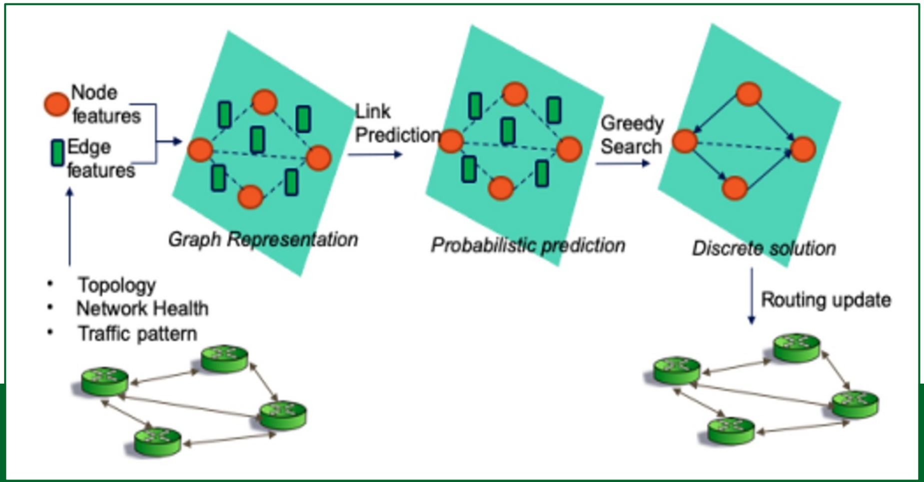
Framework for Integrating Machine Learning Methods for Path-Aware Source Routing

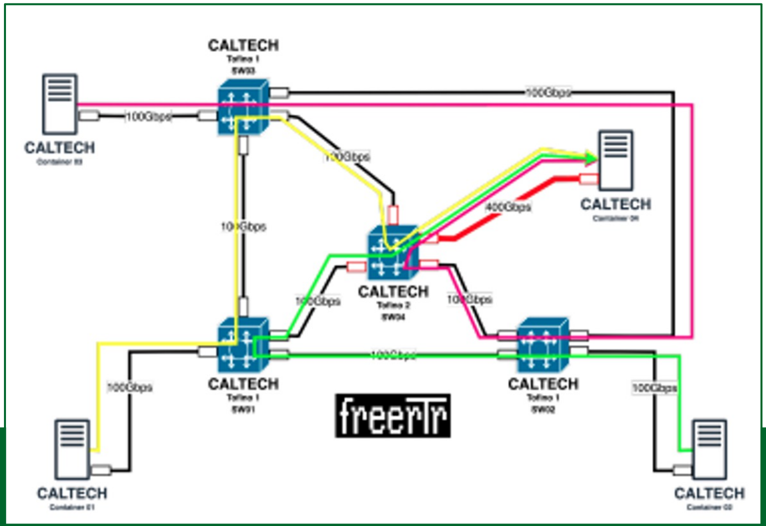
Anees Al-Najjar, Domingos Paraiso, Mariam Kiran, Cristina Dominicini, Everson Borges, Rafael Guimaraes, Magnos Martinello, Harvey Newman

ORNL IS MANAGED BY UT-BATTELLE LLC
FOR THE US DEPARTMENT OF ENERGY

Talk Outline



SC22 INDIS HECATE Paper



SC23 NRE Demonstration

- Introduce the Challenge we can now solve by working together
  - Network Challenge to build a ‘Truly’ Self-driving Network
- Collaboration is Key
- Talk is divided into 2 components
  - Segment Routing
  - Machine learning
- Results and Future work

The Challenge: Run Networks ‘Hotter’

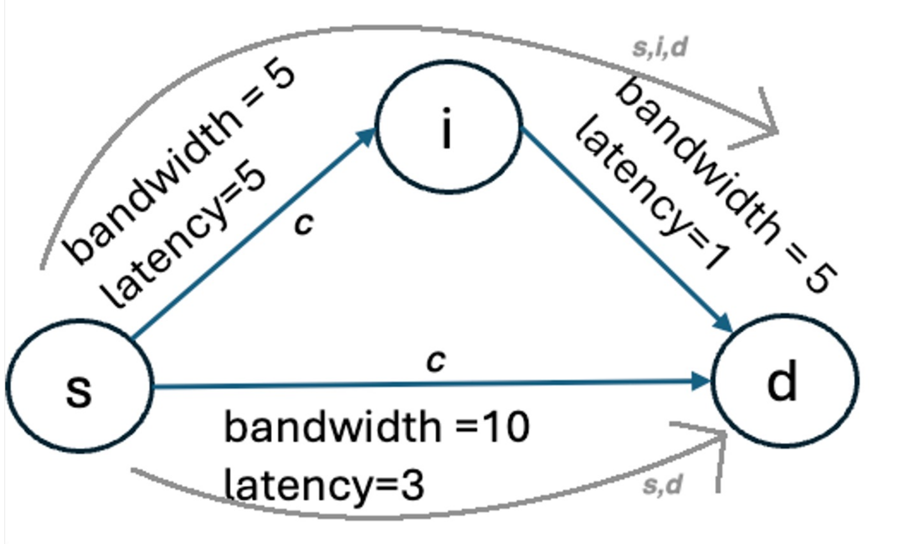
- There is an exponential increase in data production across all Science WANs
- Traffic Engineering solutions need to optimize our network in a way such that:
    - high performance throughput with minimum loss - Time sensitive flows and capacity capping
    - Latency sensitive flows - Clouds, control apis and more
    - advanced reservations using OSCARS to tackle some of the needs

- We need to think of new ways in using our network more efficiently, satisfying flow needs

Combinatorial Optimization problem:

f(x) = Bandwidth, capacity, latency, delay, ξ

The Challenge: MIN-MAX problem



Flow management is a common task needed for optimum TE

- max available bandwidth to use
- unreserved bandwidth available to use
- TE metrics for special flows

Combinatorial Optimization problem: compute edges and paths based on and incoming/outgoing flows

Min-max: allocate flows to maximize flows while minimizing congestion

Defining the problem:

Demand volume = $x_{sd}+x_{sid}$ = h

With additional parameters like latency, objective function becomes complicated

$\min_{(xsd,xsid)} F =$

$\xi x_{sd}+\xi x_{sid}$

Adding delay and link utilization

$\min_x F = x_{sd}$

$+ \quad 2x_{sid}$

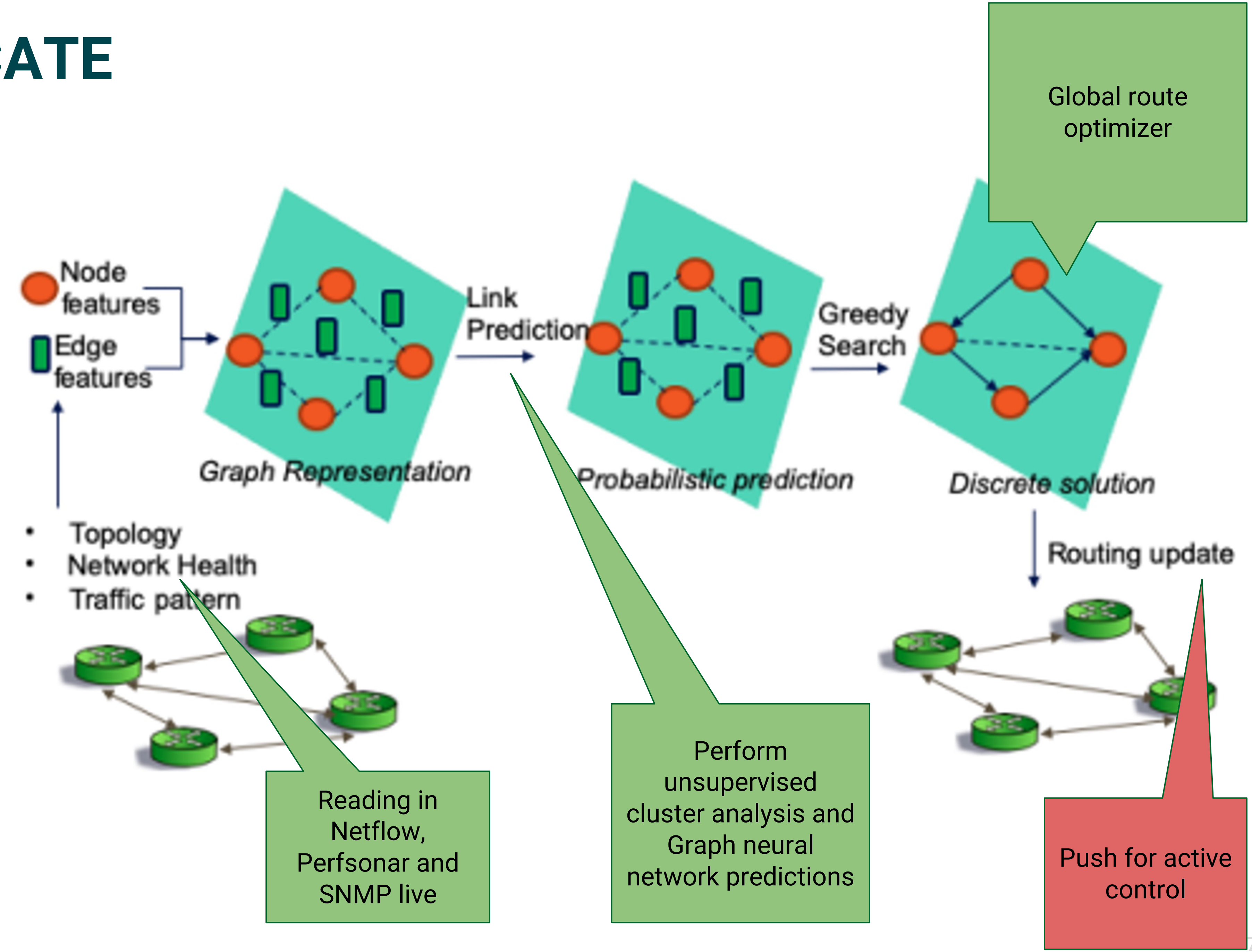
Linear programming problem, adding further constraints can become more complex

