Monitoring framework for Software-Defined Networks

Candidate: Hanieh Rajabi

Tutor: Prof. Giuseppe Bianchi

University of Rome Tor Vergata

Università di Roma Tor Vergata

Consortium GARR 5°

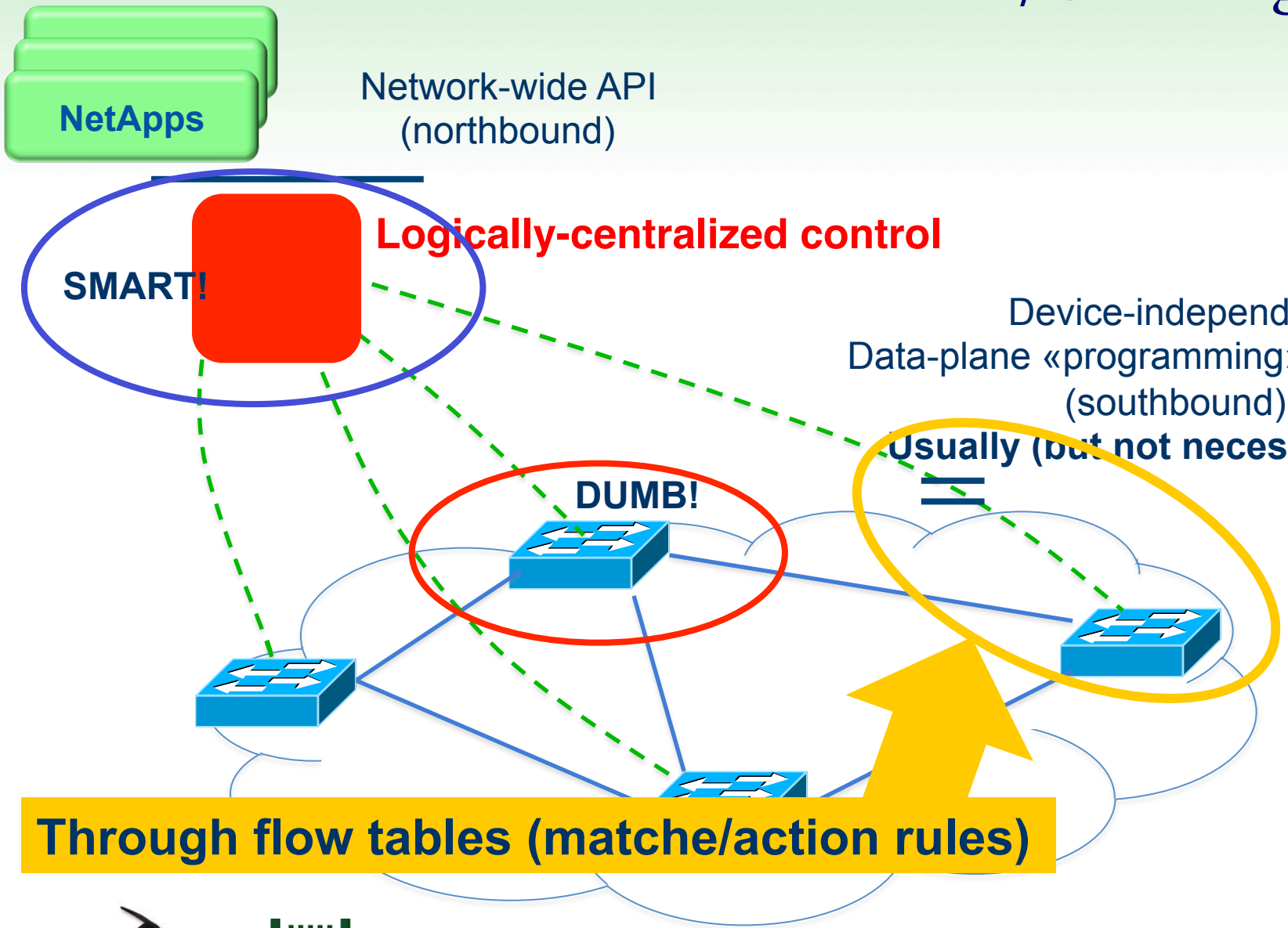
GARR

5° Borsisti Day – 13/05/2014

SDN: General overview

- Software Defined Networking paradigms
  - Smart centralized controller govern forwarding behavior.
  - device-level programming abstractions (via OpenFlow API)
  - fast and simplified deployment of comprehensive and large-scale network services
- OpenFlow: abstract model of programmable flow table
  - Match/action
  - statistics

