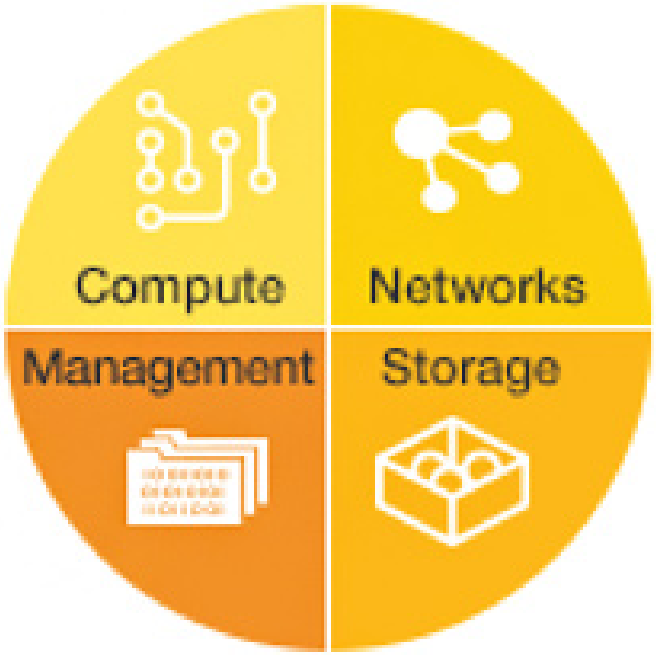
Introduction to OpenFlow, SDN & NFV

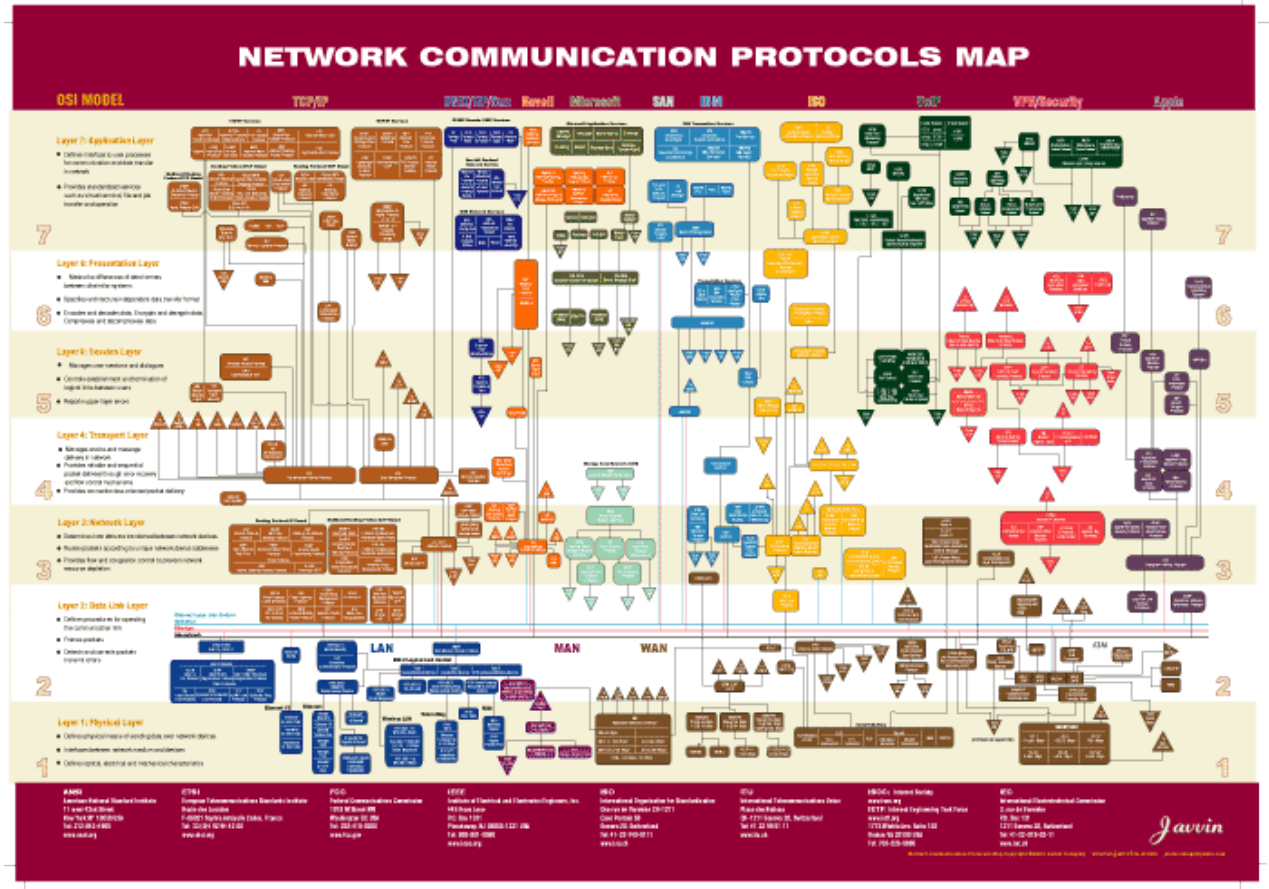
Converged Infrastructure

- Compute
- Storage
- Network



Modern Networking Complexity

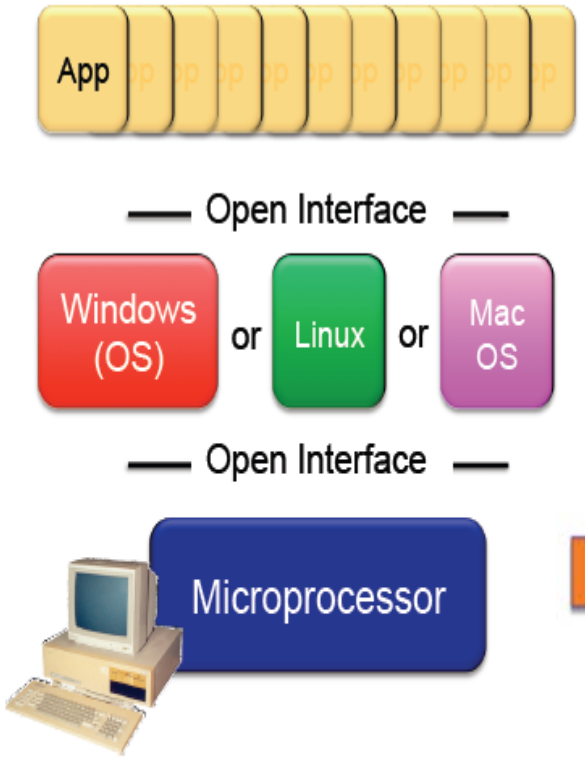
Vertically integrated
Closed, proprietary
Slow innovation



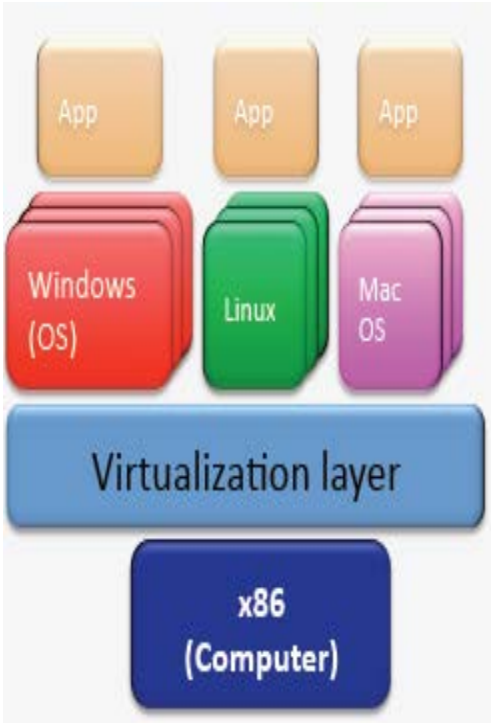
Ref: Javvin

Traditional Computing Vs Modern Computing

Vertically integrated
Closed, proprietary
Slow innovation
Small industry



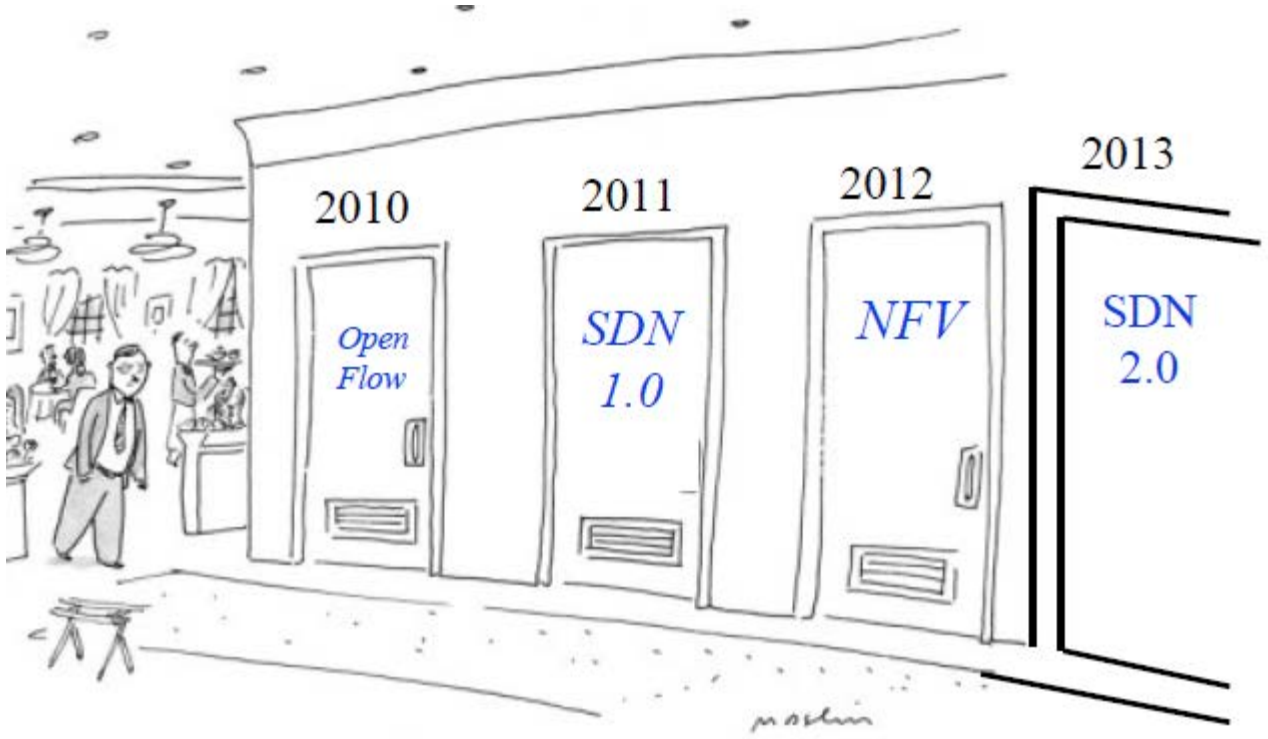
Horizontal
Open interfaces
Rapid innovation
Huge industry



Content

- **Why we need new paradigm in networking?.**
- **OpenFlow.**
- **SDN.**
- **NFV.**

OpenFlow, SDN, NFV Evolution



Source: Adopted from SDN and NFV: Facts, Extensions, and Carrier Opportunities by Prof. Raj Jain

OpenFlow

Problems with hardware-based design

- Closed Systems with no or very minimal abstractions in the network design.
- Hardware centric – usage of custom ASICs with Vendor Specific Software.
- Difficult to perform real world experiments on large scale production networks.

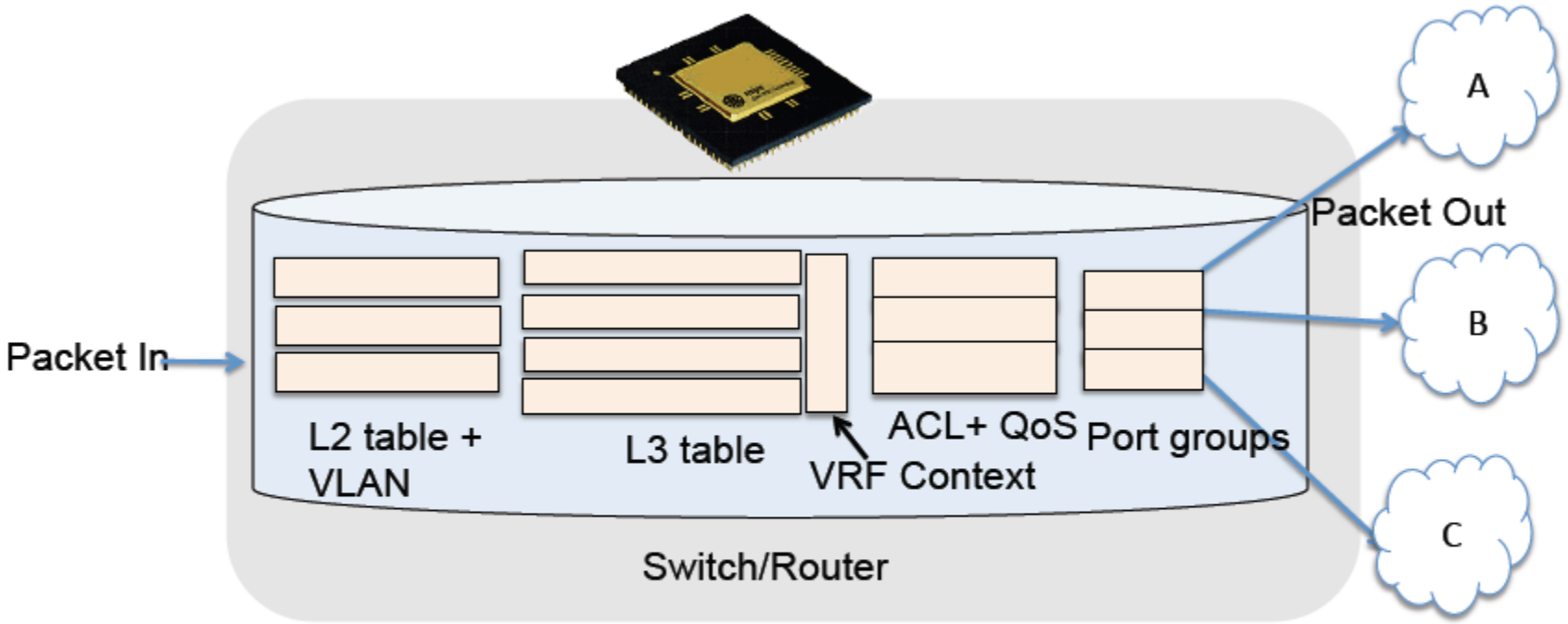
What is Open Flow

- OpenFlow is like an x86 instruction set for the network nodes.
- Provides open interface to “black box” networking node (ie. Routers, L2/L3 switch) to enable visibility and openness in network
- Separation of control plane and data plane.
  - The datapath of an OpenFlow Switch consists of a Flow Table, and an action associated with each flow entry
  - The control path consists of a controller which programs the flow entry in the flow table

Need for OpenFlow

- Layered architecture with Standard Open Interfaces
- More accessibility since software can be easily developed by more vendors
- Speed-to-market – no hardware fabrication cycles
- More flexibility with programmability and ease of customization and integration with other software applications

