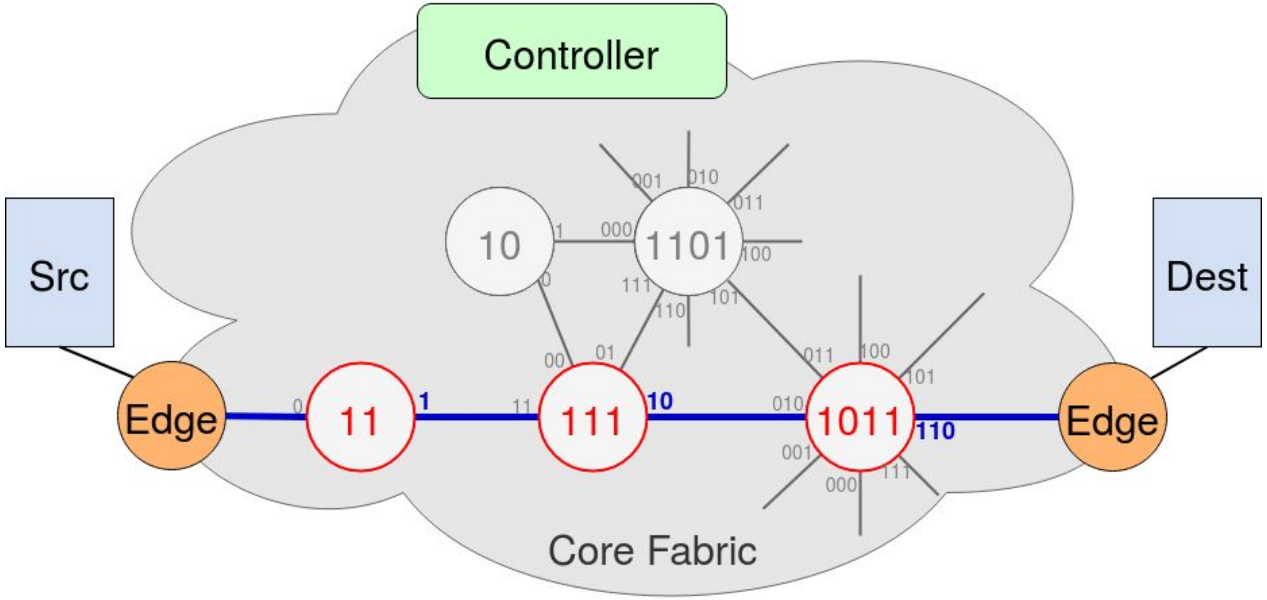
- The **Controller** chooses a **path** for a specific flow:
  - A set of switches: {0011,0111,1011}
  - and their output ports: {1 , 10, 110}