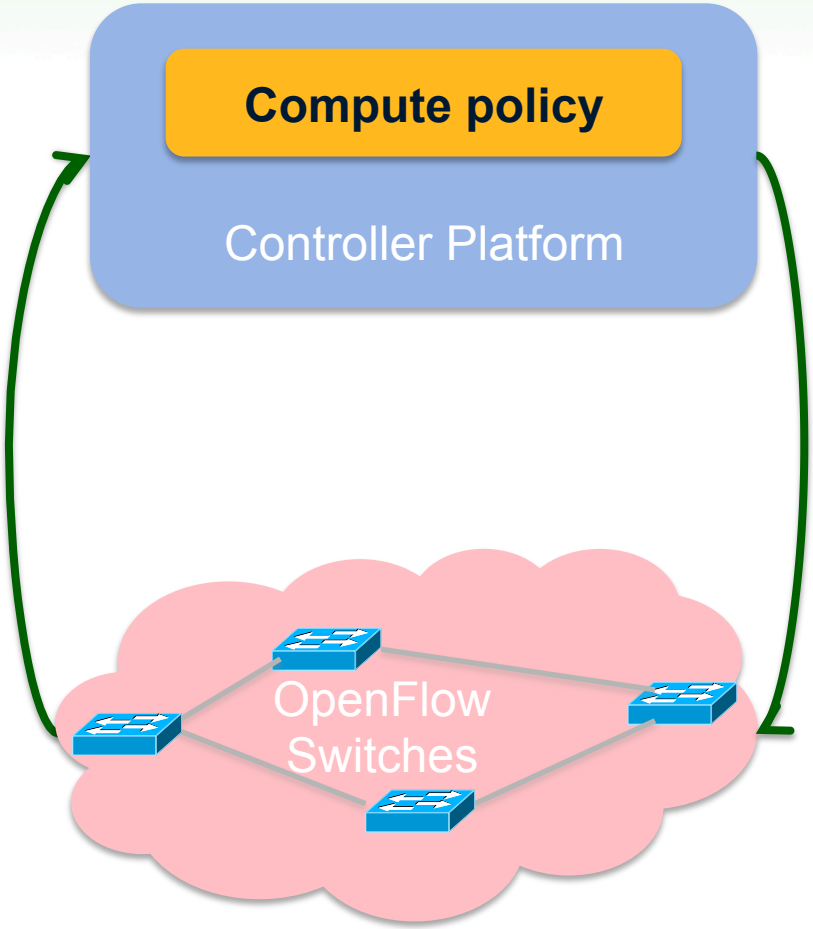
SDN/OF at a glance



fundamental 3-steps in SDN monitoring

NEED for Flow-level states visibility

Read/Monitor states & events



NEED for Network-wide visibility and control

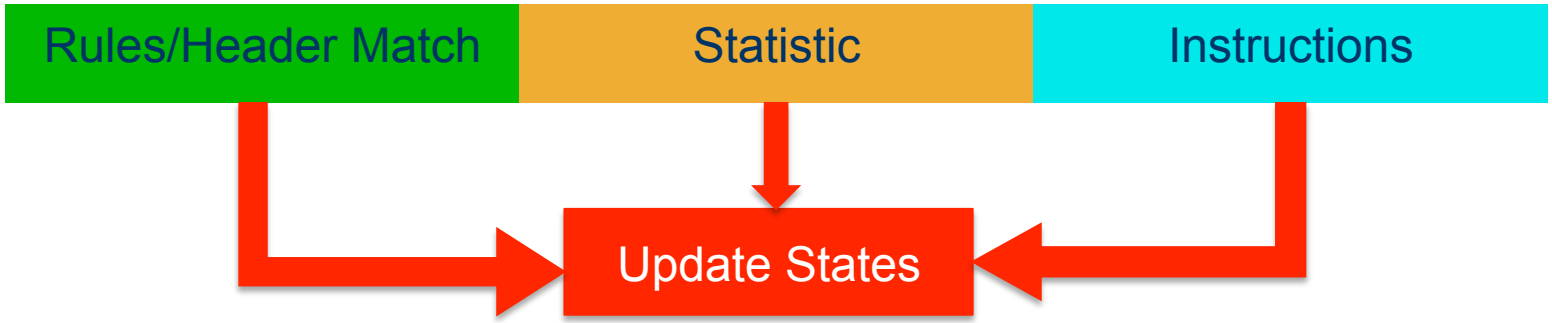
Write policy

Monitoring tasks challenges

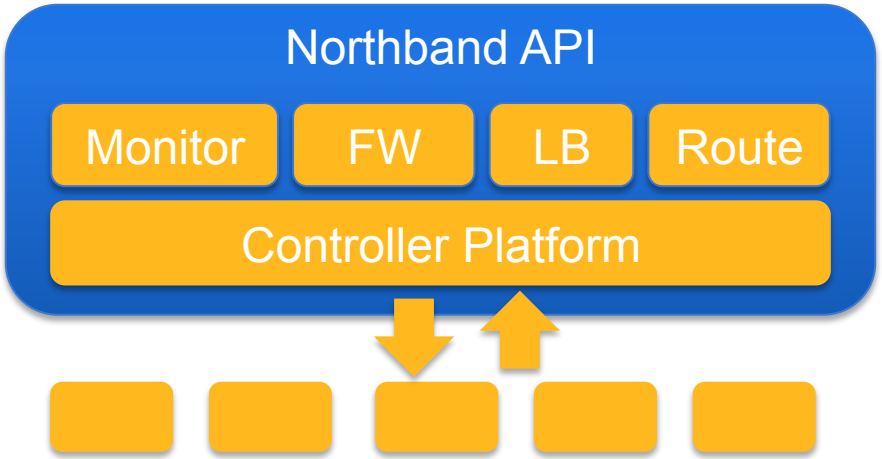
- Complex: increase volume and heterogeneity of traffic
- OF is a low level of abstraction
  - Event handling: which event for which flow
  - Sophisticated event???
  - Controller visibility
    - sees events that the switches do not know how to handle
- Limited number of forwarding rules

Monitoring tasks challenges

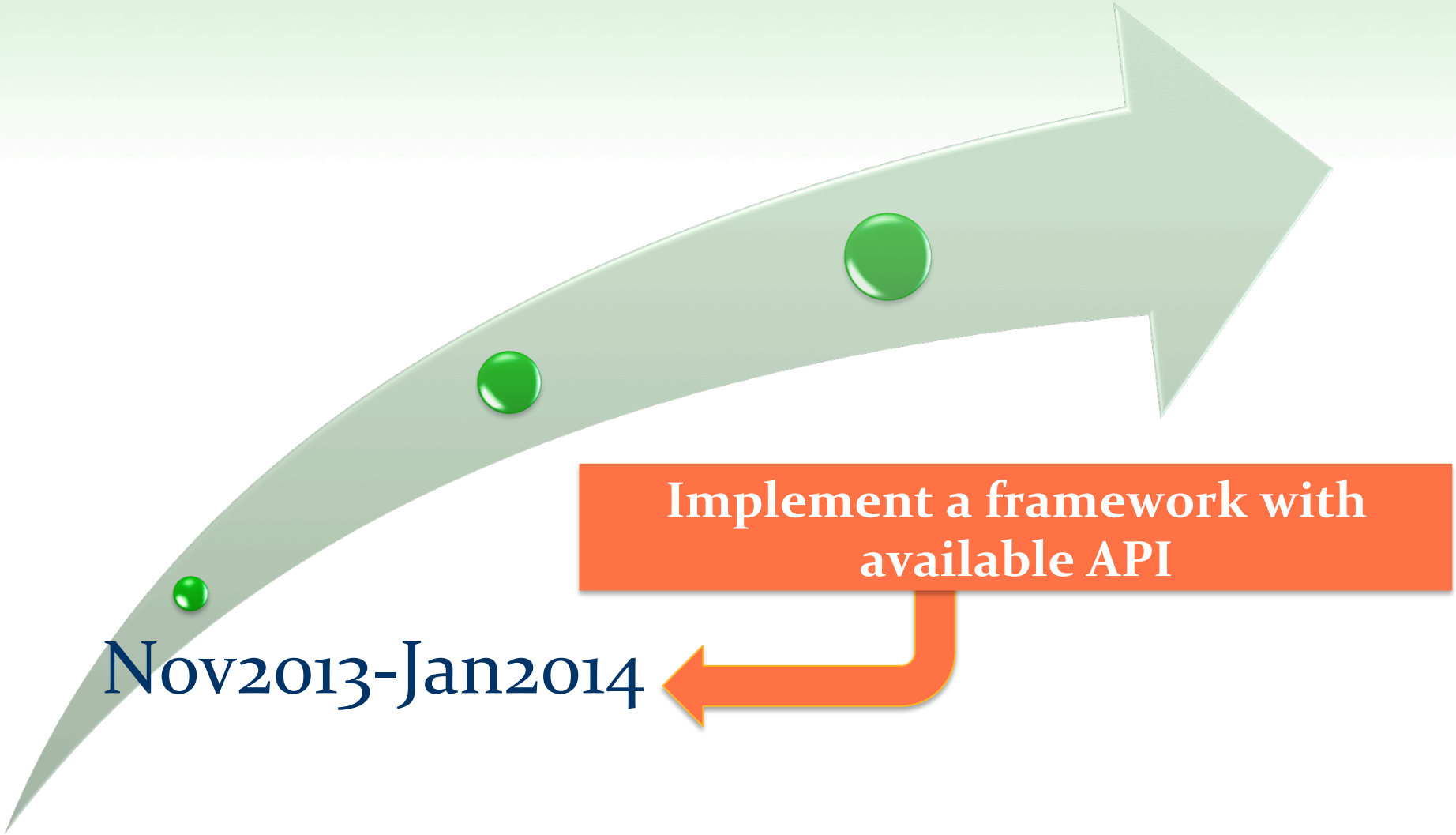
- Feedback loop from actions and statistics
  - state handling: to track attack evolution