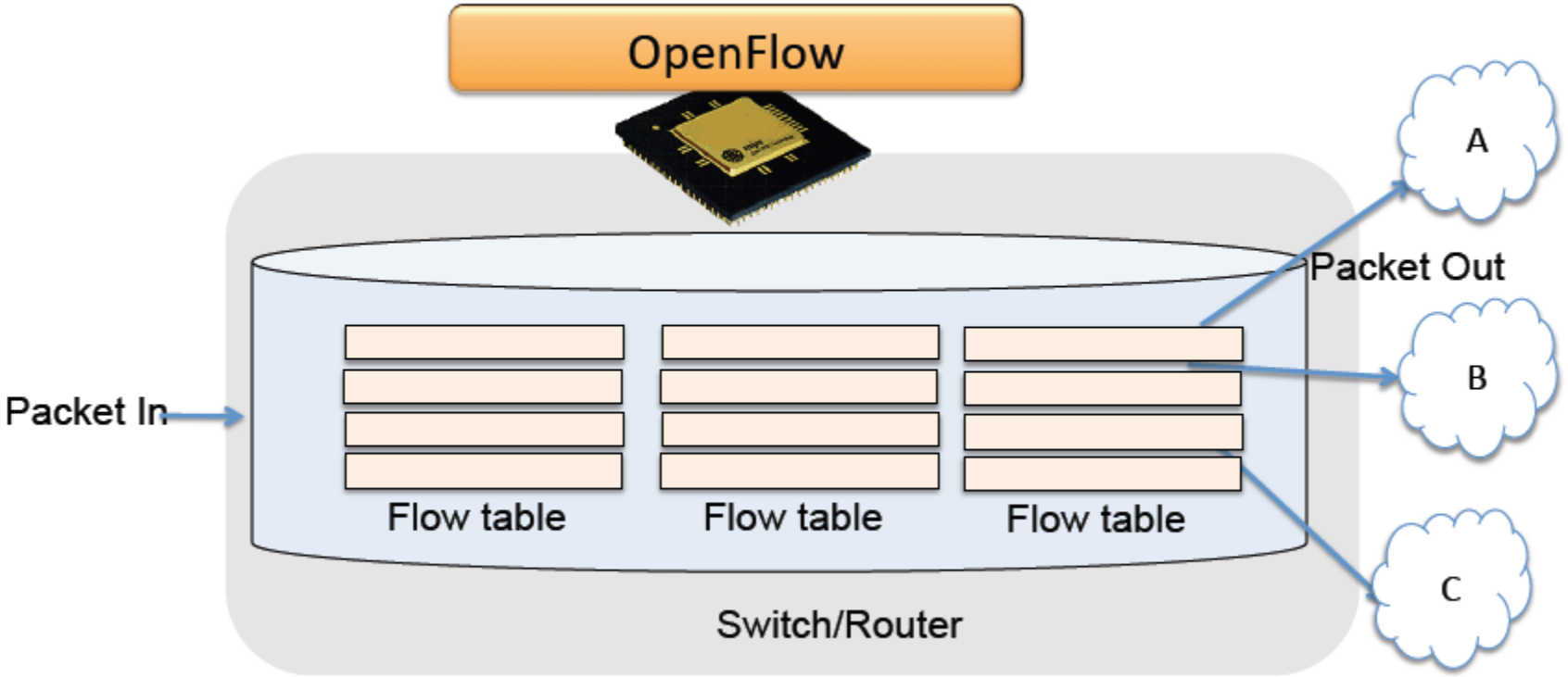
Traditional Switch Forwarding



- Fixed function
- Often expose implementation details
- Non-standard/non-existent state management APIs

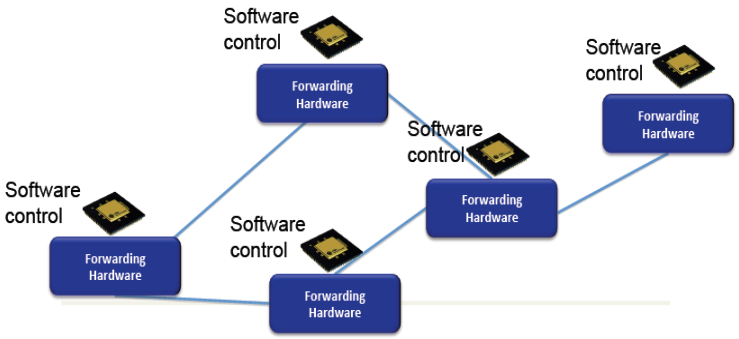
Source: Adopted from ONF11 presentation by Martin Casado

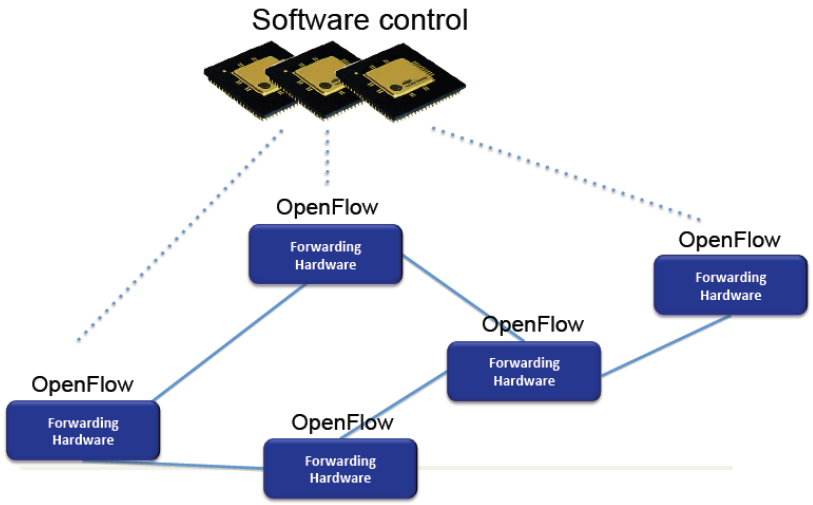
Open Flow Switch Forwarding



Source: Adopted from ONF11 presentation by Martin Casado

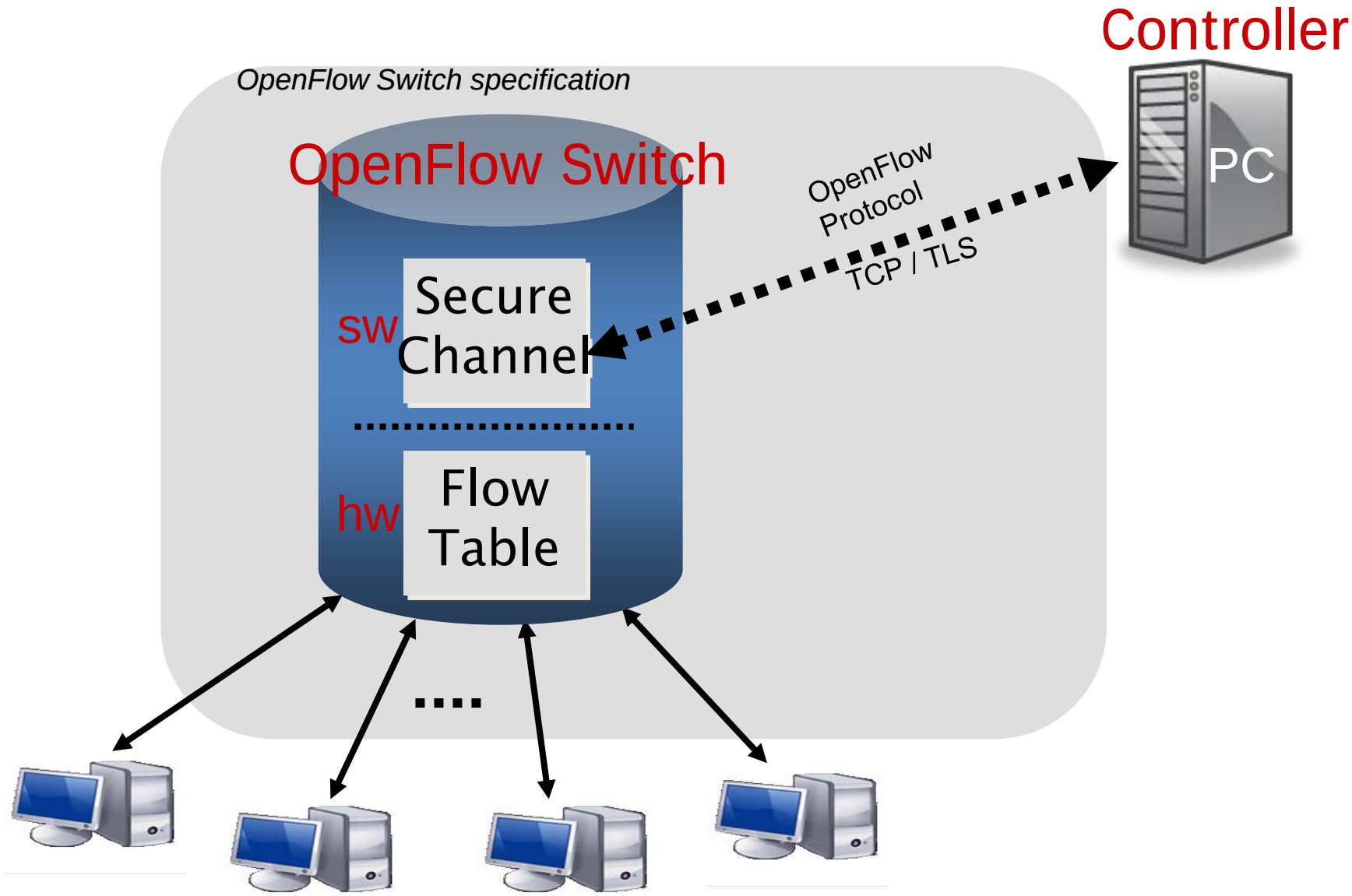
Open Flow Illustration





Source: Adopted from ONF11 presentation by Martin Casado

Components of OpenFlow Network



* Figure From OpenFlow Switch Specification

OpenFlow Controller

- Manages one or more switch via OpenFlow channels.
- Uses OpenFlow protocol to communicate with a OpenFlow aware switch.
- Acts similar to control plane of traditional switch.
- Provides a network wide abstraction for the applications on north bound.
- Responsible for programming various tables in the OpenFlow Switch.

OpenFlow Channel

- Used to exchange OpenFlow message between switch and controller.
- Switch can establish single or multiple connections to same or different controllers (auxiliary connections).
- A controller configures and manages the switch, receives events from the switch, and send packets out the switch via this interface

OpenFlow Switch

- Consists of one or more flow tables, group table and meter table.
- A single switch can be managed by one or more controllers.
- The flow tables and group table are used during the lookup or forwarding phase in order to forward the packet to appropriate port.
- Meter table is used to perform simple QOS operations like rate-limiting to complex QOS operations like DiffServ etc

Open Flow

- General Myth
  - SDN is Open Flow
- Reality
  - OpenFlow is an open API that provides a standard interface for programming the data plane switches

SDN

What is SDN?

SDN Definition

Centralization of control of the network via the

Separation of **control** logic to off-device compute, that

Enables **automation** and **orchestration** of network services via

Open **programmatic** interfaces

SDN Benefits

Efficiency: optimize existing applications, services, and infrastructure

Scale: rapidly grow existing applications and services

Innovation: create and deliver new types of applications and services and business models

Source: Adopted from SDN Central (Software-Defined Networking (SDN) Use Cases)

Need for SDN

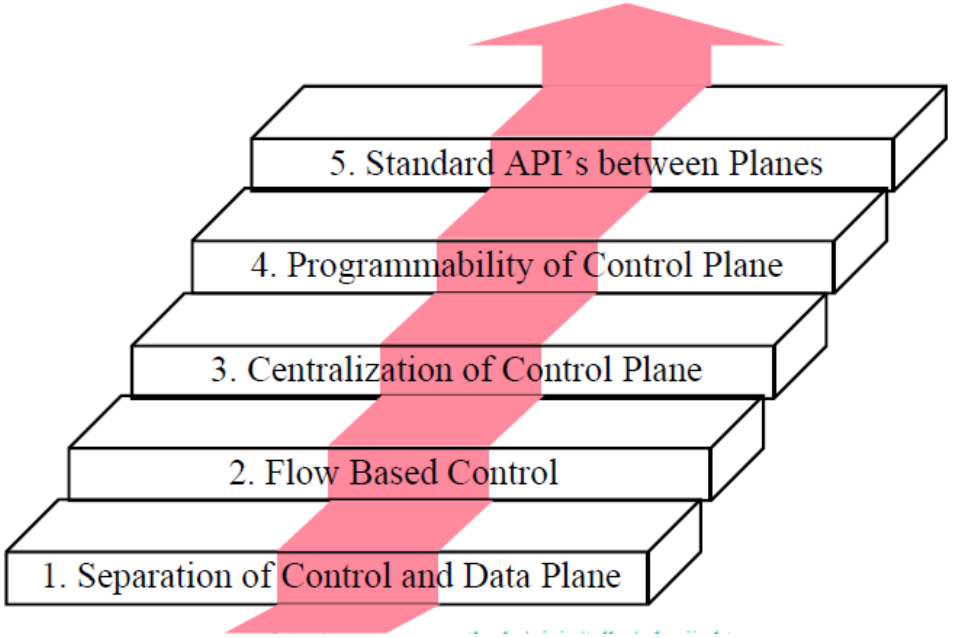
- Network Virtualization (Data Center & Cloud)– Use network resource without worrying about where it is physically located, how much it is, how it is organized, etc.
- Orchestration (Cloud) - Automated arrangement, coordination, and management of complex computer systems, middleware, and services.
- Programmable (Enterprise) - Should be able to change behavior on the fly.
- Dynamic Scaling (Cloud) - Should be able to change size, quantity
- Automation - To lower OpEx minimize manual involvement
  - Troubleshooting
  - Reduce downtime
  - Policy enforcement
  - Provisioning/Re-provisioning/Segmentation of resources

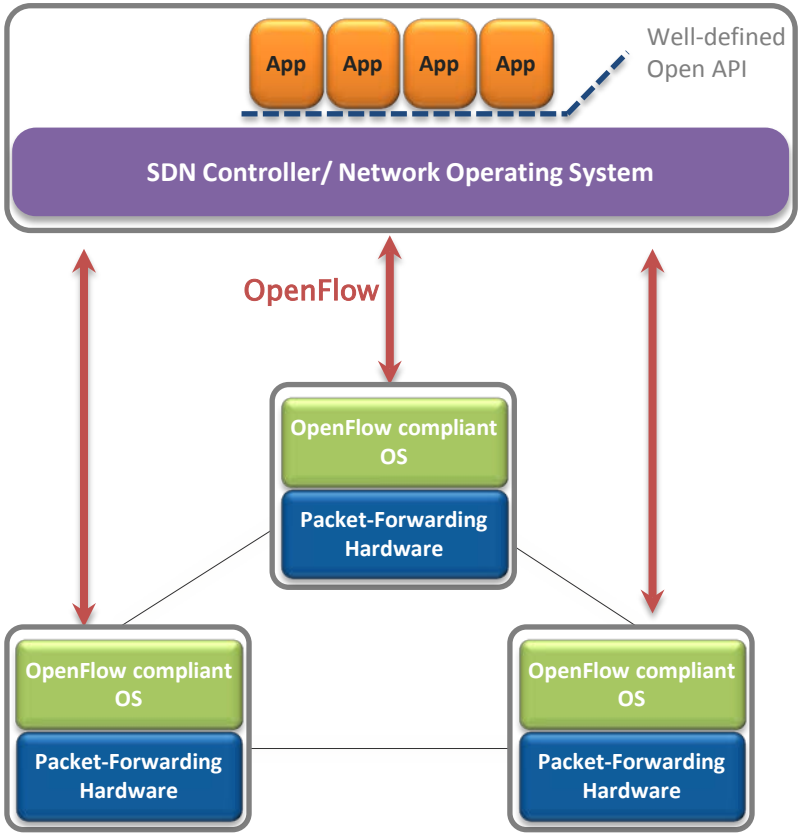
Source: Adopted from Introduction to Software Defined Software Defined Networking (SDN) Networking (SDN) by Prof. Raj Jain

