P4 Implementation of a Stateful Data Plane and its Application to Failure Recovery

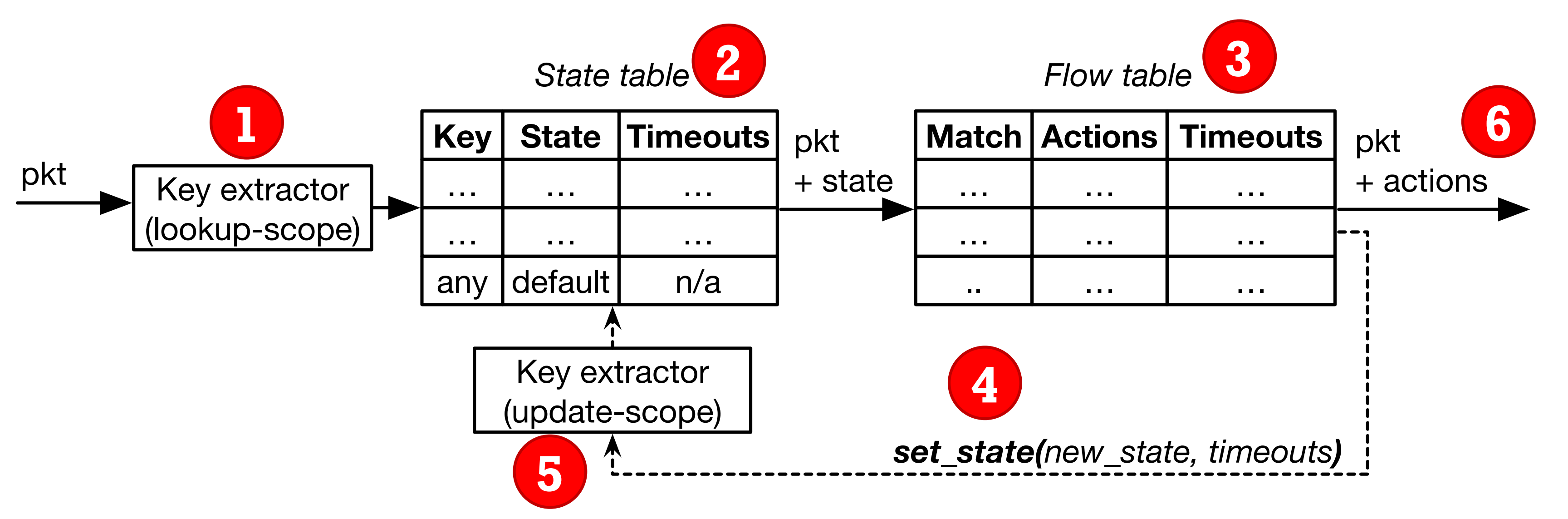
A. Capone, C. Cascone, L. Pollini , D. Sanvito
Politecnico di Milano

http://www.openstate-sdn.org

Introduction

OpenState is a stateful pipeline design (originally developed as an OpenFlow extension), that allows packets to be forwarded on the basis of "flow-states", maintained and updated by the fast path itself as a consequence of packet-level events (i.e. table match) and timers. The demo presents an application for failure resiliency that exploits the fast adaptation of the forwarding behavior in the data path. This application provides i) a **programmable detection mechanism** based on switches' periodic link probing and ii) a **fast reroute of traffic flows** even in case of distant failures, regardless of controller availability. **It can guarantee short (i.e. few ms) failure detection and recovery delays, with a configurable trade off between overhead and failover responsiveness.**

OpenState Architecture


State table
Flow table
pkt
Key extractor (lookup-scope)
Key | State | Timeouts
any | default | n/a
pkt + state
Match | Actions | Timeouts
pkt + actions
Key extractor (update-scope)
set_state(new_state, timeouts)
1
2
3
4
5
6

Steps:

1) "Lookup-scope" used to extract a "flow key" from packet headers
2) State table queried using the flow key
   – Return 0 (default state) if no entry is found
3) Match on state metadata
4) Set-state action to update/insert values in the state table
5) "Update-scope" can be defined to perform <u>cross-flow state updates</u>
6) Packet sent to the next stage in the pipeline

State table

Key	State	Timeouts
A,B,w,z	1	Idle, hard, rollback state
…	…	…
…	…	…
* (any)	0 (default)	n/a

Key: Equivalent to an exact match performed always on the same fields defined by the lookup/update scope.