- Policy composition
  - Modularize controller

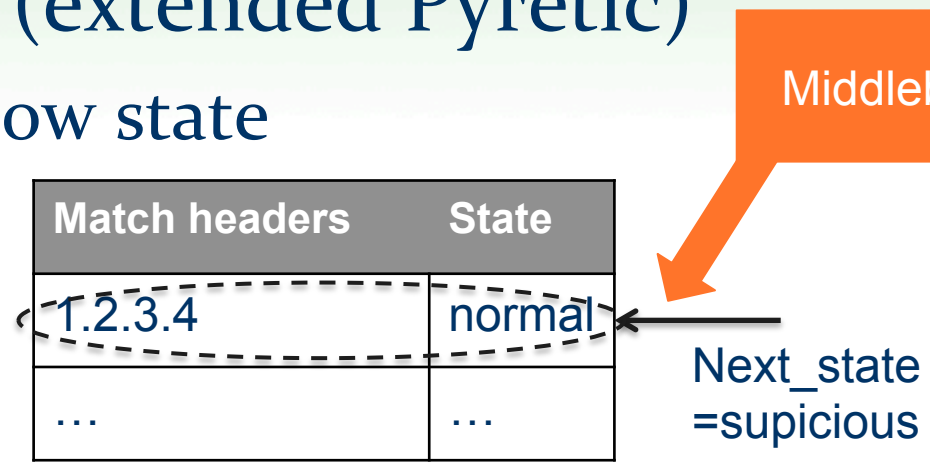


Time schedule



Existing controller: Resonance

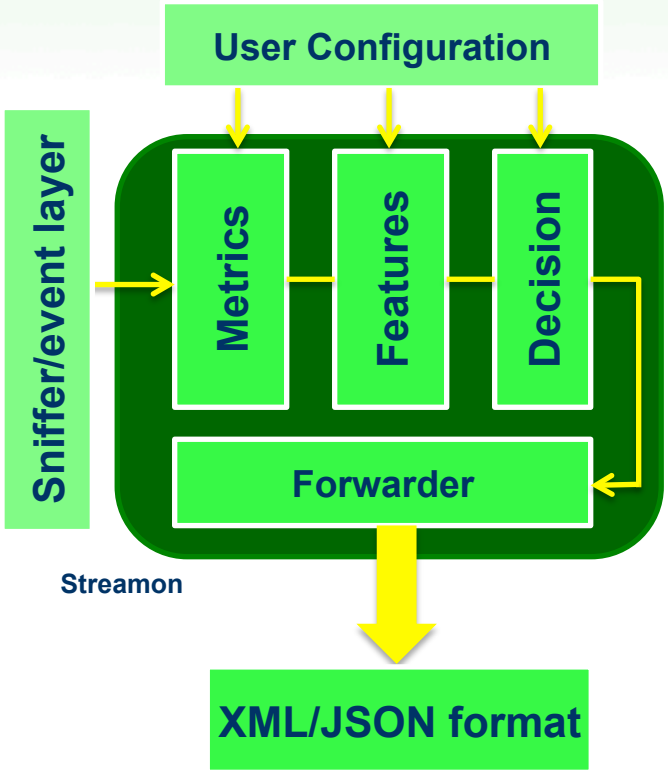
- Resonance controller (extended Pyretic)
  - keeps tracks of each flow state
    - A python dictionary



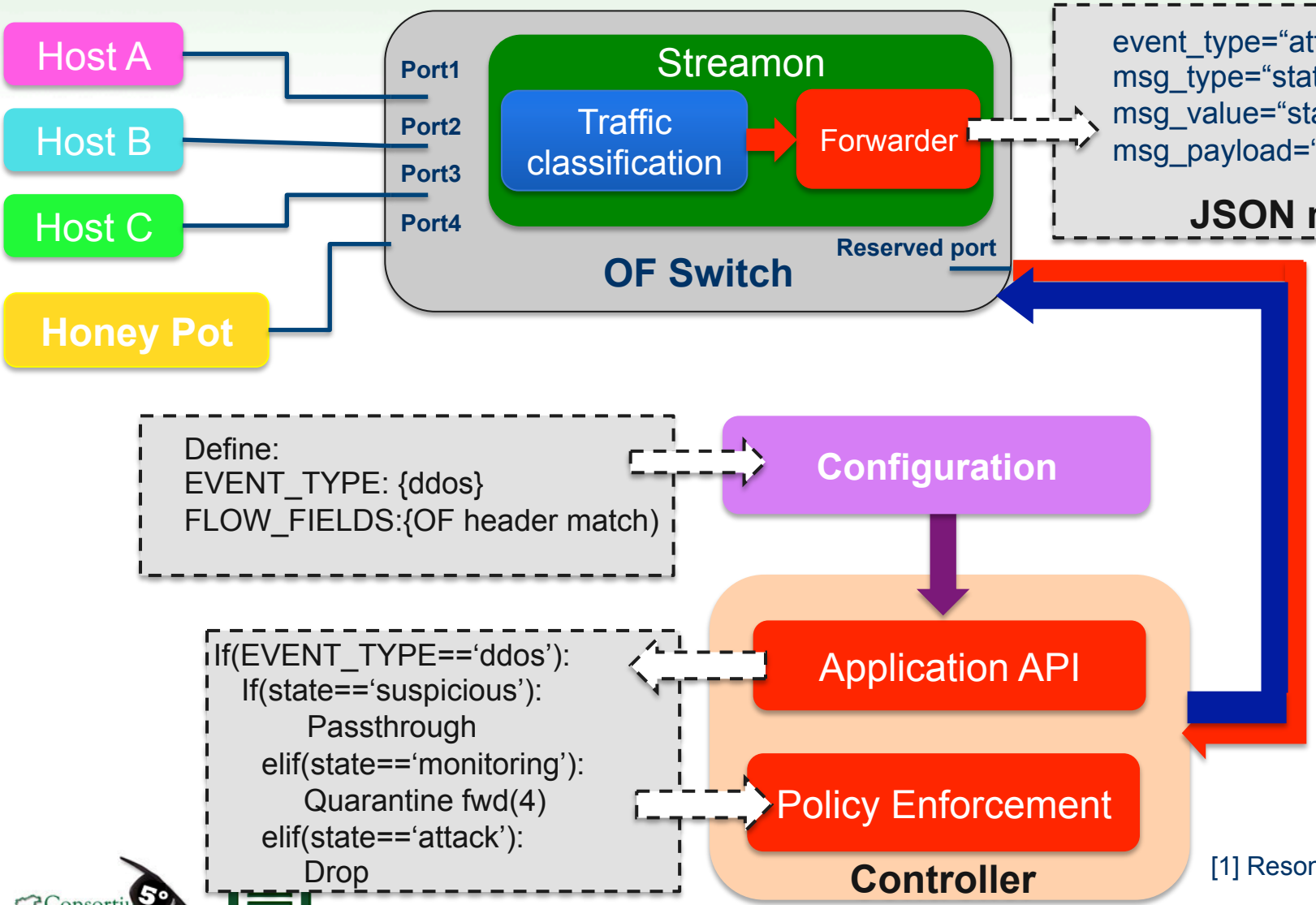
  - Is a state driven controller
    - Update the current state **but**
    - need the third application for state transition
      - E.g. next state, event_type
  - Support of event composition(parallel, sequential)
    - E.g. ddos >> ratelimiter

Streamon: Monitoring Application

- **a software-defined stream-based monitoring API**
- Programmable monitoring probes
  - exploits eXtended Finite State Machines (XFSM)
  - Sketch data structure(bloom filter and counter bloom filter)
  - Packet-per-packet basis
  - high level and configurable API
    - Through high-level XML- like language
  - Configurable output.



Demo: using Mininet-Streamon-Resonance



[1] Resonance controller

Recap: where is the intelligence?

