EWSDN ’15 – Oct 2, 2015

Traffic Management Applications for Stateful SDN Data Plane

Carmelo Cascone^, Luca Pollini*, Davide Sanvito*, Antonio Capone^

ANTLab - Politecnico di Milano^
Dipartimento di Elettronica, Informazione
e Bioingegneria

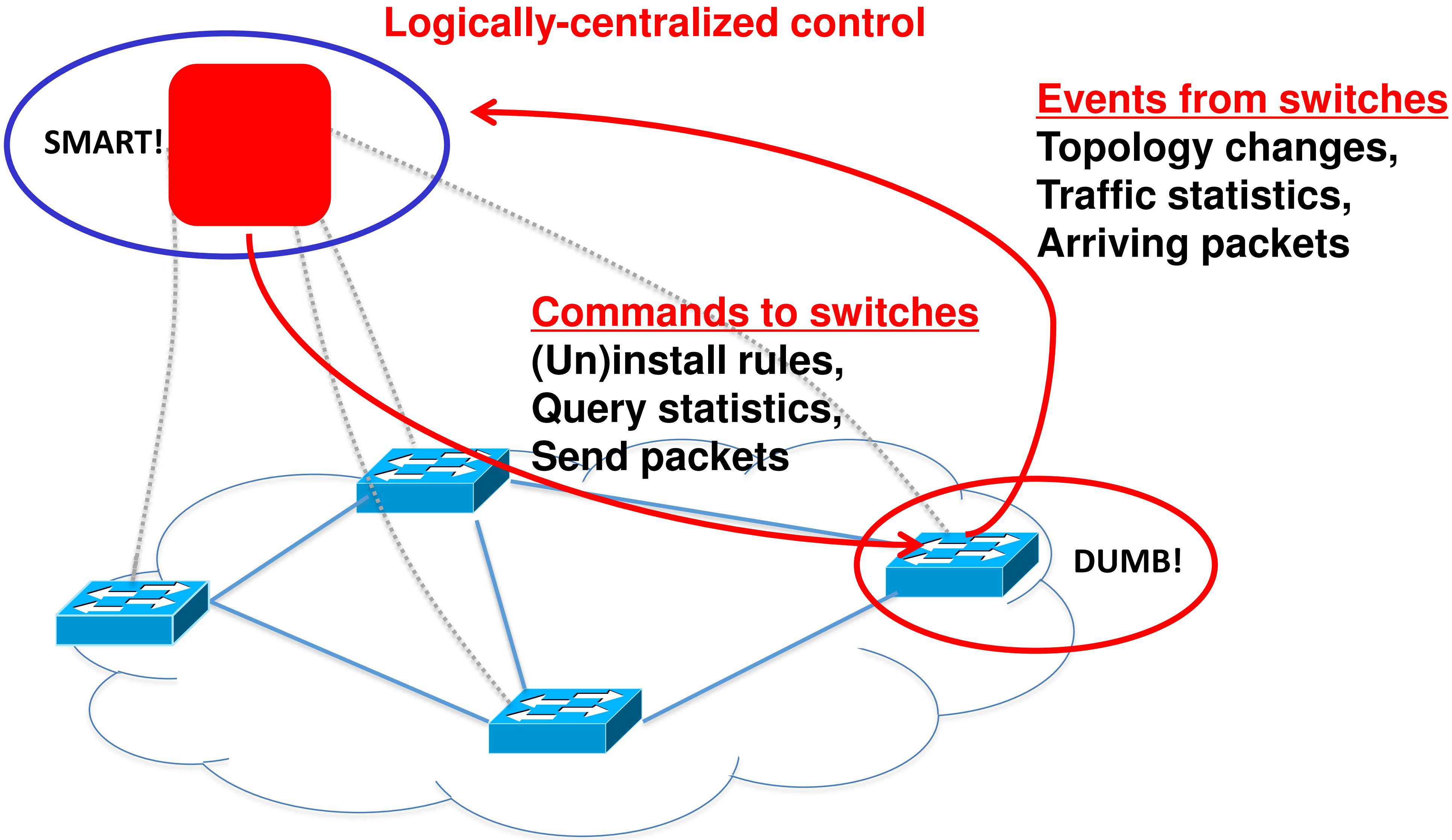
CNIT*
Consorzio Nazionale Interuniversitario
per le Telecomunicazioni

Supported by EU project:
Beba
BEhavioural BAsed forwarding

Goal

- **Highlight shortcomings of current SDN-OpenFlow paradigm**
- **Present a new “stateful” data plane model**
- **Motivate this need with 2 application examples**
  - Failure recovery
  - Forwarding consistency

OpenFlow recap



Logically-centralized control

SMART!

DUMB!

Events from switches

Topology changes,
Traffic statistics,
Arriving packets

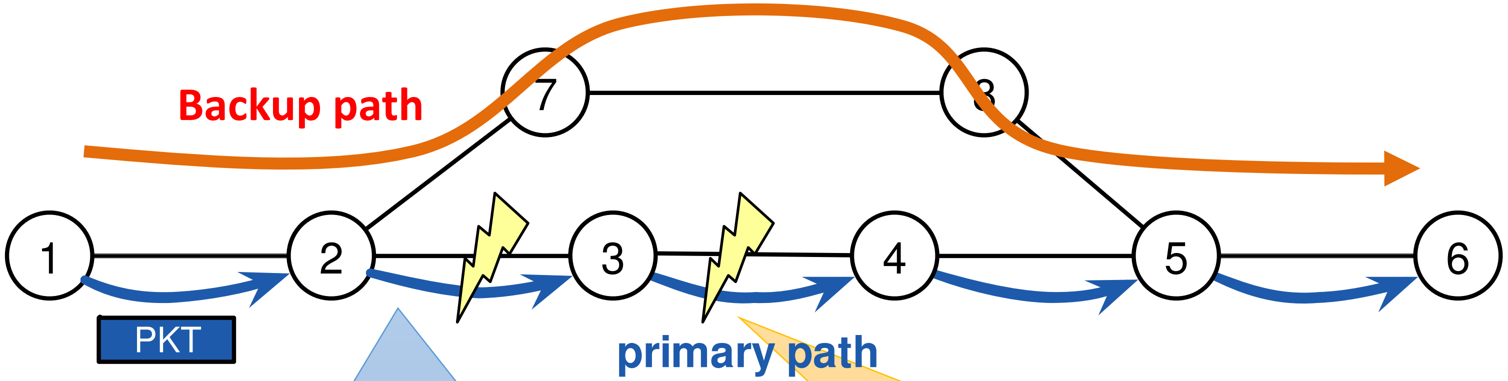
Commands to switches

(Un)install rules,
Query statistics,
Send packets

Centralized control: we know the pros but…

- **Control latency**
  - Switch-controller RTT
  - Controller processing
- **Signaling overhead**
  - First packet to the controller (Internet dominated by very short flows)
  - Flow statistics gathering

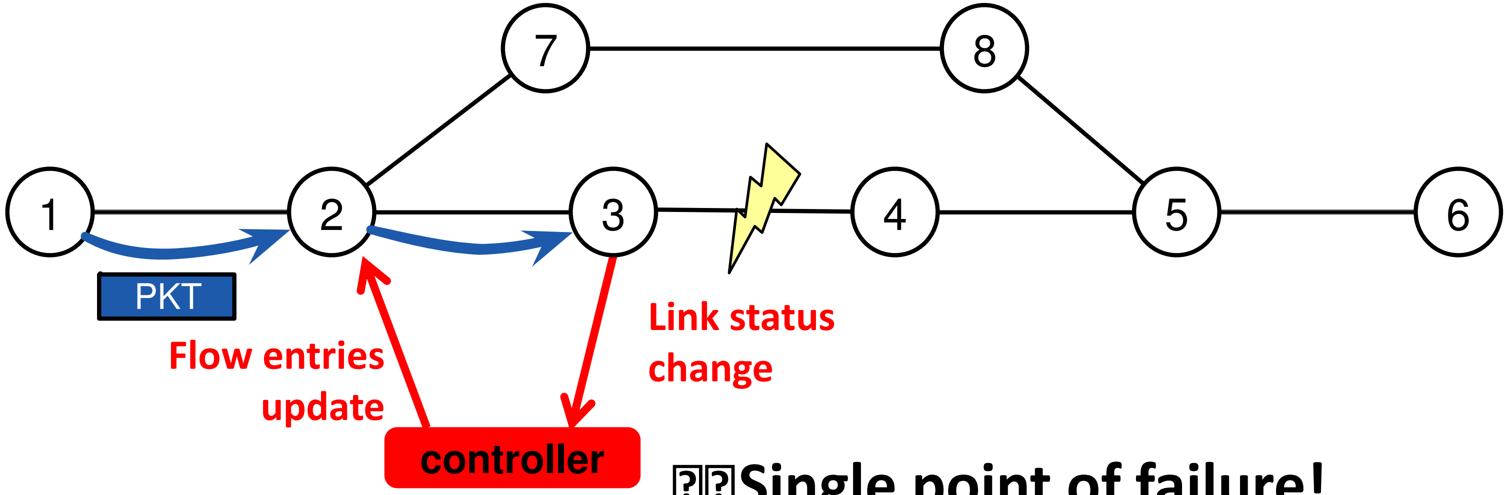
Example: failure recovery in OpenFlow (1)



“Fast-failover”:
Local reroute based on port status (OpenFlow 1.1+)

Weak! What if a local reroute in not available?