State: All flows start with state 0 (last row). Entries are populated by means of set-state action or by the controller.

Timeouts: Similarly to OpenFlow, **idle & hard timeouts can be defined** for each entry. A programmer can optionally specify a **"rollback state"** (non default) to be used when a timeout expires.

http://github.com/OpenState-SDN/openstate.p4

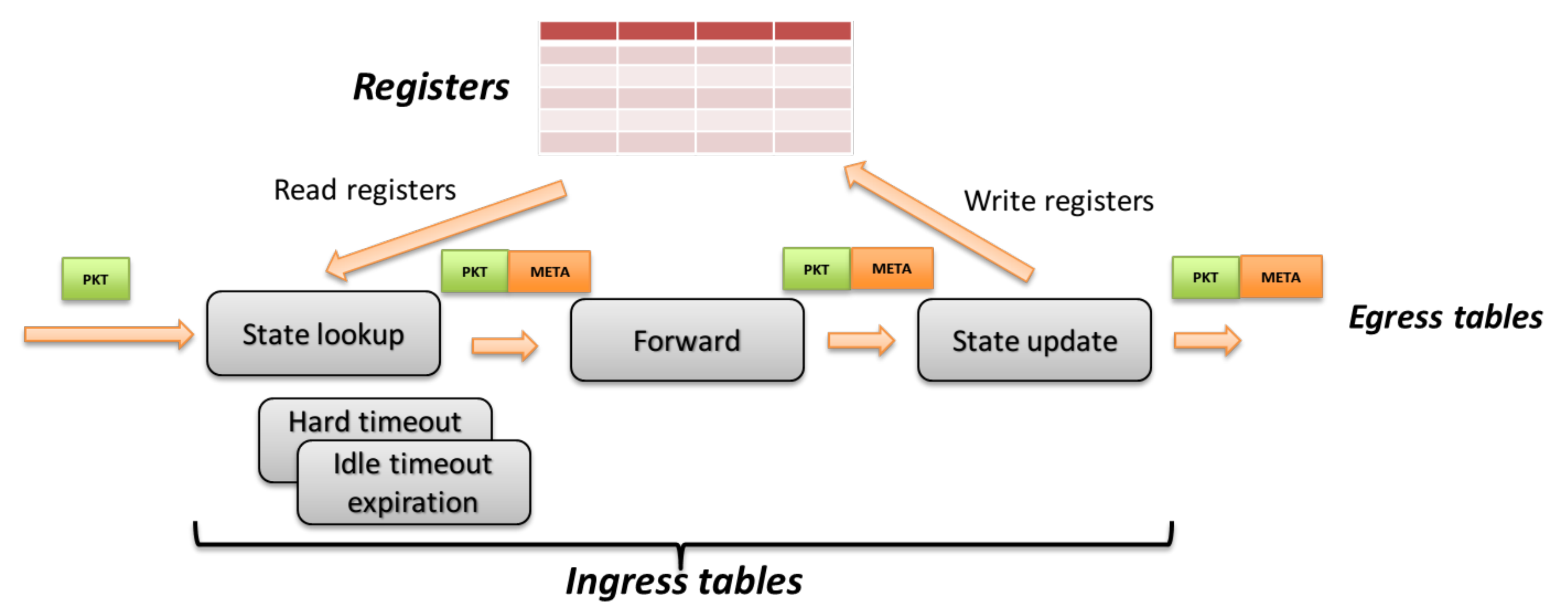
openstate.p4

An OpenState equivalent design can be obtained in P4 by combining:

- Registers (stateful memories);
- Hash value generators (functions that operate on a stream of bytes from a packet to produce an integer);
- Definition of new header types, tables and actions.

