# FOCUS: Function Offloading from a Controller to Utilize Switch Power

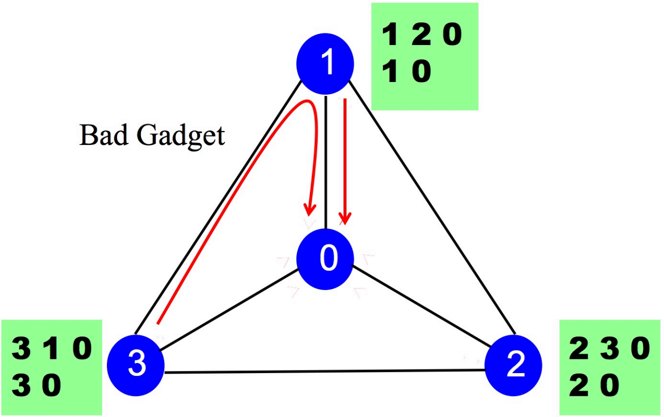
Ji Yang*, **Zhenyu Zhou***, Theophilus Benson, Xiaowei Yang, Xin Wu and Chengchen Hu

@IEEE NFV-SDN
Nov, 2016

*These authors contributed equally to this work.

# Introduction

- Traditional Networks
  - ❑ Distributed
  - ❑ Hard to manage
  - ❑ No global view for decisions



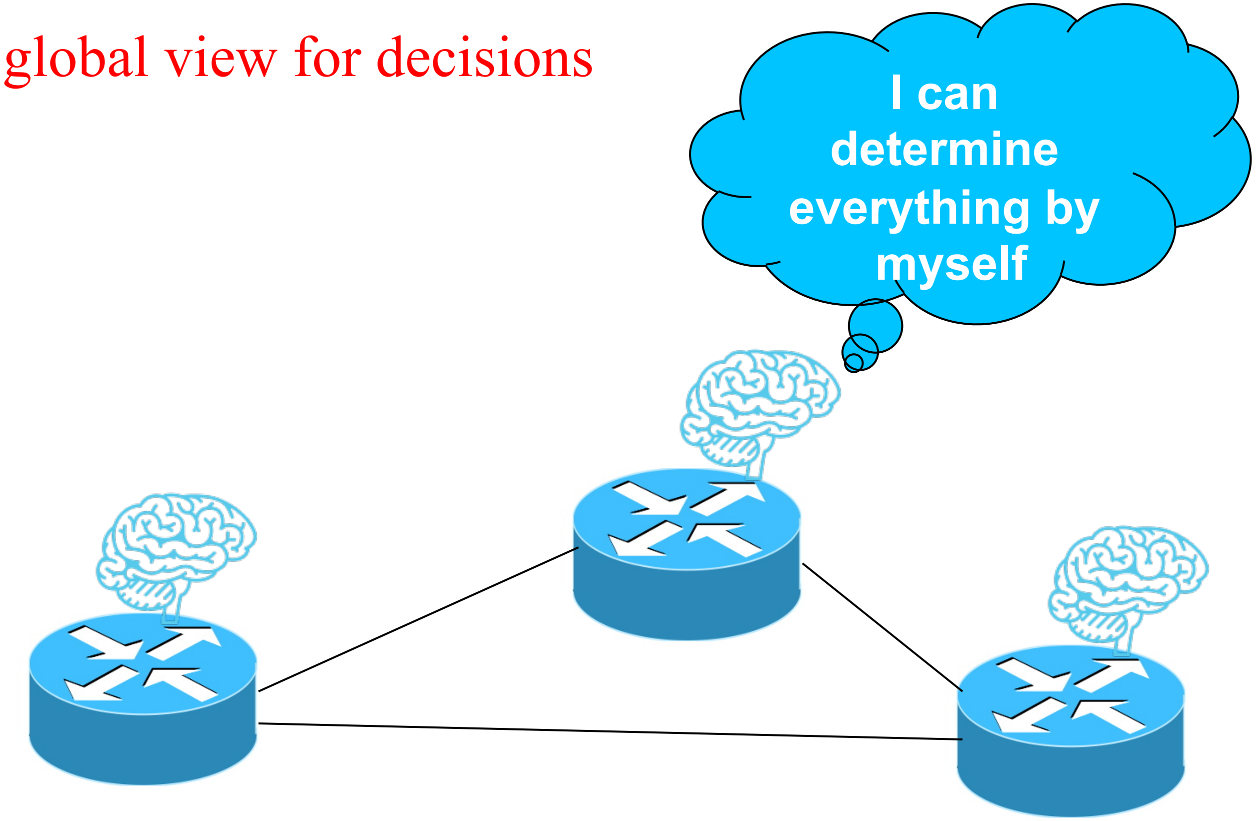
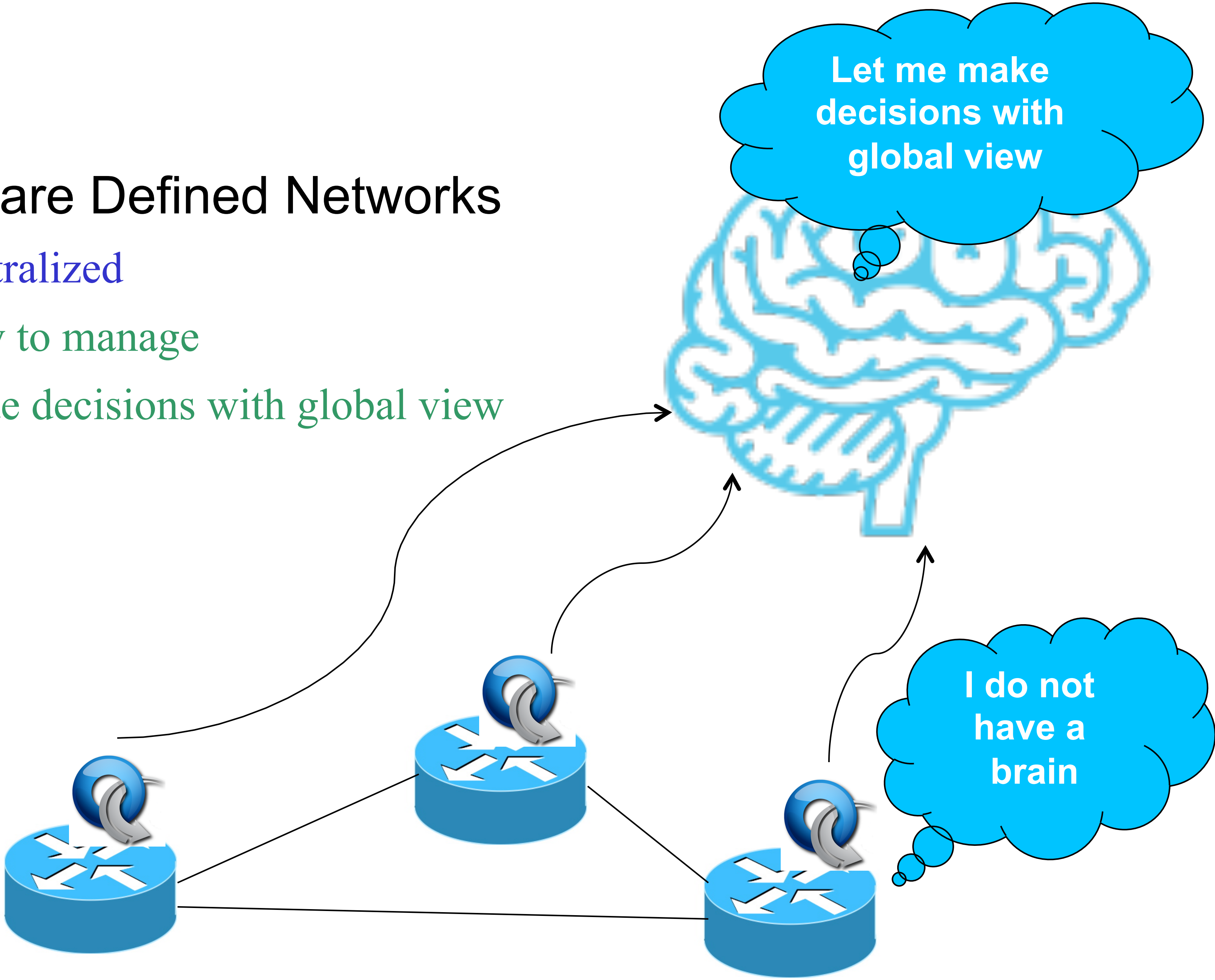
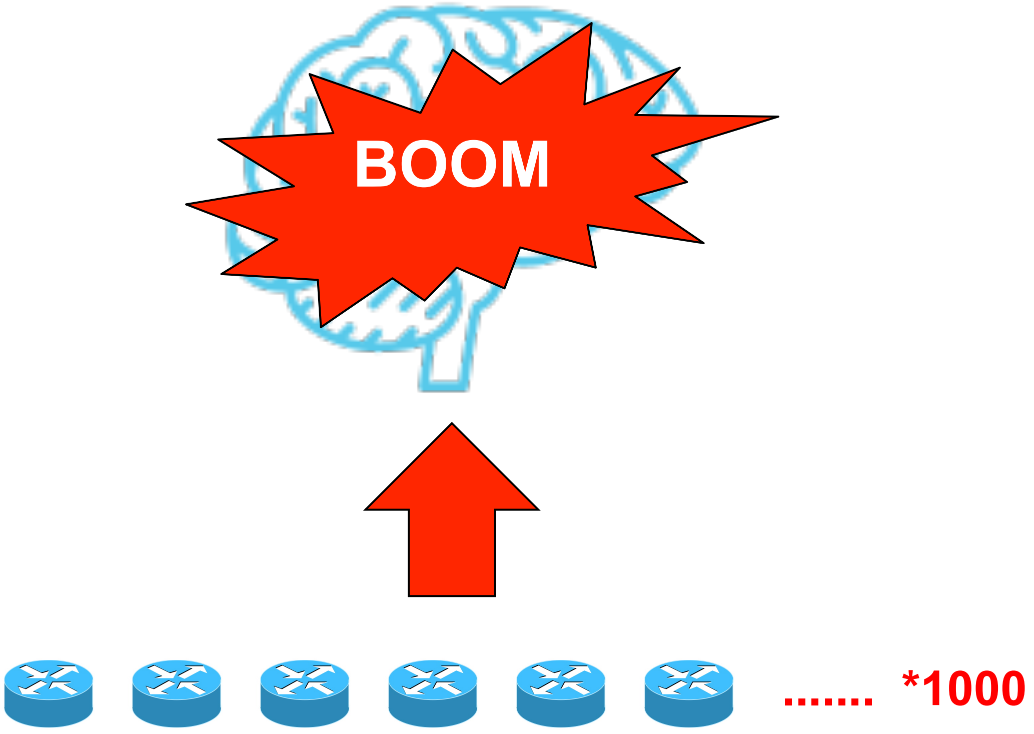