Example: failure recovery in OpenFlow (2)

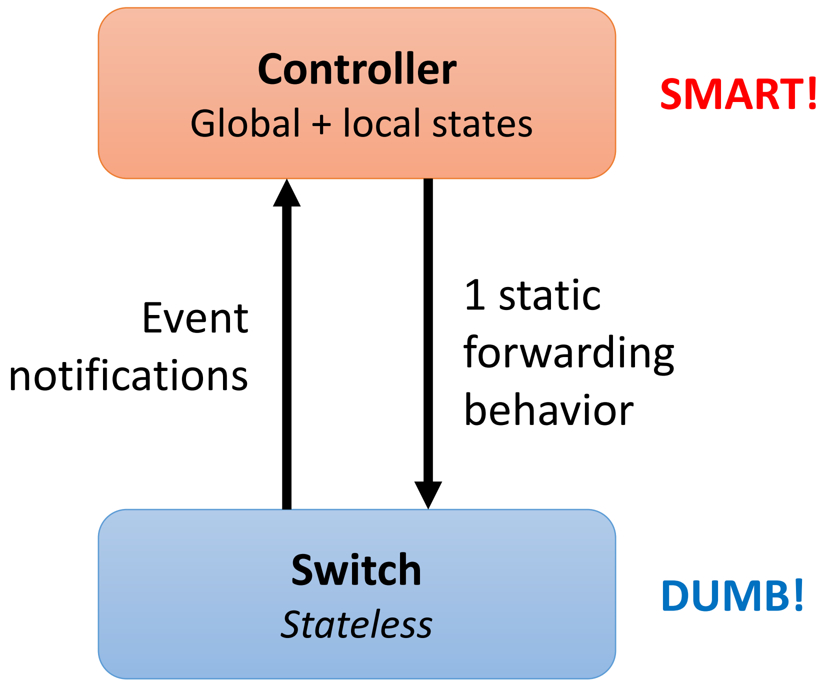


□□**Single point of failure!**

- **Can rely on controller intervention, but:**
  - Long recovery latency (> 50ms)
    - detection + signaling + flow table update
  - Failure of control channel
  - Signaling congestion (e.g. multiple failures, disasters)

Towards a new behavioral data plane model

Stateless model
(e.g. OpenFlow)



Controller
Global + local states

SMART!

Event notifications

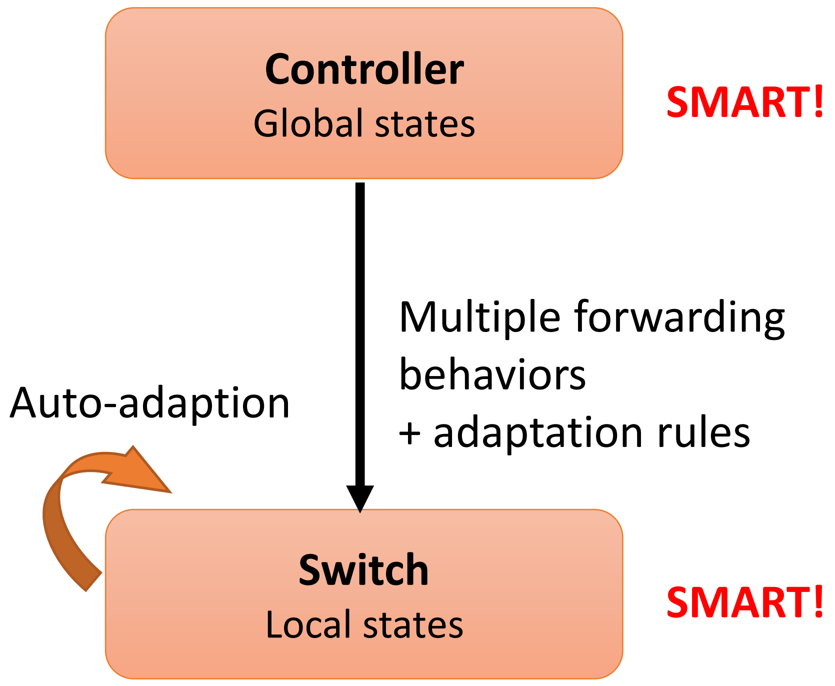
1 static forwarding behavior

Switch
Stateless

DUMB!

Control enforcing paradigm

Stateful (behavioral) model



Controller
Global states

SMART!

Multiple forwarding behaviors
+ adaptation rules

Auto-adaption

Switch
Local states

SMART!

Control delegation paradigm

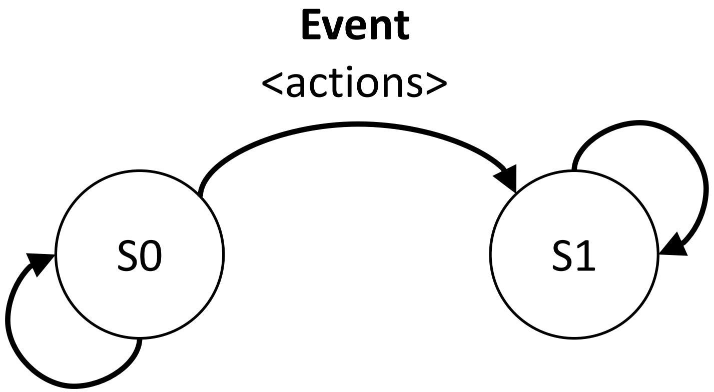
Easier said than done

- We need a switch abstraction and API which is…
  - **High performance**: control tasks executed at wire-speed (packet-based events)
  - **Platform-independent**: consistent with vendors’ needs for closed platforms
  - **Low cost and immediately viable**: based on commodity HW

- Apparently, far beyond OpenFlow switches…
- Our finding: much closer to OpenFlow than expected

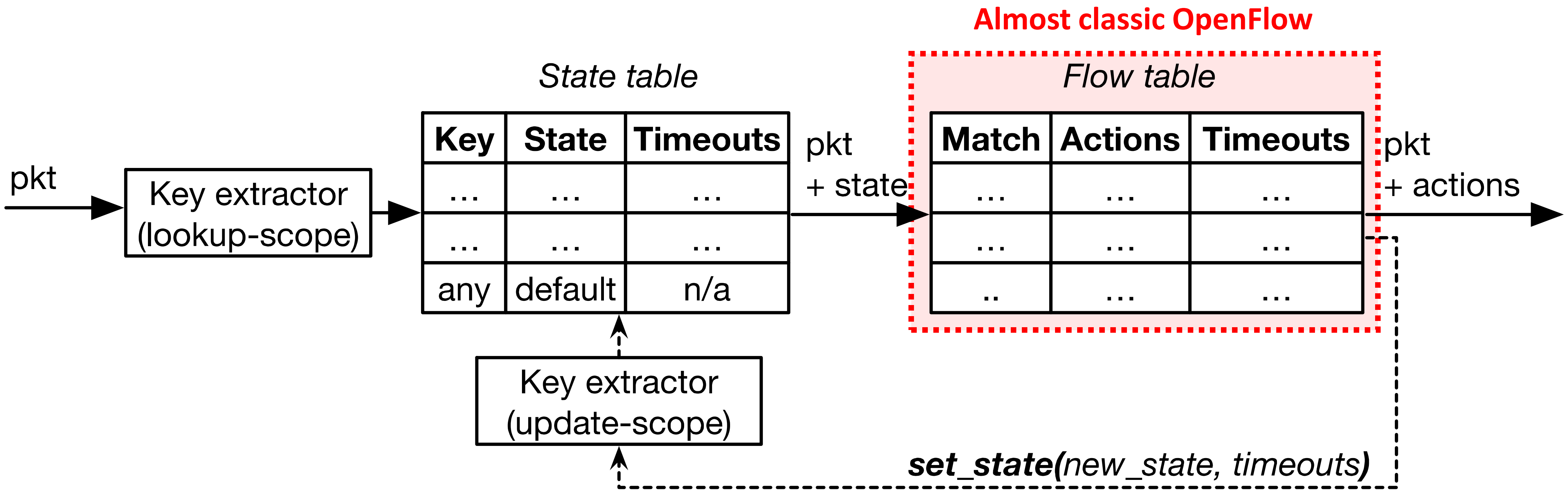
Our approach: OpenState

- **Idea: forward packets based on "flow states"**
  - Maintained by the switch
  - Autonomously updated as a consequence of local events (i.e. match, timers)
- **FSM-like forwarding model**
- **Minimal extension to OpenFlow**