Control flow at a glance:

- Hash generators produce a index to access registers
- State lookup → copy from registers to packet metadata
- State update → copy from action parameters to registers



Application Example: Failure Recovery

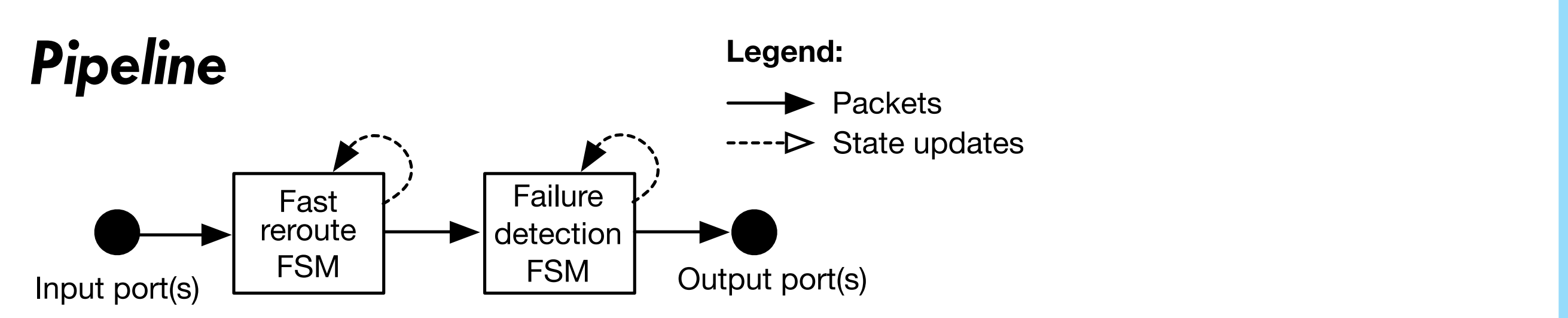
Path pre-planning: primary and backup paths for each possible failure scenario are pre-computed and provisioned to switches at boot time.

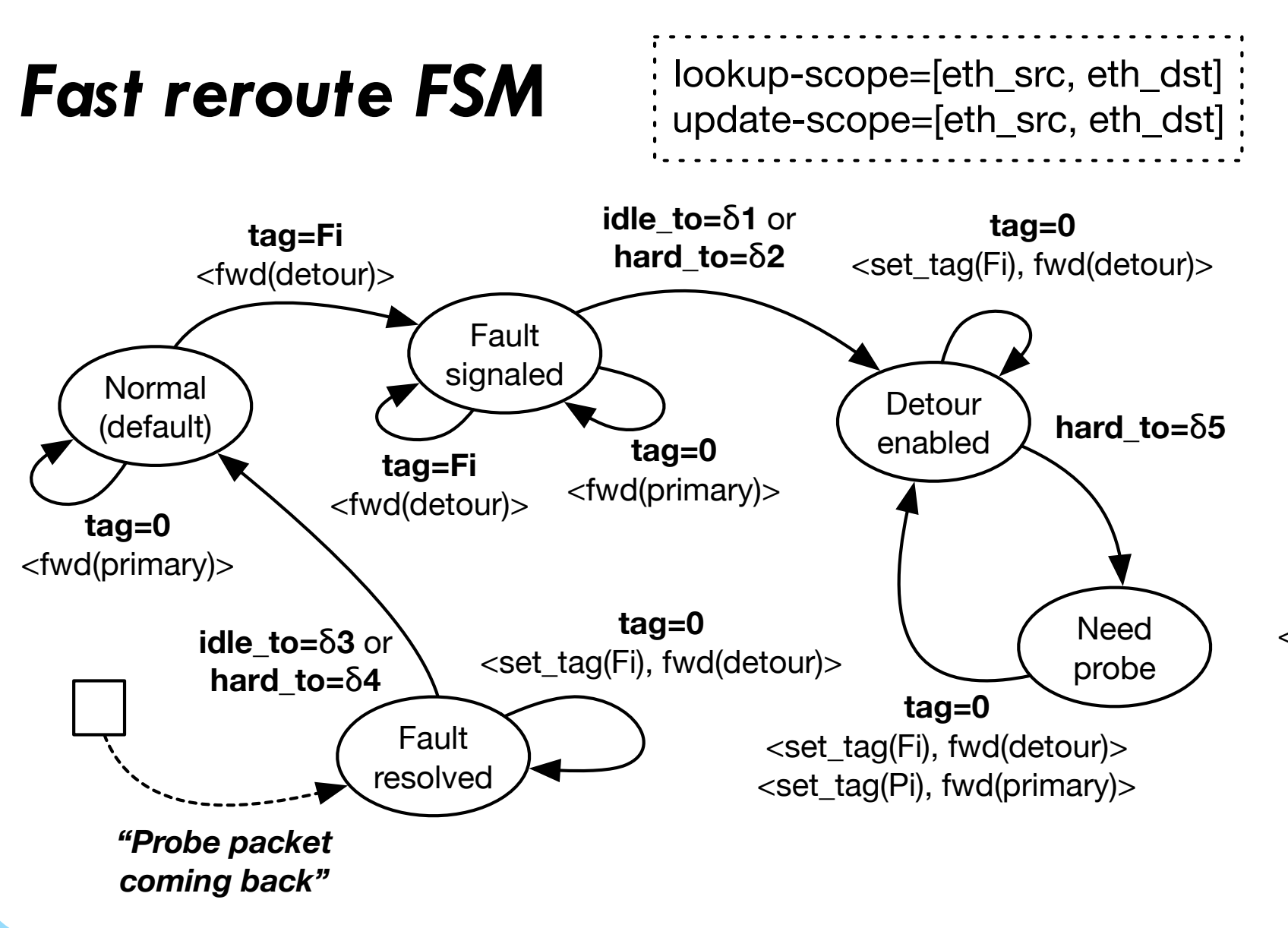
Tag based forwarding: packet labels are used to distinguish between different forwarding behaviors, in order to perform:

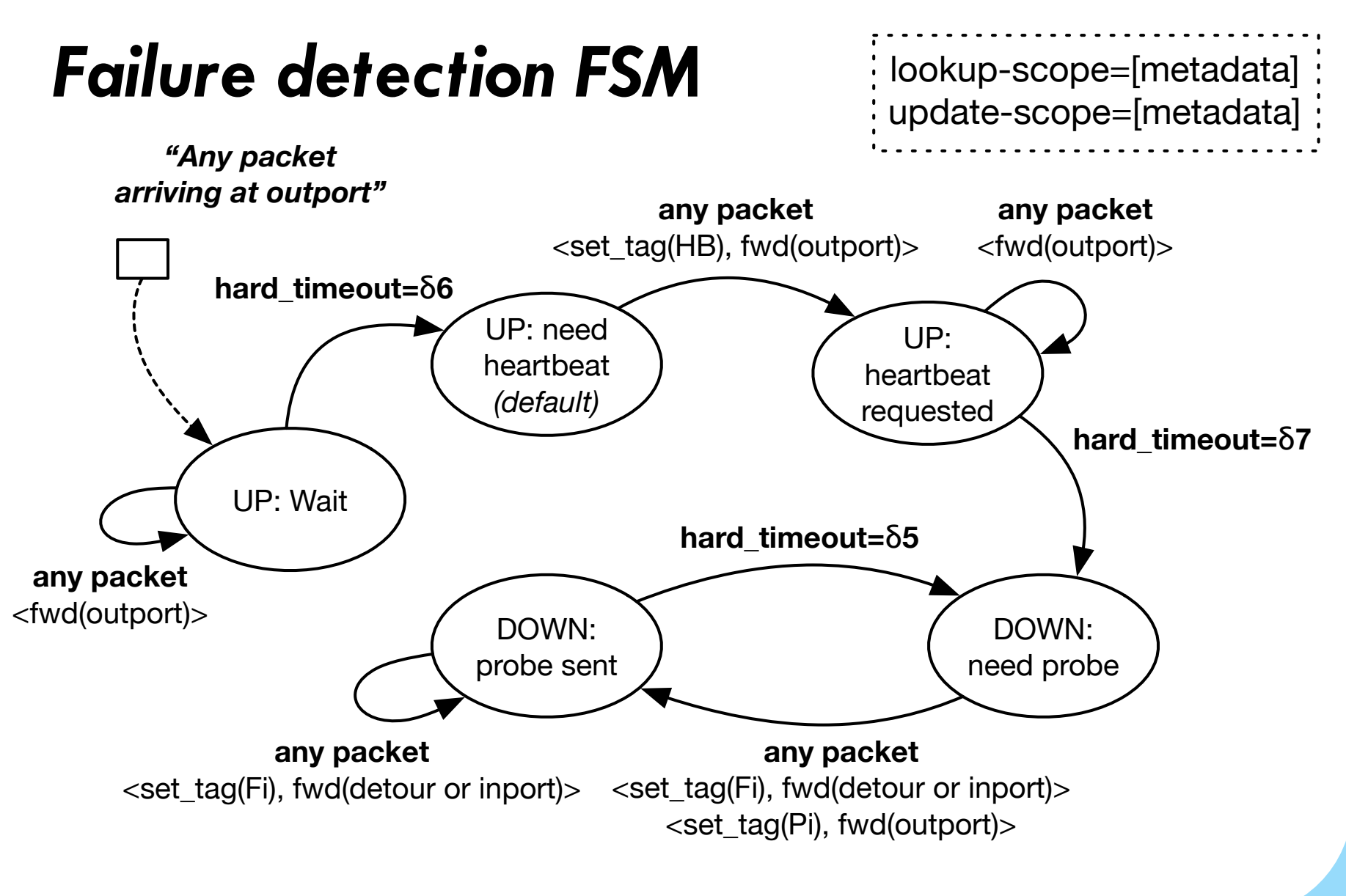
- Normal forwarding (tag=0)
- Heartbeat-based link-level failure detection (tag=HB)
- Switch-to-switch failure signaling (tag=Fi)
- Probing to check path availability after a failure (tag=Pi)