Need for SDN

- Visibility - Monitor resources, connectivity.
- Performance - Optimize network device utilization
  - Traffic engineering/Bandwidth management
  - Capacity optimization
  - Load balancing
  - High utilization
- Multi-tenancy (Data Center / Cloud)- Tenants need complete control over their addresses, topology, and routing, security
- Service Integration (Enterprise)- Load balancers, firewalls, Intrusion Detection Systems (IDS), provisioned on demand and placed appropriately on the traffic path

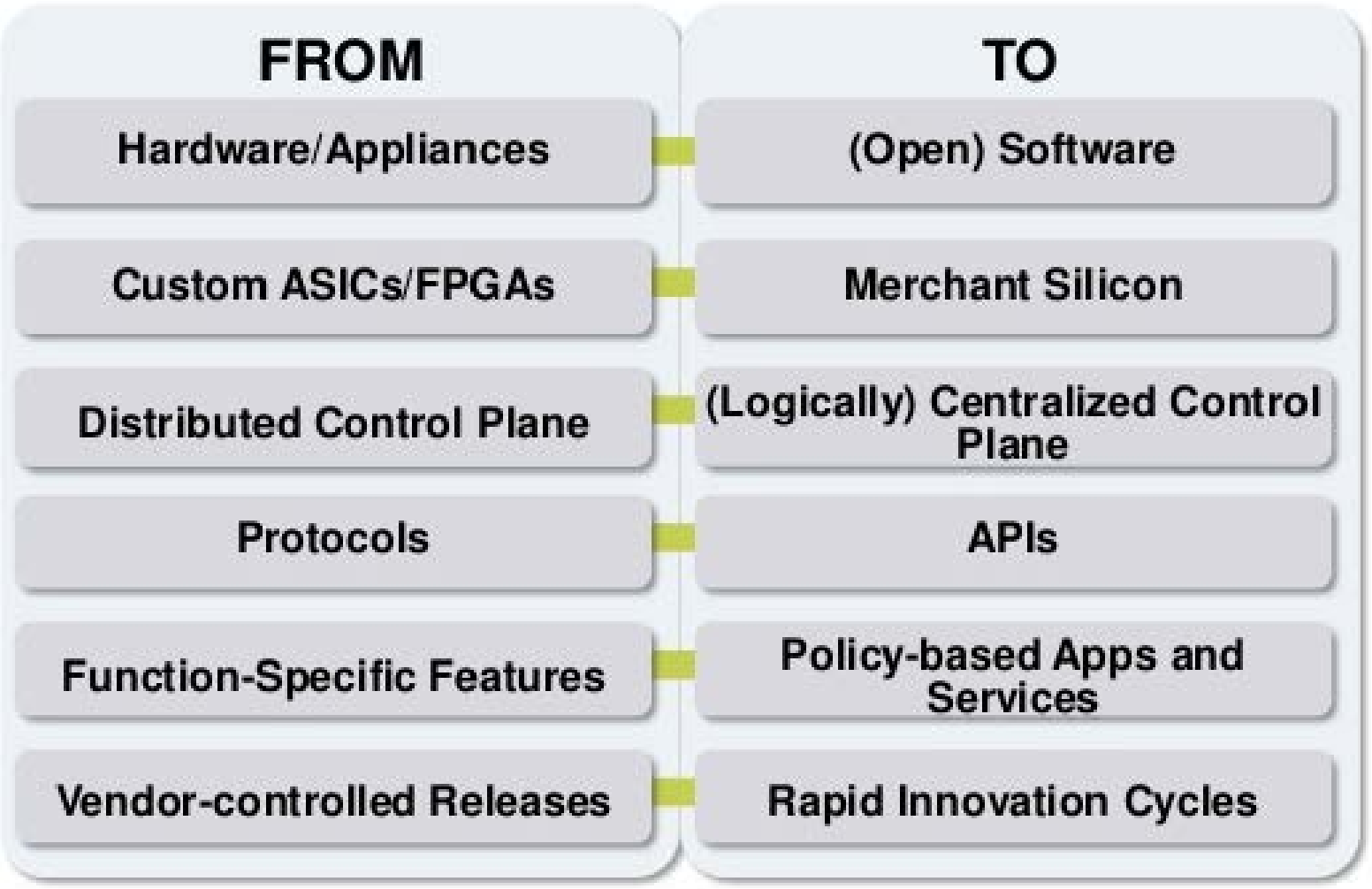
SDN Innovation & Components





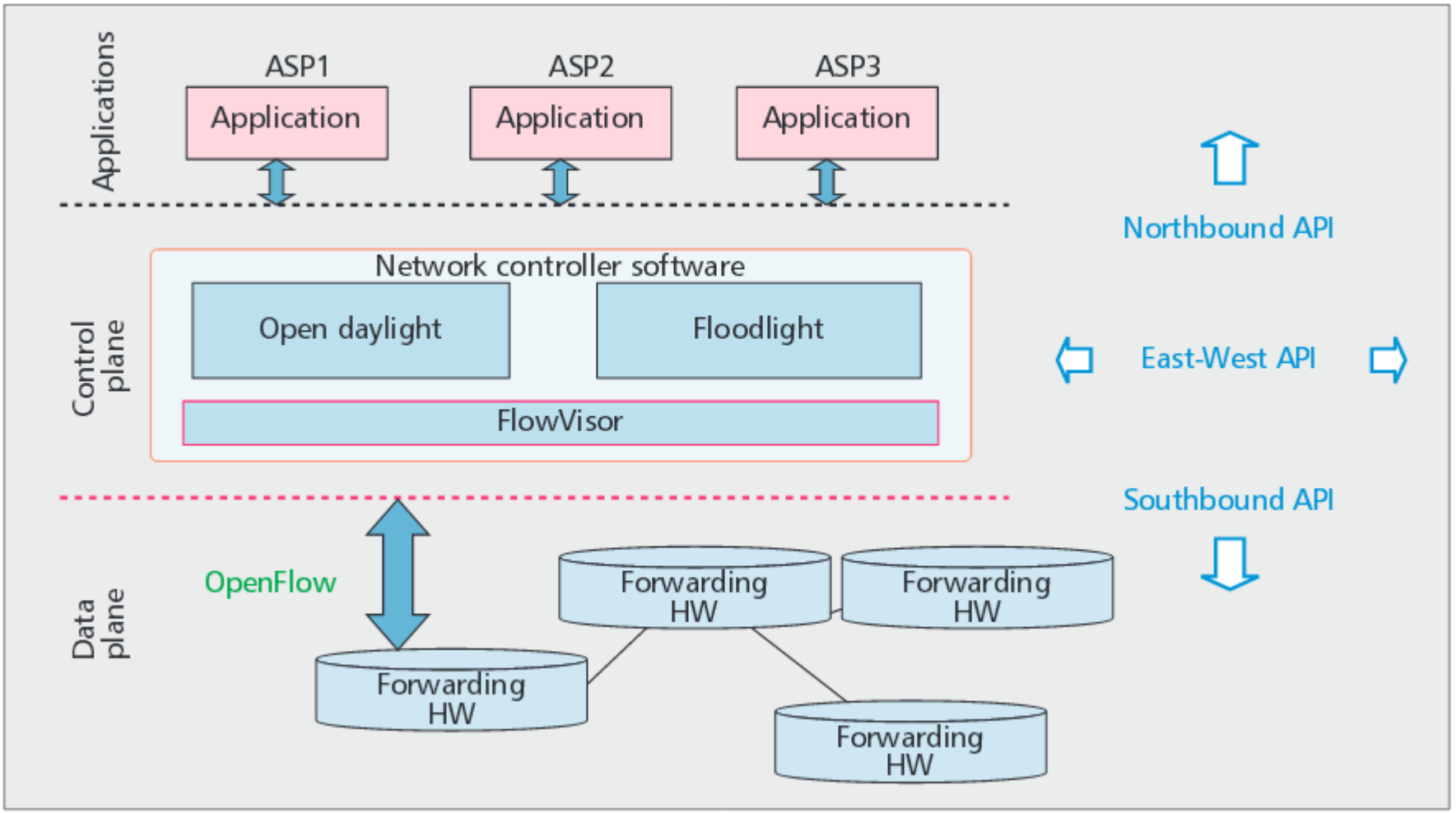
Source: Adopted from SDN Central (Software-Defined Networking (SDN) Use Cases)

SDN Approach



Source: Adapted from ONS 12 Presentation by Dan Pitt

SDN API

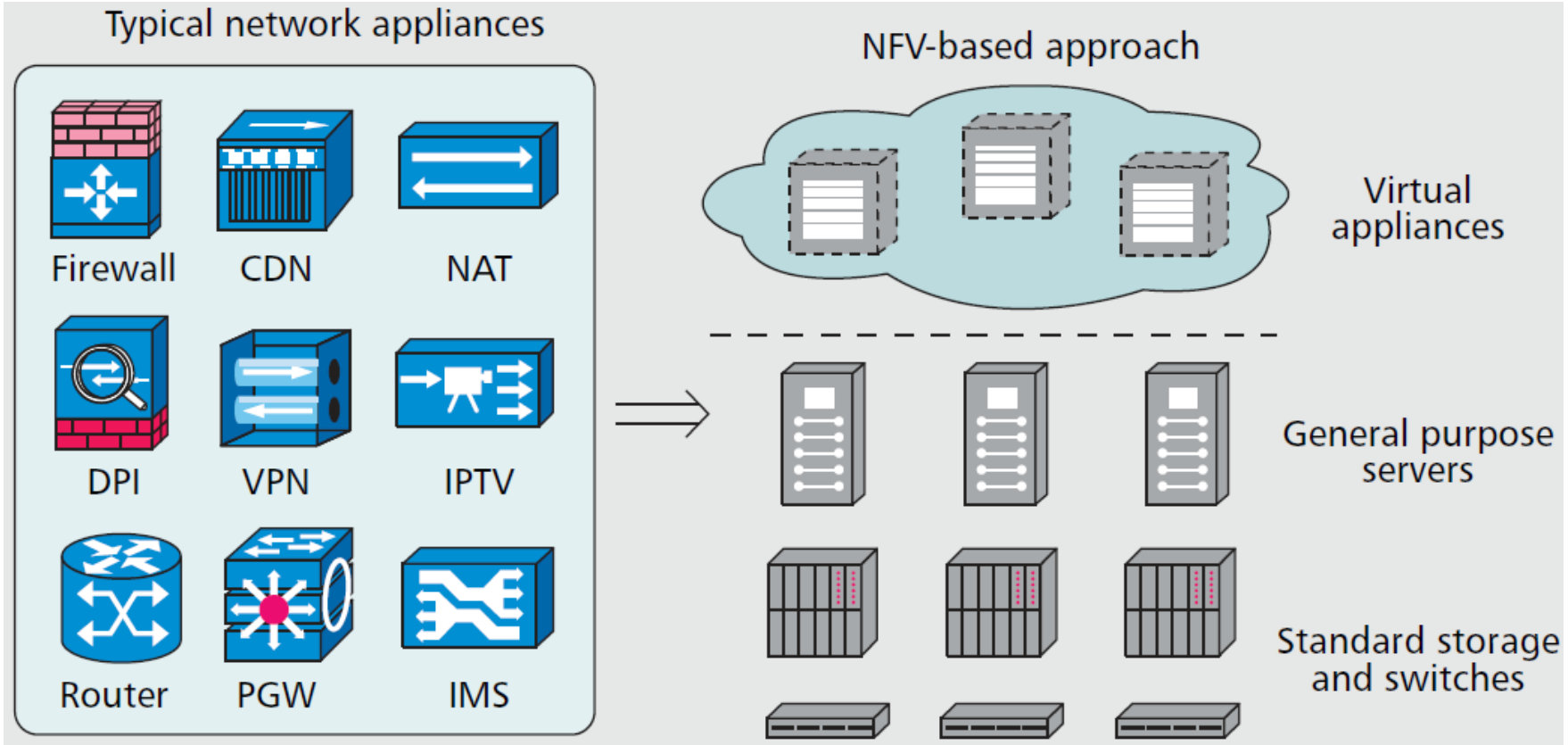

Applications
ASP1
Application
ASP2
Application
ASP3
Application
Northbound API
Control plane
Network controller software
Open daylight
Floodlight
FlowVisor
East-West API
Southbound API
Data plane
OpenFlow
Forwarding HW
Forwarding HW
Forwarding HW
Forwarding HW

NFV

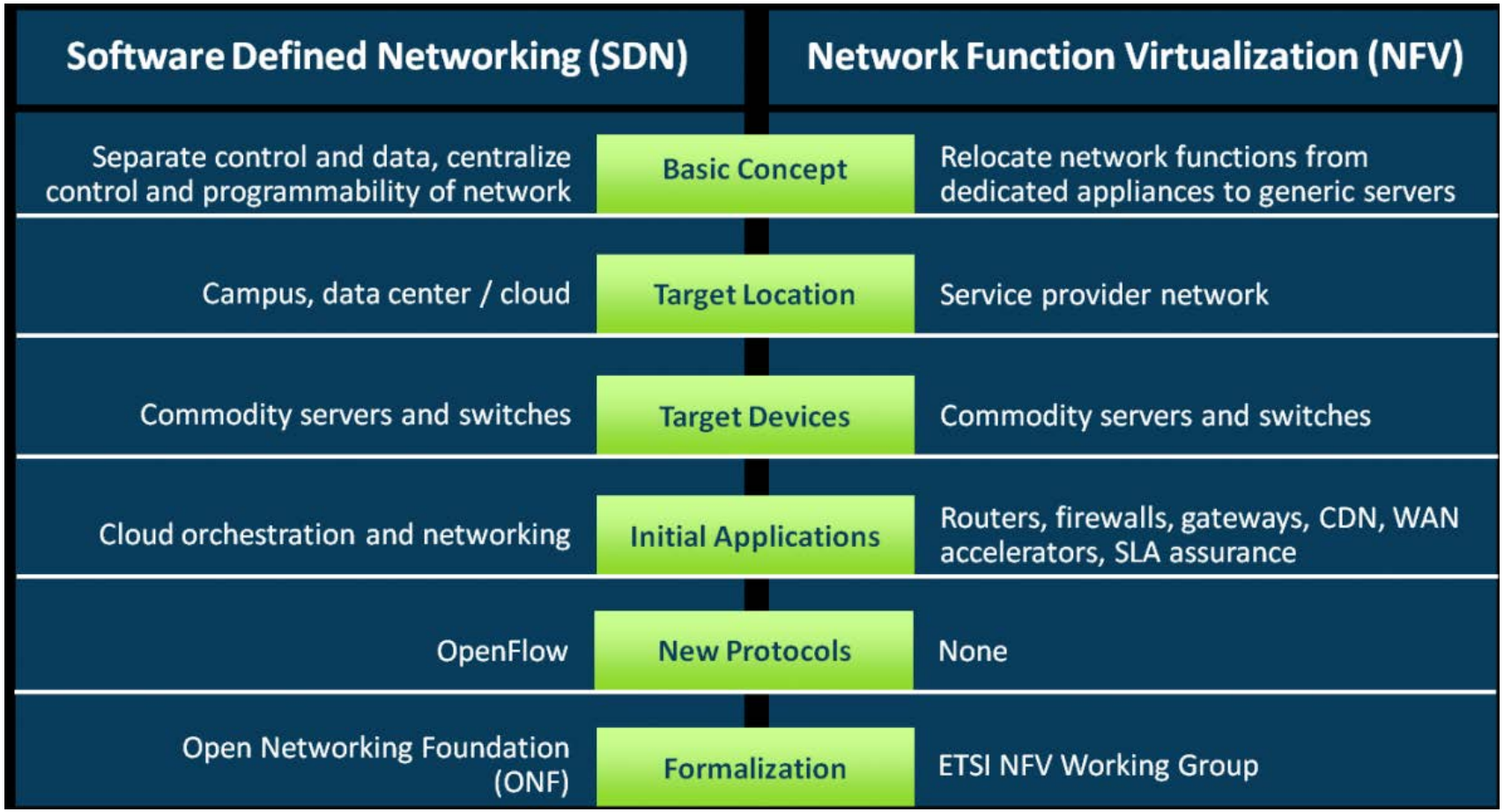
What is NFV?