- [CCR '14] G. Bianchi, M. Bonola, A. Capone, C. Cascone, **"OpenState: programming platform-independent stateful OpenFlow applications inside the switch"**, ACM SIGCOMM Comp. Comm. Rev., April 2014
- [HPSR '15] S. Pontarelli, M. Bonola, G. Bianchi, A. Capone, C. Cascone, **"Stateful OpenFlow: Hardware Proof of Concept",** IEEE High Performance Switching and Routing, July 2015

OpenState: 2 table approach


Almost classic OpenFlow
pkt
Key extractor (lookup-scope)
State table
pkt + state
Flow table
pkt + actions
Key extractor (update-scope)
set_state(new_state, timeouts)

State table

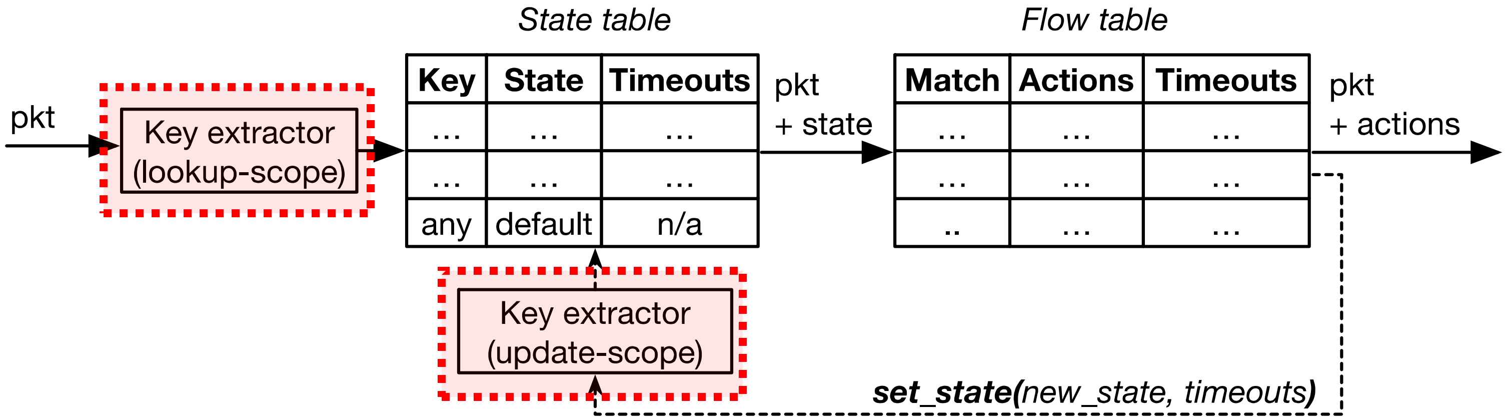
Key	State	Timeouts
…	…	…
…	…	…
any	default	n/a

Flow table

Match	Actions	Timeouts
…	…	…
…	…	…
..	…	…

Flow key extractors

- **Used to match/access the state table**
  - Lookup or update phase
- **Scope = ordered list of header fields**
  - E.g. {ip_src} → 32 bit flow key
  - E.g. {eth_src, eth_dst} → 96 bit flow key



State table

Key	State	Timeouts
…	…	…
…	…	…
any	default	n/a

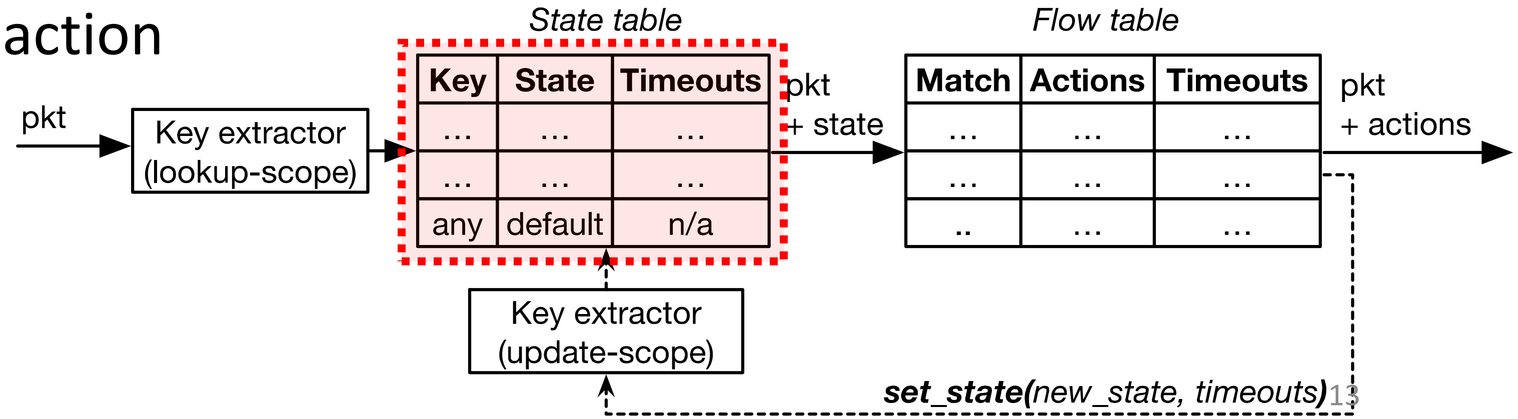
Flow table

Match	Actions	Timeouts
…	…	…
…	…	…
..	…	…

***set_state**(new_state, timeouts)*

State table

- **Exact match on flow key**
  - Efficient implementation in RAM (vs. TCAM)
- **DEFAULT state if table miss**
- **Optional timeouts**
  - Idle or hard: equivalent to OpenFlow
  - <= 1ms granularity
  - Rollback state when timeout expires
  - Configured by set_state() action



State table

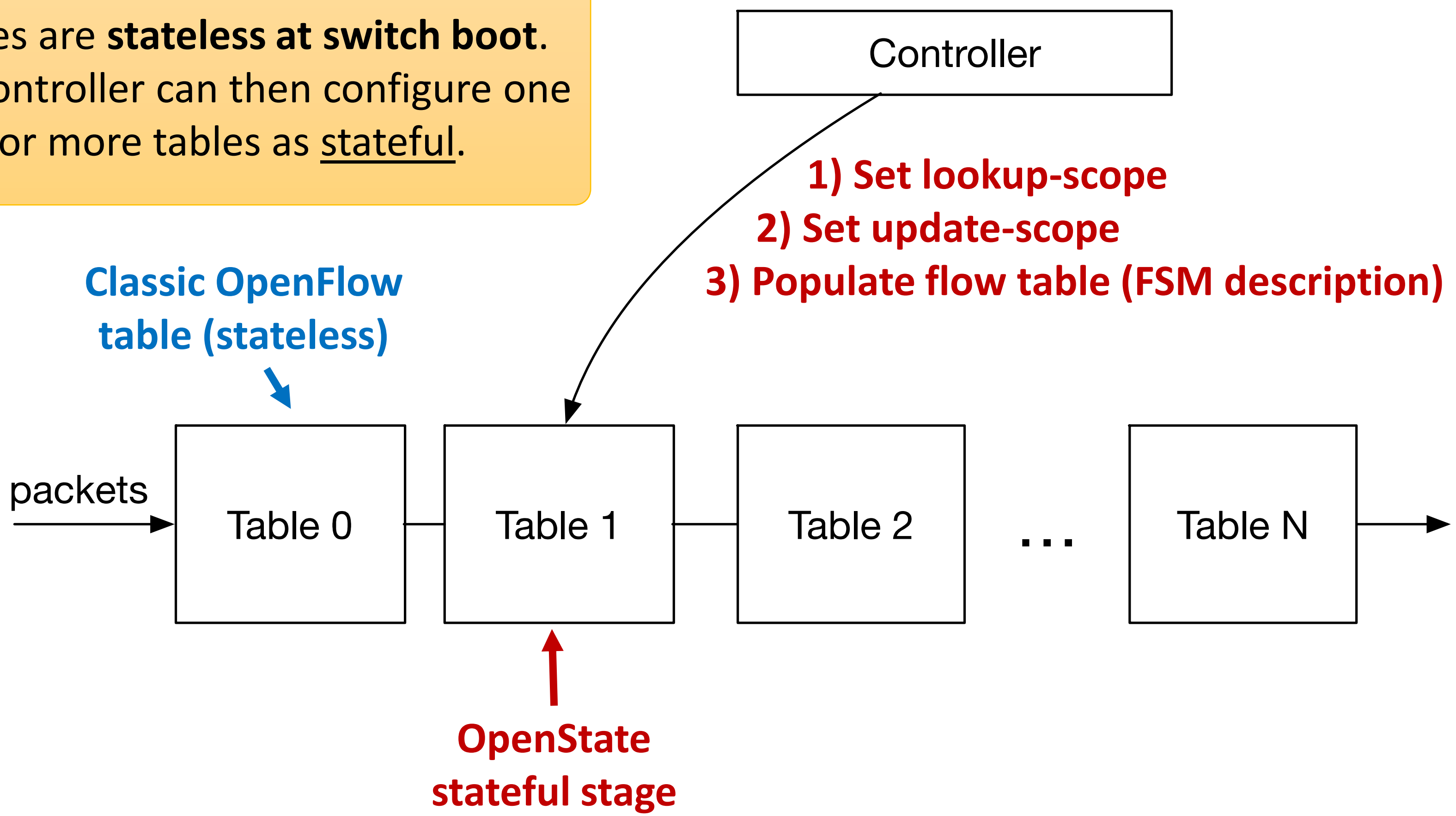
Key	State	Timeouts
…	…	…
…	…	…
any	default	n/a