Guaranteed failover delay: depends on target's timestamp resolution.

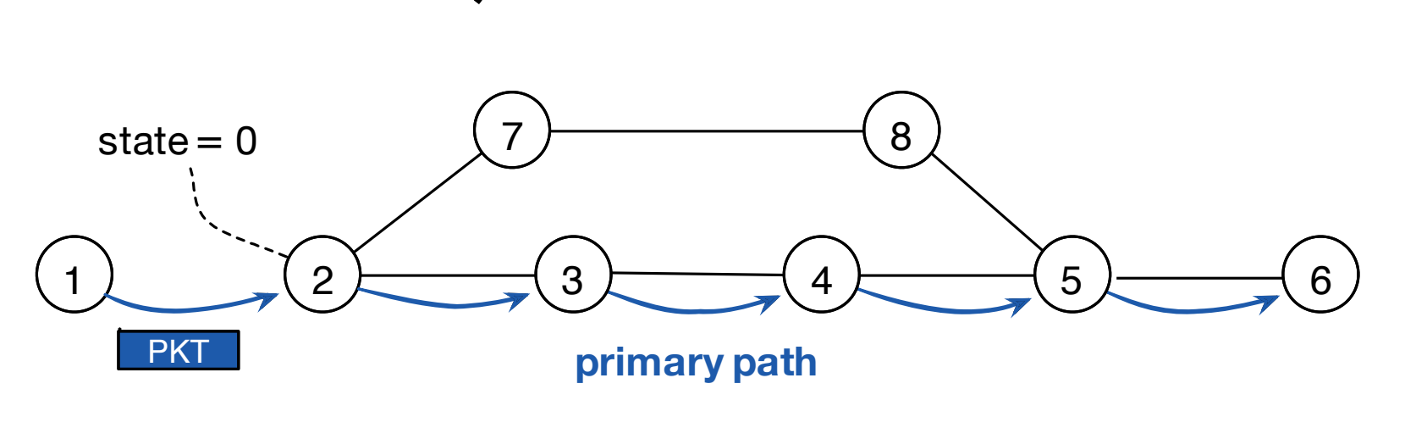
Minimize packet reordering: timeout-based flowlet-aware mechanism to postpone the path failover up to the expiration of a burst of packets.