nodeID polynomials

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

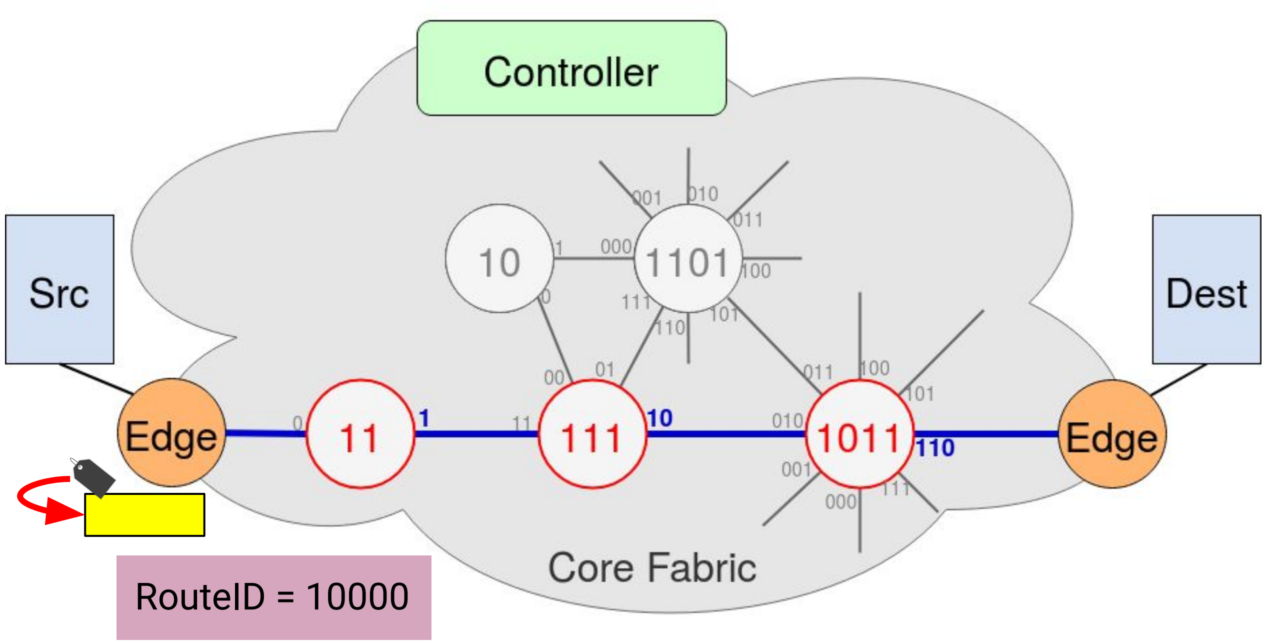
portID polynomials

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



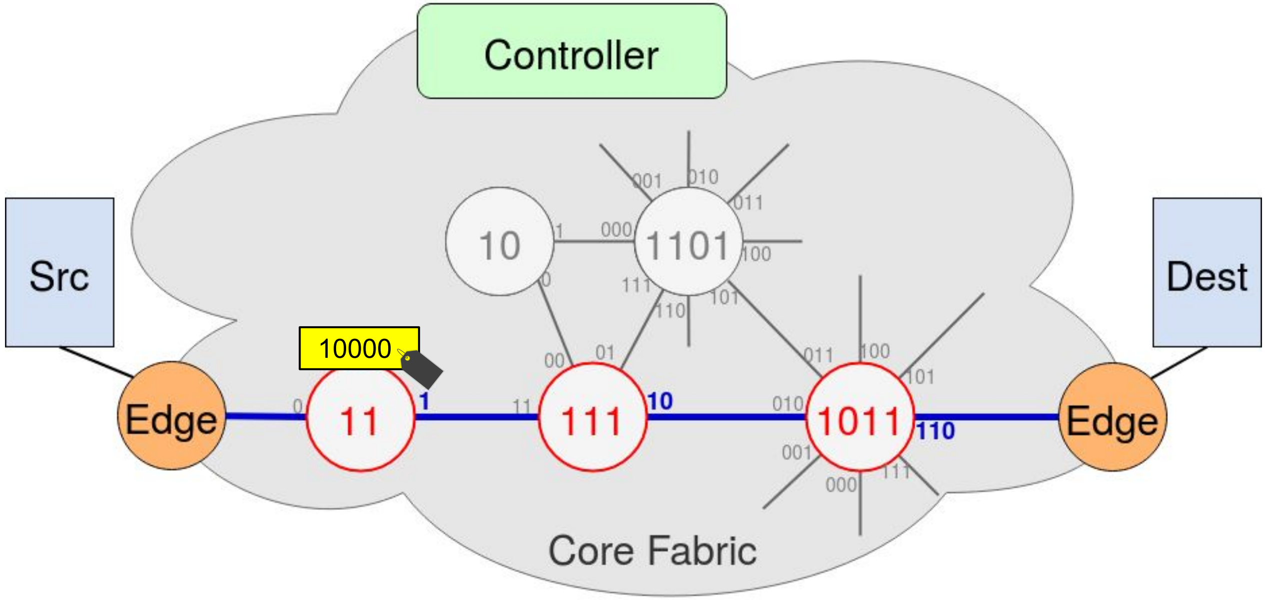
How does PolKA work?

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



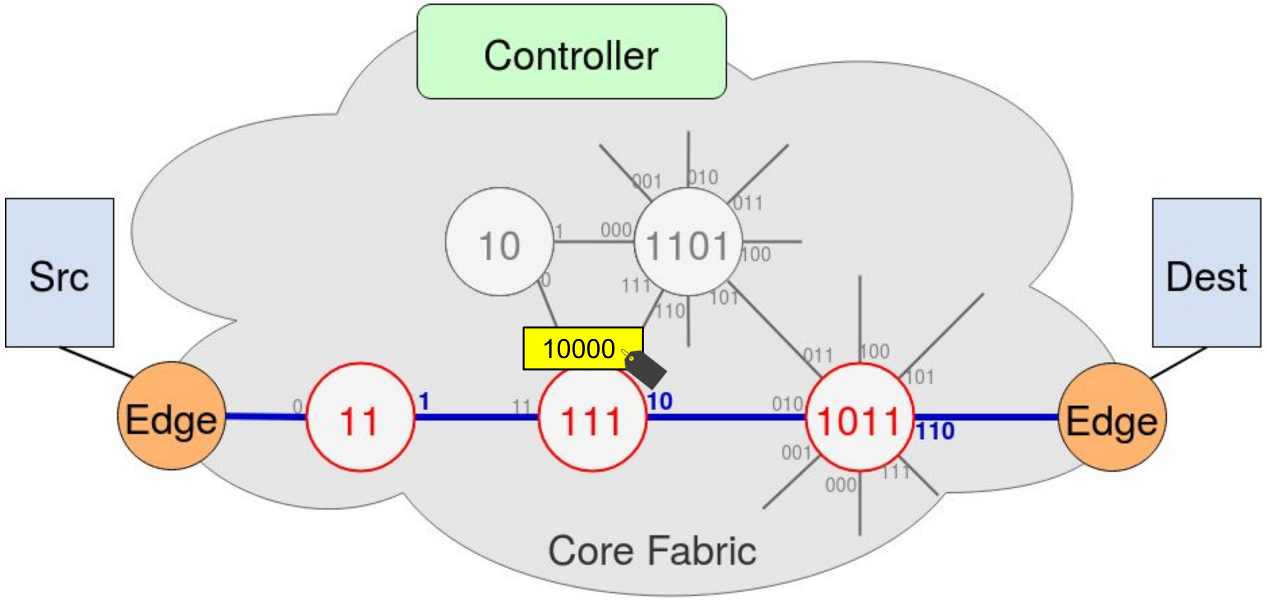
How does PolKA work?