Flow table

Match	Actions	Timeouts
…	…	…
…	…	…
..	…	…

Pipeline configuration

Tables are **stateless at switch boot**. The controller can then configure one or more tables as stateful.



Classic OpenFlow table (stateless)

1) Set lookup-scope
2) Set update-scope
3) Populate flow table (FSM description)

OpenState stateful stage

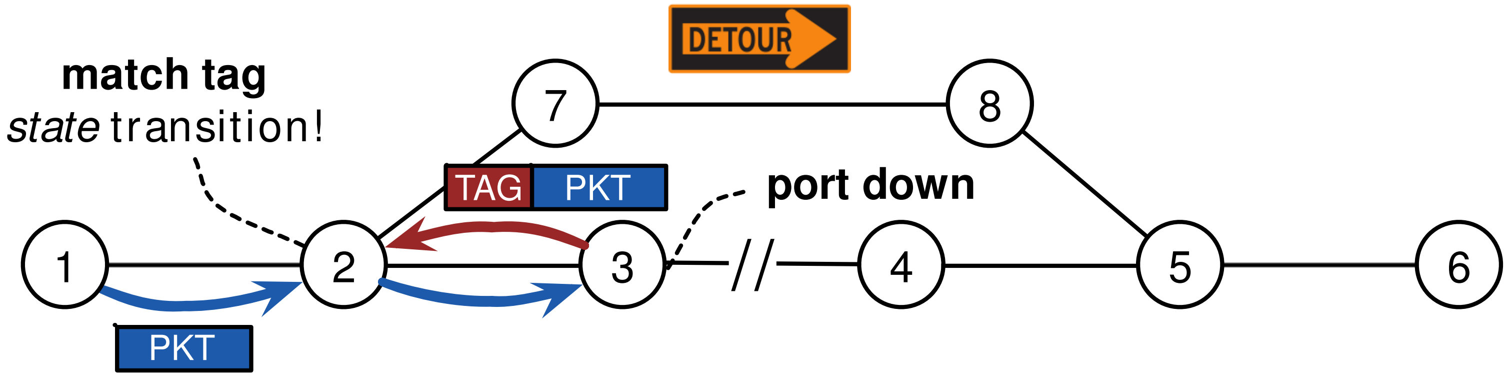
Open source: http://www.openstate-sdn.org

- **Running code: softswitch + controller**
  - Based on CPqD ofsoftswitch13, RYU
  - Initial support to Open vSwitch based on “learn()” action
- **Protocol specification**
  - OpenFlow 1.3 Experimenter Extension (PDF available)
- **Mininet-based application examples**
  - MAC learning, port knocking firewall, failure Recovery, DDoS detection and mitigation, load balancing
- **Download & try!**

Failure recovery

Failure recovery with OpenState

- Tags (e.g. MPLS labels) used to distinguish between **different forwarding behaviors**
- Upon failure, packets are **“bounced back”** with special tag
  - – until matched against a node able to respond to that specific failure
- **Periodic probe** to re-establish forwarding on the primary path

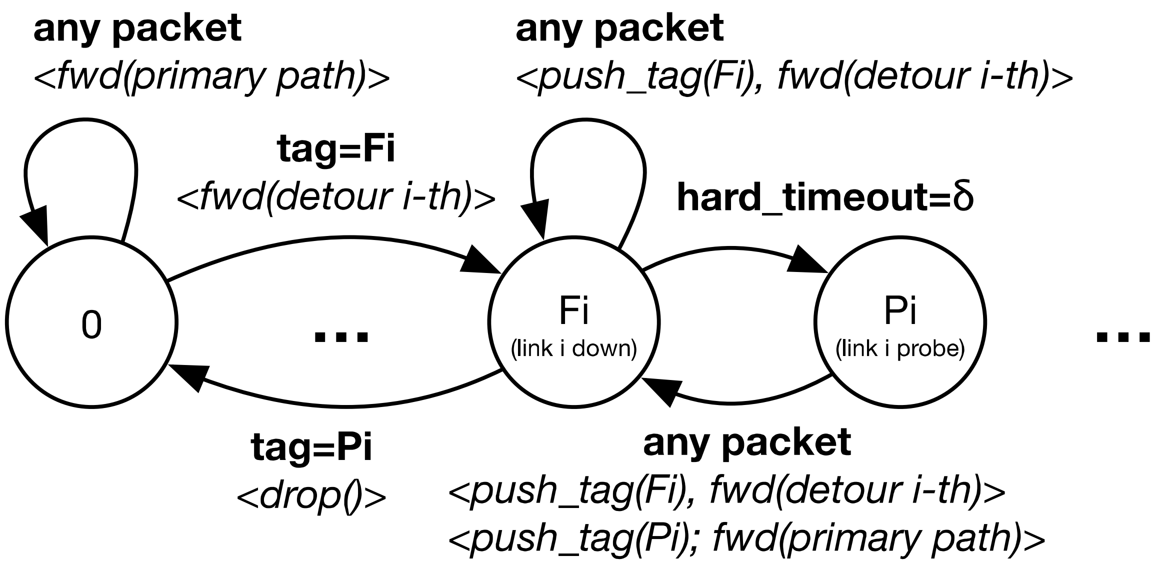


➔ **No extra signaling/packet loss after failure detection**

➔ **Controller not involved (besides initial provisioning)**

Behavioral model (FSM)

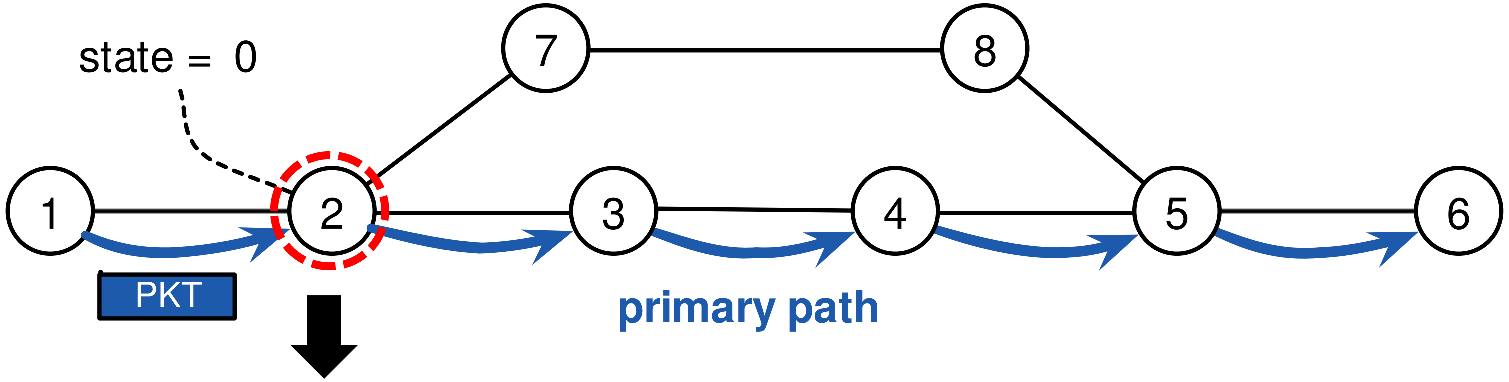
- Each flow (lookup-scope) has an associated state (tag)
  - **0 (default)** → all good, forward on primary path
  - **Fi** node *i* unreachable → forward on detour *i-th*
  - **Pi** node *i* must be probed → send 1 probe to node *i*