Image referenced from: http://dimacs.rutgers.edu/Workshops/NextGenerationNetworks/slides/Wilfong1.pdf

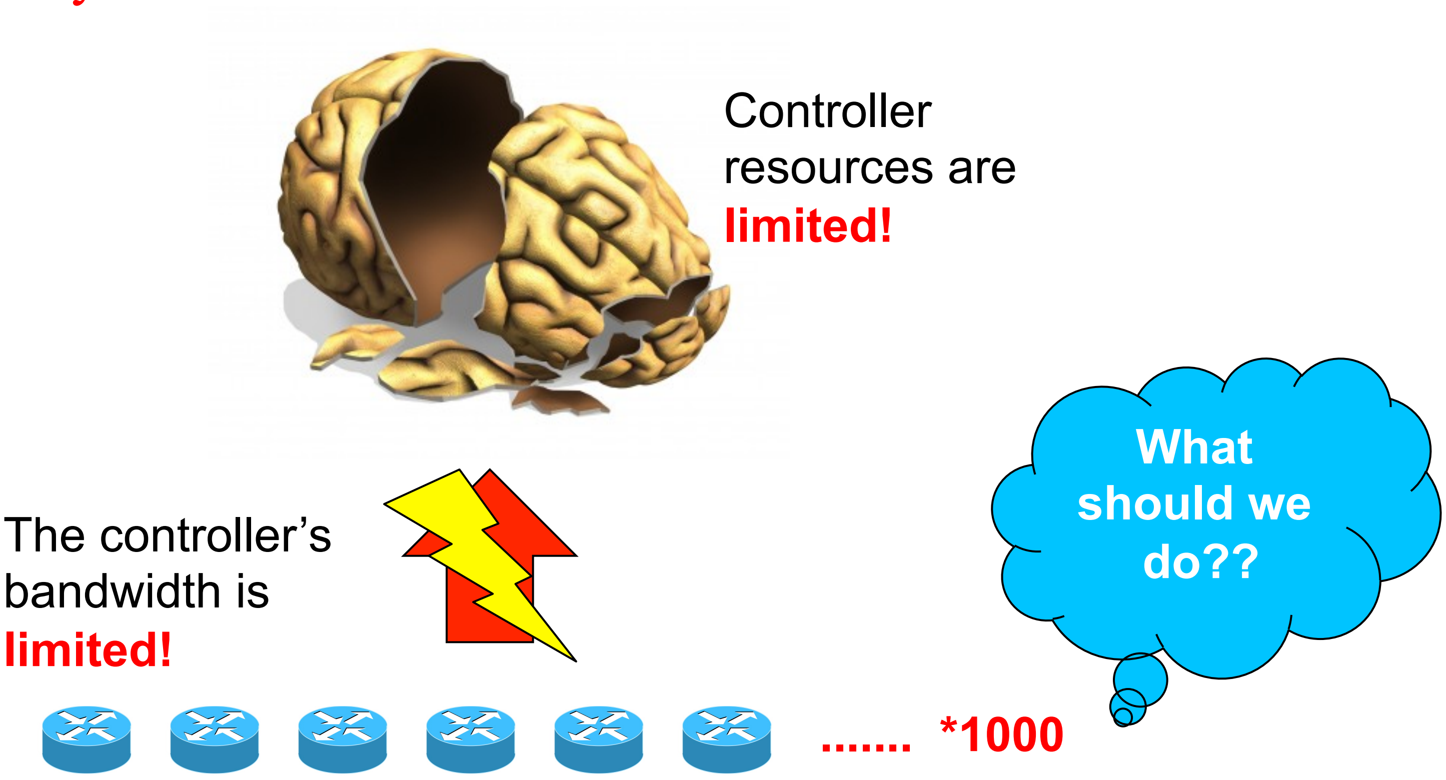
- Software Defined Networks
  - Centralized
  - Easy to manage
  - Made decisions with global view