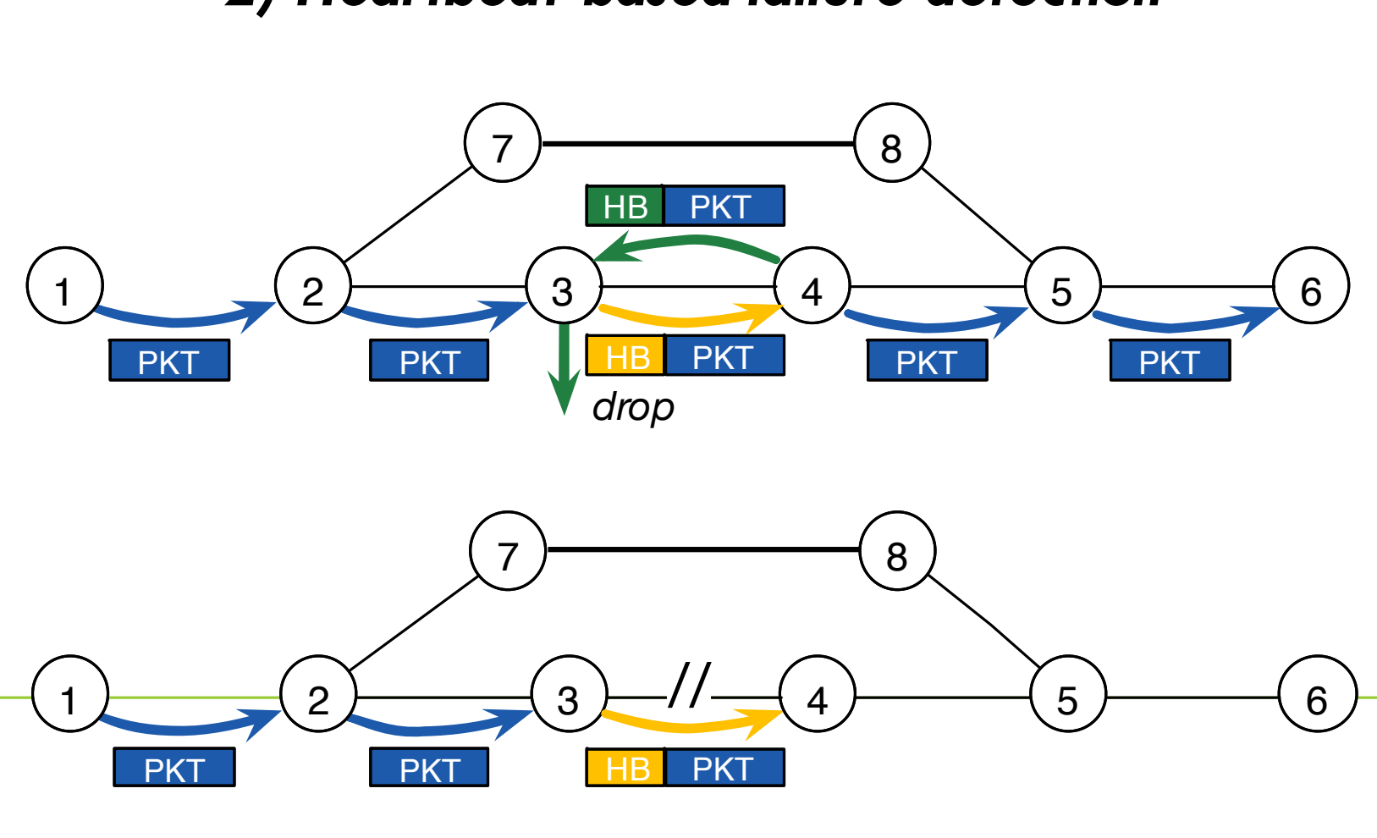

Failure detection FSM
lookup-scope=[metadata]
update-scope=[metadata]
"Any packet arriving at outport"
hard_timeout=δ6
any packet <set_tag(HB), fwd(outport)>
any packet <fwd(outport)>
UP: need heartbeat (default)
UP: heartbeat requested
hard_timeout=δ7
UP: Wait
hard_timeout=δ5
any packet <fwd(outport)>
DOWN: probe sent
DOWN: need probe
any packet <set_tag(Fi), fwd(detour or inport)>
any packet <set_tag(Fi), fwd(detour or inport)> <set_tag(Pi), fwd(outport)>