NEED for Flow-level states visibility

Comp

Contro

Winning SDN idea for network-wide states and «big picture» decisions

Events from switches
Topology changes,
Traffic statistics,
Arriving packets

Commands to switches
(Un)install rules,
Query statistics,
Send packets

Nutshell: controller is responsabile for flow state handling

Control plane scalability and reliability

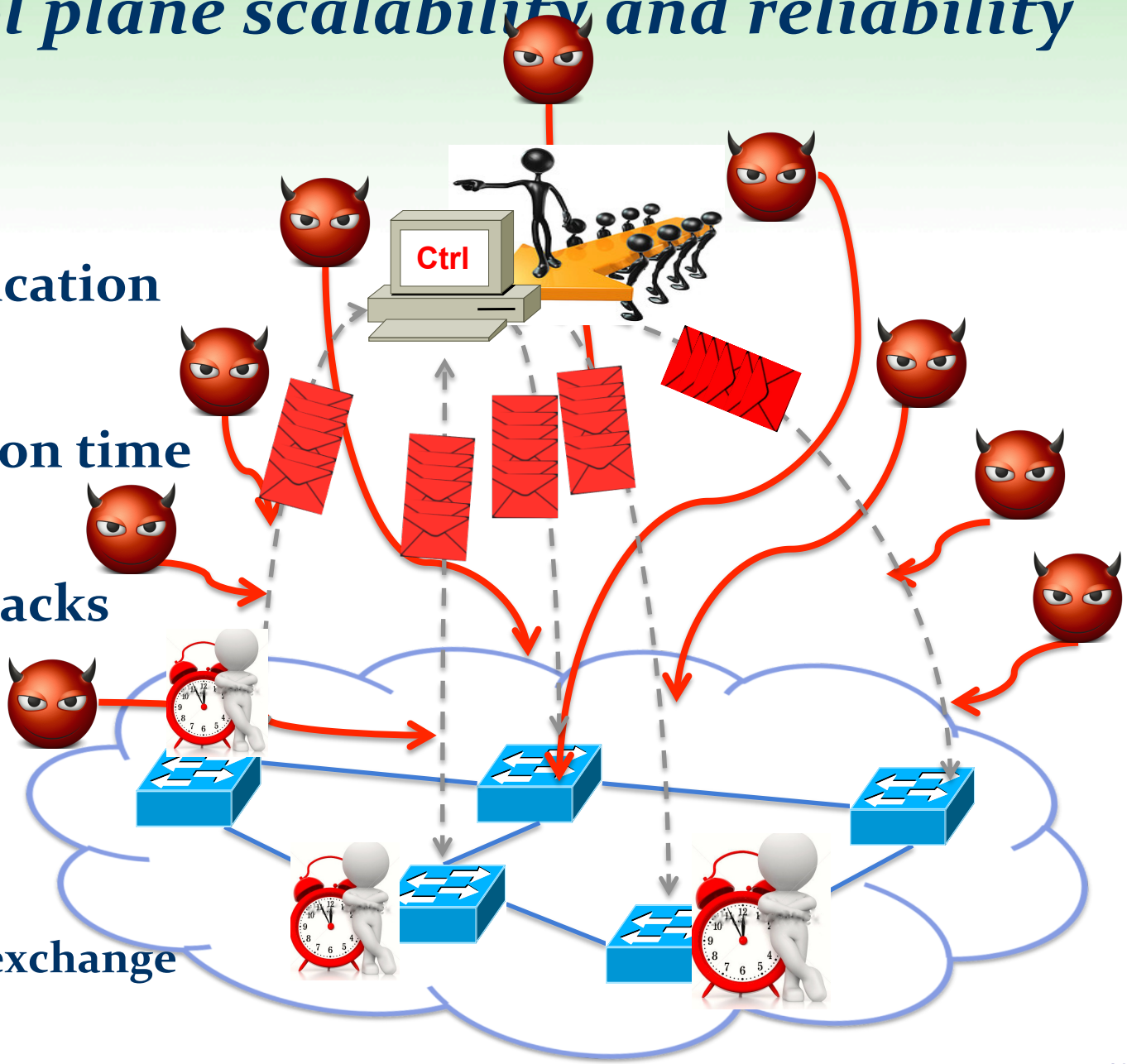
BUT

Needs communication

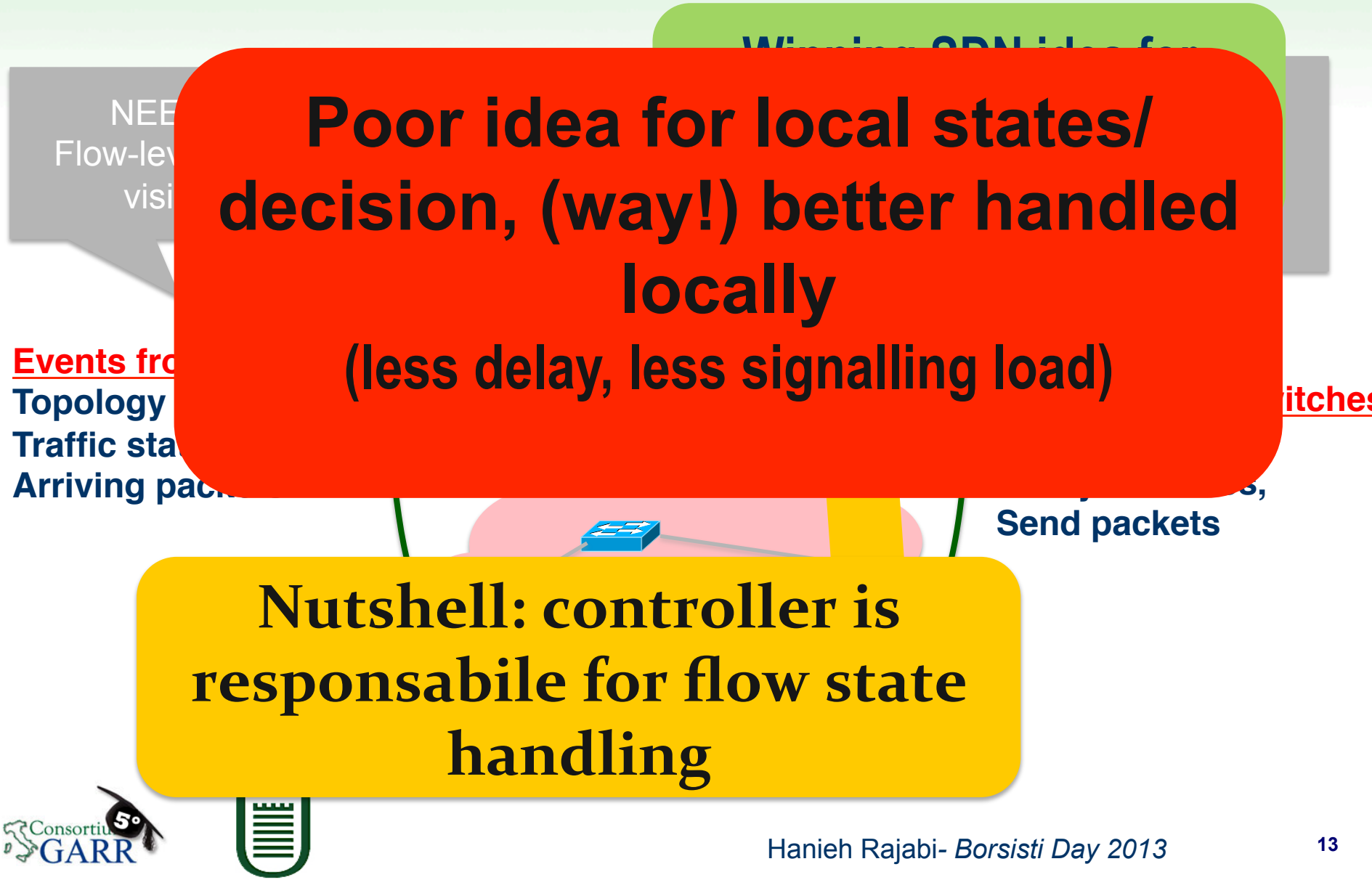
Increased reaction time

Target to the attacks

Huge num of msg exchange



Recap: where is the intelligence?