- Software Defined Networks
  - Scalability



- Software Defined Networks
  - Scalability



Controller resources are **limited!**

The controller's bandwidth is **limited!**

....... *1000

What should we do??

# Introduction

- Existing Solutions
  - Hardware optimization (DevoFlow [SIGCOMM’11])
    - Inflexible
  - Distributed control planes (Beehive [SoSR’16])
    - Control traffic overhead

# Introduction

- Existing Solutions
  - Turning on legacy functions (Open vSwitch [NSDI'15])
    - Losing visibility and control
  - Executing arbitrary code in the switches (Kandoo [HotSDN'12])
    - Heavyweight
- Our Approach
  - Delegating functions into the switches

# Roadmap

Introduction

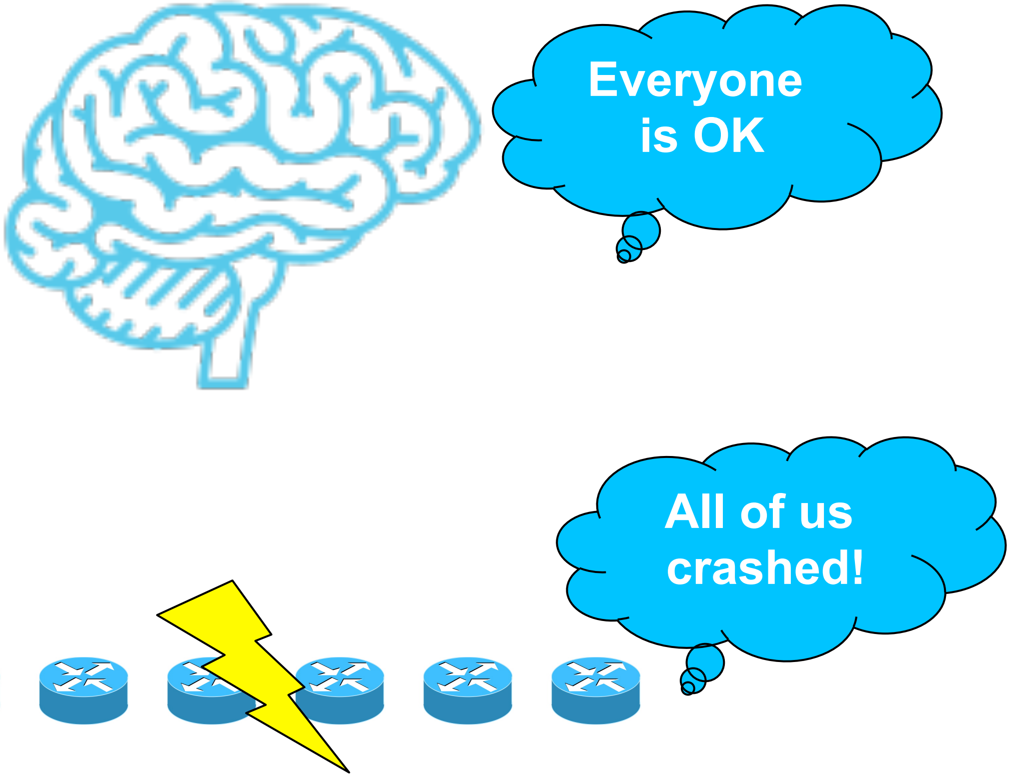
Challenges and Solution