1) Normal conditions

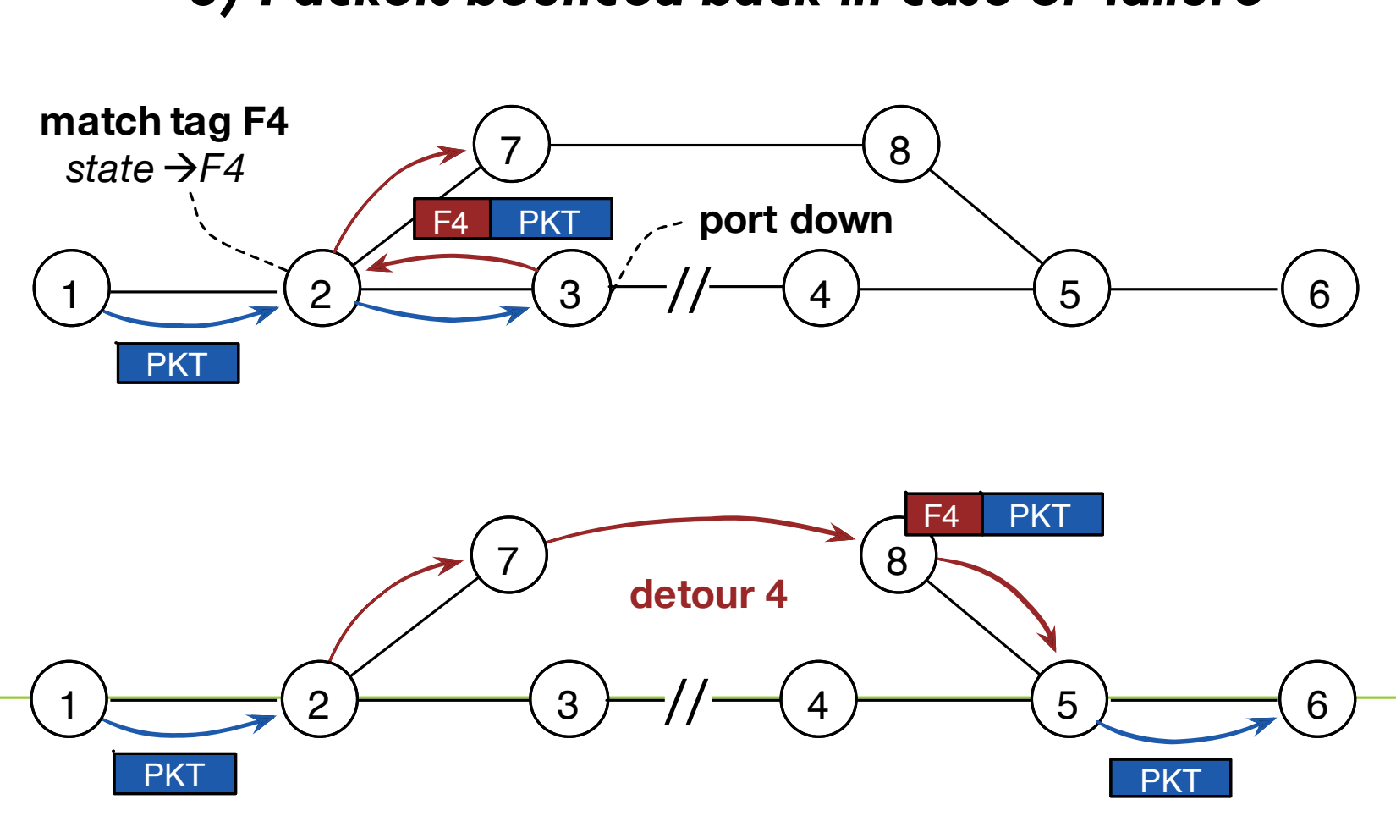


* As long as packets are received from a given port, that port can be used to reliably transmit other packets. When no packets are received for a given interval, a node can request its neighbor to send an "heartbeat". If no reply is received within an interval, the port is declared down.