- **Network Functions Virtualization** (NFV) is a network architecture concept that proposes using IT virtualization related technologies, to virtualize entire classes of network node functions into building blocks that may be connected, or chained, together to create communication services.

Hardware-based VS. NFV-based

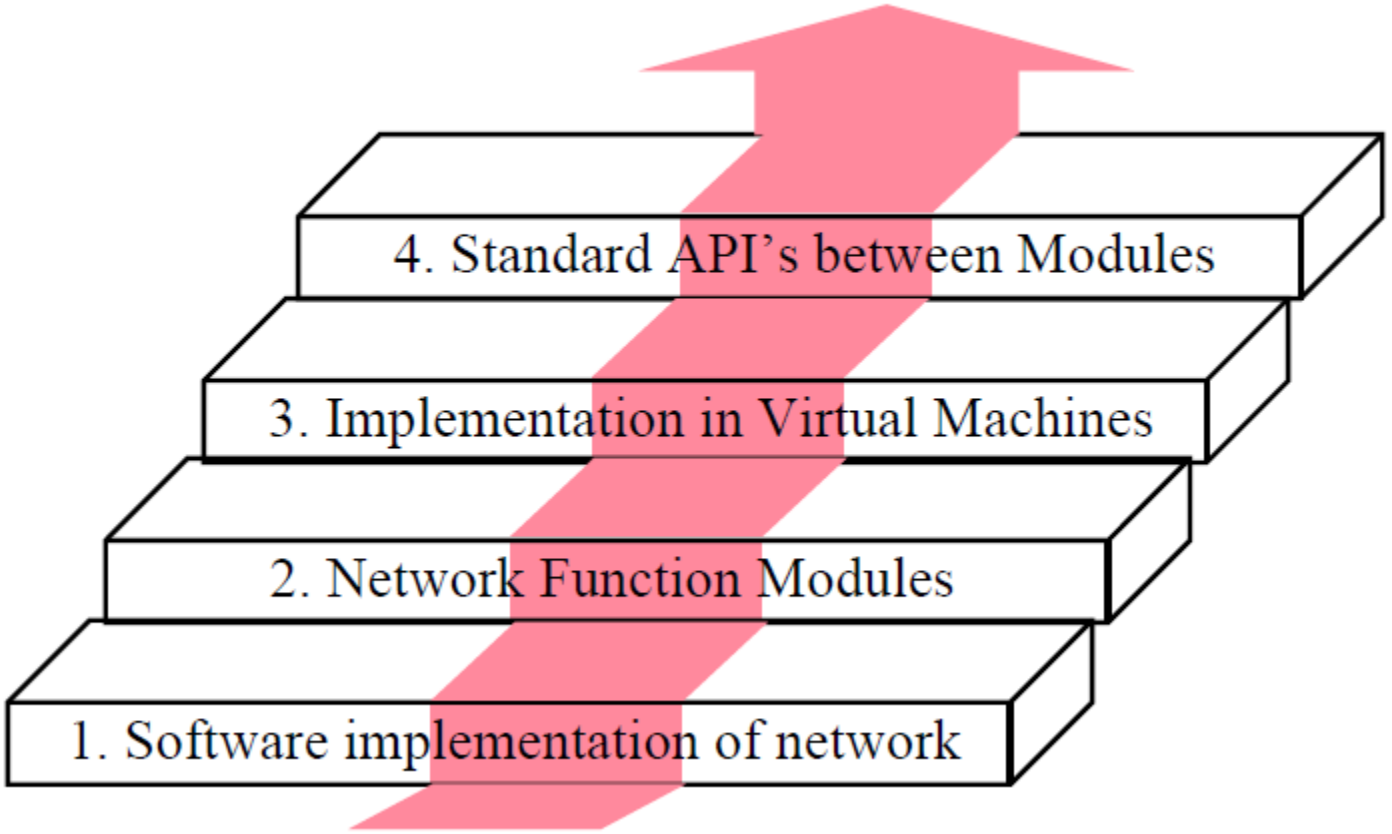


NFV vs SDN



Source: Adopted from http://www.overturenetworks.com/blog/2013/04/12/network-function-virtualization-and-software-defined-networking-whats-difference

NFV Innovations



Source: Adopted from SDN and NFV: Facts, Extensions, and Carrier Opportunities by Prof. Raj Jain

NFV Components

Network Function (NF): Functional building block with a well defined interfaces and well defined functional behavior

Virtualized Network Function (VNF): Software implementation of NF that can be deployed in a virtualized infrastructure

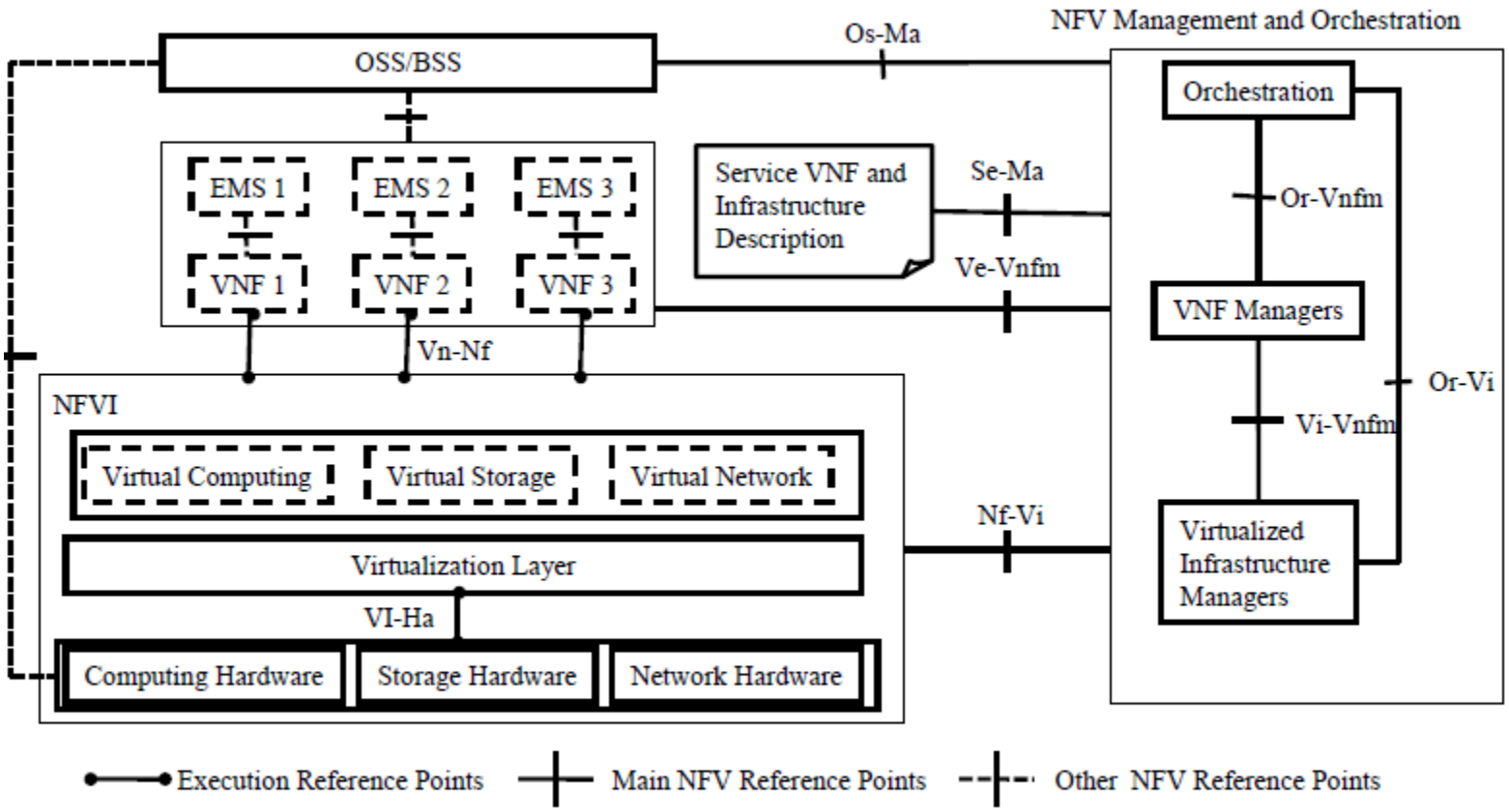
VNF Set: Connectivity between VNFs is not specified, e.g., residential gateways

VNF Forwarding Graph: Service chain when network connectivity order is important, e.g., firewall, NAT, load balancer

NFV Infrastructure (NFVI): Hardware and software required to deploy, mange and execute VNFs including computation, networking, and storage.

Source: Adopted from SDN and NFV: Facts, Extensions, and Carrier Opportunities by Prof. Raj Jain

NFV Architecture



Ref: ETSI, "Architectural Framework," Oct 2013,

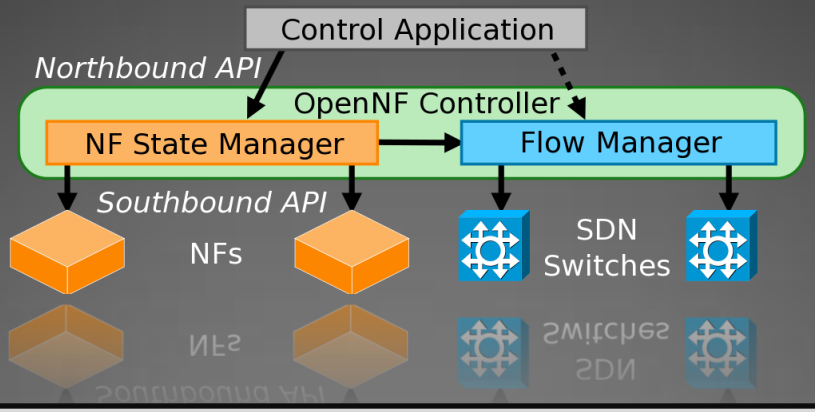
Source: Adopted from SDN and NFV: Facts, Extensions, and Carrier Opportunities by Prof. Raj Jain

Points

- NFV provides virtualization, orchestration, scaling, automation, hardware independence etc..
- NFV and SDN are complementary and independent frameworks.
- NFV doesn’t mandata control plane and Data plane separation and hence OpenFlow is not mandated in NFV.
- Lot of Network function has been demonstrated by carriers already.

References / Reading List

- B. Martinussen (Cisco), "Introduction to Software Defined Networks (SDN)," April 2013, http://www.cisco.com/web/europe/ciscoconnect2013/pdf/DC_3_SDN.pdf
- http://www.sdncentral.com/sdn-use-cases /
- Open Data Center Alliance Usage Model: Software Defined Networking Rev 1.0," http://www.opendatacenteralliance.org/docs/Software_Defined_Networking_Master_Usage_Model_Rev1.0.pdf
- SDN and NFV: Facts, Extensions, and Carrier Opportunities by Prof. Raj Jain
- OpenFlow Switch Specication V 1.4.
- Software Defined Software Defined Networking (SDN) Networking (SDN) by Prof. Raj Jain
- ETSI, "NFV - Update White Paper," Oct 2013, http://portal.etsi.org/NFV/NFV_White_Paper2.pdf
- ETSI, "Architectural Framework," Oct 2013
- ETSI, "NFV Use Cases,"
- Intel, "Open simplified Networking Based on SDN and NFV," 2013, 7 pp., http://www.intel.com/content/dam/www/public/us/en/documents/whitepapers/sdn-part-1-secured.pdf
- SDN Central (Software-Defined Networking (SDN) Use Cases)



OpenNF: Enabling Innovation in Network Function Control

WISCONSIN
UNIVERSITY OF WISCONSIN–MADISON

Aaron Gember-Jacobson, Chaithan Prakash, Raajay Viswanathan, Robert Grandl, Junaid Khalid, Sourav Das, Aditya Akella

Network functions (NFs)

- Perform sophisticated *stateful* actions on packets/flows

WAN optimizer

Caching proxy

Intrusion detection system (IDS)

NF trends

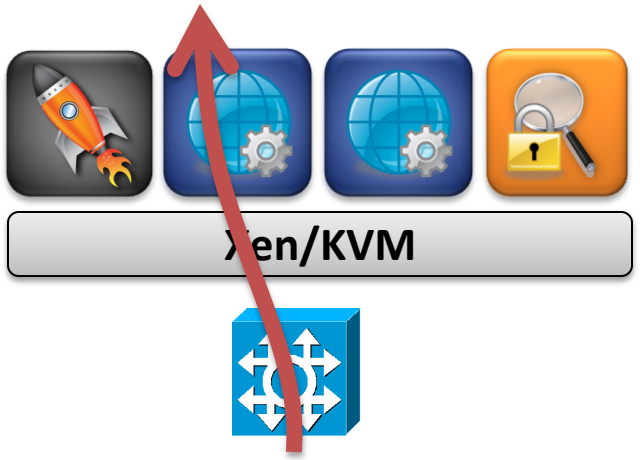
- NFV → dynamically allocate NF instances

NF trends

- NFV → dynamically allocate NF instances

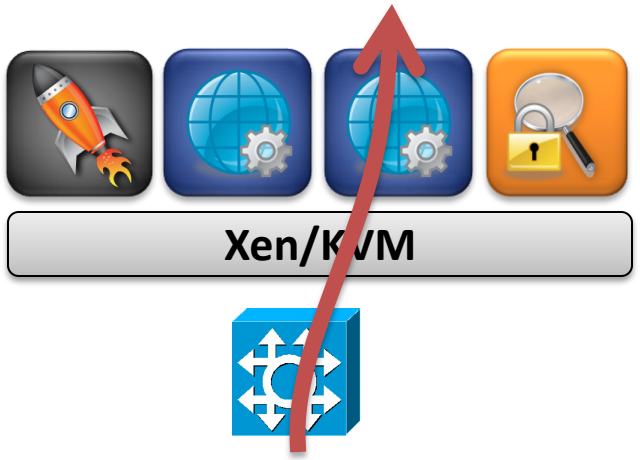
NF trends

- NFV → dynamically allocate NF instances
- SDN → dynamically reroute flows



NF trends

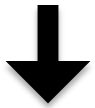
- NFV → dynamically allocate NF instances
- SDN → dynamically reroute flows



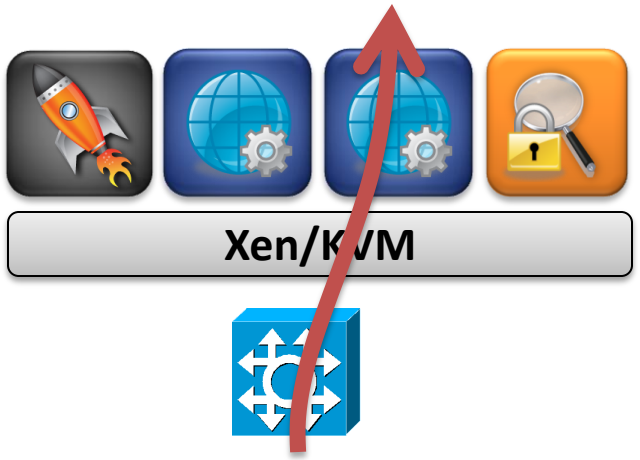
NF trends

- NFV → dynamically allocate NF instances

- SDN → dynamically reroute flows



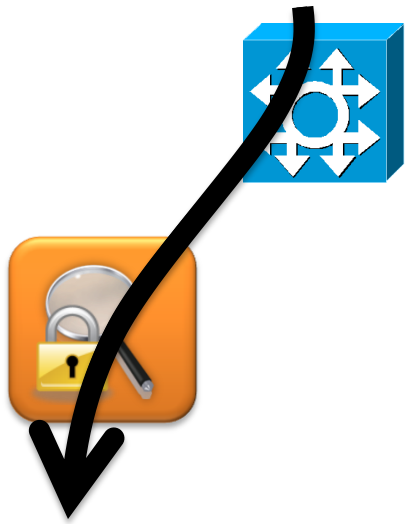
Dynamic reallocation of packet processing