Architecture and Examples

Evaluation

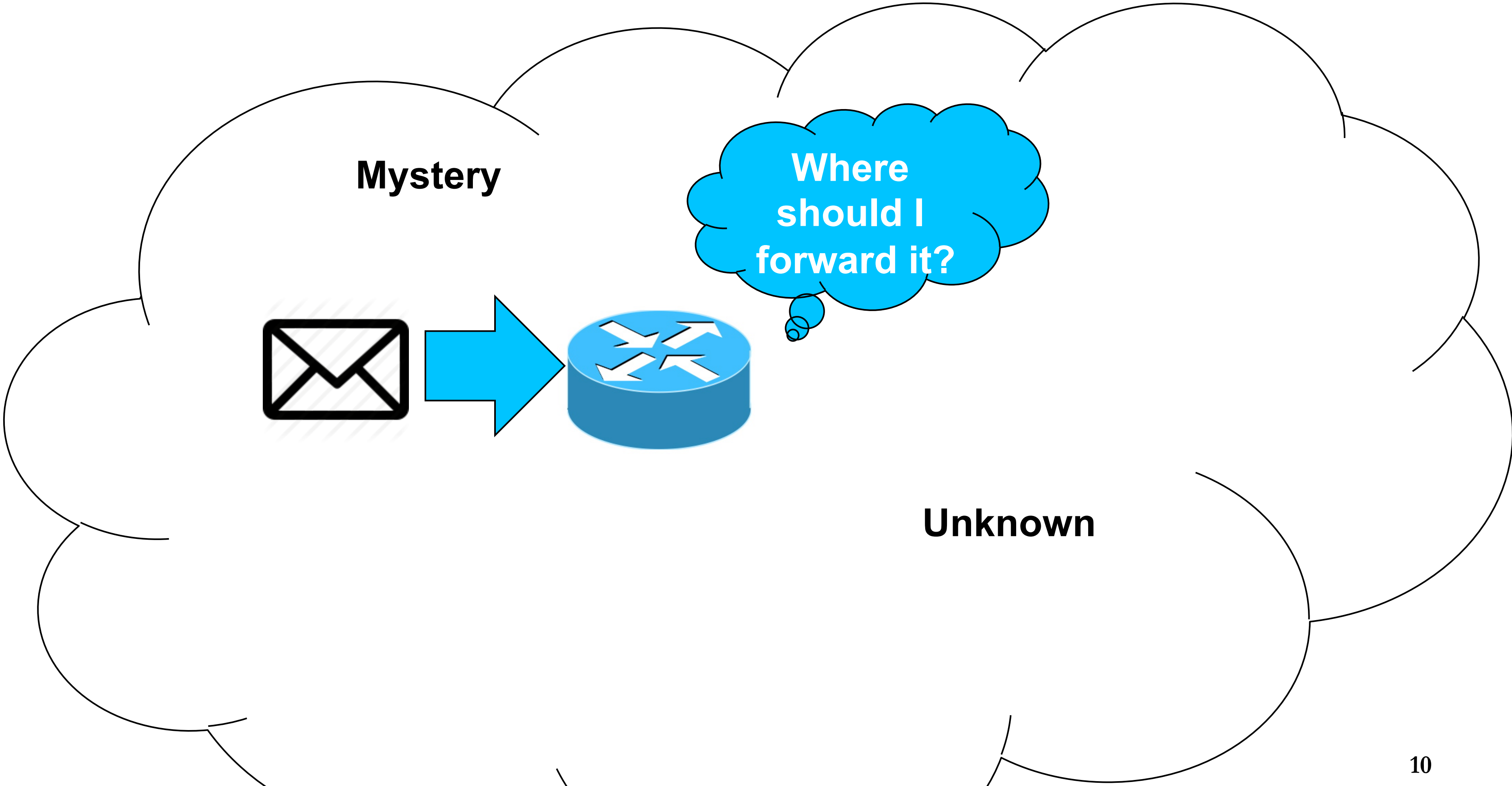
# Challenges

- Global Visibility
  - The controller should keep the identical visibility as before



# Challenges

- Local Decisions
  - The abstraction should act solely on local information



# Challenges

- No Hardware Modifications
  - Implementing the solution using the switch software
  - Leveraging the CPU power from the switches again
    - More flexible
    - Policies are not limited
- But
  - How to cooperate with the controller?
  - How to avoid defining a new programming language?

- What is inside a switch?

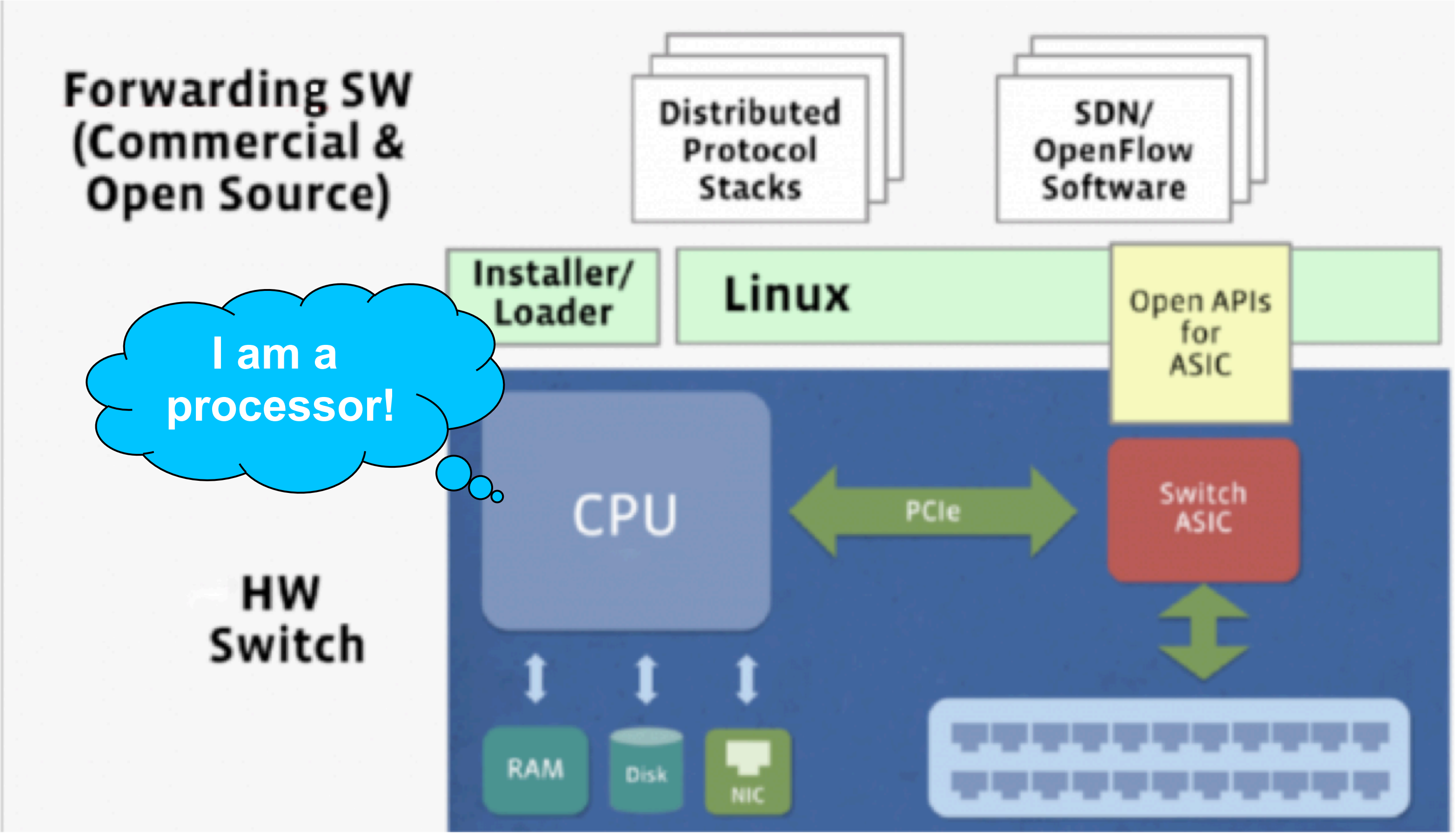


Image referenced from: https://sreeninet.wordpress.com/2014/07/23/open-compute-networking-project/