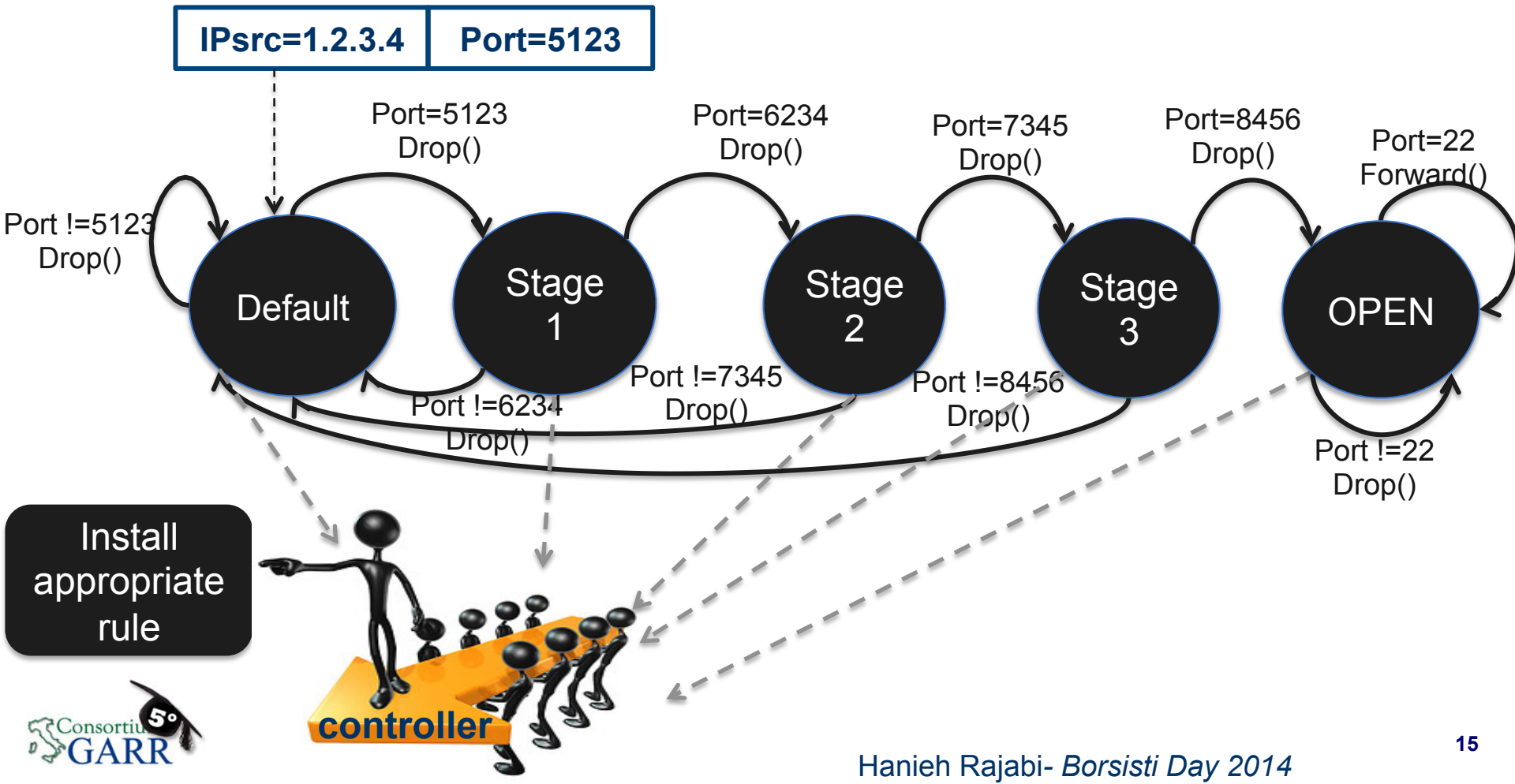


Solution: delegate control to devices

- Some control functions **inside** the devices (more intelligent!)
  - Keeping central SDN controller to full control of all delegated
  - Scalability is mostly achieved *by design*
    - *Reduction in reaction time (network wire speed and per packet basis)*
- greater involvement from the data-plane layer.

Example: how you'd implement an OF port knocking firewall using Mealy machine?

- knock «code»: 5123, 6234, 7345, 8456



Can transform in a flow table? Yes!

XFSM table

Match fields		Actions	
state	event	action	Next-state
DEFAULT	Port=5123	drop	STAGE-1
STAGE-1	Port=6234	drop	STAGE-2
STAGE-2	Port=7345	drop	STAGE-3
STAGE-3	Port=8456	drop	OPEN
OPEN	Port=22	forward	OPEN
OPEN	Port=*	drop	OPEN
*	Port=*	drop	DEFAULT

1 state machine: all flows (same knocking sequence)
7 states -> depend on xfsm program!

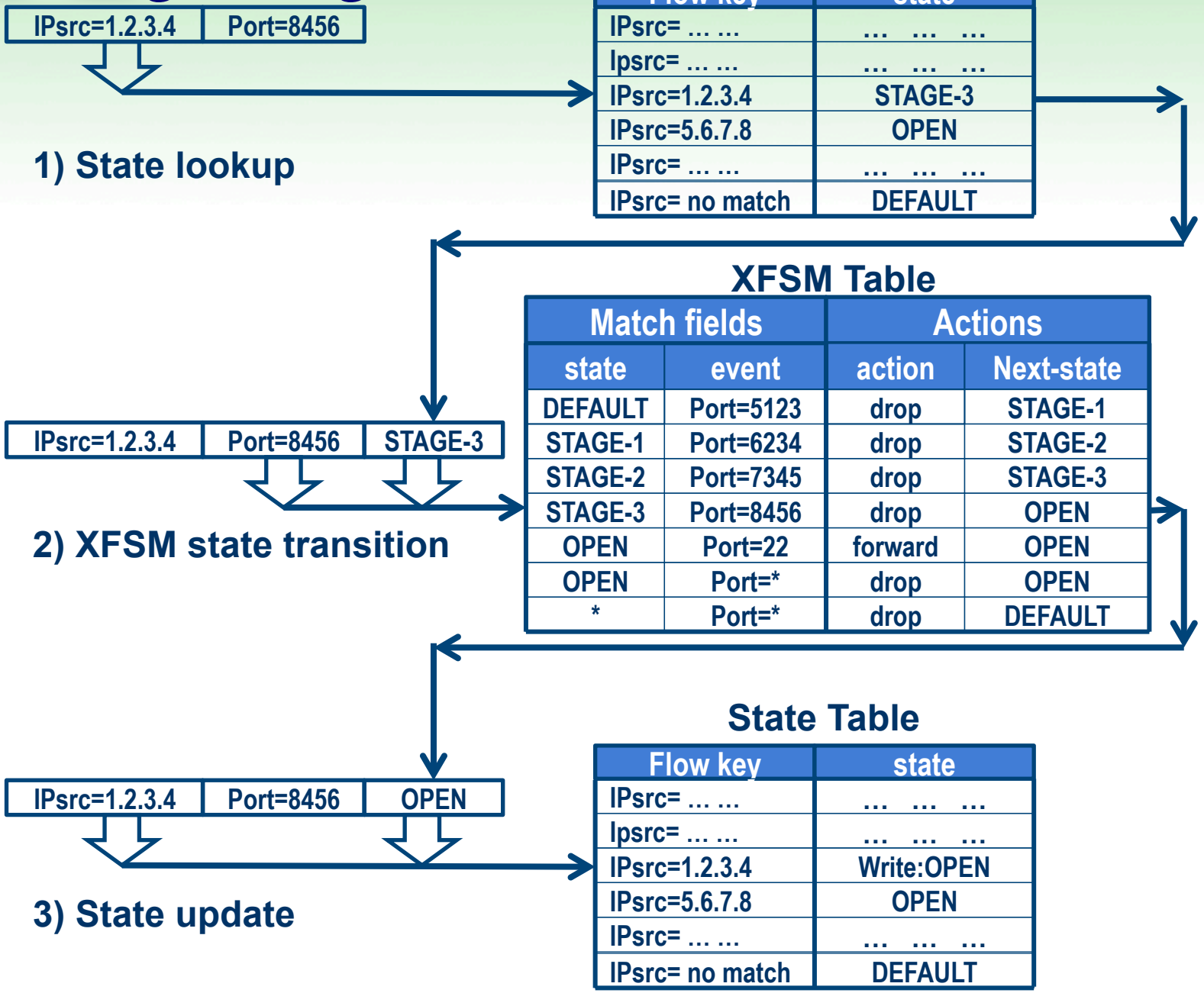
If next state were an OF instruction, this would be a STANDARD OF1.3+ flow table!
TCAM implementation