Example: elastic NF scaling

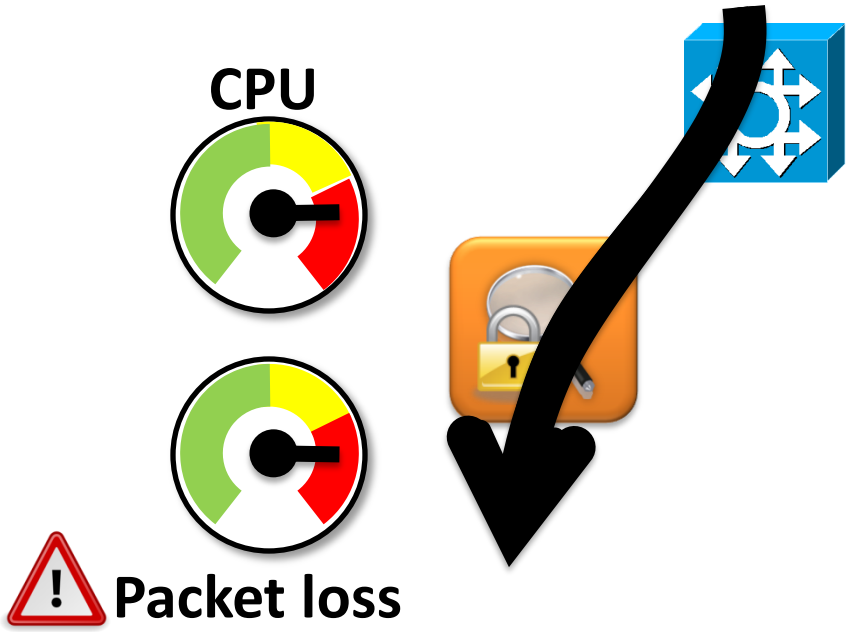
Example: elastic NF scaling

1. Satisfy performance SLAs



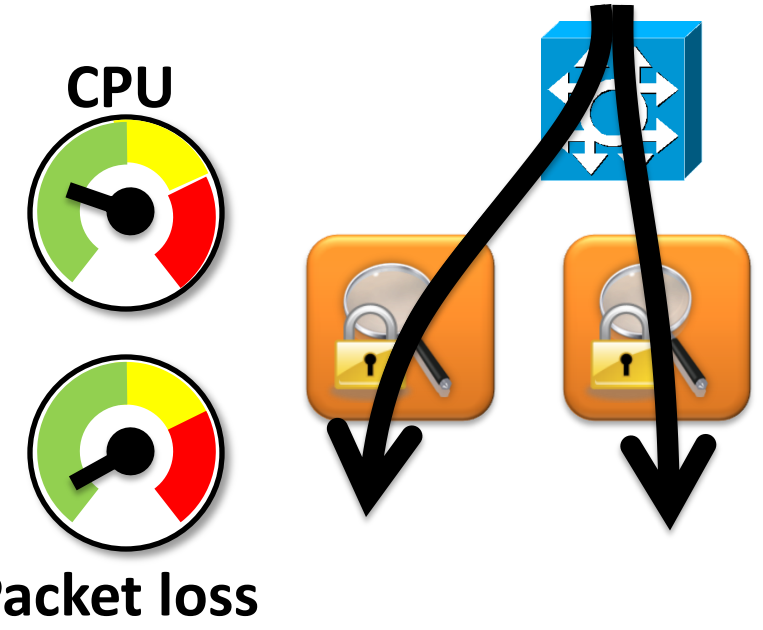
Example: elastic NF scaling

1. Satisfy performance SLAs



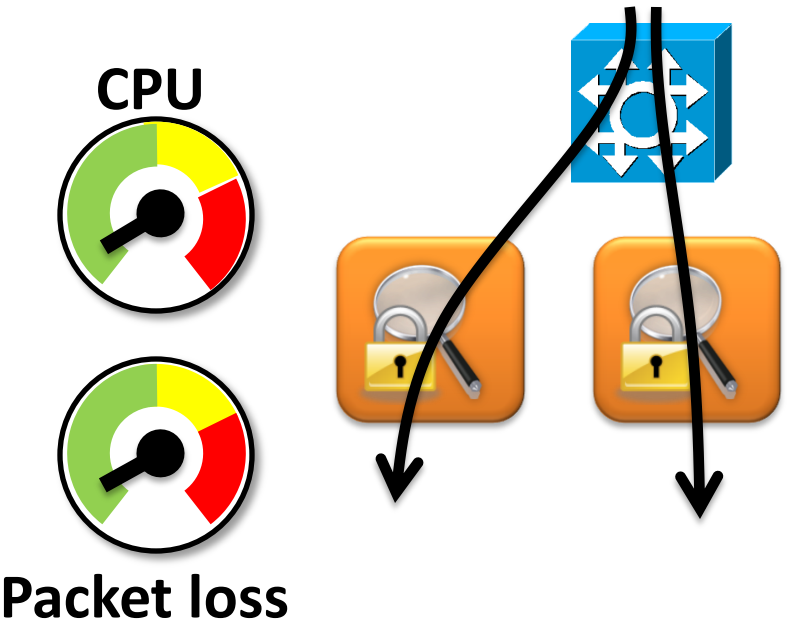
Example: elastic NF scaling

1. Satisfy performance SLAs



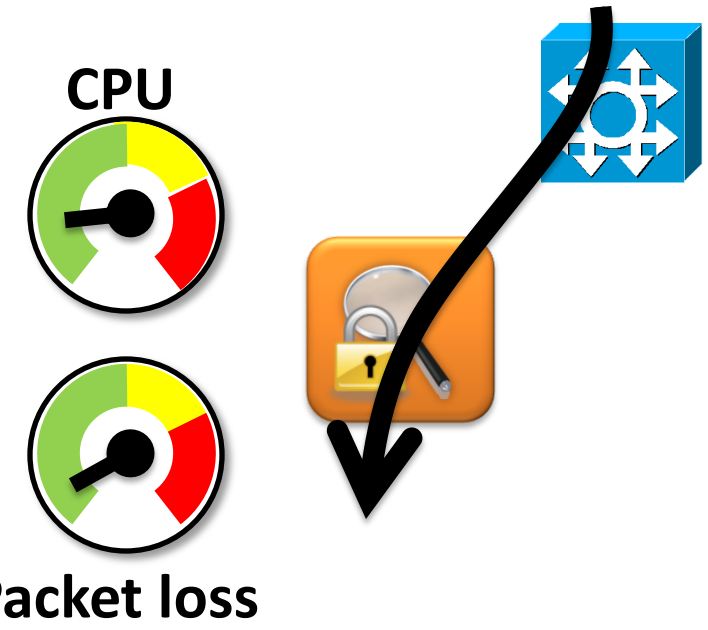
Example: elastic NF scaling

1. Satisfy performance SLAs
2. Minimize operating costs



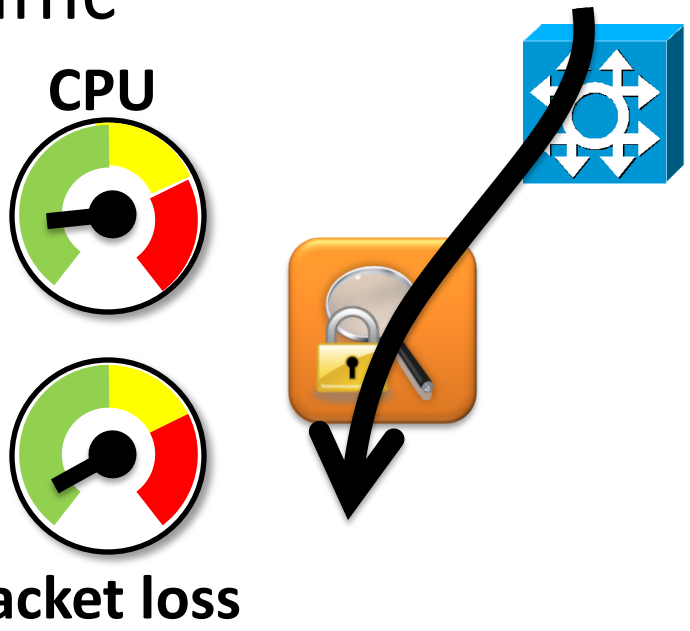
Example: elastic NF scaling

1. Satisfy performance SLAs
2. Minimize operating costs



Example: elastic NF scaling

1. Satisfy performance SLAs
2. Minimize operating costs
3. Accurately monitor traffic



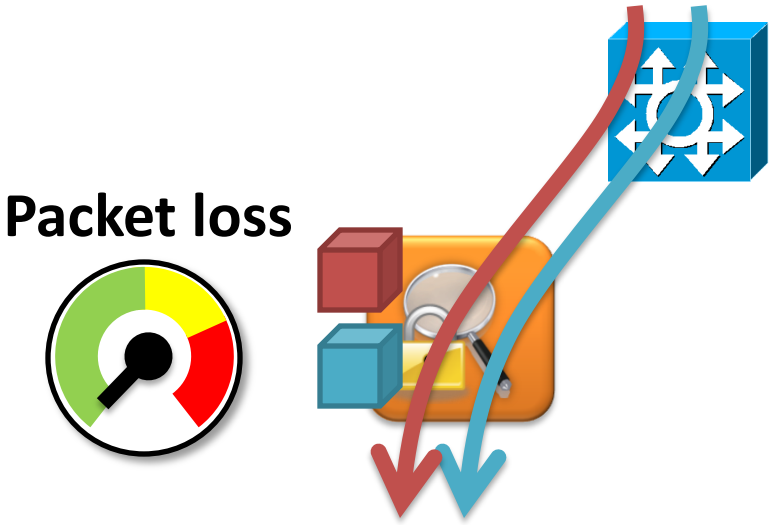
Problem: NFV+SDN is insufficient

To **simultaneously**…

1. Satisfy performance SLAs
2. Minimize operating costs
3. Accurately monitor traffic

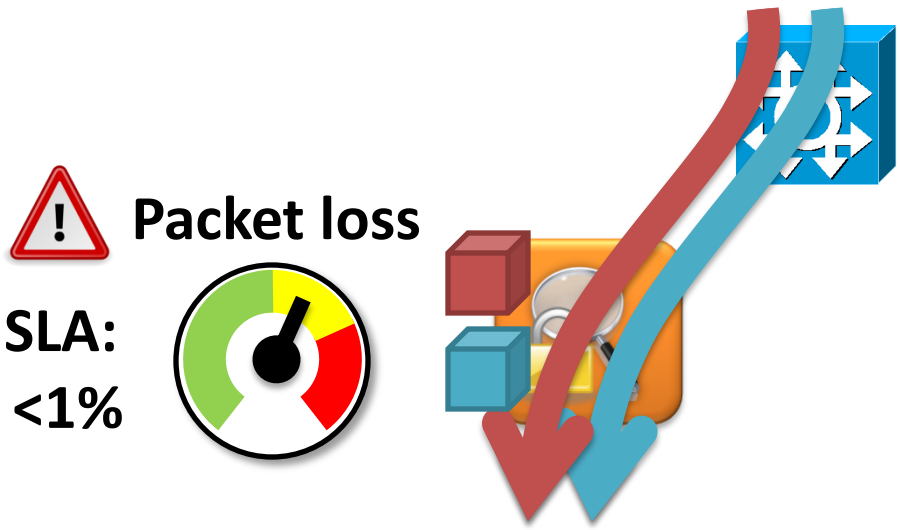
Cannot effectively implement
new services or abstractions!