Using ML to ‘learn’ optimal objective function, but there are network constraints

- New objective functions can be self-learned
- Network engineering constraints:
    - Flow tables are limited by size and details on management of specific flows
    - Update rules dynamically to actively change flow patterns
    - Here, PolKA helped us use Source Routing for active flow changes

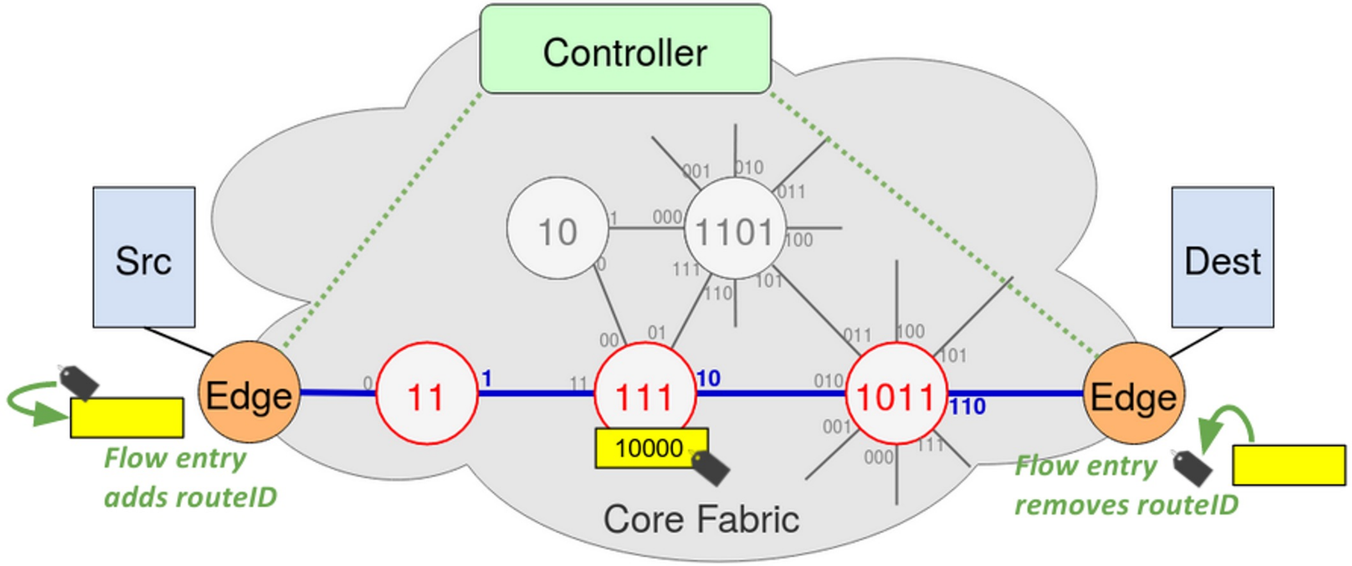
Precap On HECATE

How can we instigate the 'Change'?



PolKA- Source Routing

- Better than traditional table-based routing include a reduction in network states and the optimal use of network capacity
- The route label represents an ordered list of output ports. Each hop executes the forwarding operation by popping the first
- PolKA uses RNS to determine route labels and polynomial identification numbers using Chinese Remainder Theorem.



C. Dominicini et al., "PolKA: Polynomial Key-based Architecture for Source Routing in Network Fabrics," 2020 6th IEEE Conference on Network Softwarization (NetSoft), Ghent, Belgium, 2020.

How does Polynomial Key-based Architecture work?

- Three polynomials:
  - routeID: a route identifier calculated using the CRT.
  - nodeID: to identify each core node.
    - Irreducible polynomial which is a prime number representation in GF2
  - portID: to identify the port or a set of ports on each core node.

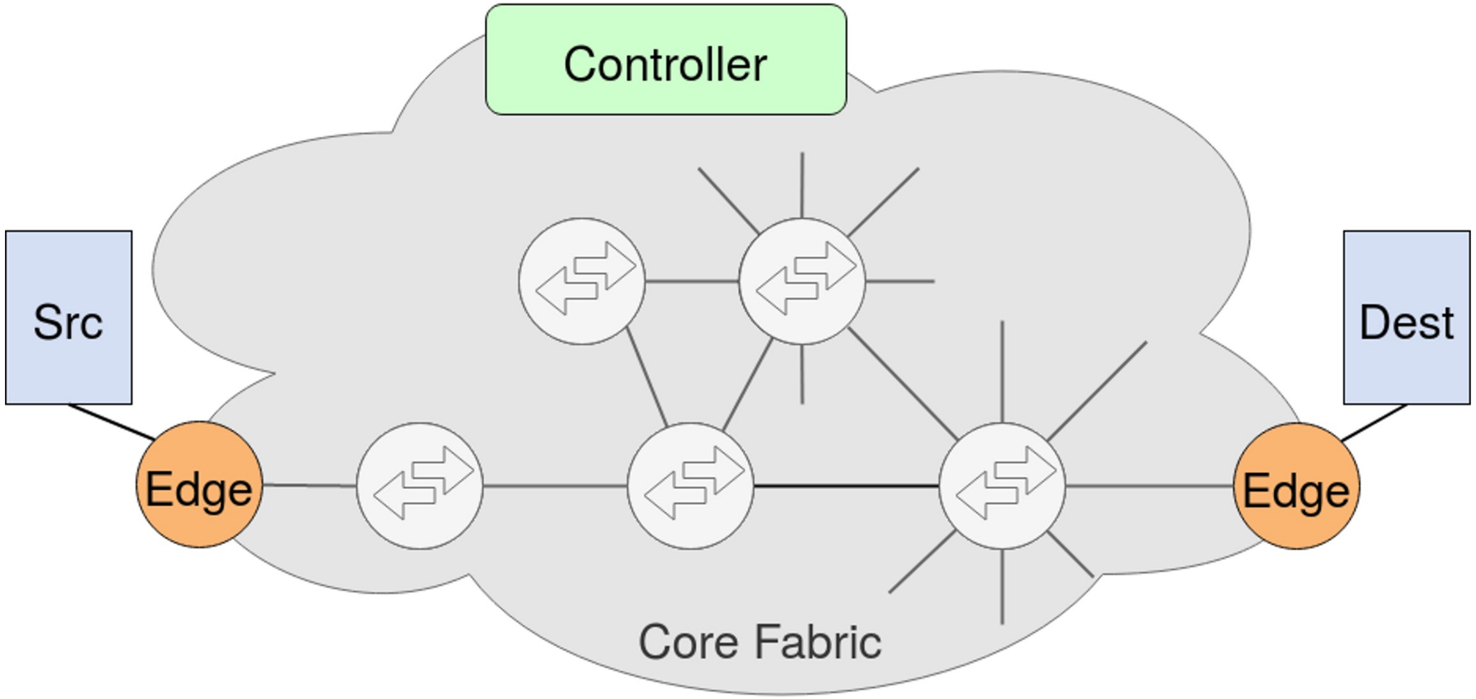
The forwarding uses a mod operation (remainder of division):