Failure recovery

Example

Normal conditions (no failures)



lookup-scope=[eth_src, eth_dst]
update-scope=[eth_src, eth_dst]

L2 flows

State table

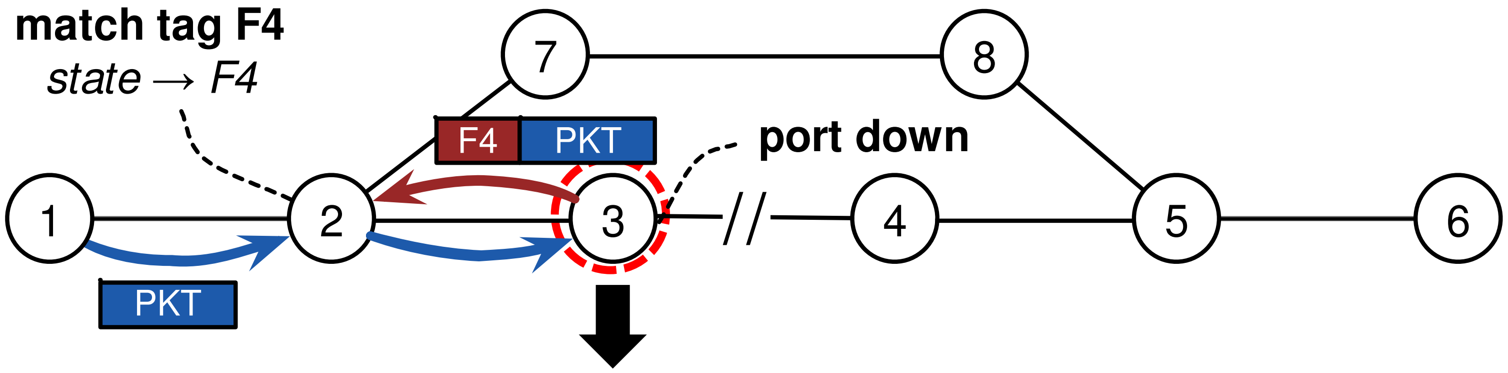
Key	State
…	…
…	…
* (any)	0

Flow table

Match	Instructions
src=1, dst=6, state=0	fwd(3)
…	…
…	…

Example

Packets "bounced back" in case of failure



Match	Instructions
src=1, dst=6	Group(1)
…	…
…	…

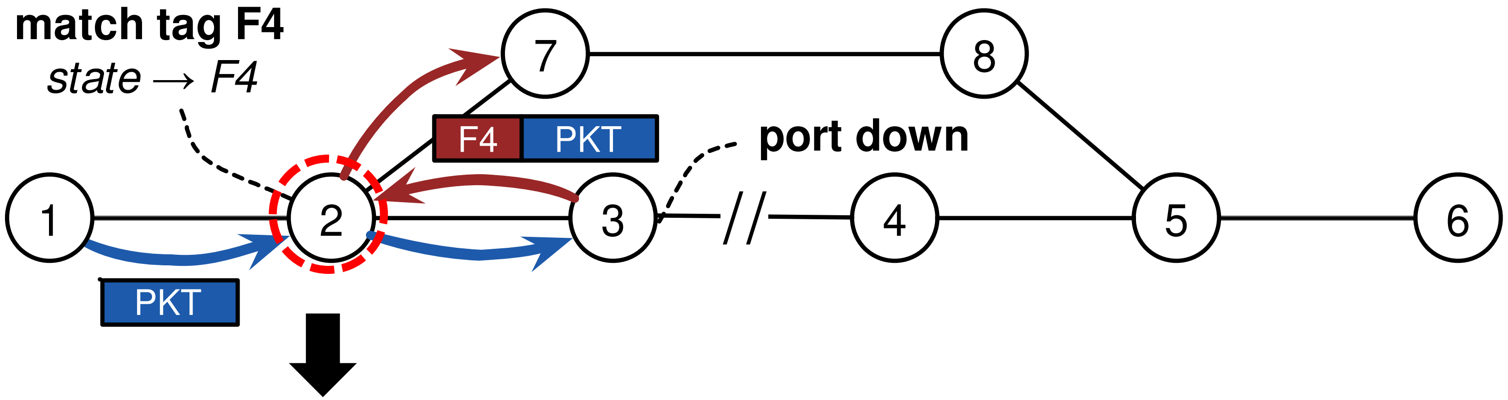
Group table

ID	Type	Action buckets
1	FAST-FAILOVER	<output(2)>, <push_tag(F4), output(1)>,
…	…	…

Failure recovery

Example

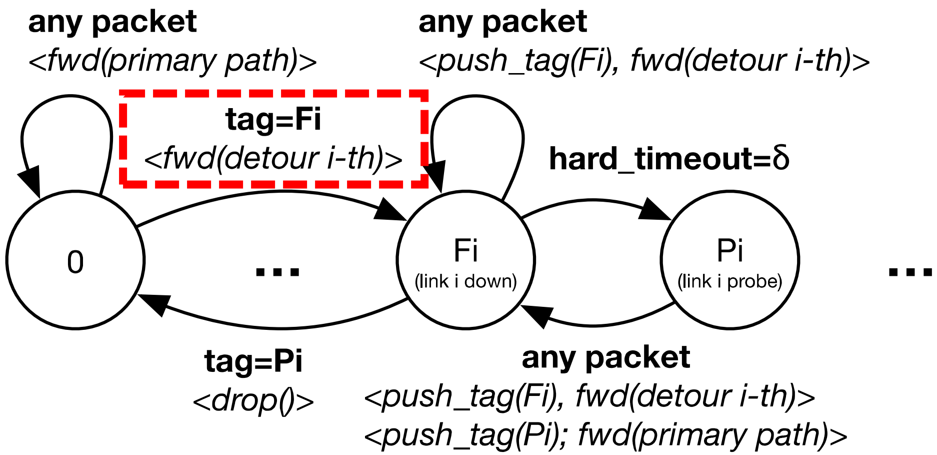
State transition at a pre-determined reroute node



Match	Instructions
…	…
src=1, dst=6, state=0	fwd(3)

src=1, dst=6, tag=F4	set_state(F4, hard_to=10s,
	hard_rollback=P4)
	fwd(7)