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



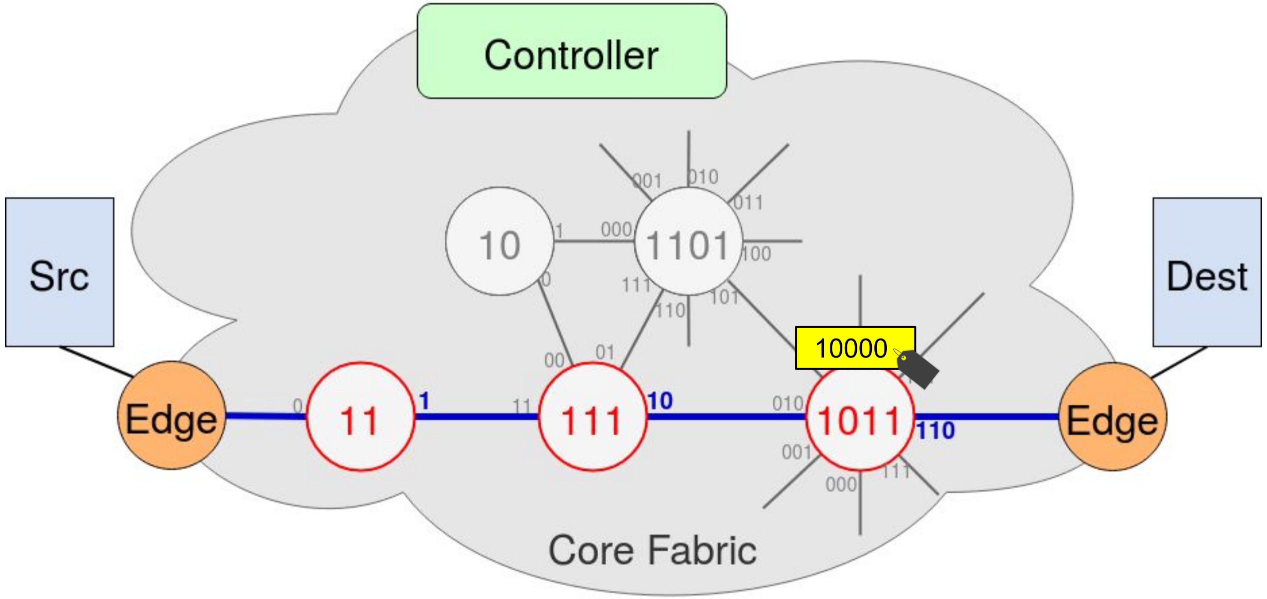
How does PolKA work?