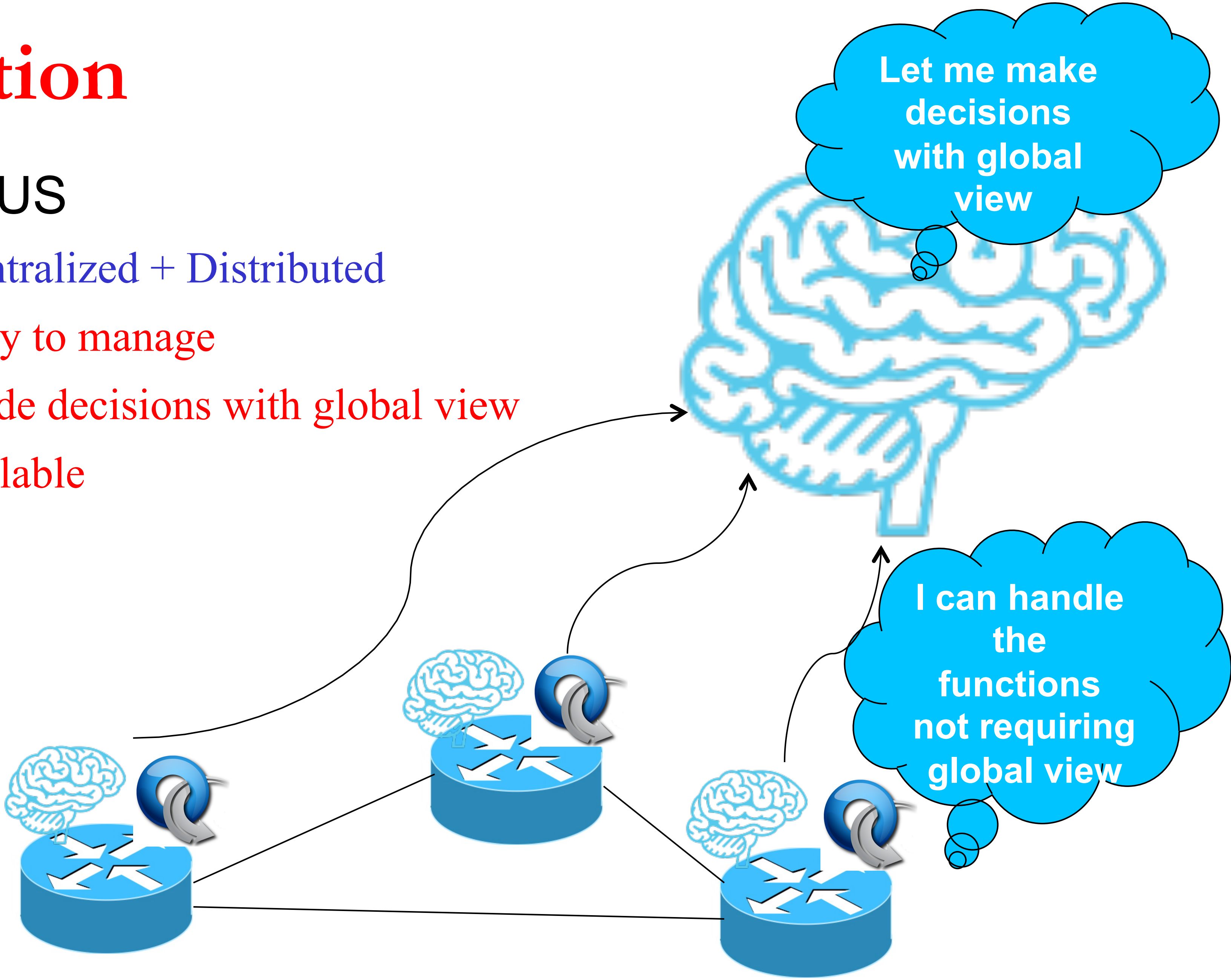
# Solution

- FOCUS: Function Offloading from a Controller to Utilize Switch Power
  - Offloads a subset of control functions into the switches' software stack
  - Defines a small set of APIs for offloading
  - Observation: not all control functions need global view
  - Example applications: ARP, LLDP and elephant flow detection

- "Subset"
  - Stable local functions
    - Remain **stable** over time as long as the network configuration does not change
    - Only require input **local** to a switch to compute

# Solution

- FOCUS
  - Centralized + Distributed
  - Easy to manage
  - Made decisions with global view
  - Scalable


Let me make decisions with global view
I can handle the functions not requiring global view