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

Oak Ridge National Laboratory

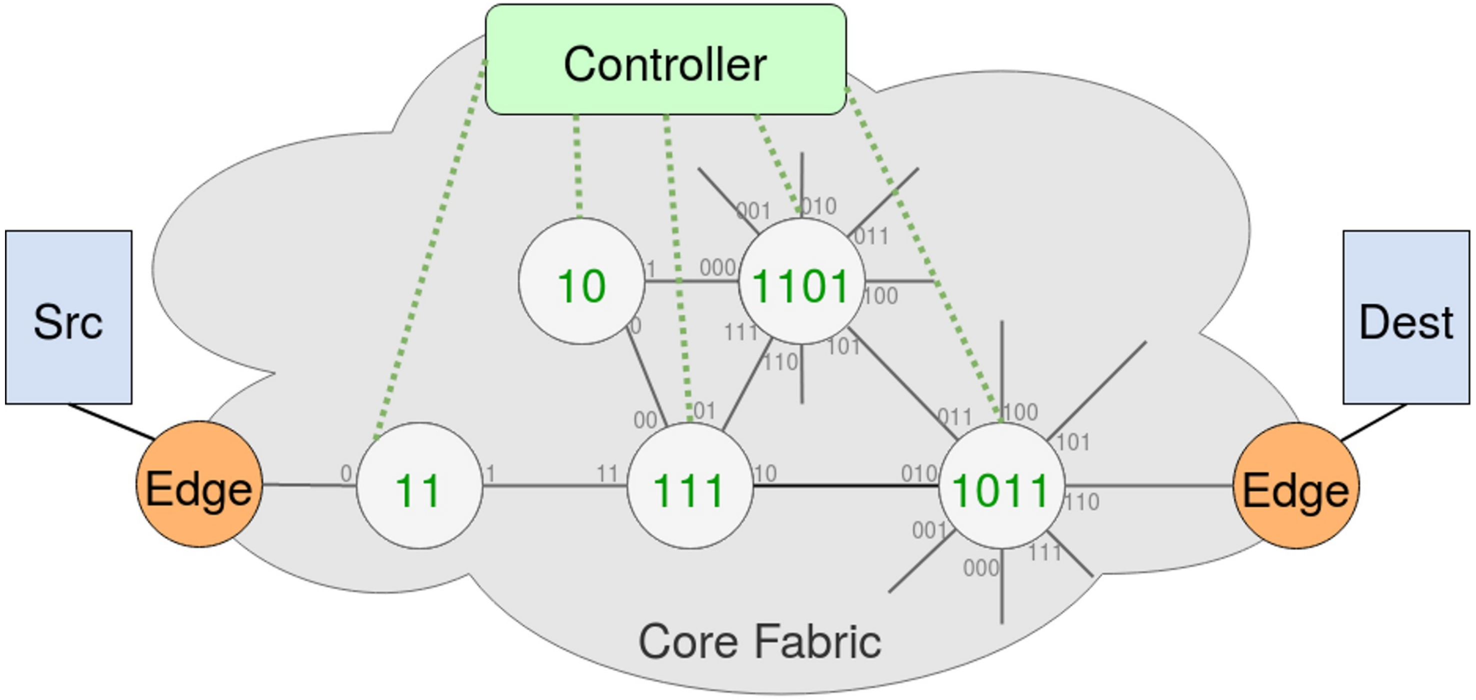
Simple example of how PolKA works

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



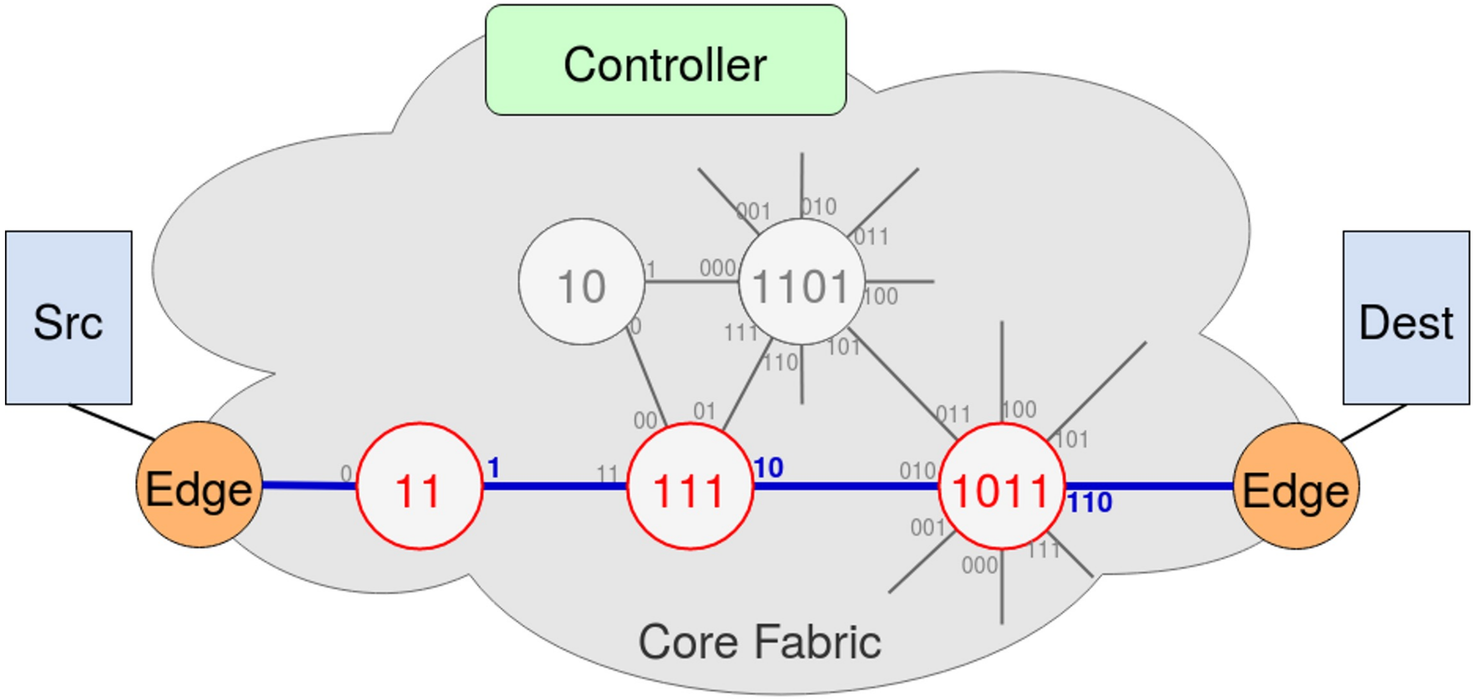
Configuration phase of PolKA network

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



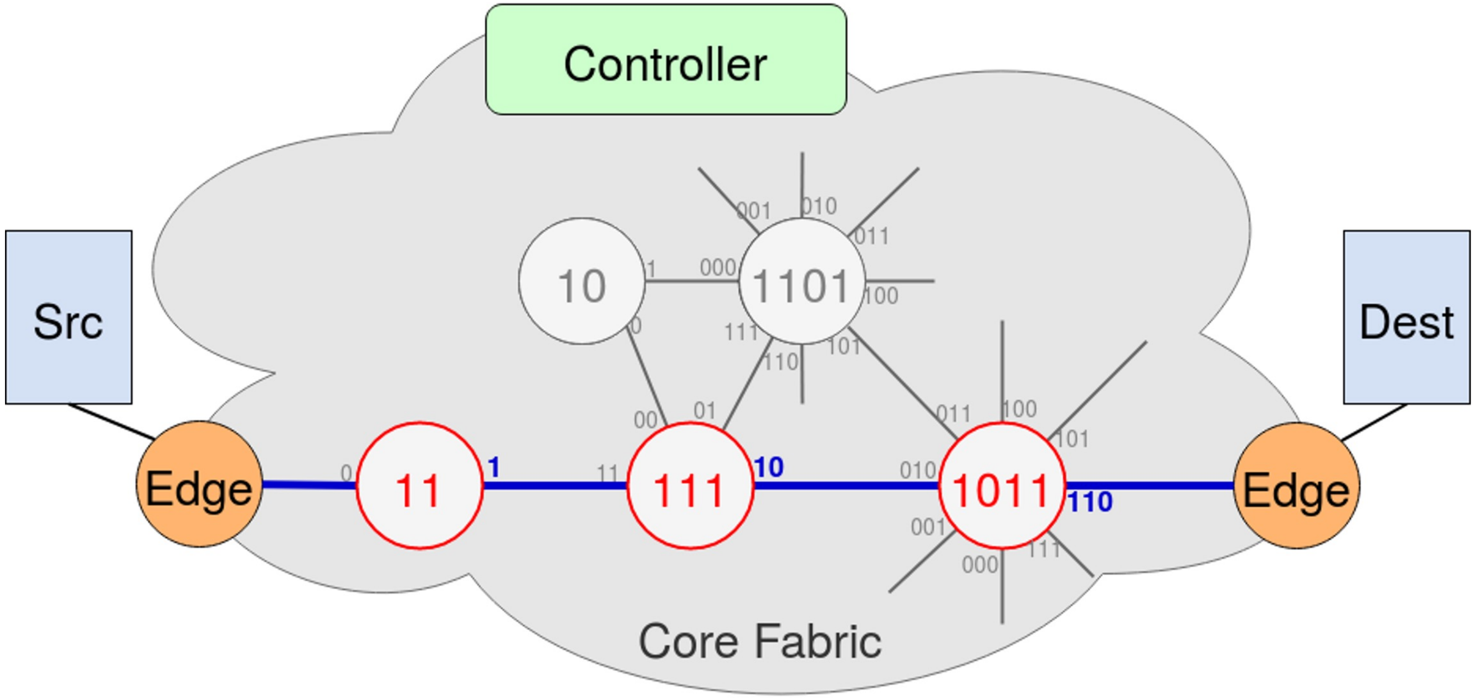
Selecting a path for flow assignment

- 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}



Nodes and ports in their polynomial representation

- 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}



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

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

Computing the routeid with CRT

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

1	=	$\langle 10000 \rangle_{0011}$
10	=	$\langle 10000 \rangle_{0111}$
110	=	$\langle 10000 \rangle_{1011}$

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

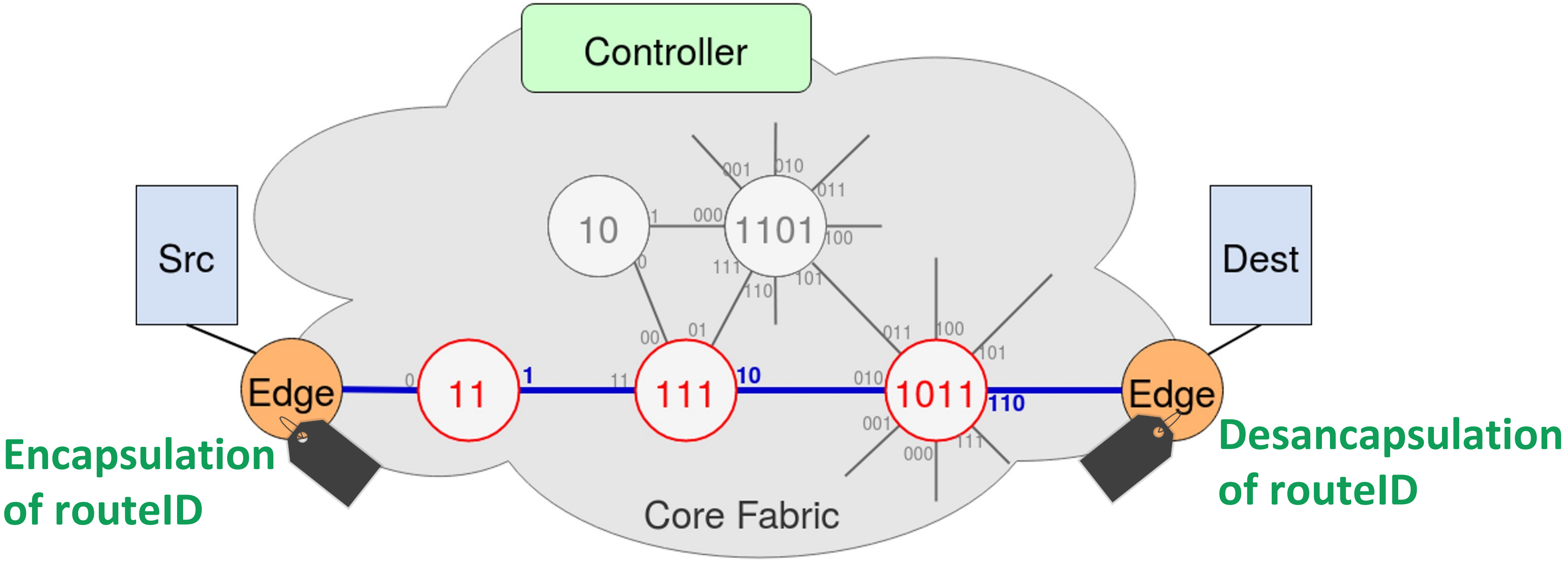
$$o_1(t) = 1$$
$$o_2(t) = t = 10$$
$$o_3(t) = t^2 + t = 110$$