Why NFV + SDN falls short



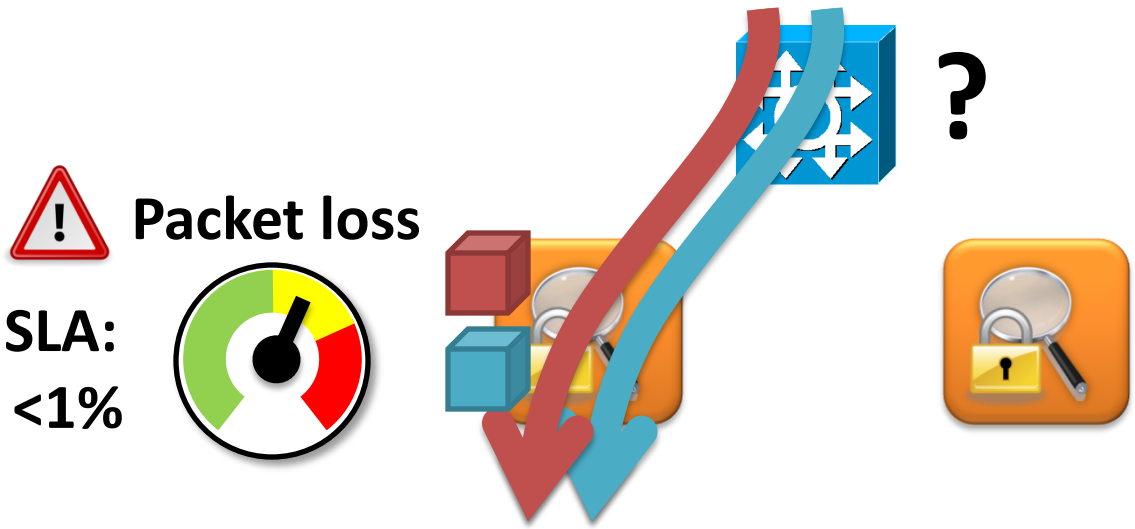
1. SLAs 2. Cost 3. Accuracy

Why NFV + SDN falls short



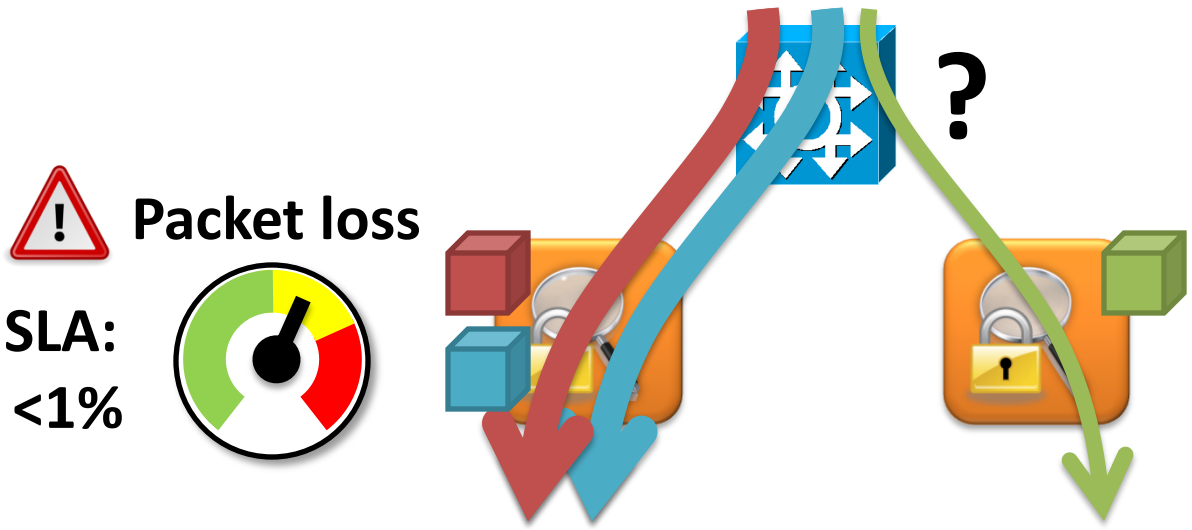
1. SLAs 2. Cost 3. Accuracy

Why NFV + SDN falls short



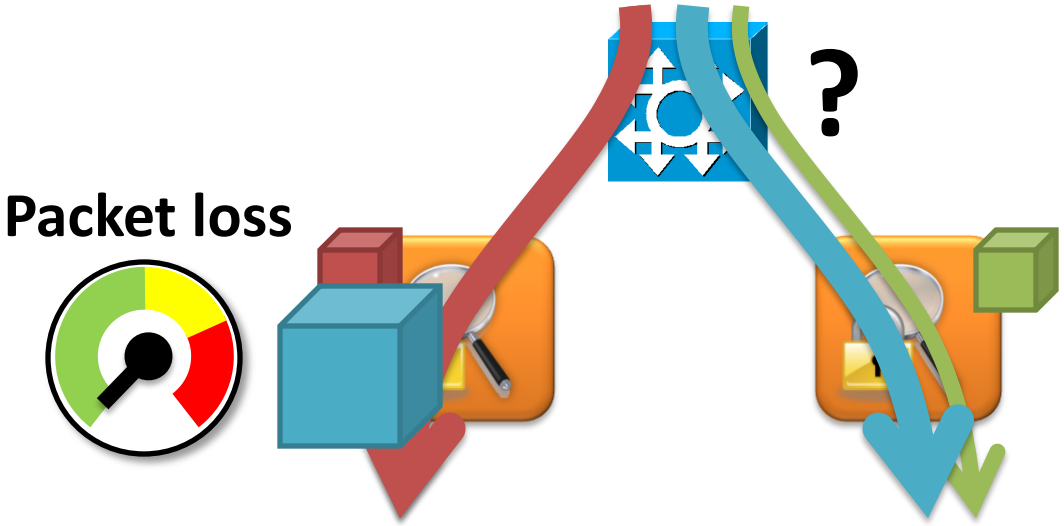
1. SLAs 2. Cost 3. Accuracy

Why NFV + SDN falls short



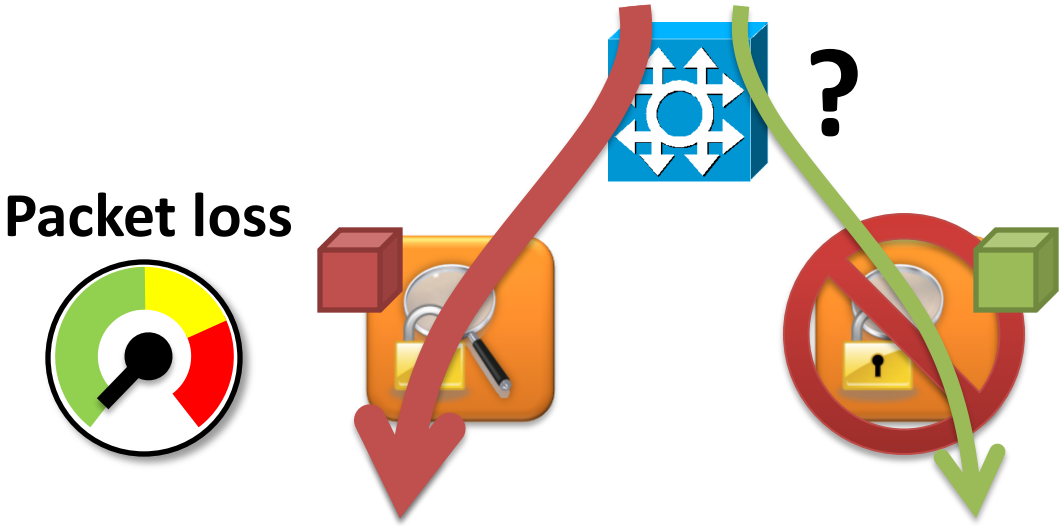
	1. SLAs	2. Cost	3. Accuracy
Reroute new flows *[Stratos - arXiv:1305.0209]*	✗	✓	?

Why NFV + SDN falls short



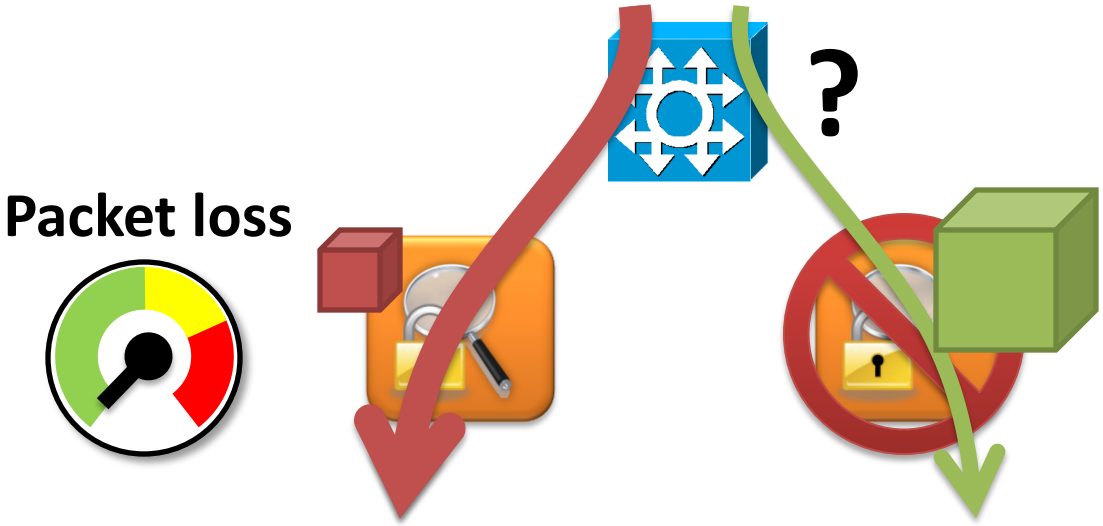
	1. SLAs	2. Cost	3. Accuracy
Reroute new flows *[Stratos - arXiv:1305.0209]*	✗	✓	?
Reroute existing flows *[SIMPLE - SIGCOMM '13]*	✓	✓	✗

Why NFV + SDN falls short



	1. SLAs	2. Cost	3. Accuracy
Reroute new flows *[Stratos - arXiv:1305.0209]*	✗	✓	?
Reroute existing flows *[SIMPLE - SIGCOMM '13]*	✓	✓	✗

Why NFV + SDN falls short



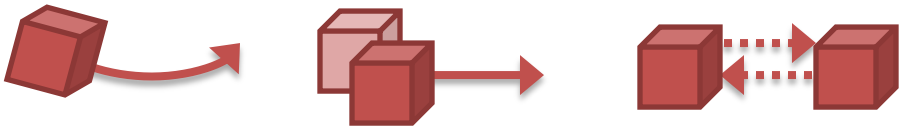
	1. SLAs	2. Cost	3. Accuracy
Reroute new flows *[Stratos - arXiv:1305.0209]*	✗	✓	?
Reroute existing flows *[SIMPLE - SIGCOMM '13]*	✓	✓	✗

Why NFV + SDN falls short

	1. SLAs	2. Cost	3. Accuracy
Reroute new flows *[Stratos - arXiv:1305.0209]*	✗	✓	?
Reroute existing flows *[SIMPLE - SIGCOMM '13]*	✓	✓	✗
Wait for flows to die *[Stratos - arXiv:1305.0209]*	✓	✗	✓

SLAs + cost + accuracy: What do we need?

- Quickly move, copy, or share internal NF state alongside updates to network forwarding state



- Guarantees: loss-free, order-preserving, …

Also applies to other scenarios

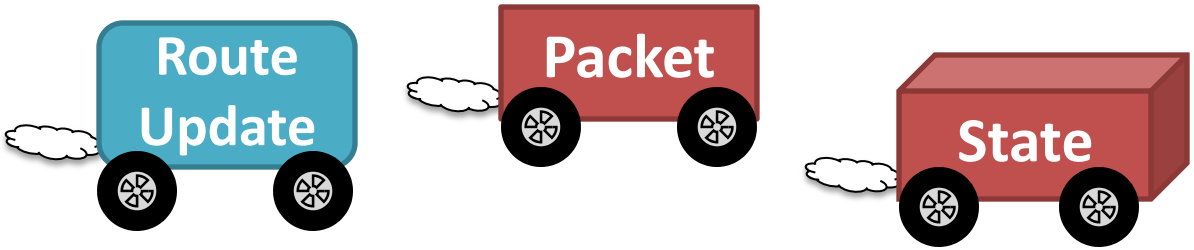
Outline

- Motivation and requirements
- Challenges
- OpenNF architecture
  - State export/import
  - State operations
  - Guarantees
- Evaluation