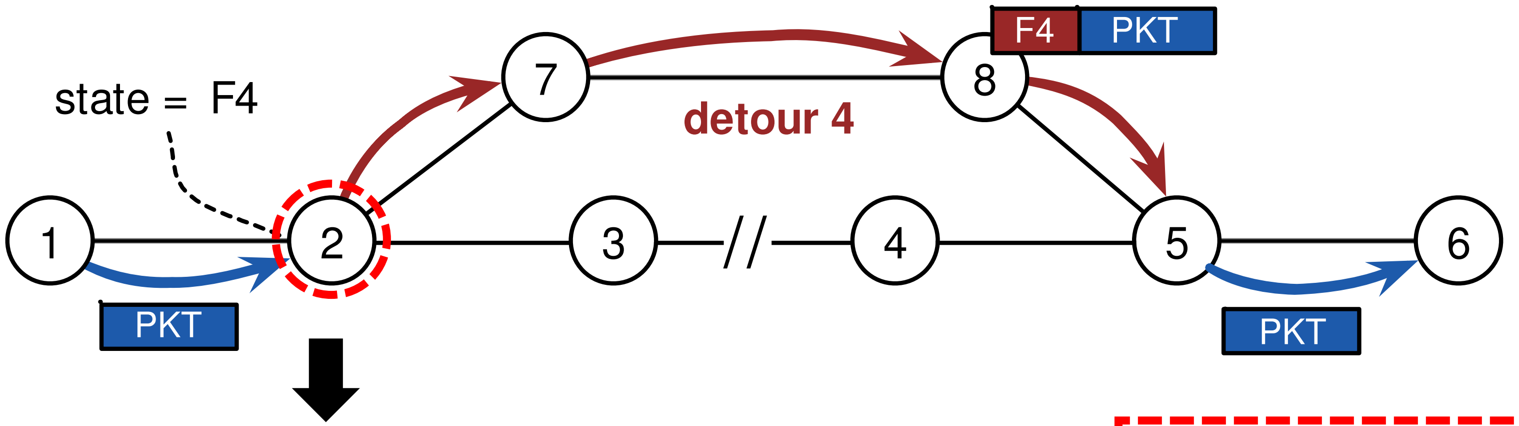
…	…



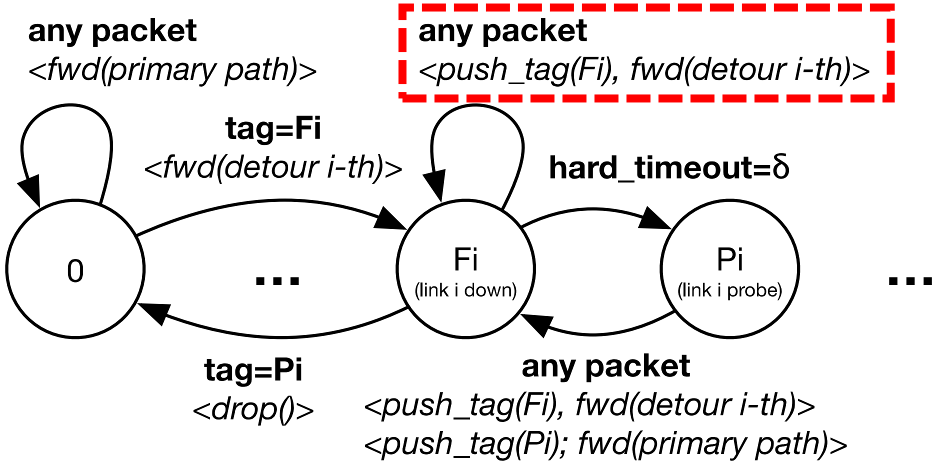
Failure recovery

Example

Detour path enabled

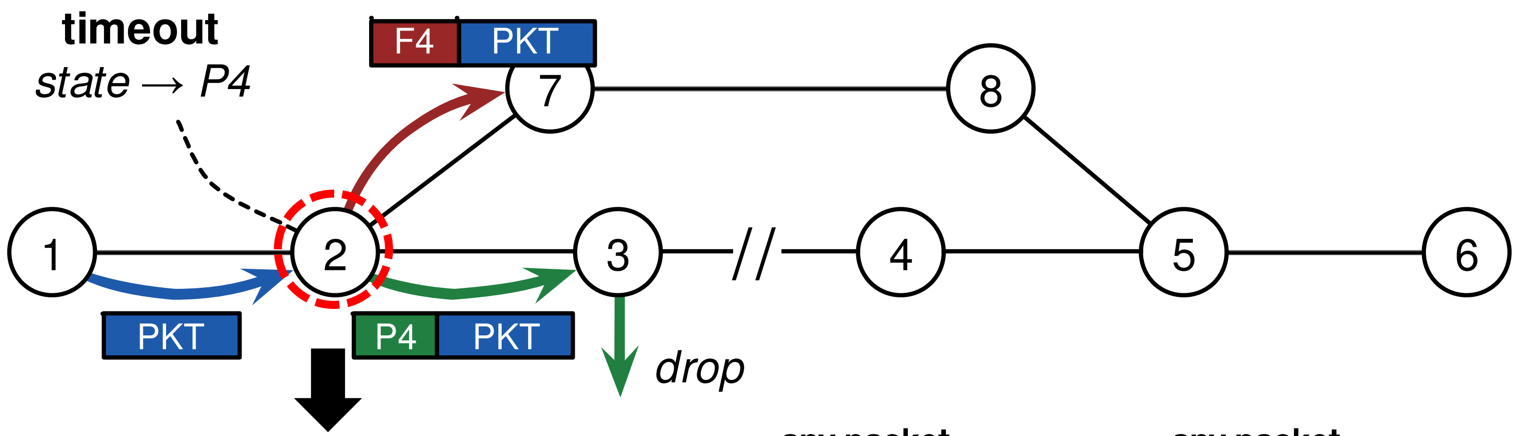


Match	Instructions
…	…
src=1, dst=6, state=F4	push_tag(F4), fwd(7)
…	…
…	…

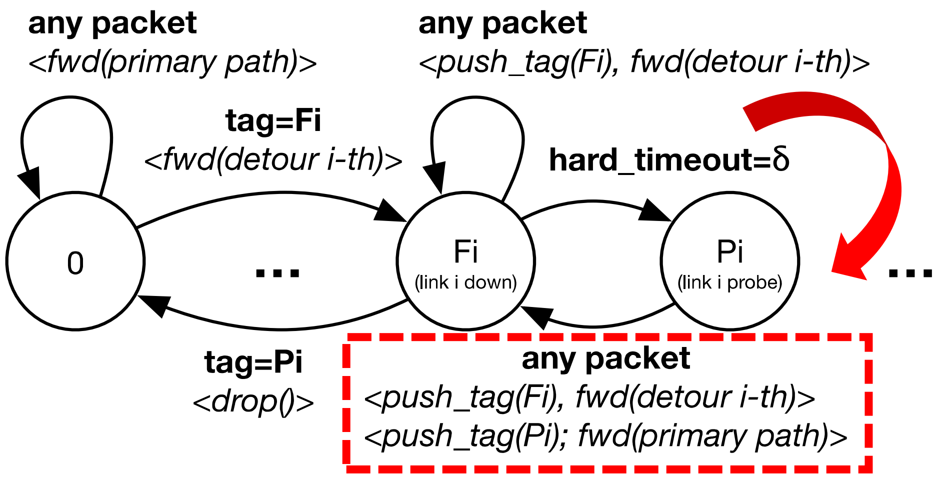


Example

State hard timeout to generate probe packets



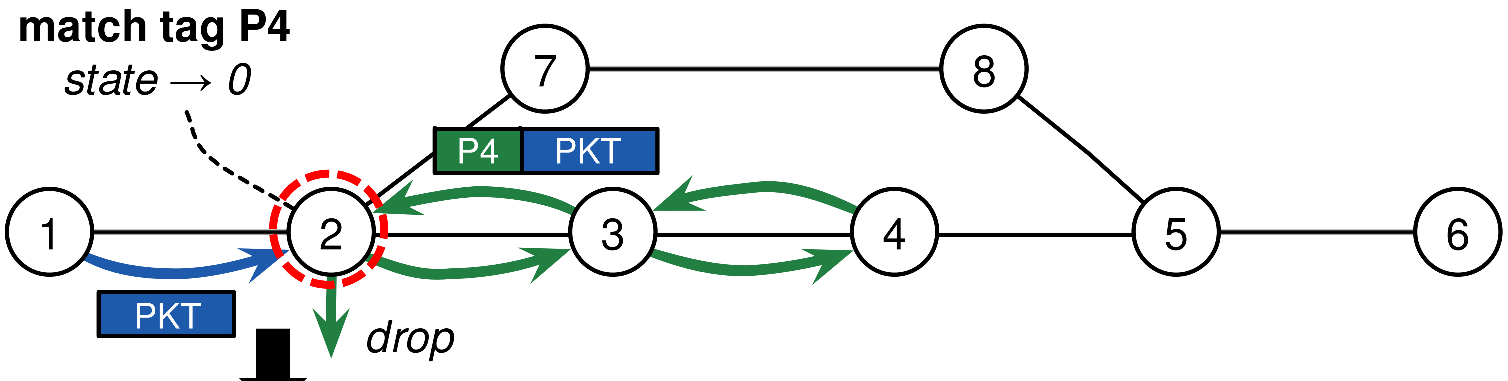
Match	
…	…
…	…
…	…
src=1, dst=6, state=P4	set_state(F4, hard_to=10s, hard_rollback=P4), <push_tag(F4), fwd(7)> <push_tag(P4), fwd(3)>