And what about flow states?

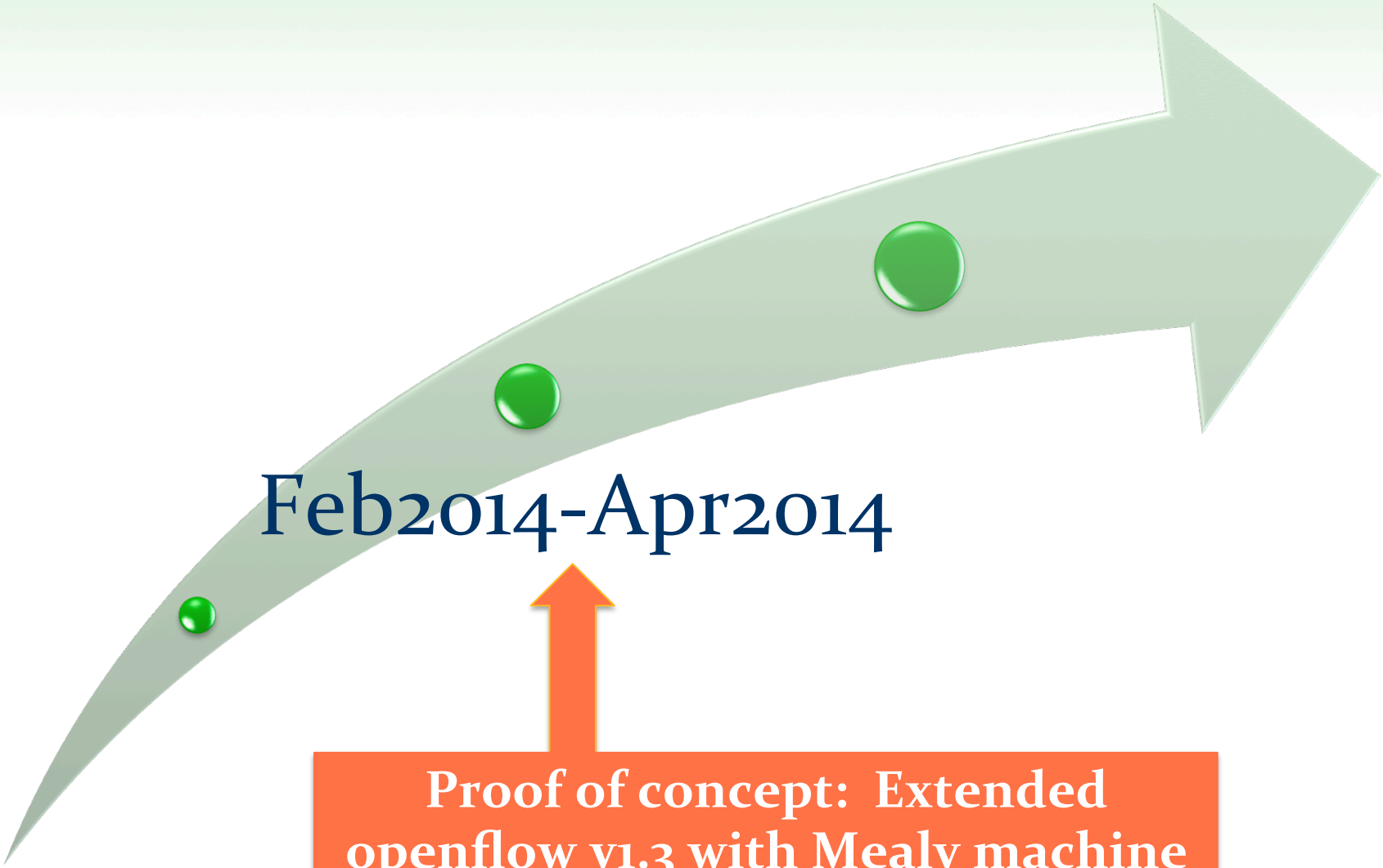
Flow key	state
IPsrc= … …	… … …
Ipsrc= … …	… … …
IPsrc=1.2.3.4	STAGE-3
IPsrc=5.6.7.8	OPEN
IPsrc= … …	… … …
IPsrc= no match	DEFAULT