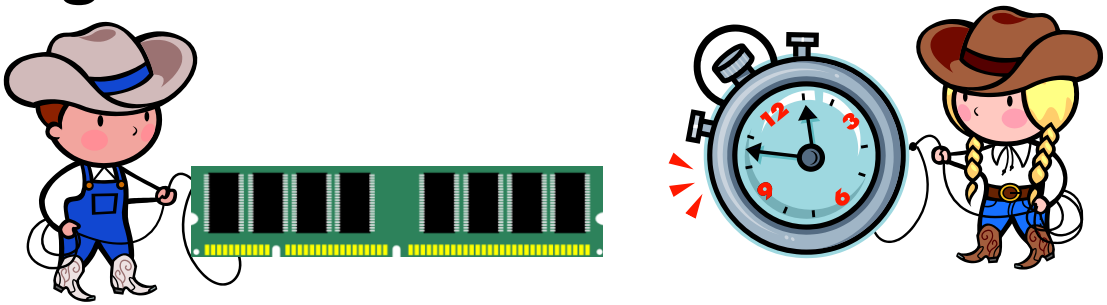
Challenges

1. Supporting many NFs with minimal changes

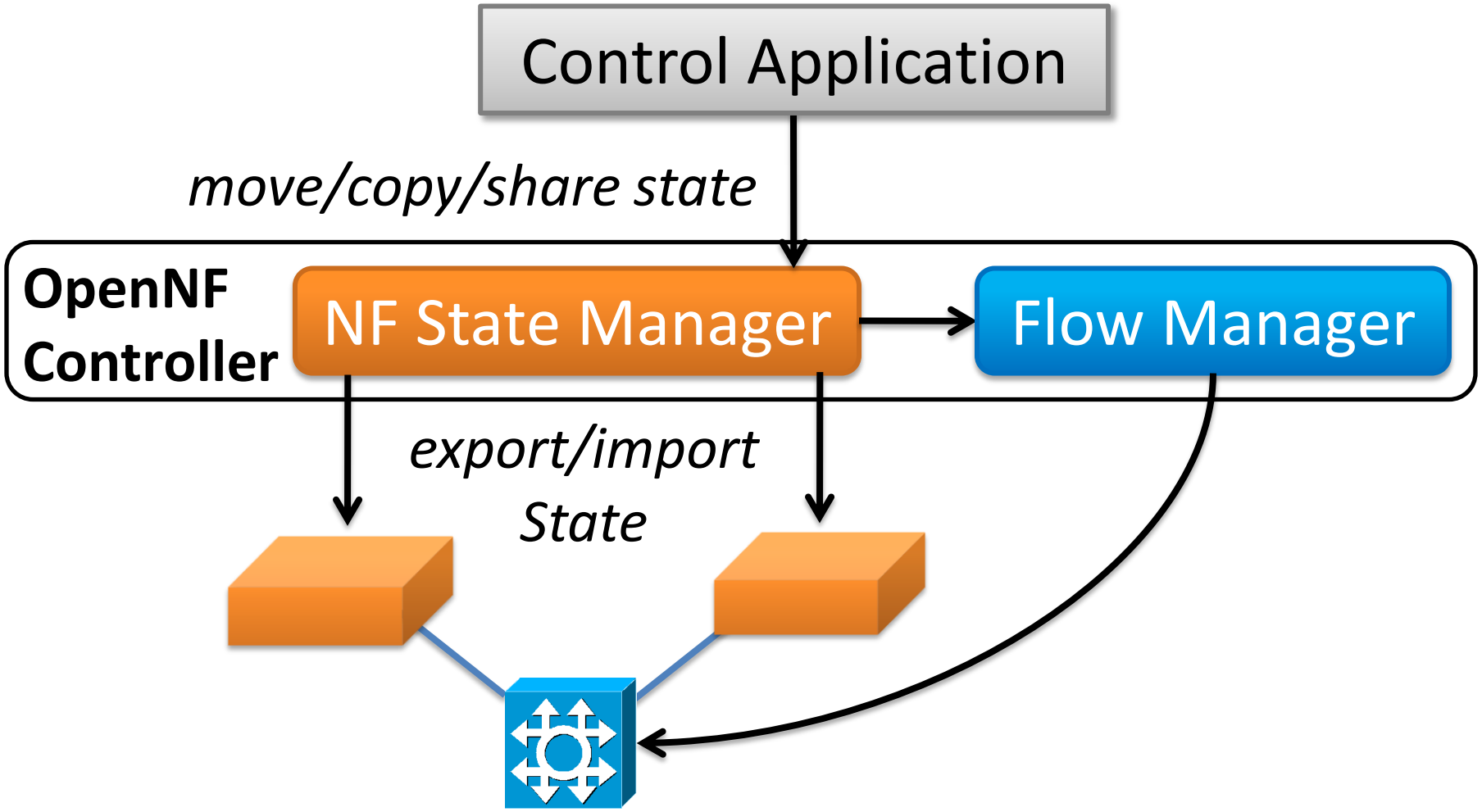
2. Dealing with race conditions



3. Bounding overhead

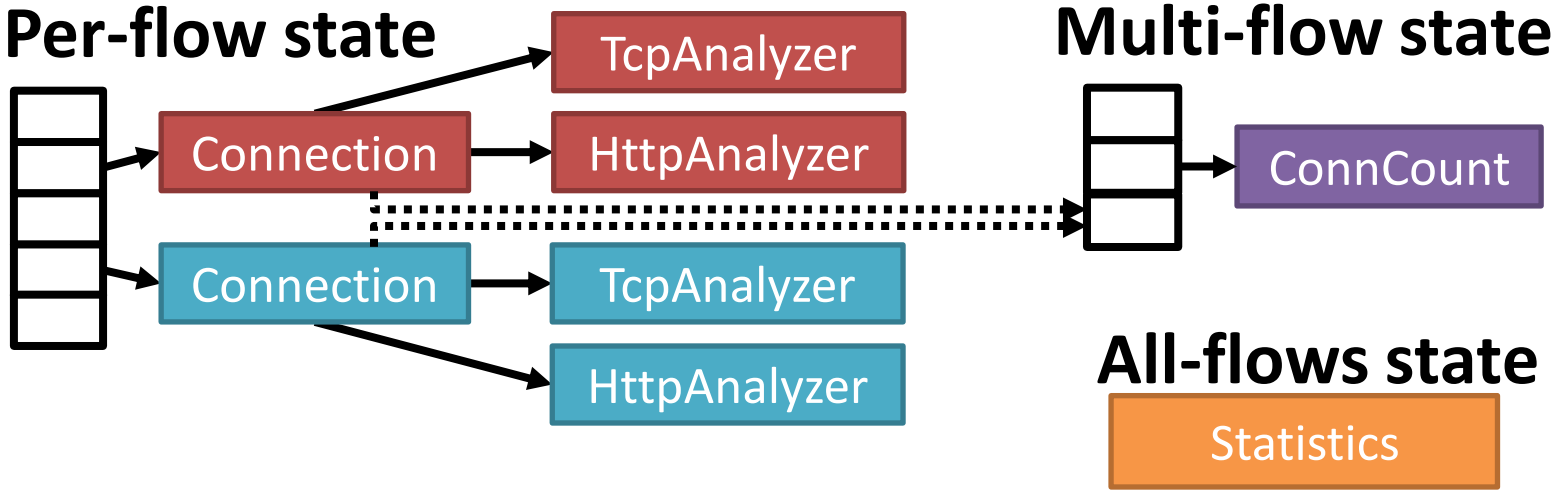


OpenNF overview



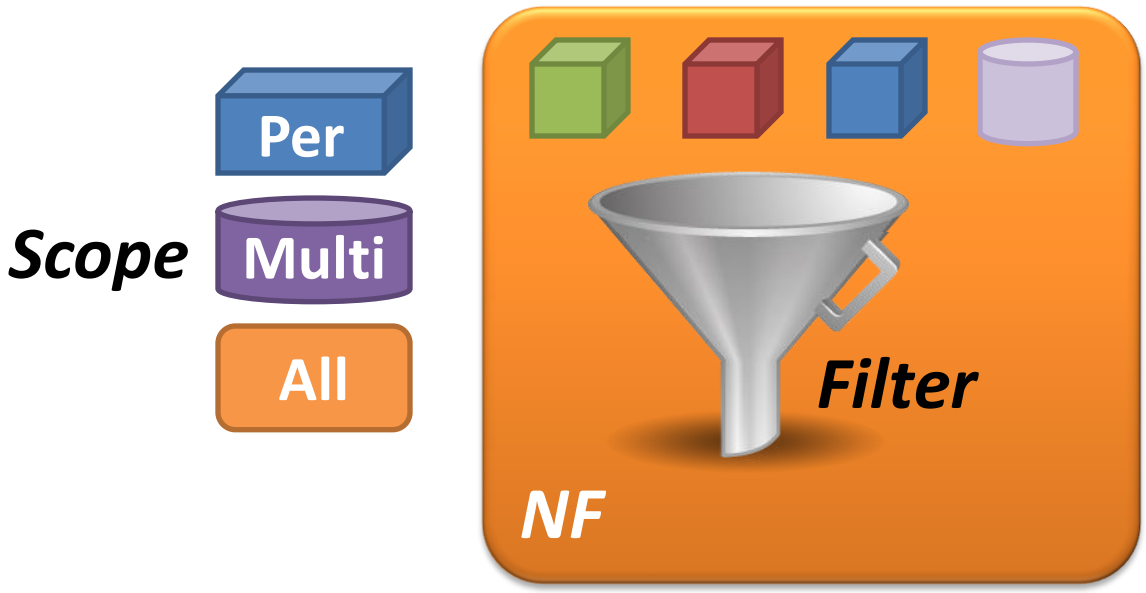
NF state taxonomy

State created or updated by an NF applies to either a **single flow** or a **collection of flows**