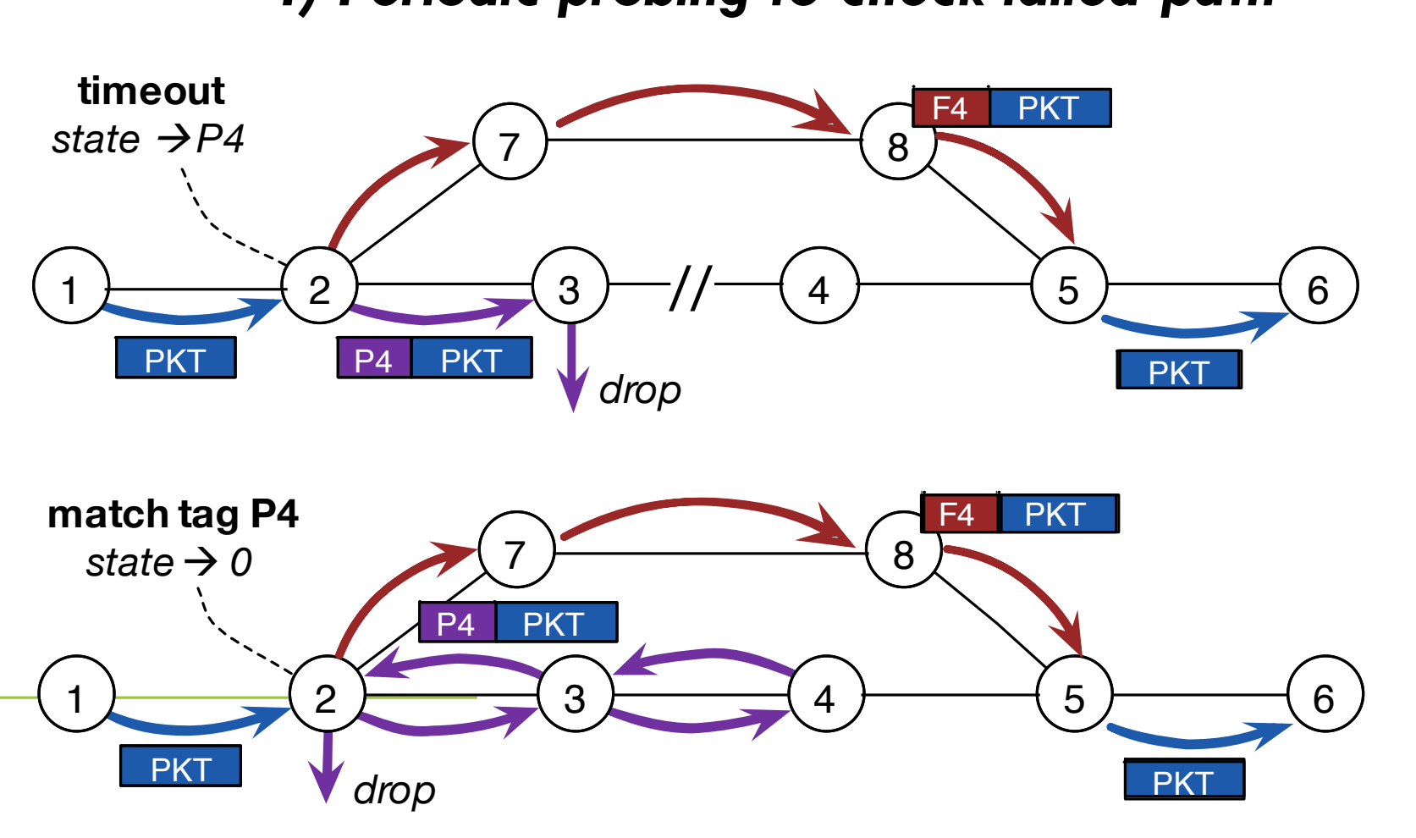
*2) Heartbeat-based failure detection**



3) Packets bounced back in case of failure



4) Periodic probing to check failed path



POLITECNICO MILANO 1863

Beba BEhavioural BAsed forwarding
www.beba-project.eu

References

[1] G. Bianchi, M. Bonola, A. Capone, C. Cascone, "OpenState: programming platform-independent stateful OpenFlow applications inside the switch," SIGCOMM CCR, April 2014.
[2] S. Pontarelli, G. Bianchi, M. Bonola, A. Capone, C. Cascone, "Stateful OpenFlow: Hardware Proof of Concept", in HPSR 2015, Budapest, July 2015.
[3] C. Cascone, L. Pollini, D. Sanvito, A. Capone. "Traffic management applications for stateful SDN data plane", in EWSDN 2015, September 2015.
[4] A. Capone, C. Cascone, A. Q.T. Nguyen, and B. Sansò, "Detour Planning for Fast and Reliable Failure Recovery in SDN with OpenState", in DRCN 2015, March 2015.