Calculate routeID with CRT

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

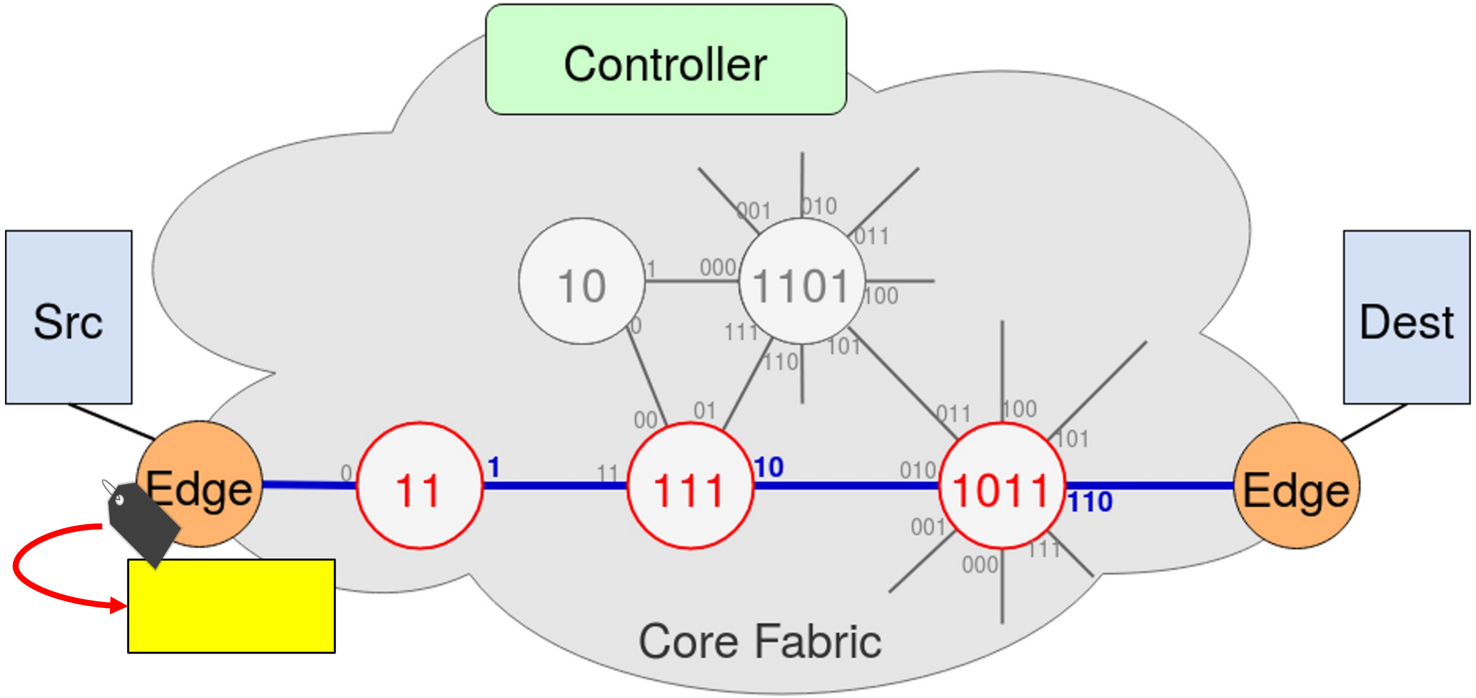
Installation of rules at the edges

- The Controller installs rules at the edges to add/remove routeIDs.