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



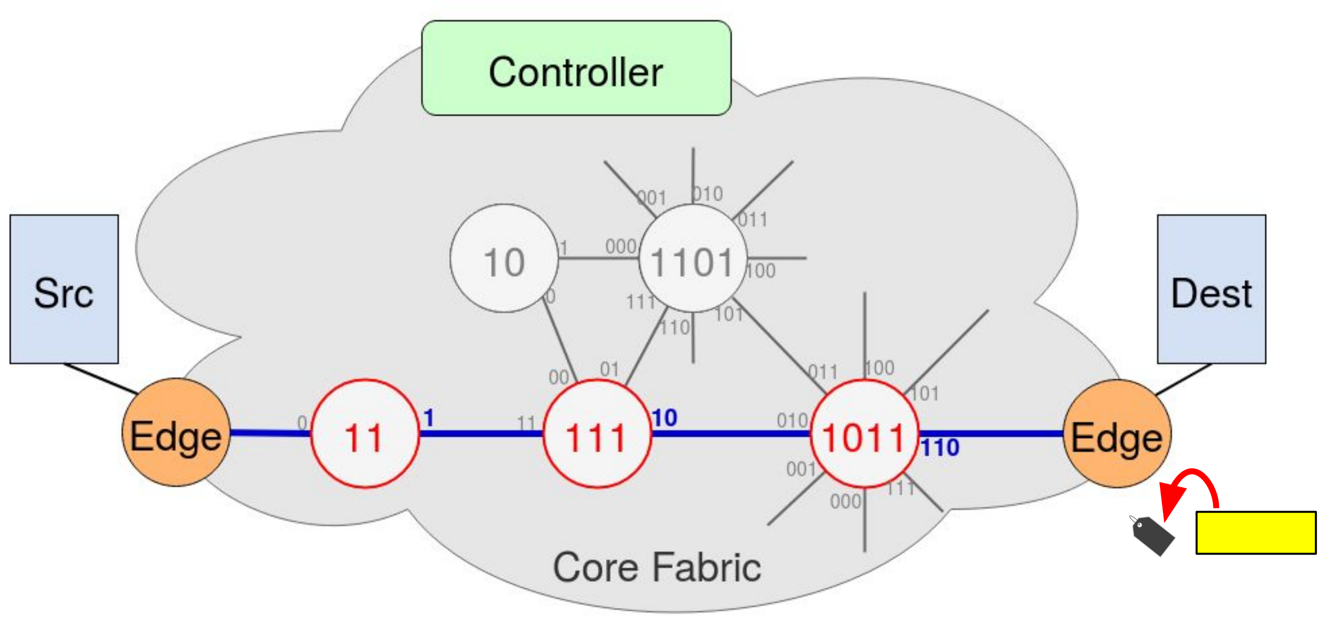
How does PolKA work?

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



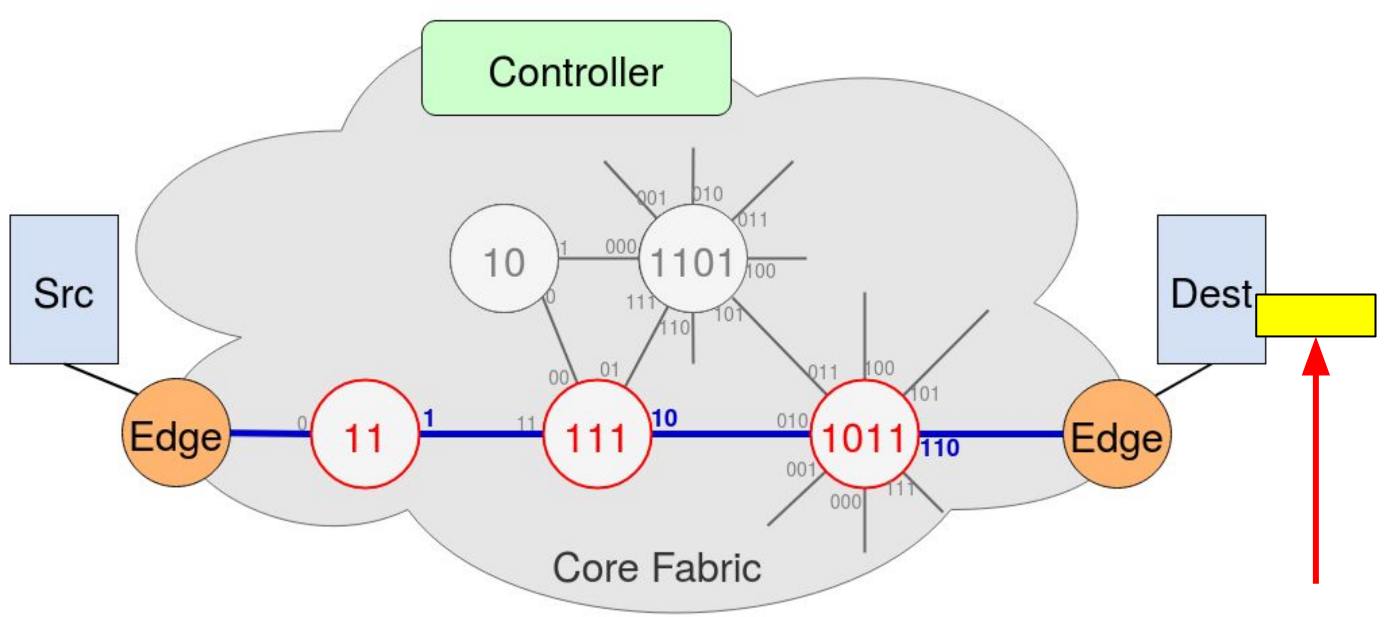
How does PolKA work?

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



How does PolKA work?

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



Thank you!

Rafael Silva Guimaraes

rafaelg@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.*