# Roadmap

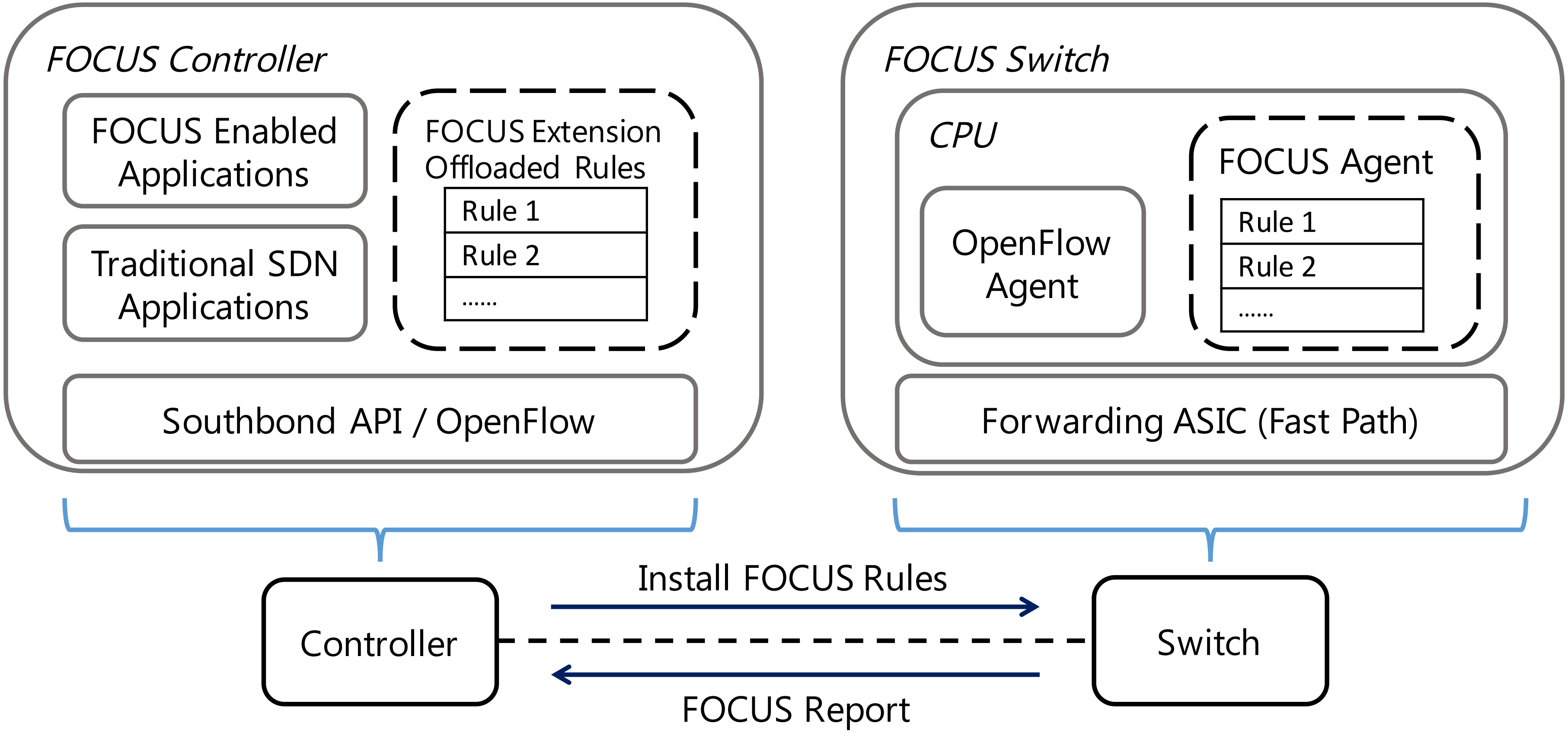
Introduction

Challenges and Solution

Architecture and Examples

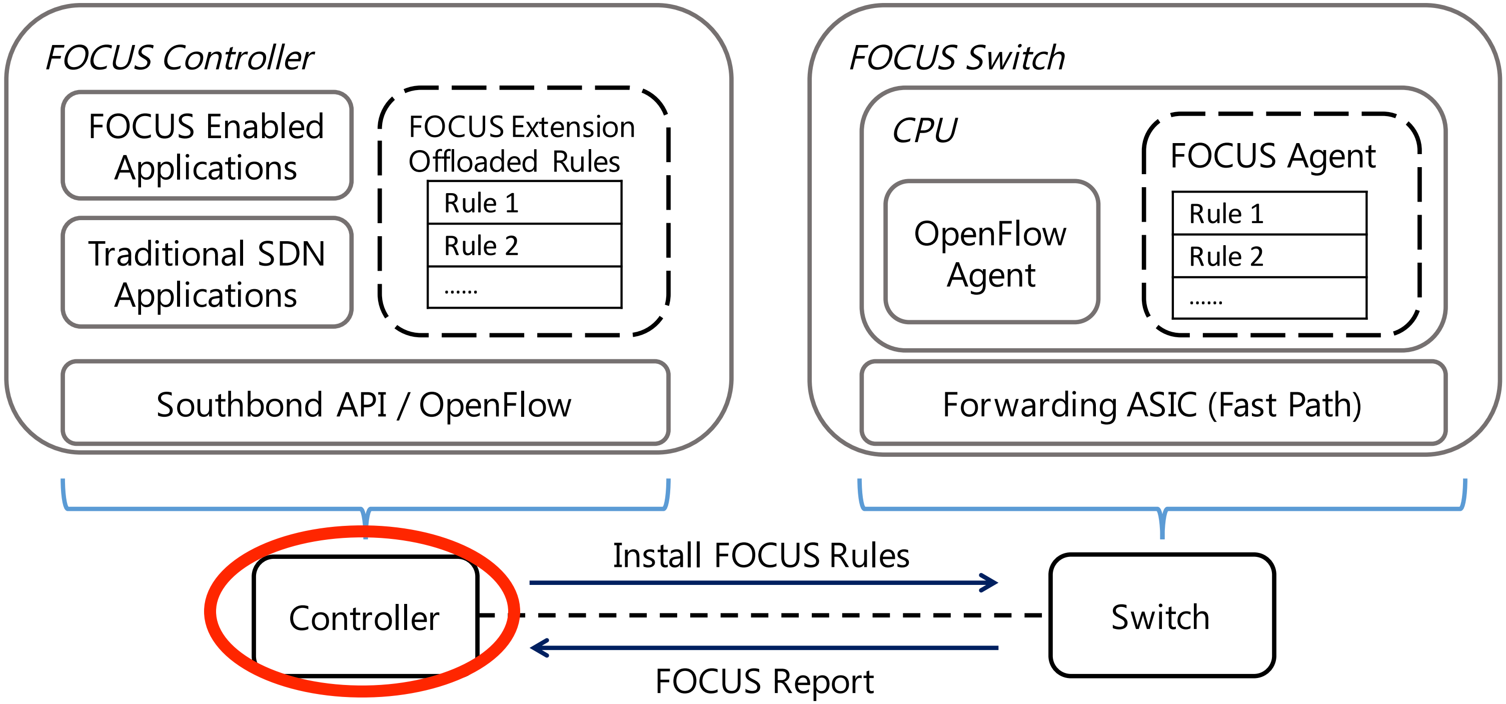
Evaluation

# Architecture



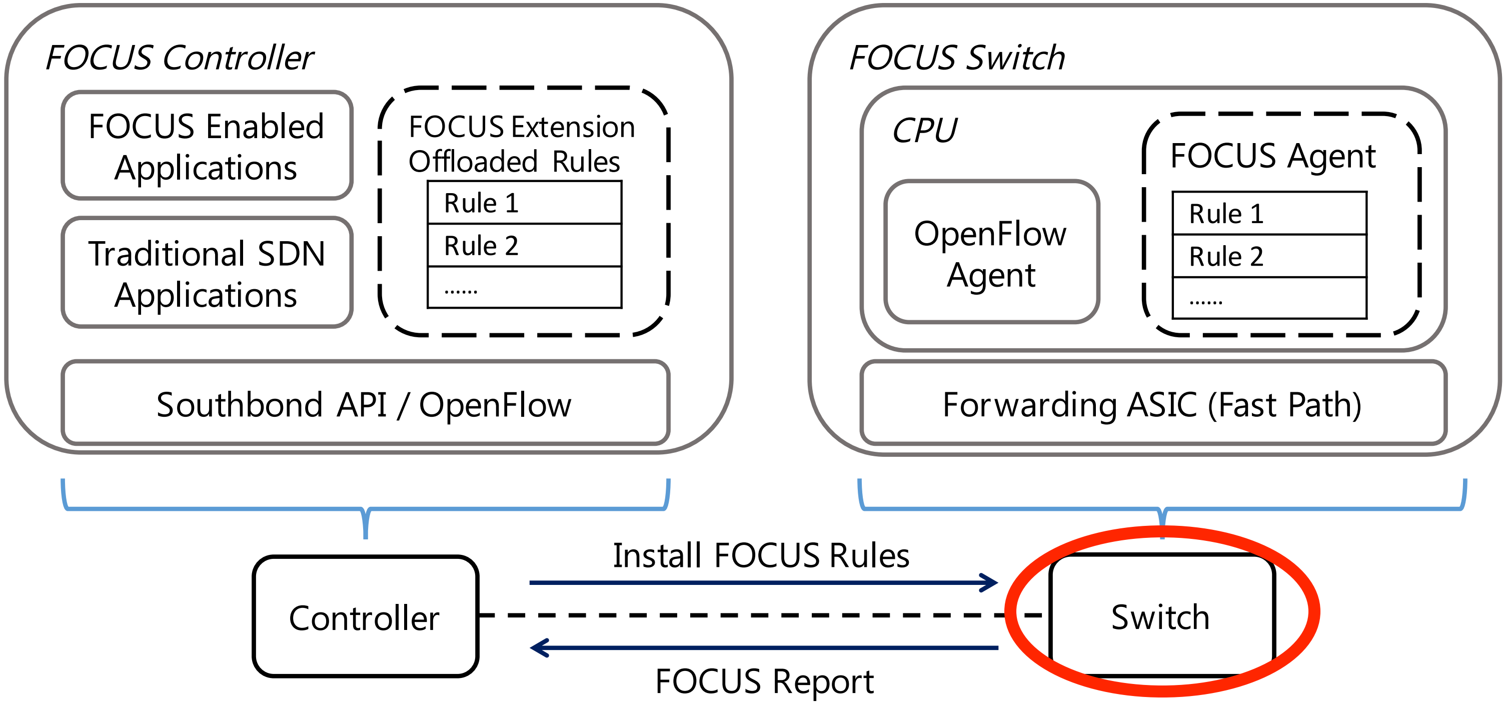
# Architecture

- FOCUS Controller
  - FOCUS enabled applications AND the traditional ones
    - The applications requiring global view are still treated as before
  - Offloaded rules table
    - Maintains the status of offloaded rules



# Architecture

- FOCUS Switch
  - FOCUS agent AND OpenFlow agent
    - Handles the FOCUS rules
    - Inside switch software stack



# FOCUS Rules

- Trigger
  - *Timer-based*: for periodically polling and sending packets.
  - *Packet matching predicate*: flexible TLV packet matching.