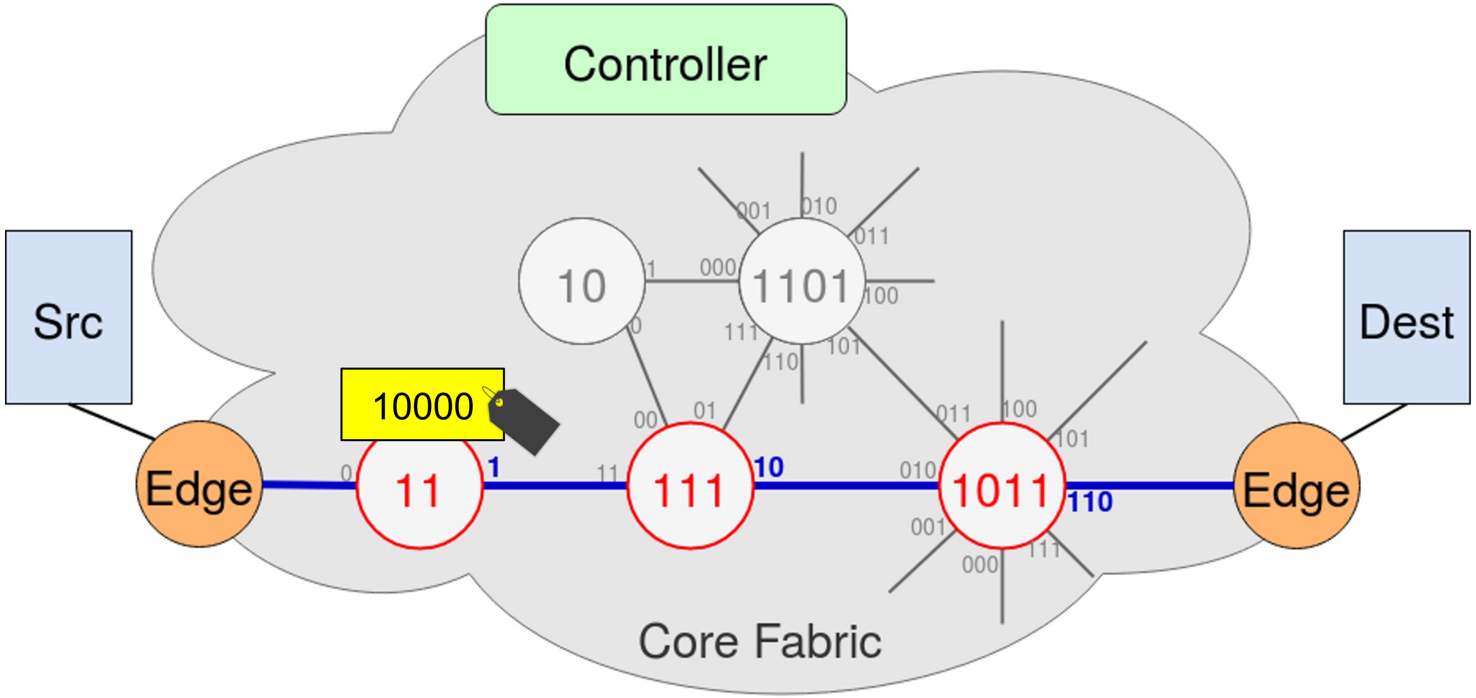
Ingress edge adds the labels

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



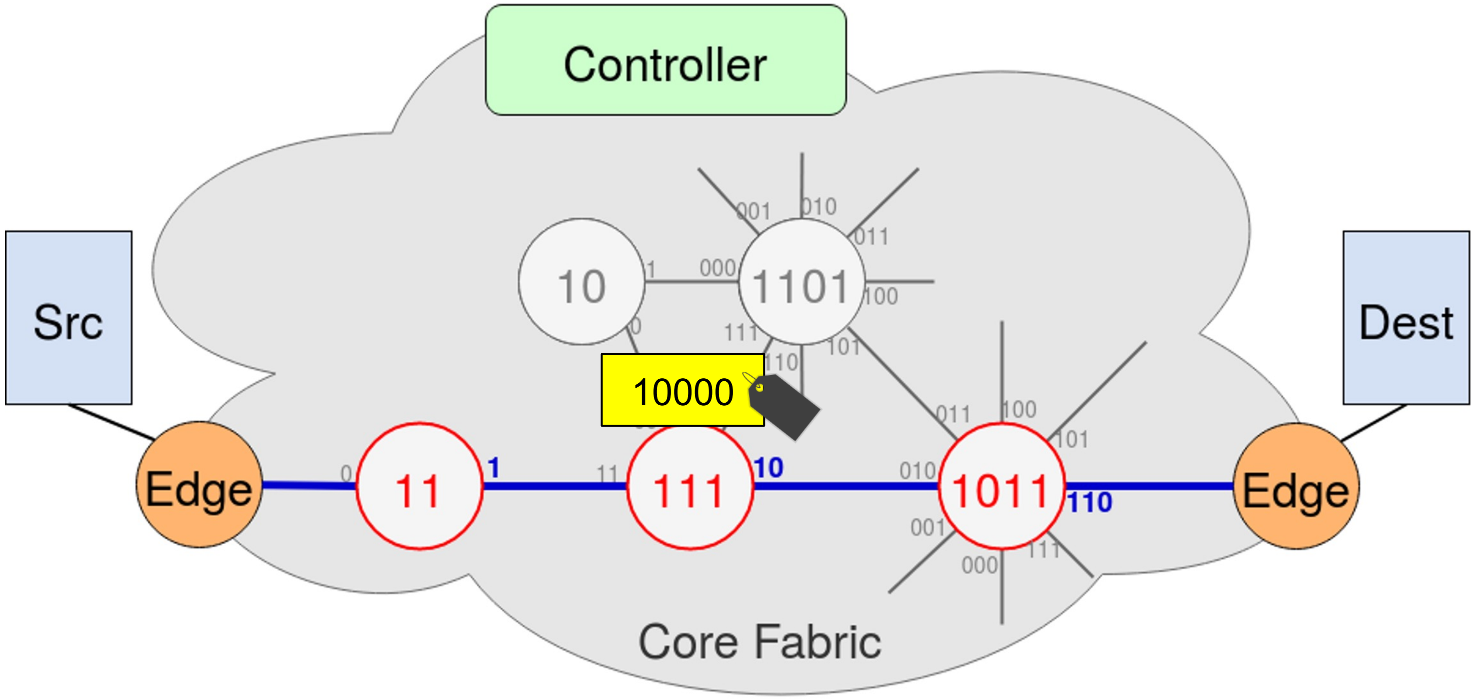
Packet forwarding at the core node

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



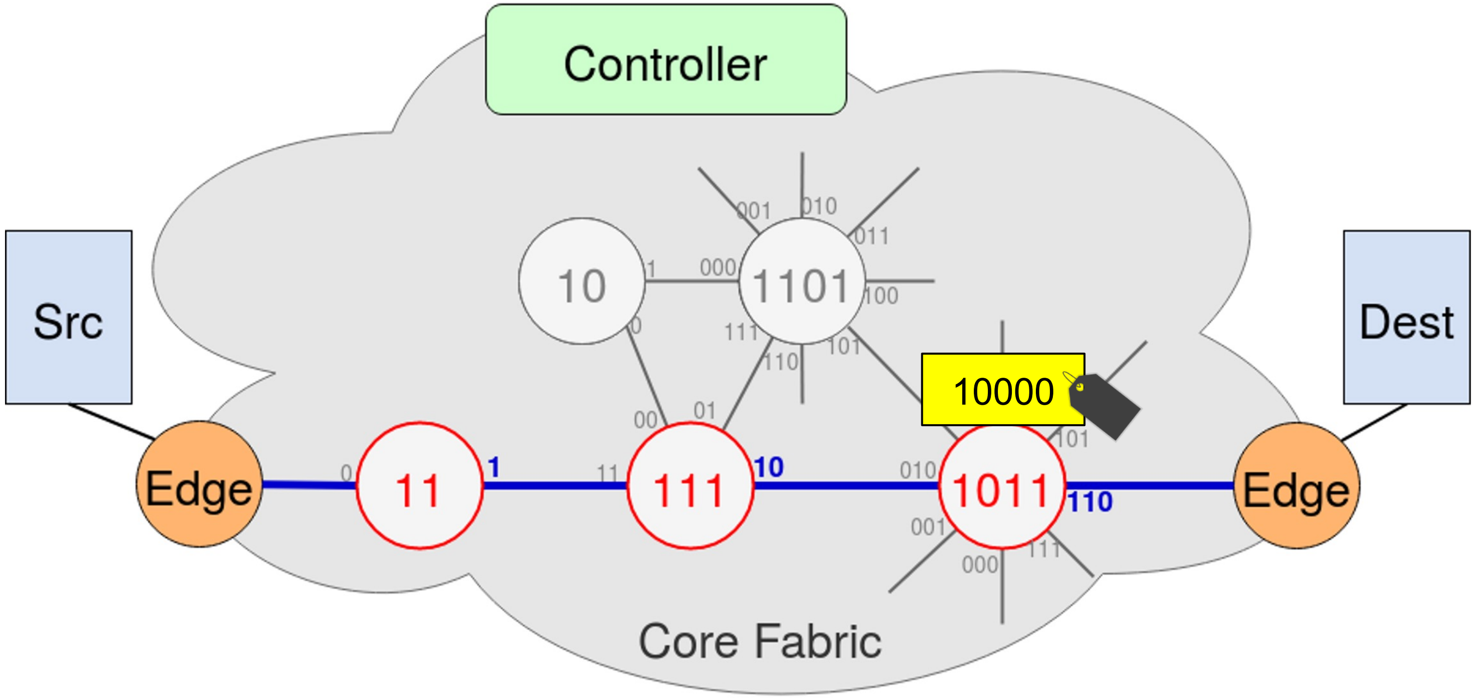
Packet forwarding at the core node

- Forwarding using mod operation: $\langle 10000 \rangle_{111}$ = 10 → output port
  - Stateless core nodes with no routeID rewrite! No tables !