Just a (Hash) Table, e.g. dleft
No need for TCAM

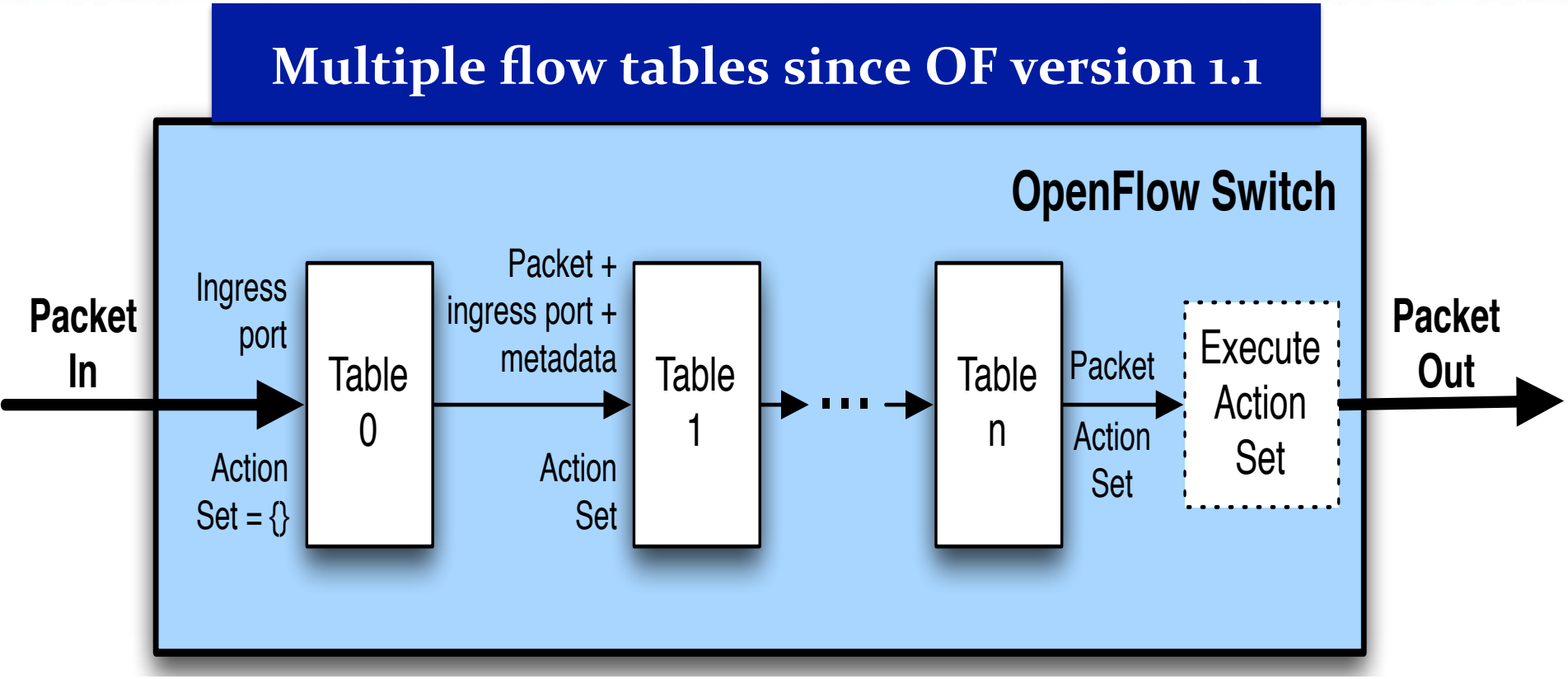
Putting all together



Time schedule



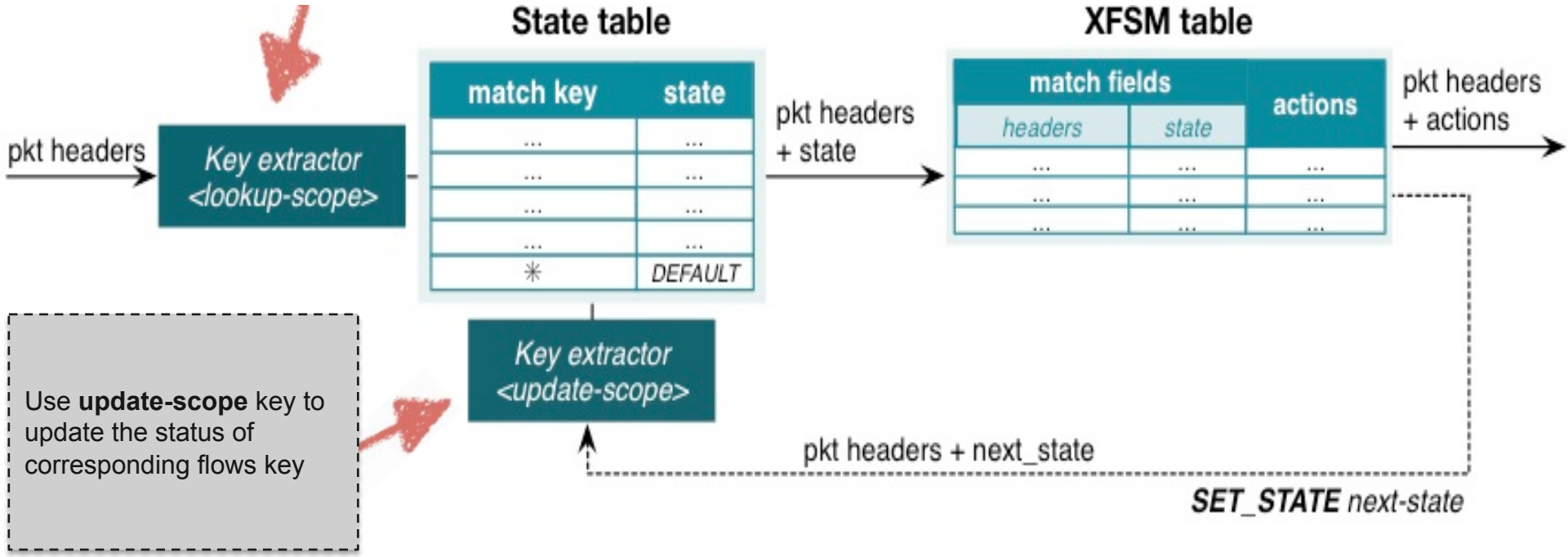
Pipeline processing in OpenFlow switch



Extend OpenFlow pipeline by State table

Bidirectional state handling

Use **lookup-scope** key to find out the actual state of flows



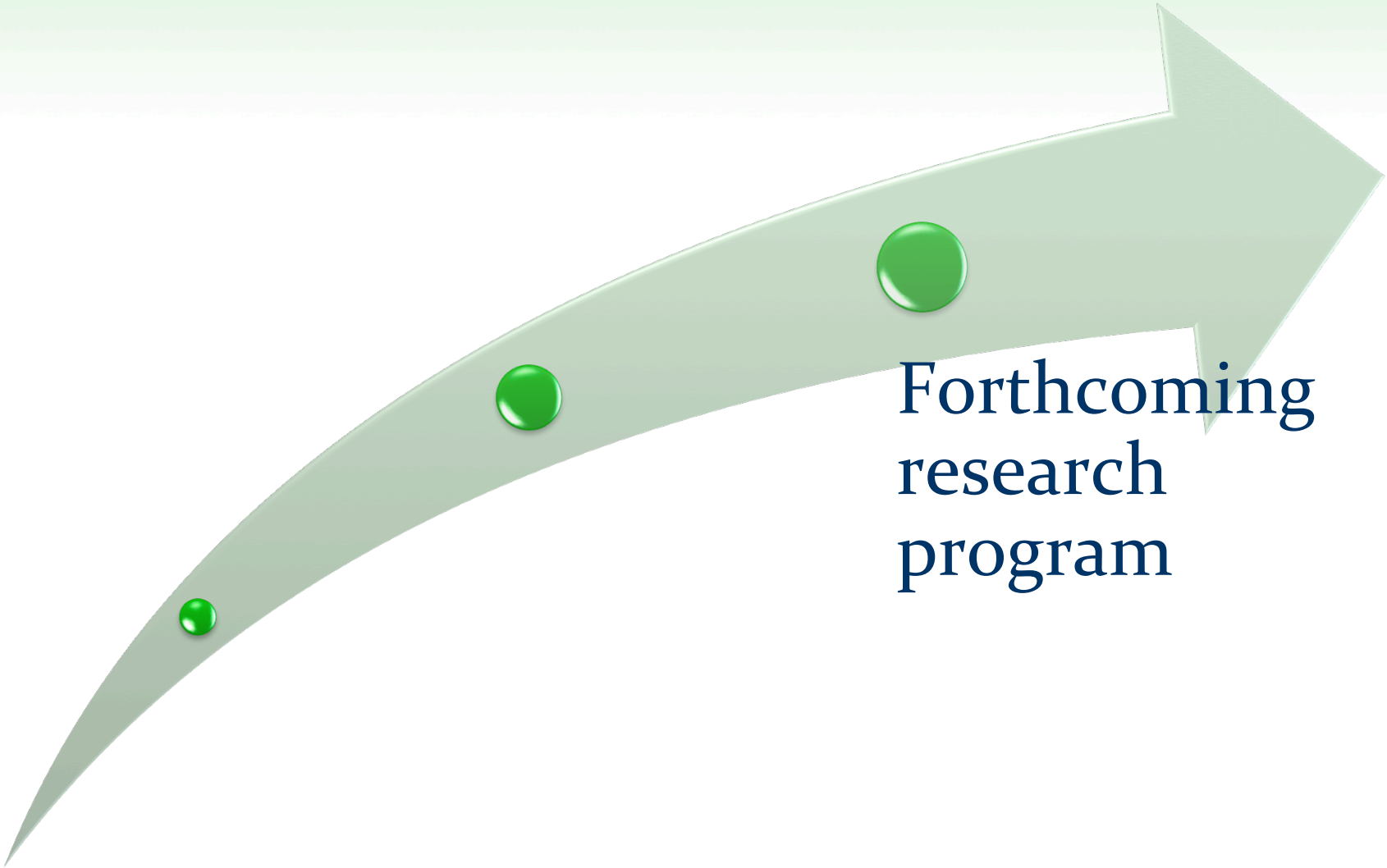
Use **update-scope** key to update the status of corresponding flows key

Pipeline processing

Details implementation

- New switch capability: OFPC_TABLE_STATEFUL
- New flow table: attached with stateful table
- New instruction: OFPIT_SET_STATE
- New controller to switch msg: OFP_STATE_MOD
- Extended existing SDN controllers to **support the advanced API**
  - Integrate XFSM's support into existing control framework (Ryu).