- Action-List
  - *Packet operations*: for accessing fields of the input packets.
  - *Flow entry operations*: for accessing the flow table entries.
  - *Message operations*: for communicating with the controller.

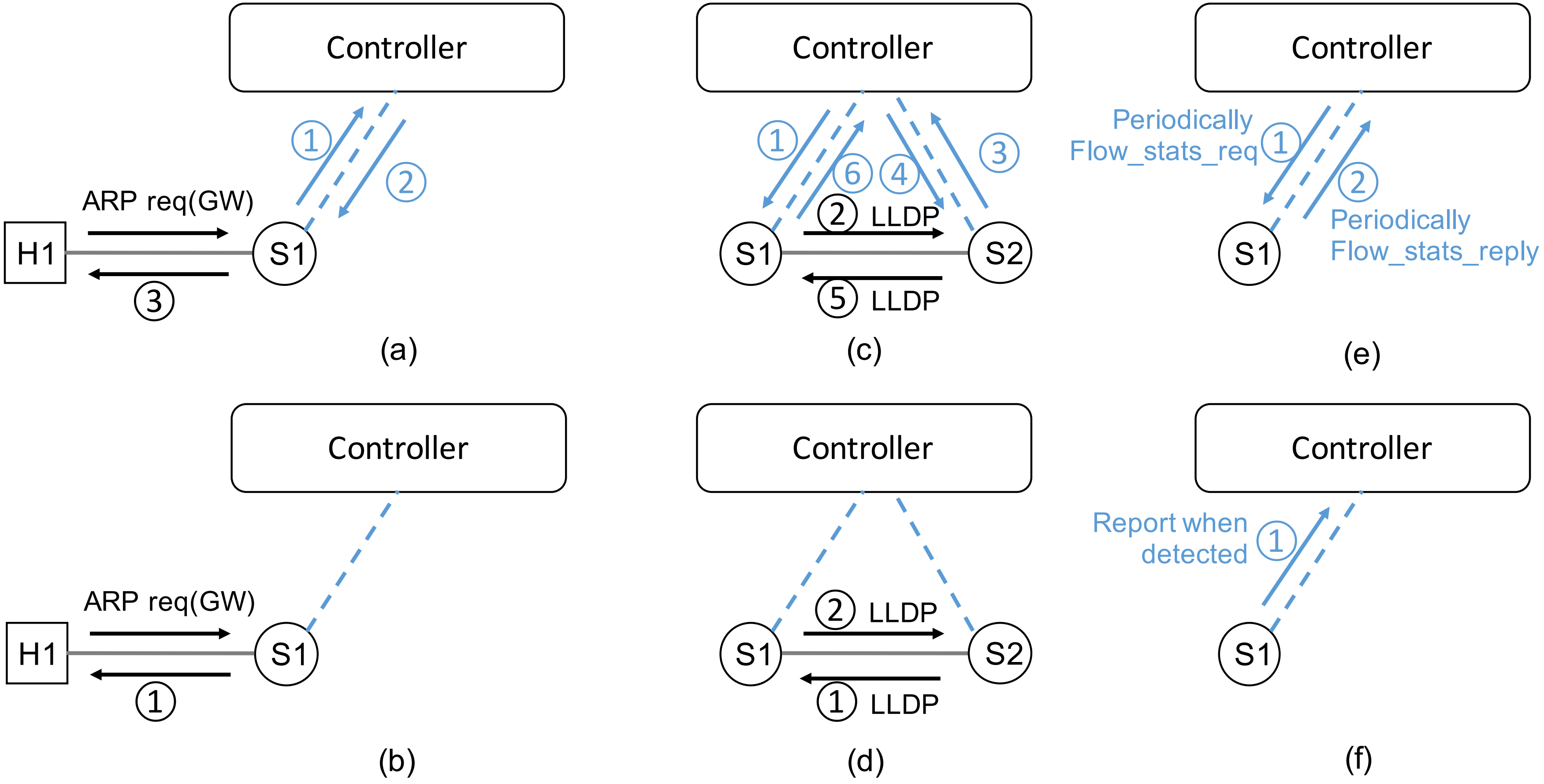
- Timeout
  - Informs the controller of whether a rule is still active.

| | | |
|---|---|---|
| Trigger | Action List | Timeout |

# Examples

- Comparison of OpenFlow with FOCUS Workflow
  - Host Discovery (ARP, ICMP for TTL expiration)
  - Topology Maintenance (LLDP)
  - Traffic Statistics (elephant flow detection)