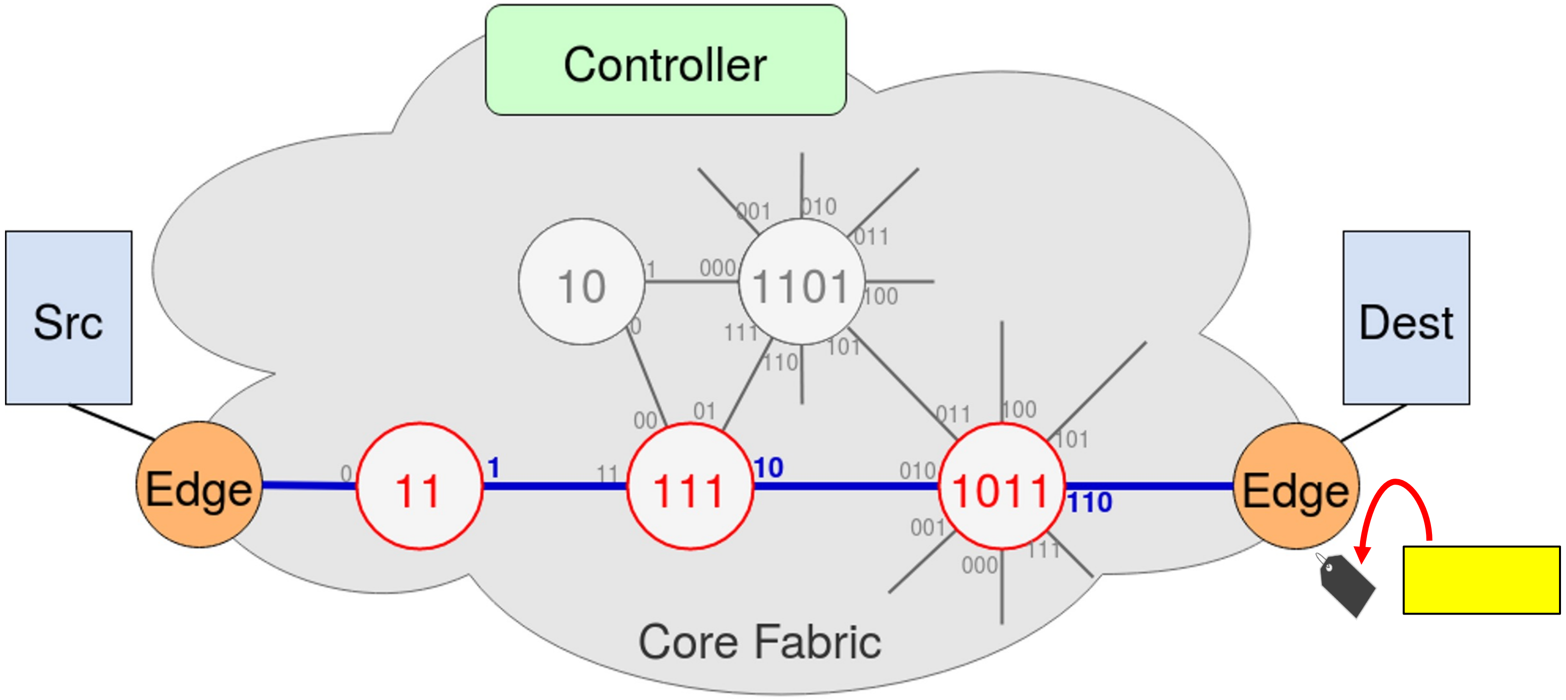
Packet forwarding at the core node

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



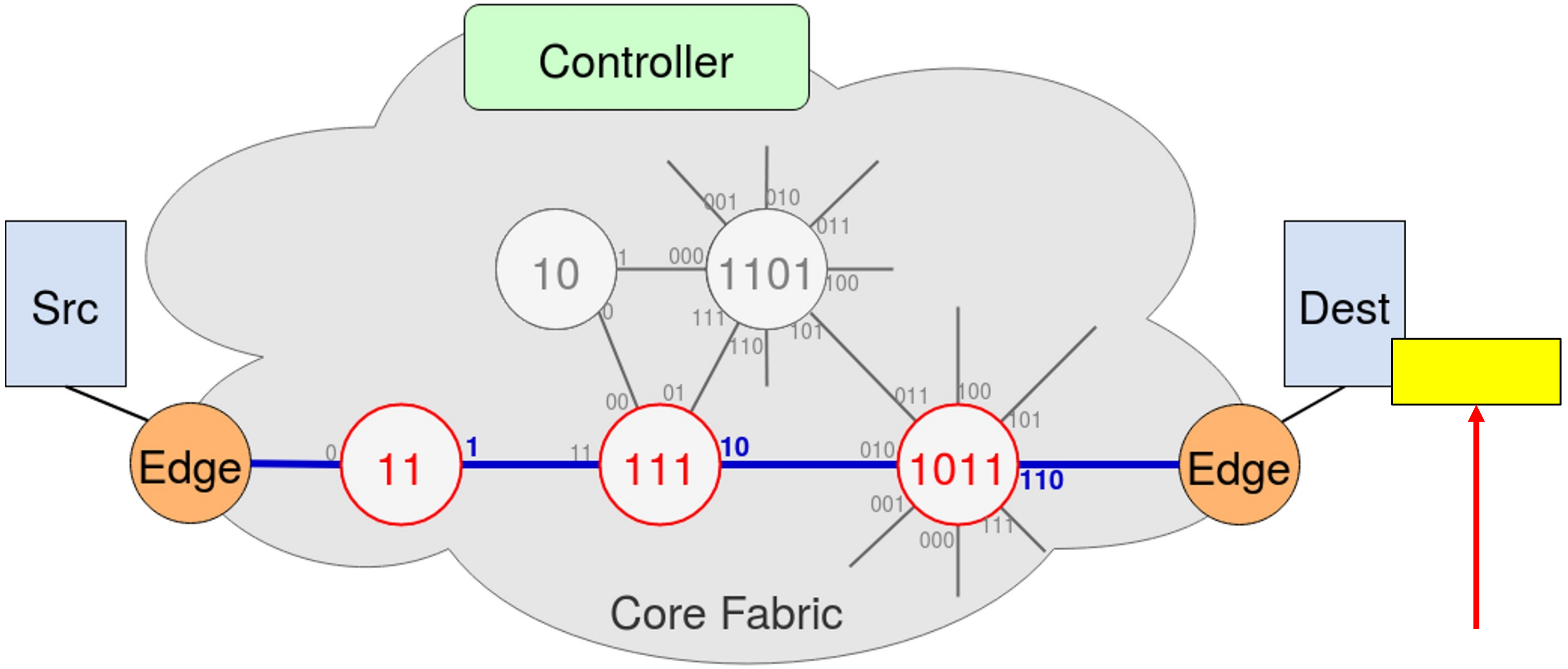
Egress edge removes the label

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



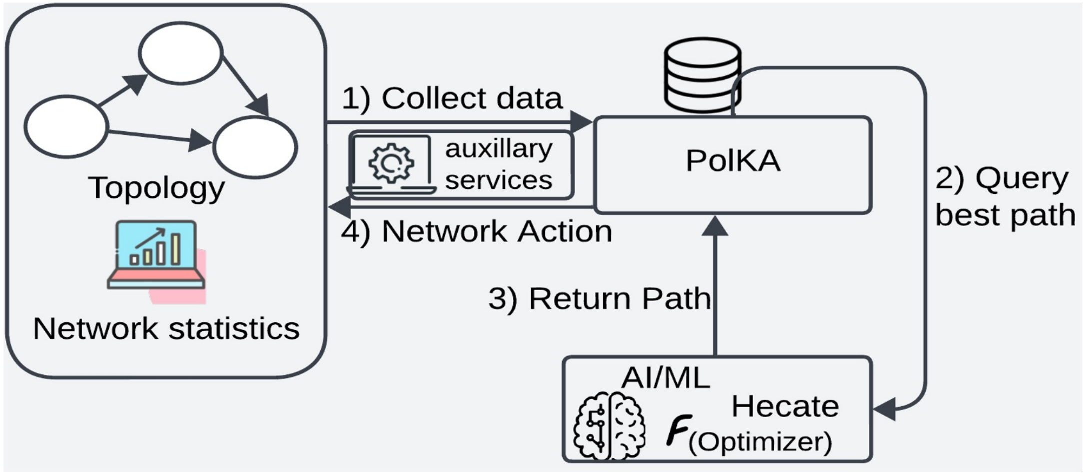
Egress edge removes the label

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



Merging PolKA and Hecate through APIs

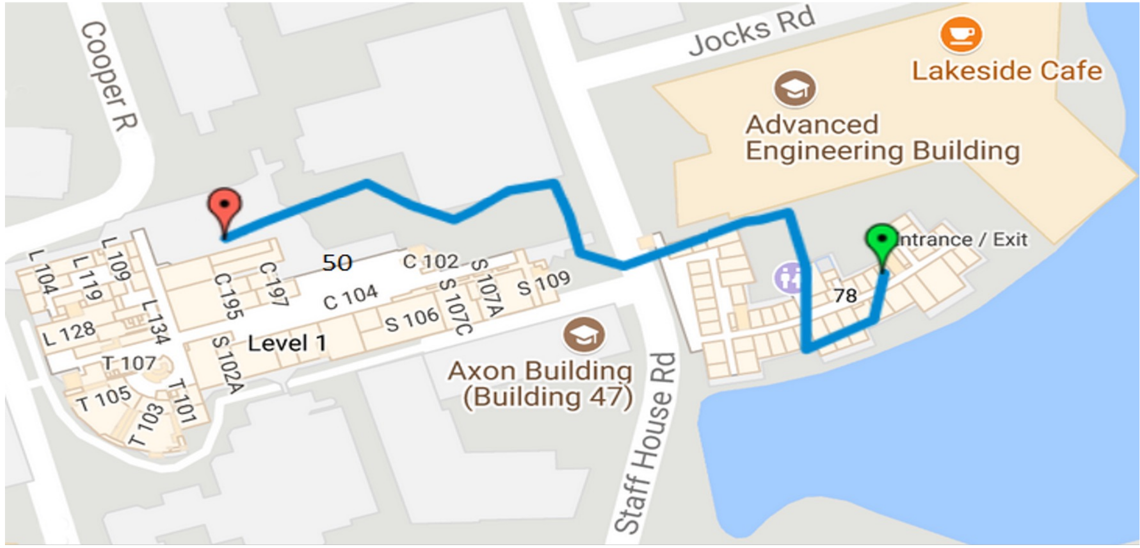
- The proposed framework enables efficient adaptive routing via leveraging multiple network service, including:
  - PolKA SR routing service
  - Hecate AI-Network Driven service
  - Optimizer module for route selection
  - Auxiliary services (e.g., scheduler, controller, etc) for orchestrating control and data messages between PolKA and Hecate



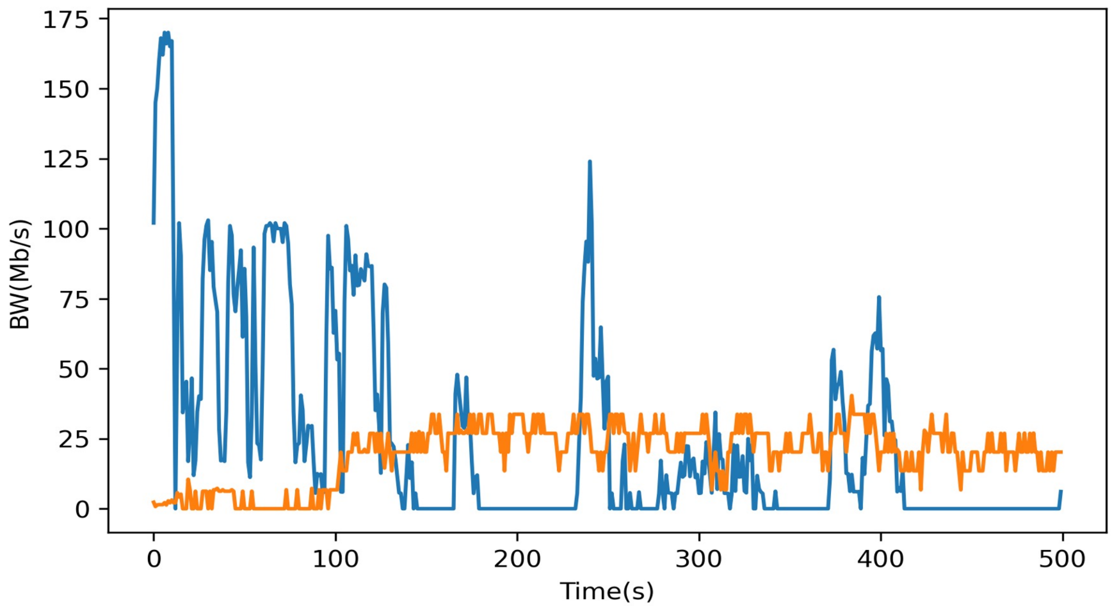
PolKA-Hecate integration framework

Using Data-driven Learning (Proof of concept)

- Real network dataset is leveraged for testing and validating Hecate service at the proposed routing framework.
  - Dataset is collected over a certain path at The University of Queensland (UQ).
  - Measuring bandwidth of different wireless networks (WiFi, and LTE)
  - Different bandwidth patterns of indoor and outdoor are collected over 500 seconds.