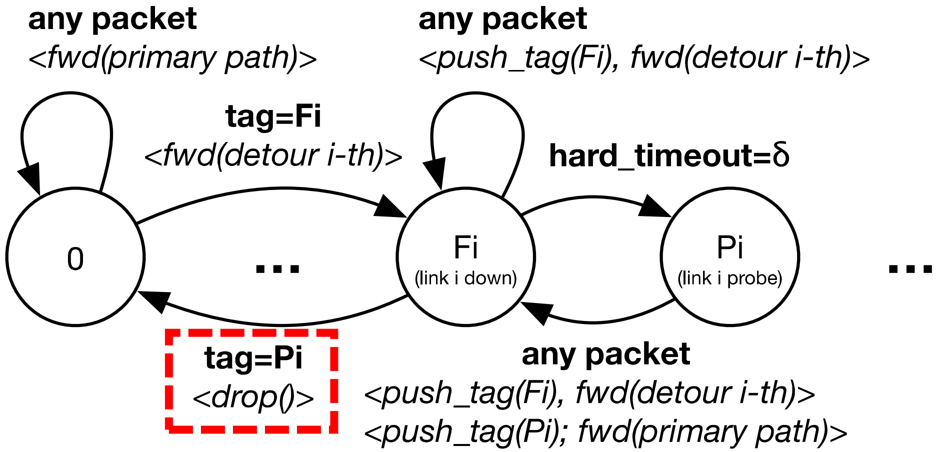
Failure recovery

Example

Primary path re-established



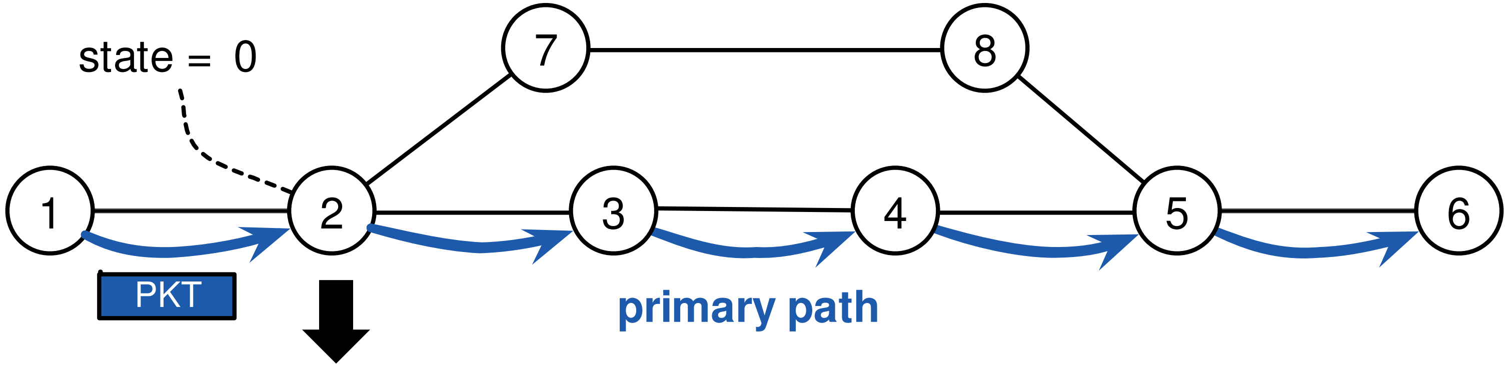
Match	
…	…
…	…
…	…
…	…
tag=P4	set_state(0), drop()



Failure recovery

Example

Failure solved



Match	Instructions
src=1, dst=6, state=0	fwd(3)
…	…
…	…

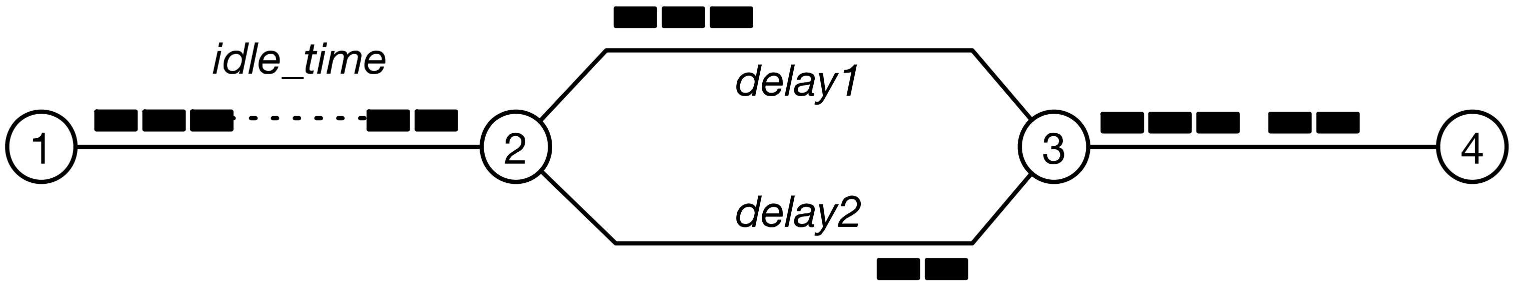
Load balancing

Load balancing in OpenFlow

- **OpenFlow SELECT group entry**
  - Packets forwarded using only one of multiple defined action buckets
  - Implementation left out to vendors (e.g. round robin, hash-based, etc)
- **Usually implemented with ECMP-like hash-based schemes**
  - Can’t decide on which header fields
  - Two or more elephant flows can collide on their hash, using the same path, hence creating a bottleneck
  - **Current OF solutions**:
    - reactive allocation (first packet to controller)
    - detection and relocation based on periodic flow statistic gathering

Better idea: flowlet-based load balancing

- **Originally introduced in FLARE (2007)***
  - Based on the idea of **switching bursts of packets** (flowlets) instead of pinning the whole flow to one path
  - **No packet reordering** if the idle time between bursts is larger than the maximum delay difference between parallels paths
  - No need to worry about elephant flows (burden shared among all paths)

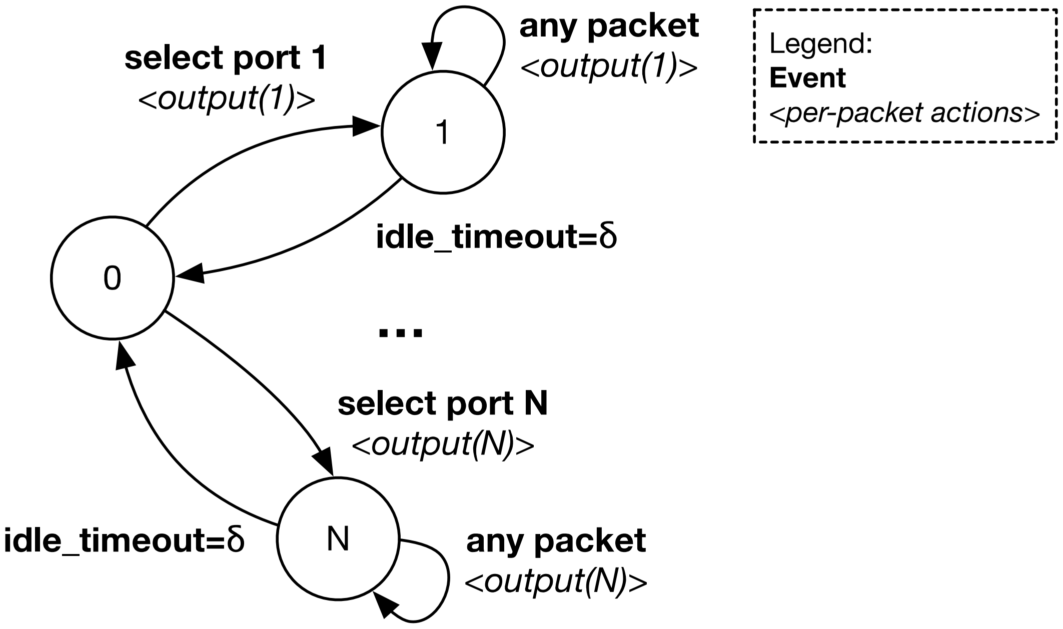


No packet reordering if *idle_time* > |*delay*1 − *delay*2|

* S. Kandula, D. Katabi, S Sinha, and A. Berger, “FLARE: Dynamic load balancing without packet reordering”. *ACM SIGCOMM Computer Communication Review*, *2007*.

OpenState-based implementation

- States used to **distinguish between consecutive bursts/instances** of the same flow
- State idle timeouts to define the **lifetime of a forwarding decision**
  - sub-RTT scales for flowlet switching



lookup_scope=[ip_src, ip_dst, tcp_src, tcp_dst]
update_scope=[ip_src, ip_dst, tcp_src, tcp_dst]

State table

Key	State	Timeouts
A,B,x,y	1	idle_to=δ
…	…	…
*	0	n/a

Flow table

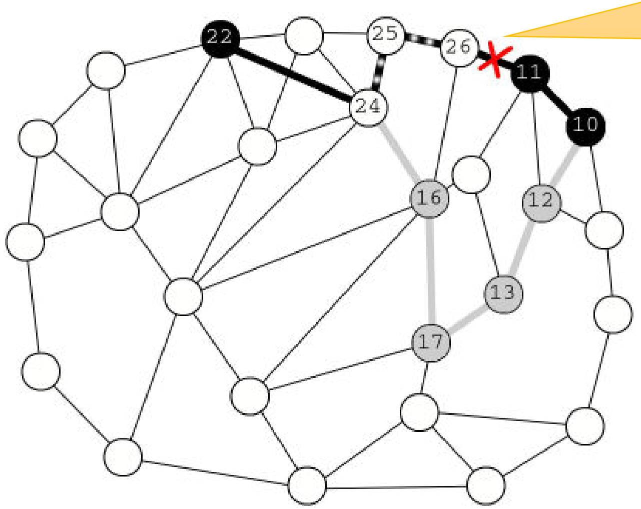
Match	Instructions
ip_dst=A, state=0	group(1)
ip_dst=B, state=0	group(2)
state=1	output(1)
state=2	output(2)
…	….
state=N	output(N)