# Examples



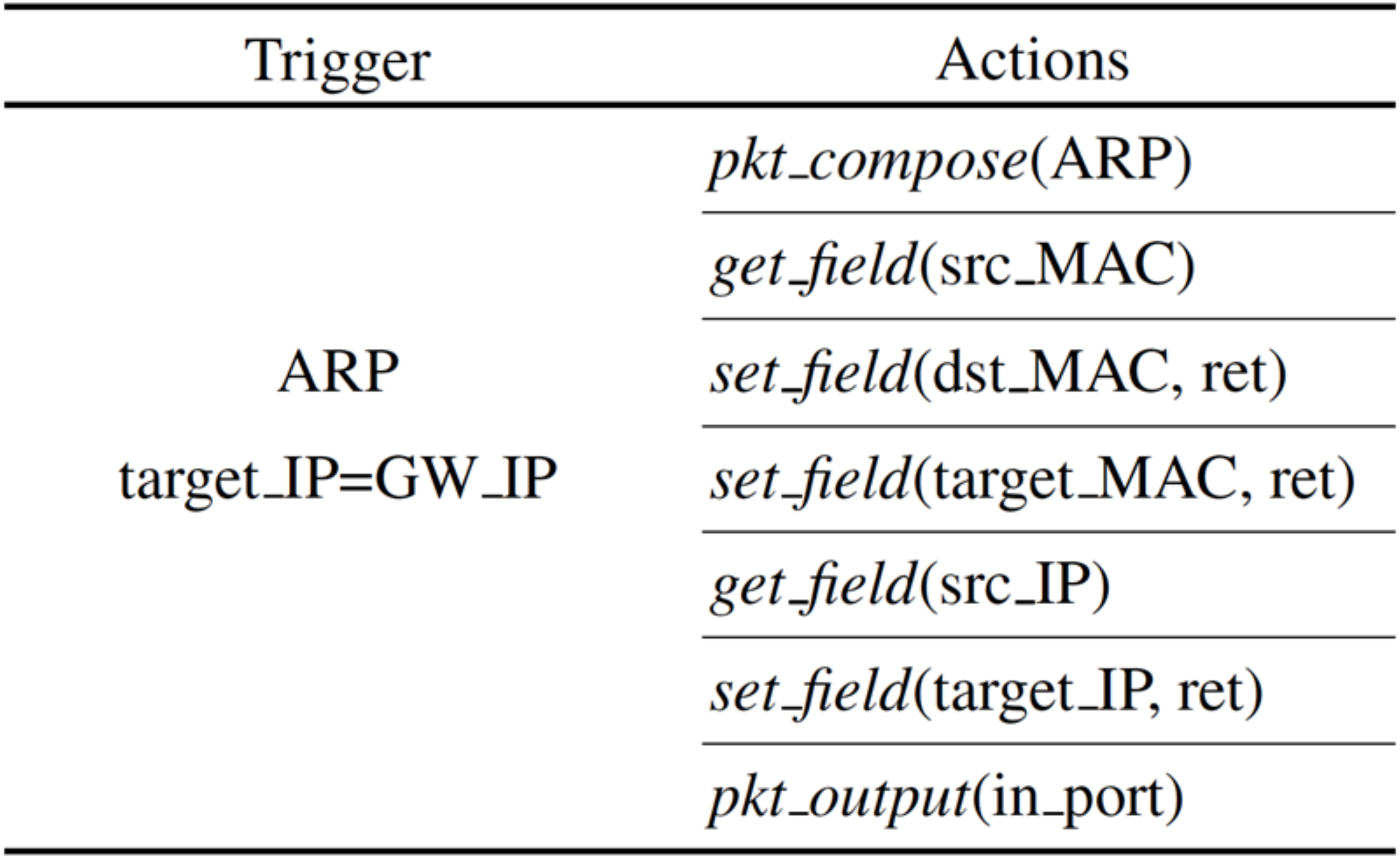
**ARP**

**LLDP**

**Elephant Flow Detection**

# Examples

- API Example (ARP Reply for Default Gateway)



| Trigger | Actions |
|---|---|
| ARP<br>target_IP=GW_IP | *pkt_compose*(ARP) |
| | *get_field*(src_MAC) |
| | *set_field*(dst_MAC, ret) |
| | *set_field*(target_MAC, ret) |
| | *get_field*(src_IP) |
| | *set_field*(target_IP, ret) |
| | *pkt_output*(in_port) |

# Roadmap

Introduction

Challenges and Solution

Architecture and Examples

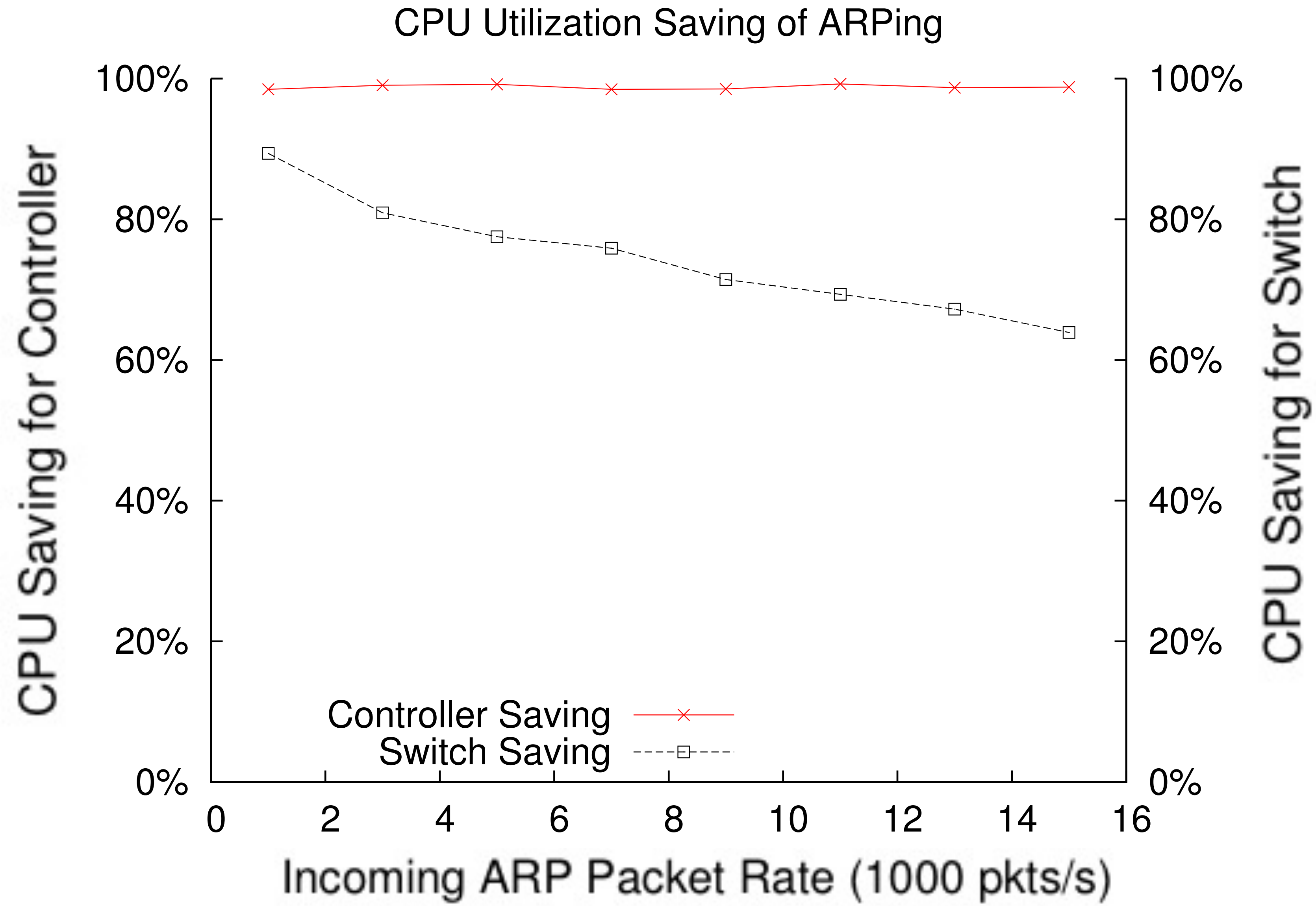
Evaluation

# Evaluation

- Setup
  - Floodlight + Mininet (Open vSwitch)
  - Topology
    - A single switch (ARP), mesh-like topology (LLDP), linear topology (elephant flow detection)

- Questions
  - Benefits for the controller
  - Costs for the switch
  - Benefits for different applications

# Evaluation: CPU Utilization



# Evaluation

- Performance Improvement
  - Computational overhead is reduced by 80%－98%
    - Reduced controller overhead
  - Communication overhead is reduced by 50%－nearly 100%
    - Reduced switch overhead
  - ARP response time is shortened by 18ms
    - Benefits for the ARP application

# Conclusion

- FOCUS improves the scalability of an SDN controller by offloading certain control functions to switches.

- FOCUS defines stable local functions.

- FOCUS reduces the CPU utilization of both controller and switch side, the number of control messages and the response time.

# Thank you!

# Questions?