Selected path for measuring links bandwidth at UQ

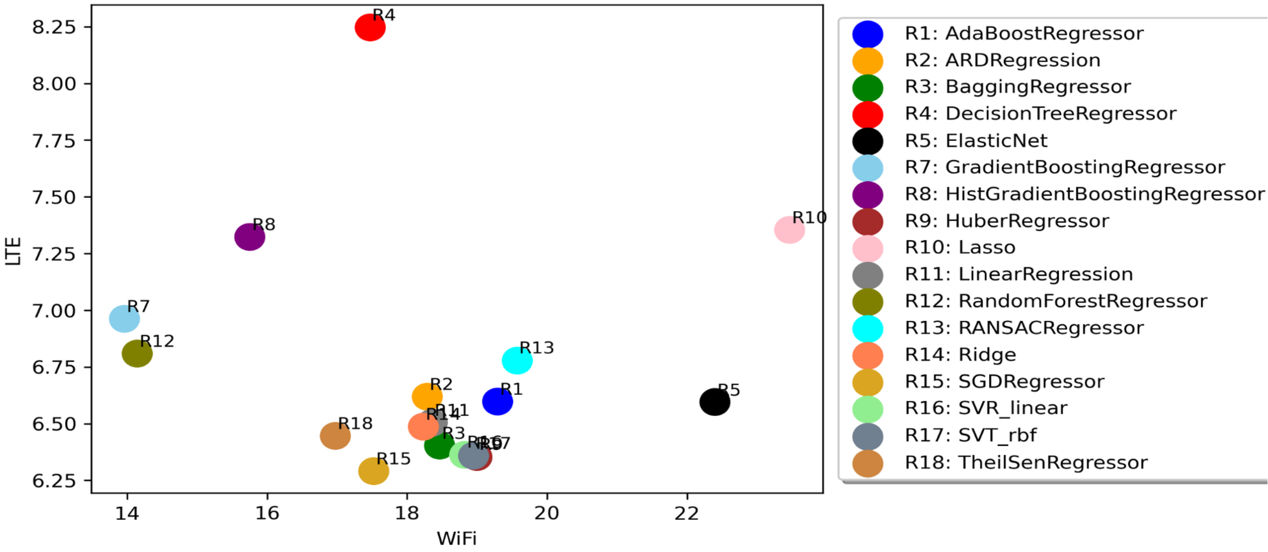


WiFi (Path 1) vs LTE (Path 2) bandwidth

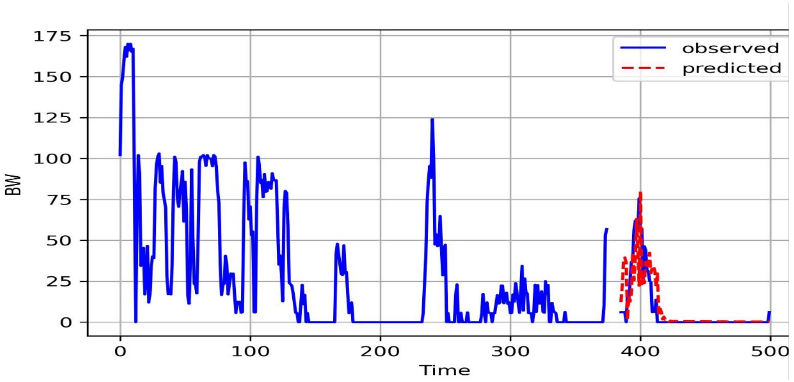
Oak Ridge National Laboratory

Exposing Supervised and Prediction Methods

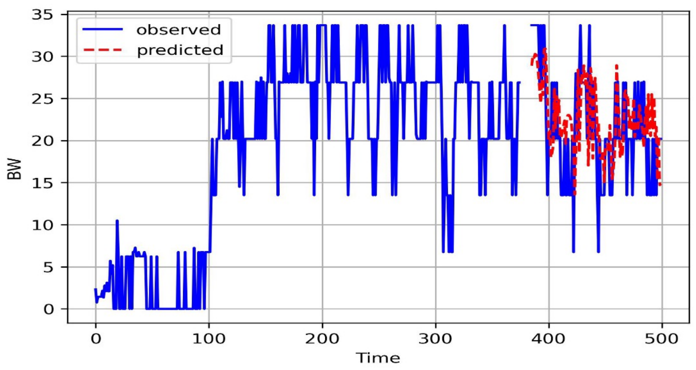
- Hecate APIs exposed 18 ML regressors that can estimate bandwidth and return optimum routing information
- Multiple regressors are explored for predicting next bandwidth measurement based previous measurements
  - 10 history values t(i)-to t(i-9) used to predict t(i+1)
- UQ dataset are utilized for training and testing the models
  - the dataset is split to 75:25
- default models hyperparameters used
- RMSE is opted as a performance metric



RMSE of multiple regression models applied on the bandwidth of WiFi (Path 1) and LTE (Path 2).



Observed and predicted WiFi (Path 1) bandwidth using R12:RFR

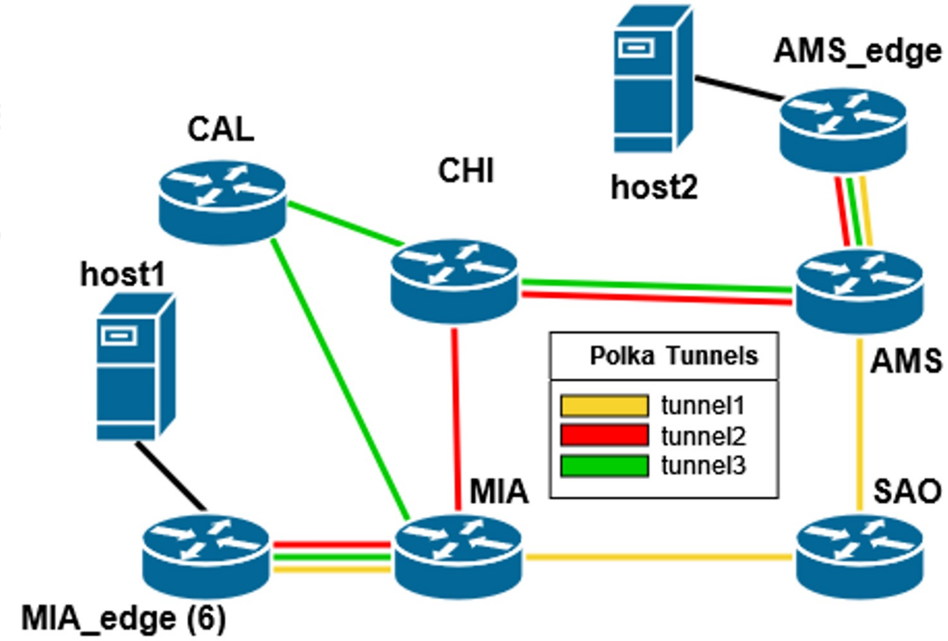


Observed and predicted LTE (Path 2) bandwidth using R12:RFR

Oak Ridge National Laboratory

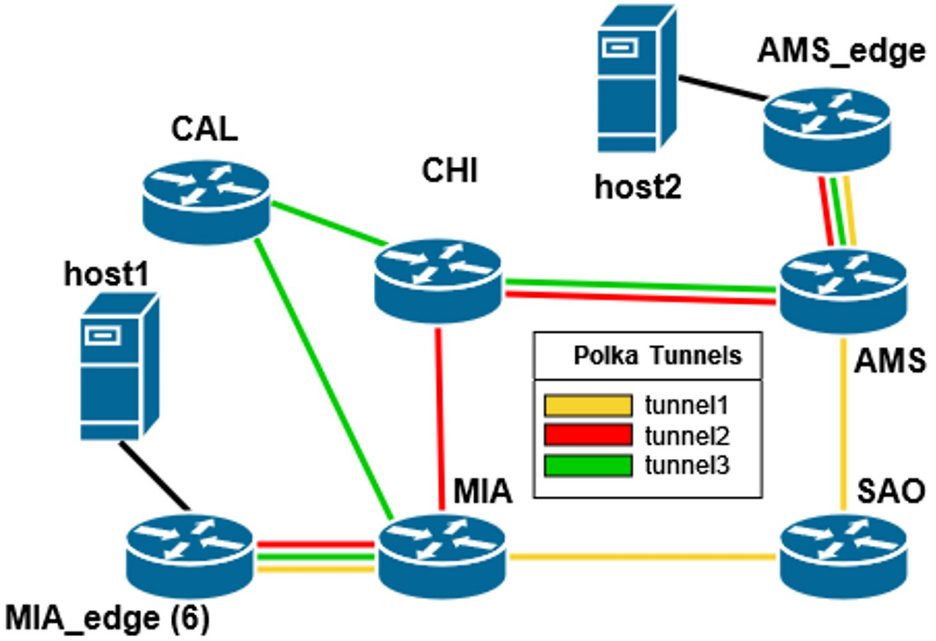
PolKA enabling Path-Aware Networking

- Path aware networking :
  - exposes **all the existing paths** in the topology to the endpoints
  - offers selection of **any available path** to the the endpoints
  - measures continuously the path performance for optimization
    - RTT
    - Latency
    - Loss
    - Link occupation
    - more



PolKA enabling Path-Aware Networking

- Path aware networking :
  - exposes **all the existing paths** in the topology to the endpoints
  - offers selection of **any available path** to the the endpoints
  - measures continuously the path performance for optimization
    - RTT
    - Latency
    - Loss
    - Link occupation
    - more



- However, since each host has its own perspective, then sub-optimal decisions can occur