Group table

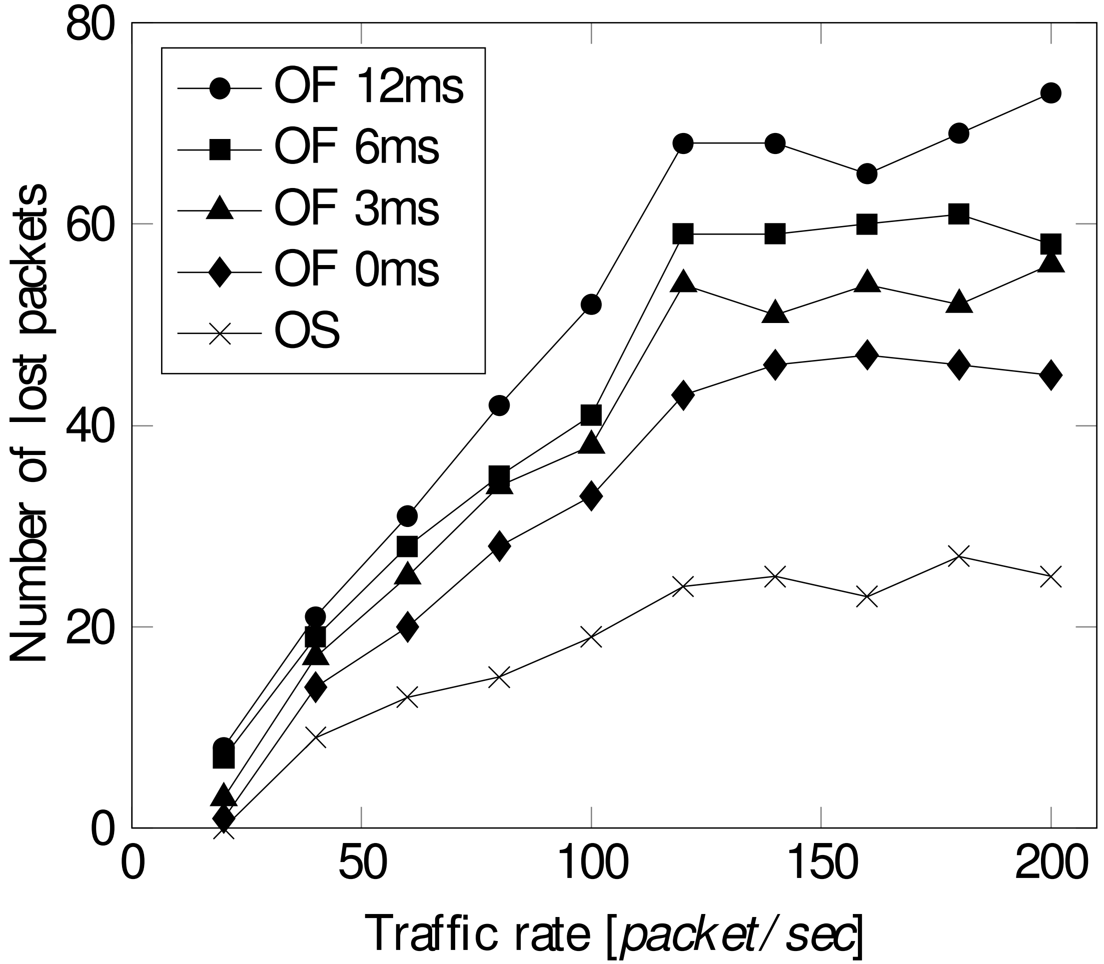
Group ID	Type	Action buckets
1	SELECT	<set_state(1, idle_to=δ), output(1)>, <set_state(2, idle_to=δ), output(2)>, …
2	SELECT	…

Example results: failure recovery

- **OF:** OpenFlow-based reactive approach, controller establishes backup path (with different switch-controller RTTs)
- **OS:** OpenState-based approach, packets bounced back upon failure



9 demands affected by link failure

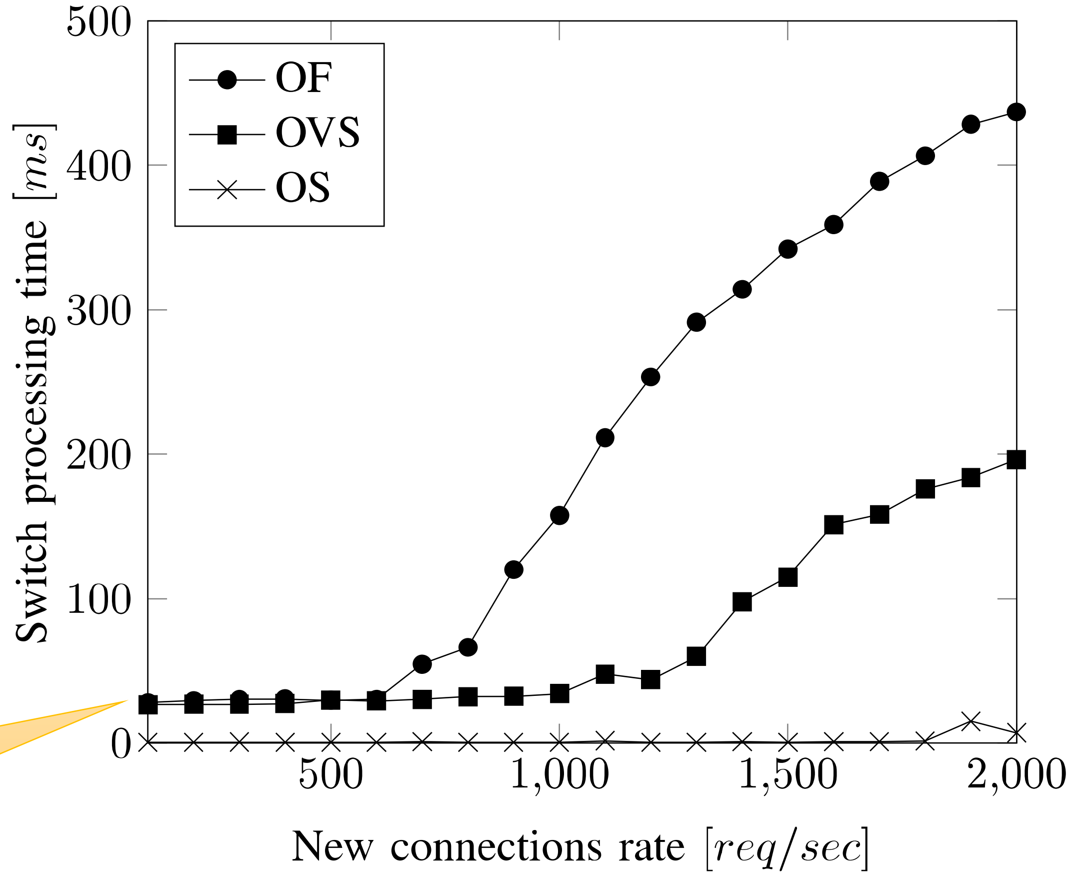


Optimal routing that minimizes bounce path based on:

A. Capone, C. Cascone, A. Q. Nguyen, and B. Sansò. "Detour planning for fast and reliable failure recovery in SDN with OpenState".
In IEEE *Design of Reliable Communication Networks (DRCN)*, March 2015

Example results: load balancing

- **<u>OF:</u>** controller-based reactive approach, new connections allocated by controller
- **<u>OVS:</u>** same as OF, but with faster switch (Open vSwitch)
- **<u>OS:</u>** OpenState-based approach



12ms
switch-controller RTT

Conclusions

- **New stateful data plane model → OpenState**
  - Control «decided» at controller, «execution» delegated to switches’ data plane)
- **Running code available at: http://www.openstate-sdn.org**
  - Openflow 1.3 extension
- **Failure recovery**
  - Switches pre-loaded with backup routing
  - MPLS labels use to perform failure signaling/path probing
  - Almost 0 packets lost after failure detection
- **Load balancing**
  - Can implement flowlet-based scheme
  - No need for elephant flows handling
  - Controller initially configure group table with optimal state idle timeouts

Beba

BEhavioural BAsed forwarding

http://www.beba-project.eu

- **Started January 2015**
- **Technical plans:**
  - Propose OpenState for standardization
  - SW switch acceleration + HW prototype
  - Advanced security, forwarding and monitoring applications
  - Data plane verification
  - Real field large scale experimentation

cnit

THALES

6WIND

NEC

KTH VETENSKAP OCH KONST

CESNET

Horizon 2020
European Union Funding
for Research & Innovation

Thanks!

carmelo.cascone@polimi.it