Time schedule



Forthcoming research program

New research direction for monitoring applications in SDN

security applications are scalable and responsive to dynamic network threats

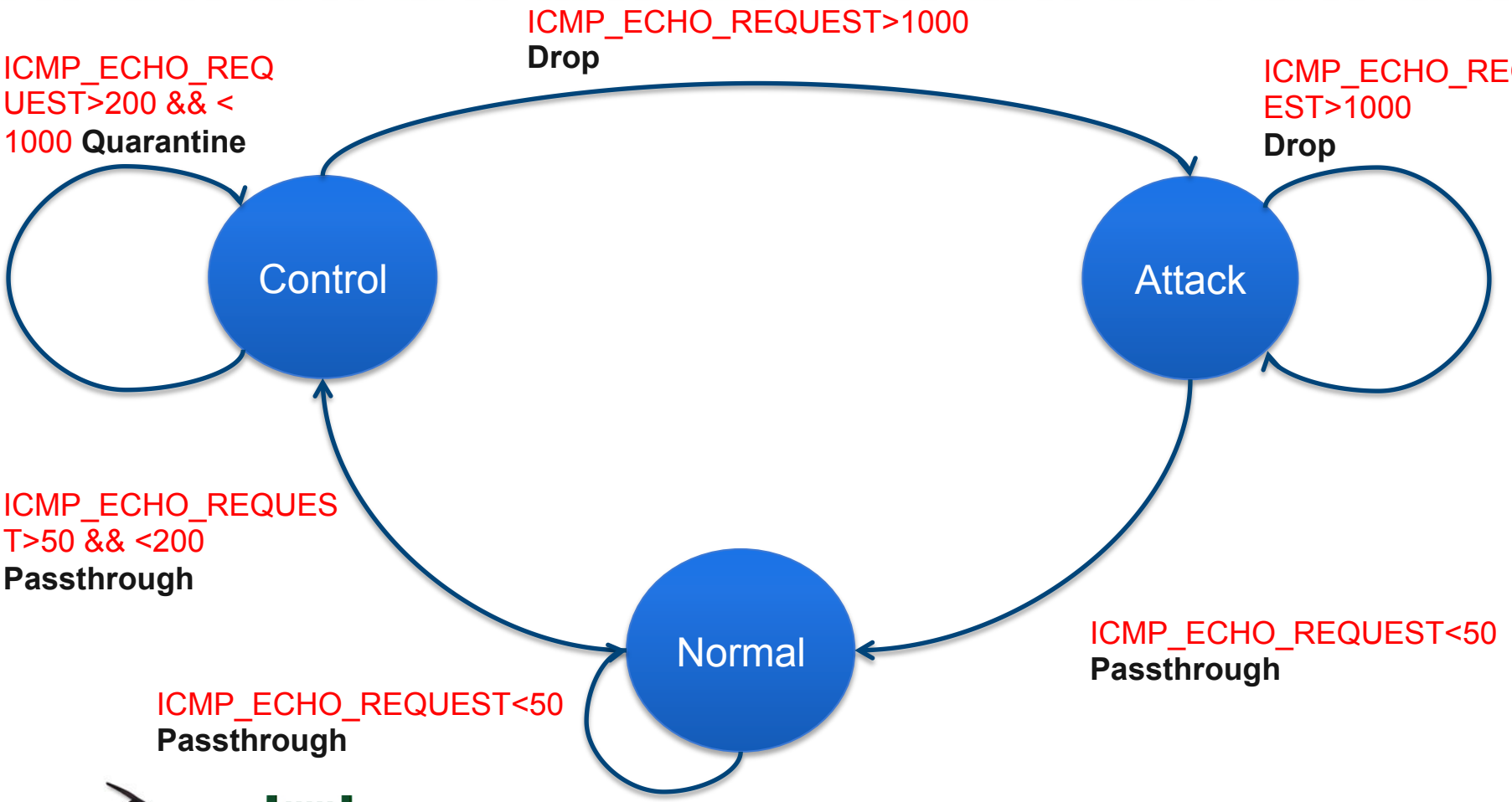
Behavioral data-plane can be in handy for monitoring purpose?

Eliminate middleboxes for security advance functionality

How to use set of building block, namely actions, events & conditions

XFSM abstraction to configure *of complex monitoring*

A Simple example: Behavioral based OF monitoring application in the switch

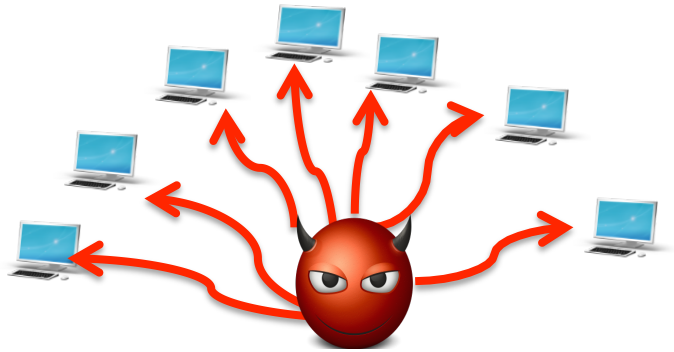


Off-the-shelf features in OpenFlow

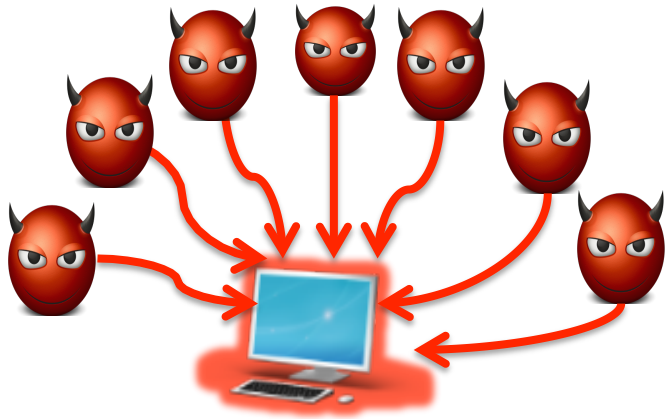
- OpenFlow switches are great at counting
  - Maintain counters for flow table, flow entry, port, queue, group, group bucket, meter and meter band (Rx/Tx of pkts/bytes, time duration).
- New features in Openflow v1.3
  - Meter bands: control the network rate and pose rate limiter

How about the traffic classification in OF switch directly?

- But can not answer is someone doing
  - a port scan? Either vertical or horizontal!



  - Or DDoS attack is running?



Source codes

- Implementation can be find at:
  - Extended of openflow v1.3 softswitch
  - https://github.com/haniehrajabi/stateful-ofsoftswitch13/tree/hani-xfsm
  - Extended Ryu controller
    - Unique available controller that supports OF v1.3
  - **https://github.com/haniehrajabi/ryu/tree/ofsoftswitch13**

Thank you

Q & A