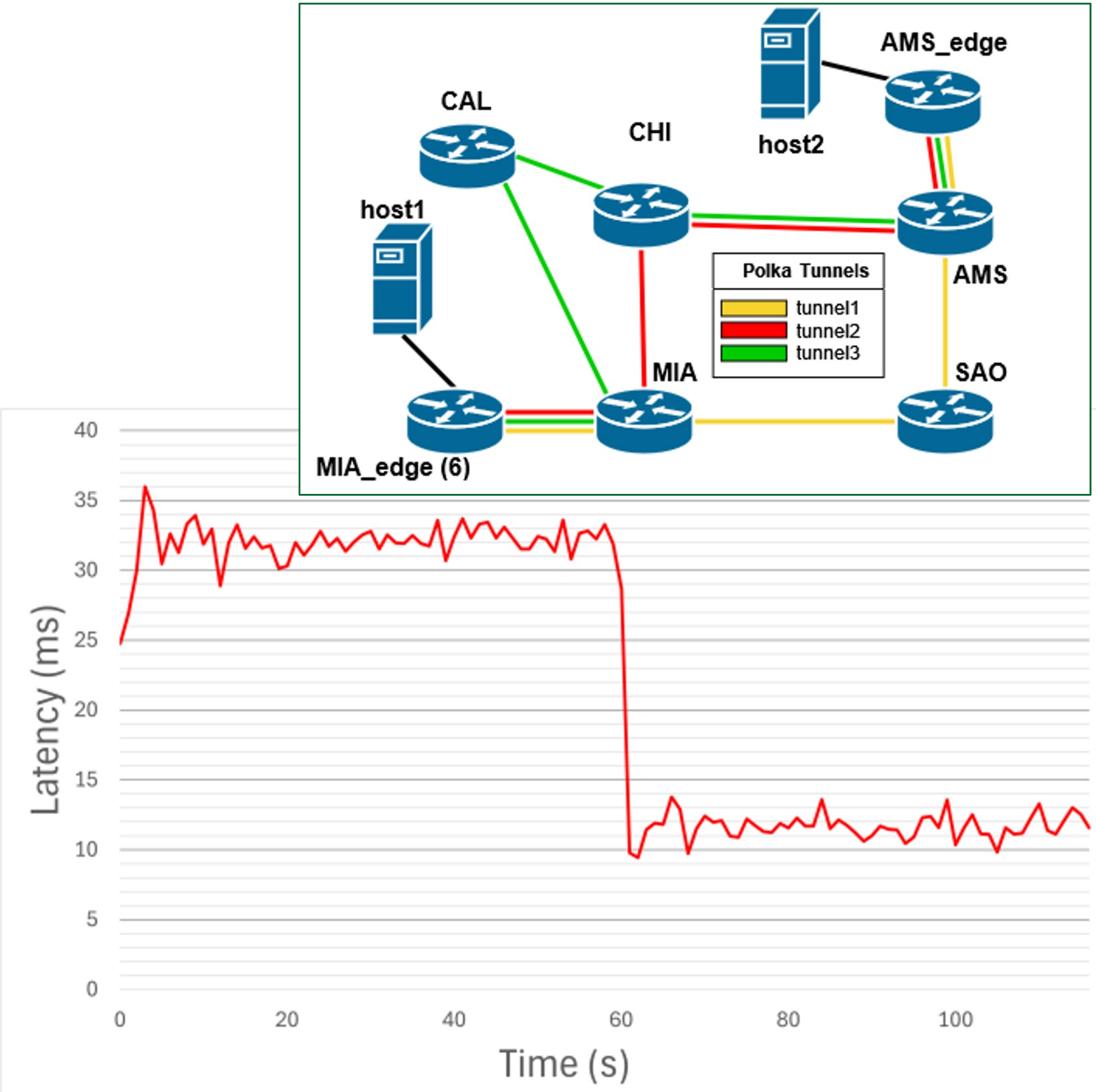
- To provide a dynamic optimization : :
  - Continuously adjust path selection (by Hecate AI) and resource allocation based on changing network conditions (performance metrics from the paths) and application needs (DIS datasets and flows duration)

Experiment 01: Agile migration to a path with lower latency

- Initially, we configure the flow to traverse the path through the MIA-SAO-AMS nodes. ICMP packets are sent between host1 and host2 to measure latency metrics.

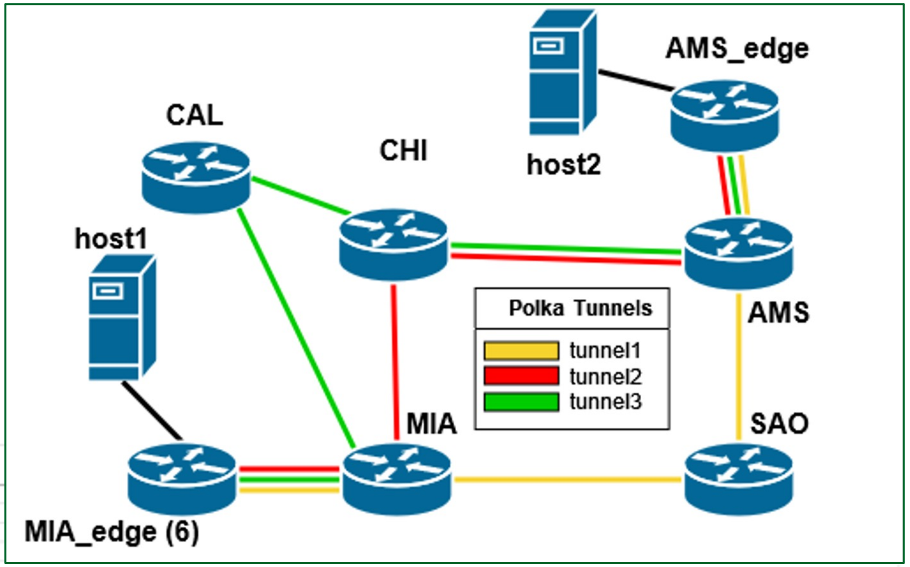
- Leveraging a path-aware network to minimize latency, HECATE identifies and selects an optimized path, MIA-CHI-AMS. In PolKA, redefining the path is only a matter of updating the routeID at the source.

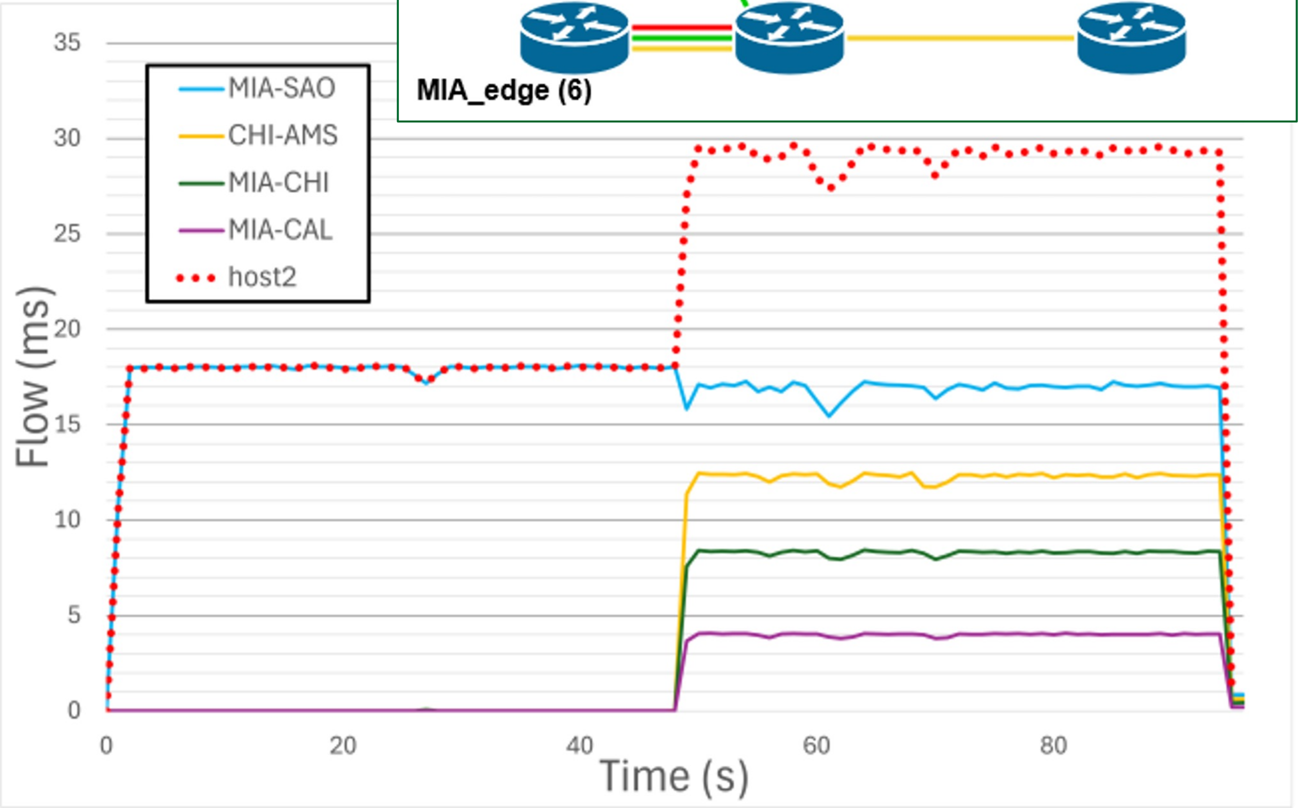
- As a result, the user perceives a better experience by reducing the latency to 10 ms.



Experiment 02: Flow aggregation with multiple paths to increase available bandwidth.

- In the topology, each path is configured with different link speeds.
- Initially, we generate TCP flows - all allocated to path 1 (yellow). This results in the maximum throughput capacity for path 1.
- HECATE collects metrics from the network, such as bandwidth, and uses this information to determine the optimal path allocation. After that, HECATE selects one flow to path 2 (red) and another to path 3 (green). For that, PolKA redefining the path by updating the routeID of each flow at the source.
- As a result, the average throughput improved, and total throughput increased as the flows utilized different paths to reach the destination host.





What's Next

HECATE exposes APIs to provide ML decisions to PolKA to actively switch paths

NRE Demo at SC Theater at 5:00pm Tuesday

PolKA Demo at Caltech Booth 2:00pm Tuesday

DOE Booth at 1:00pm Thursday

Integrating a monitoring tool to help Hecate and PolKA perform better communication of results

P4 implementation on P4 Labs and FABRIC to help push a truly self-driving network