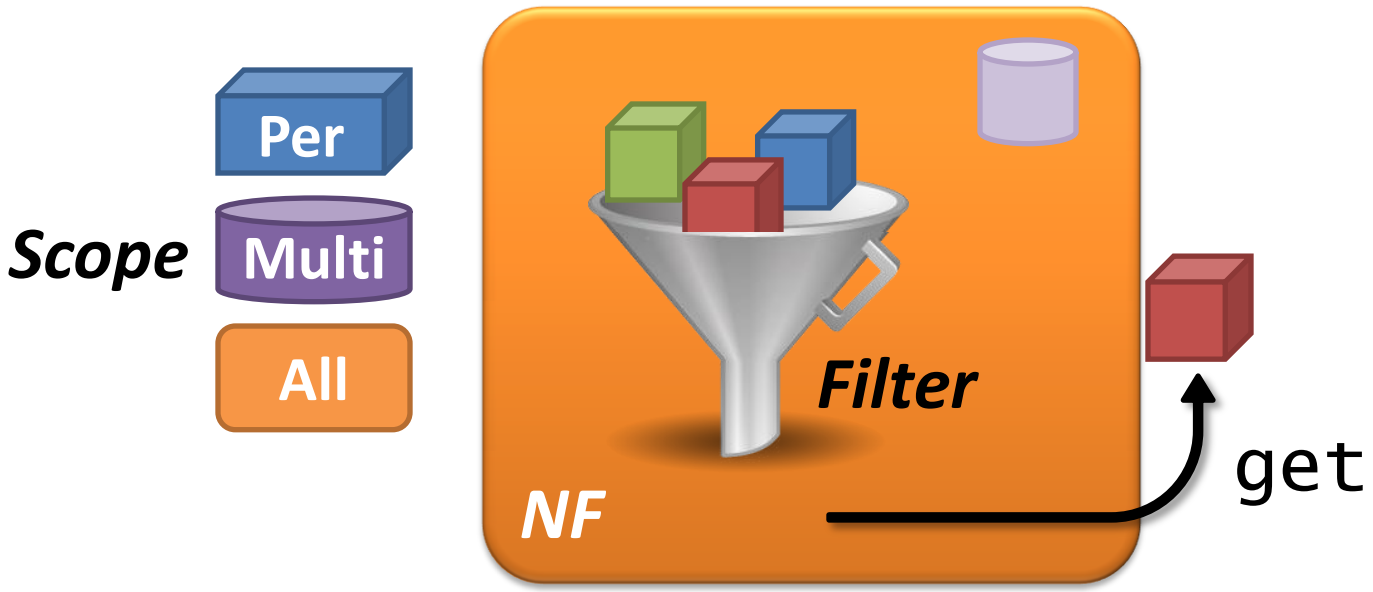
NF API: export/import state

- Functions: `get`, `put`, `delete`



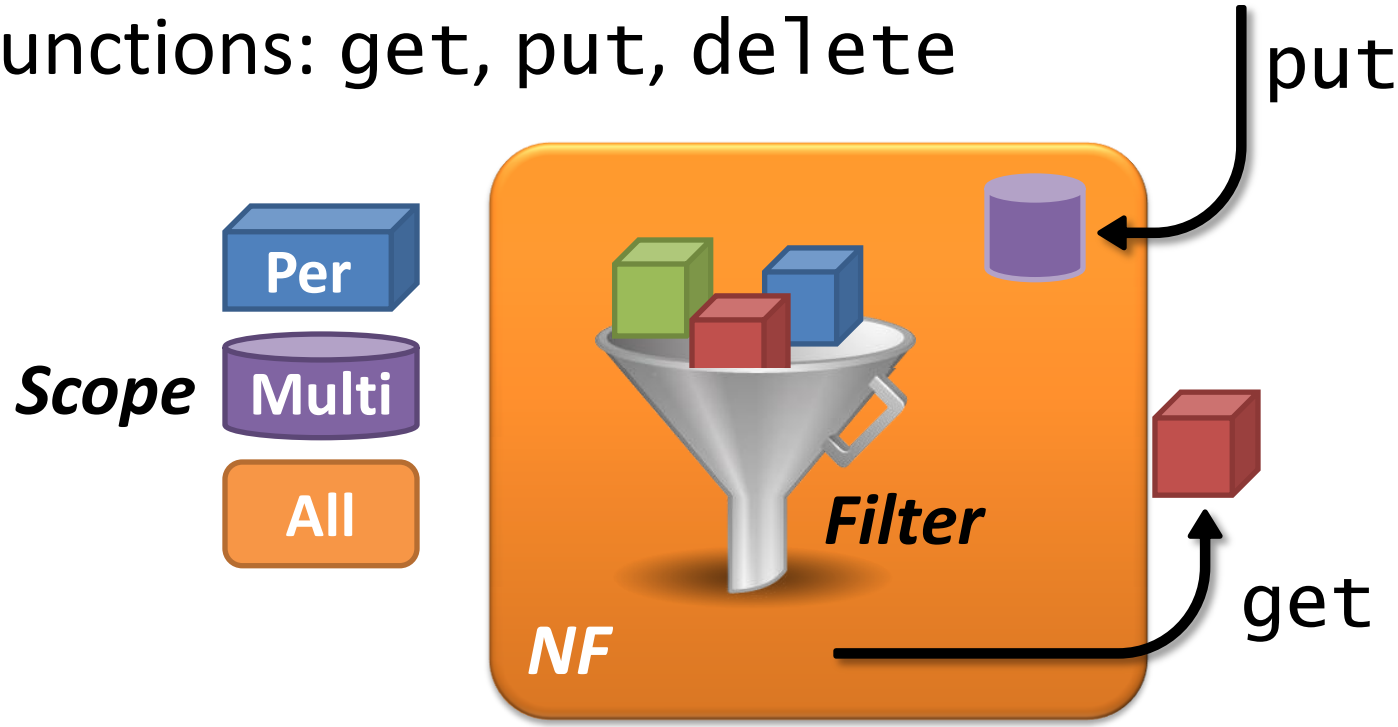
NF API: export/import state

- Functions: `get`, `put`, `delete`



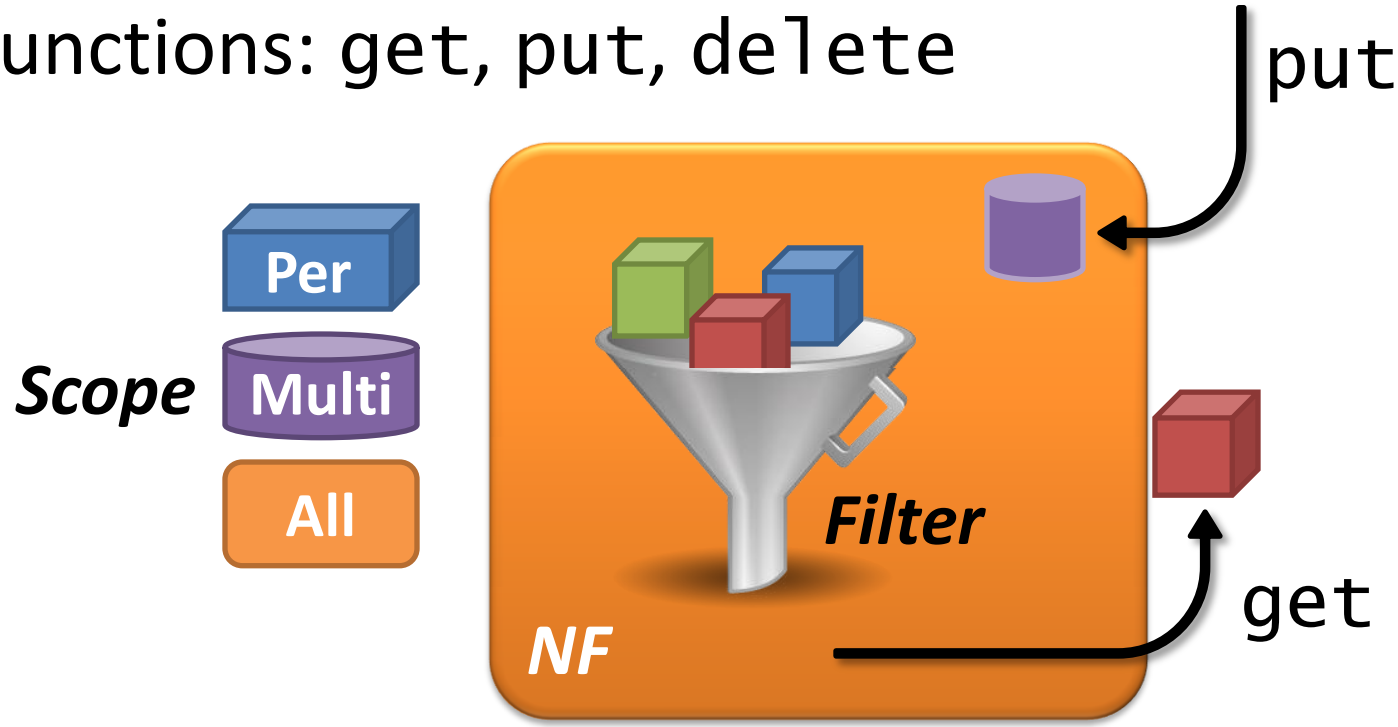
NF API: export/import state

- Functions: `get`, `put`, `delete`



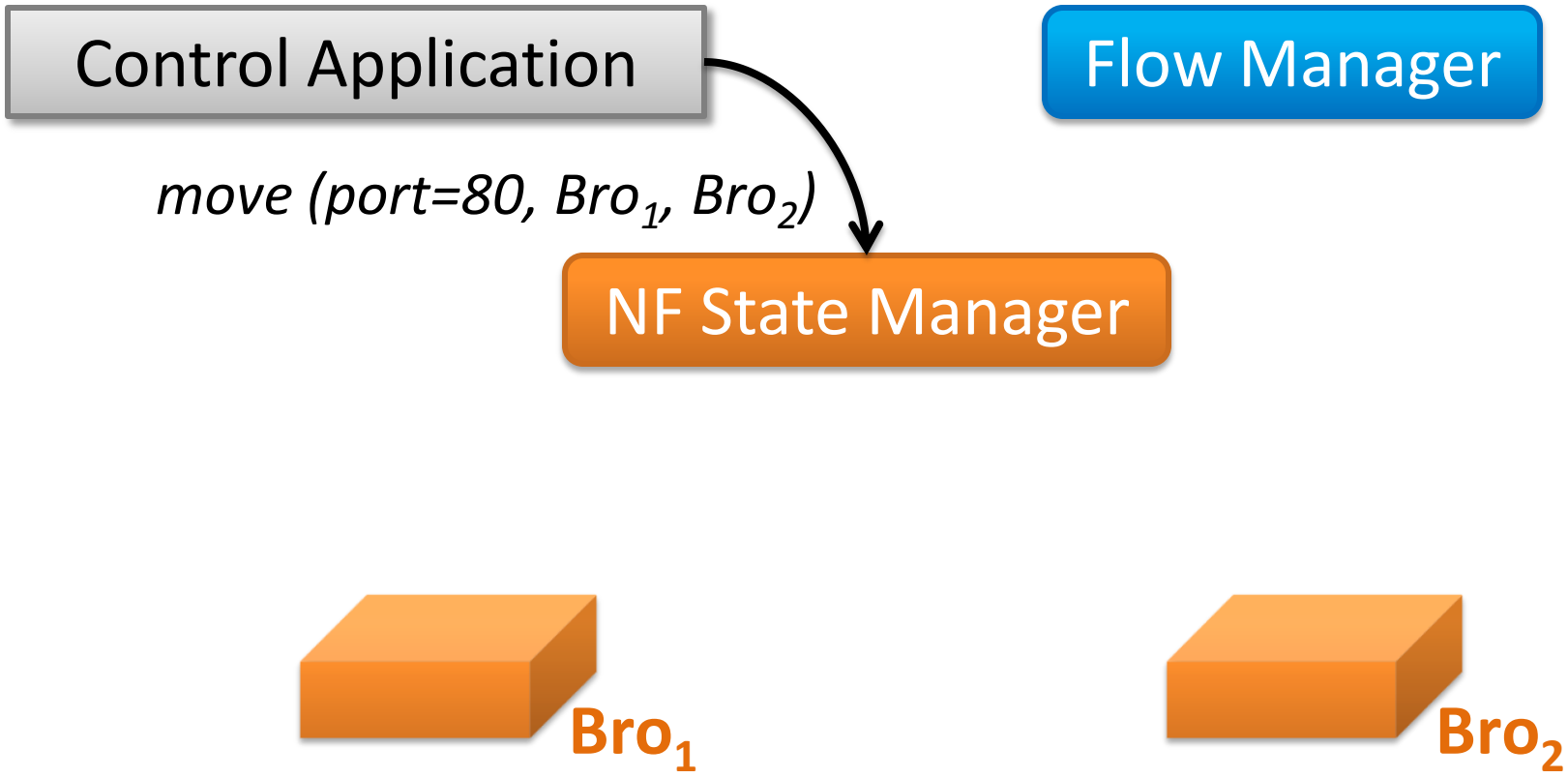
NF API: export/import state

- Functions: `get`, `put`, `delete`

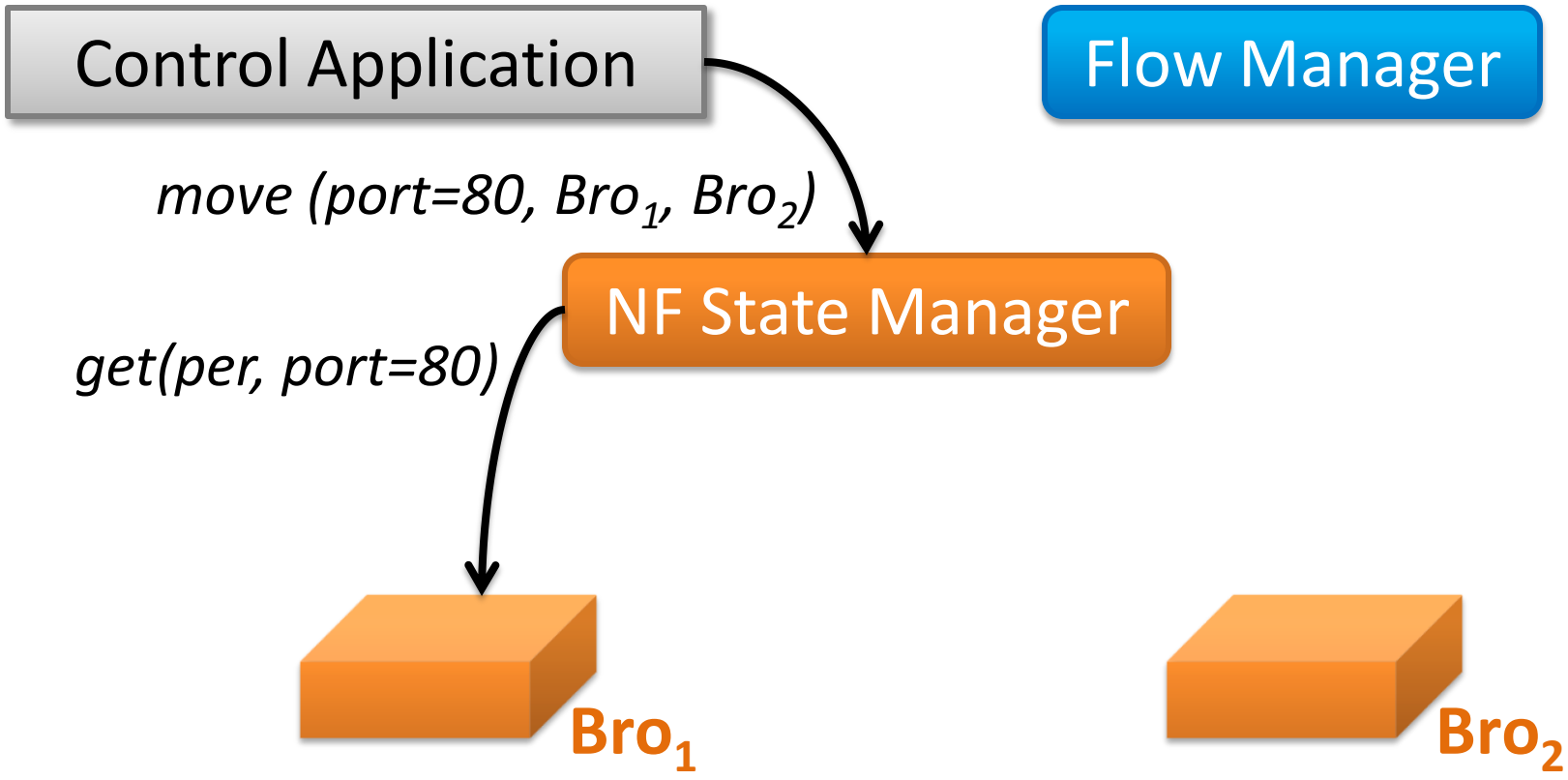


No need to expose/change internal state organization!

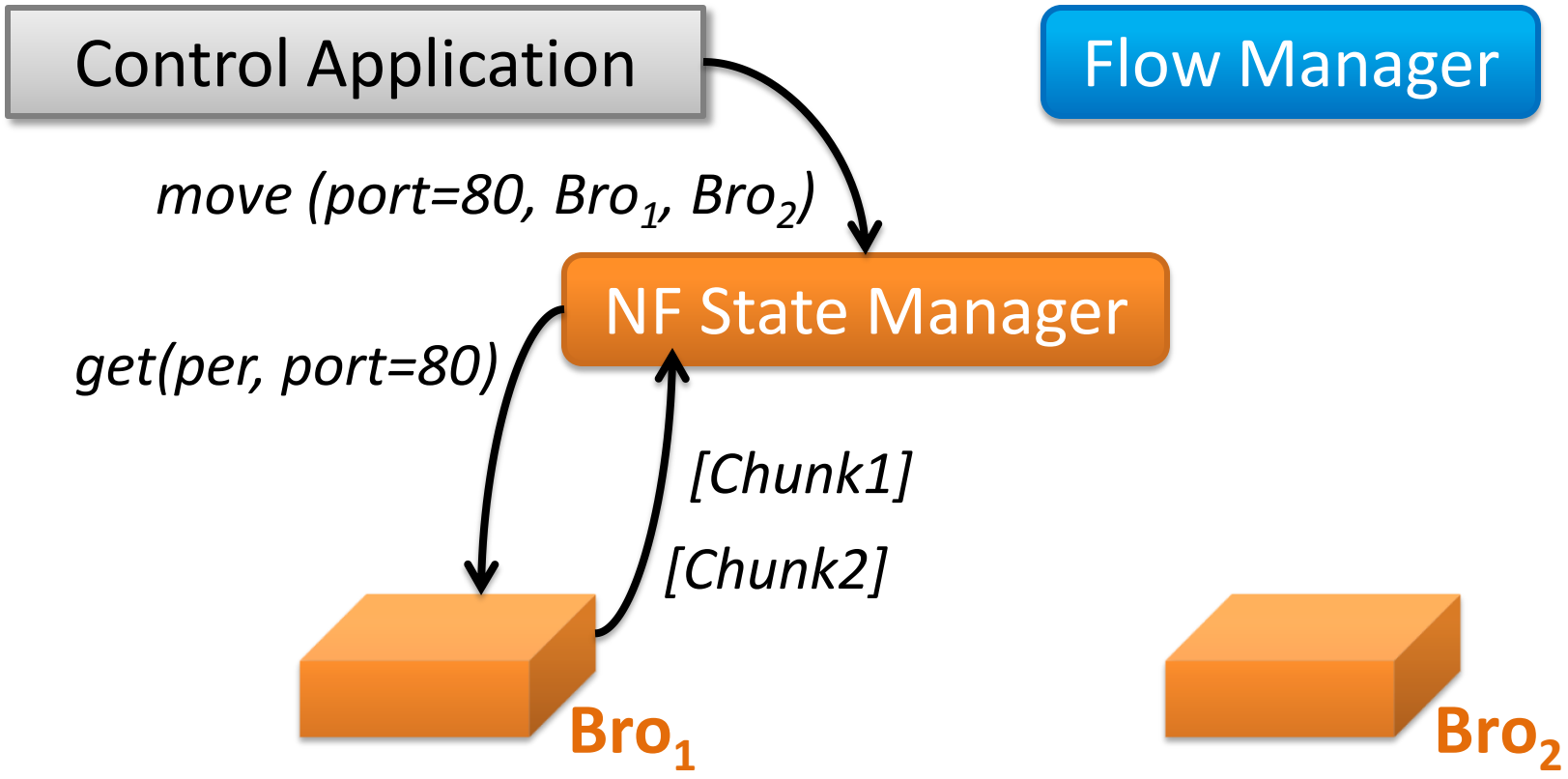
Control operations: move



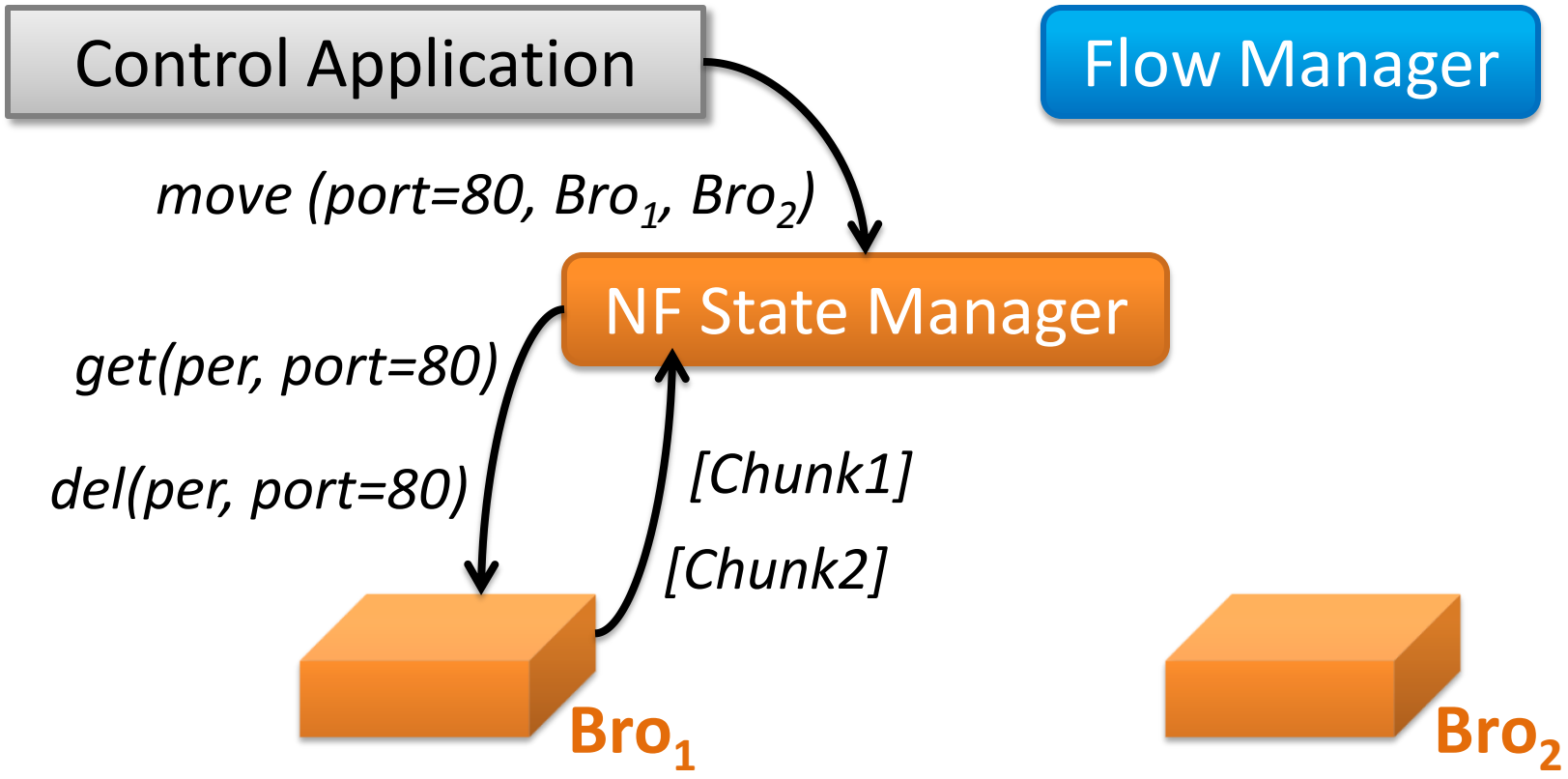
Control operations: move



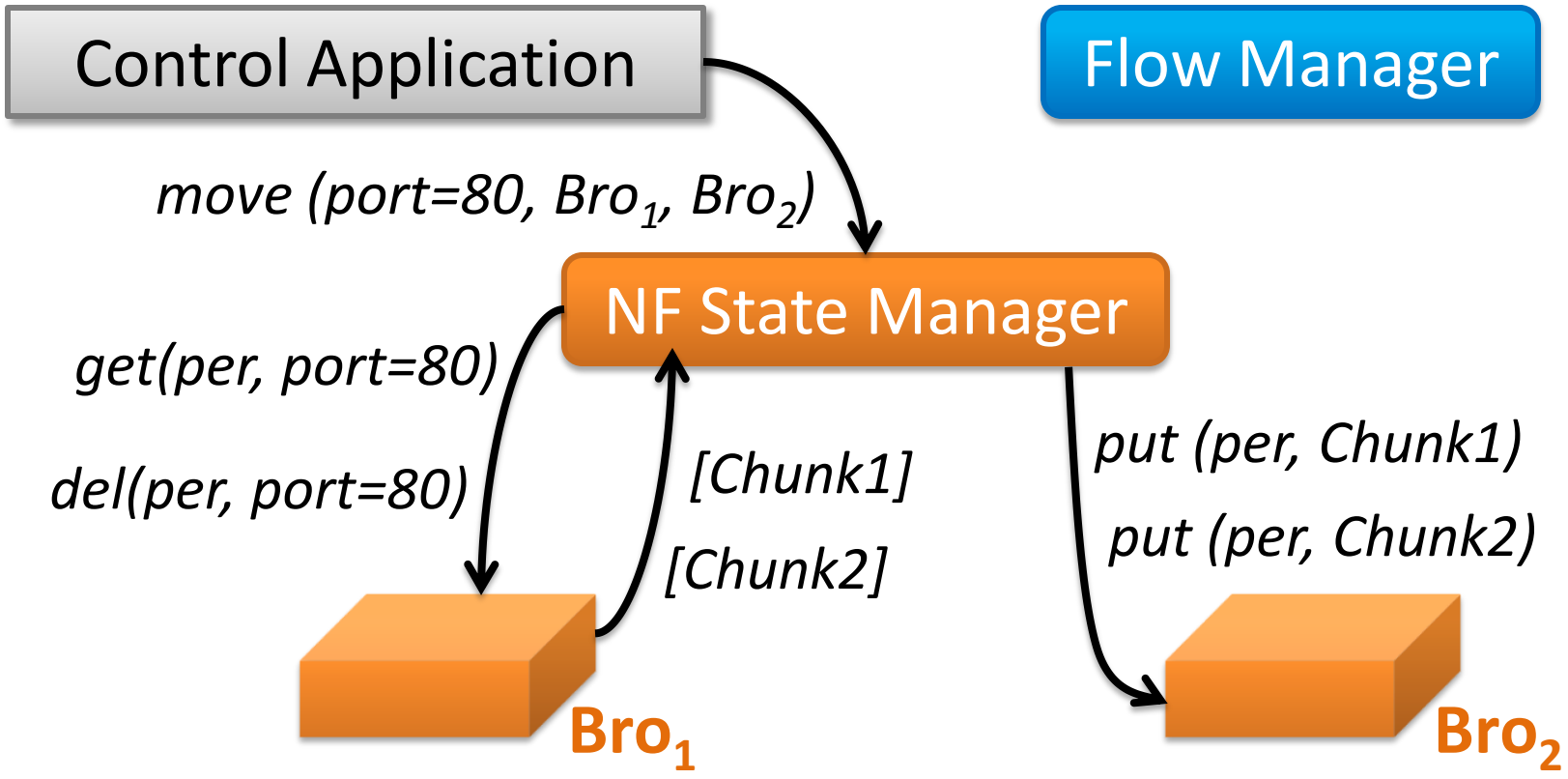
Control operations: move



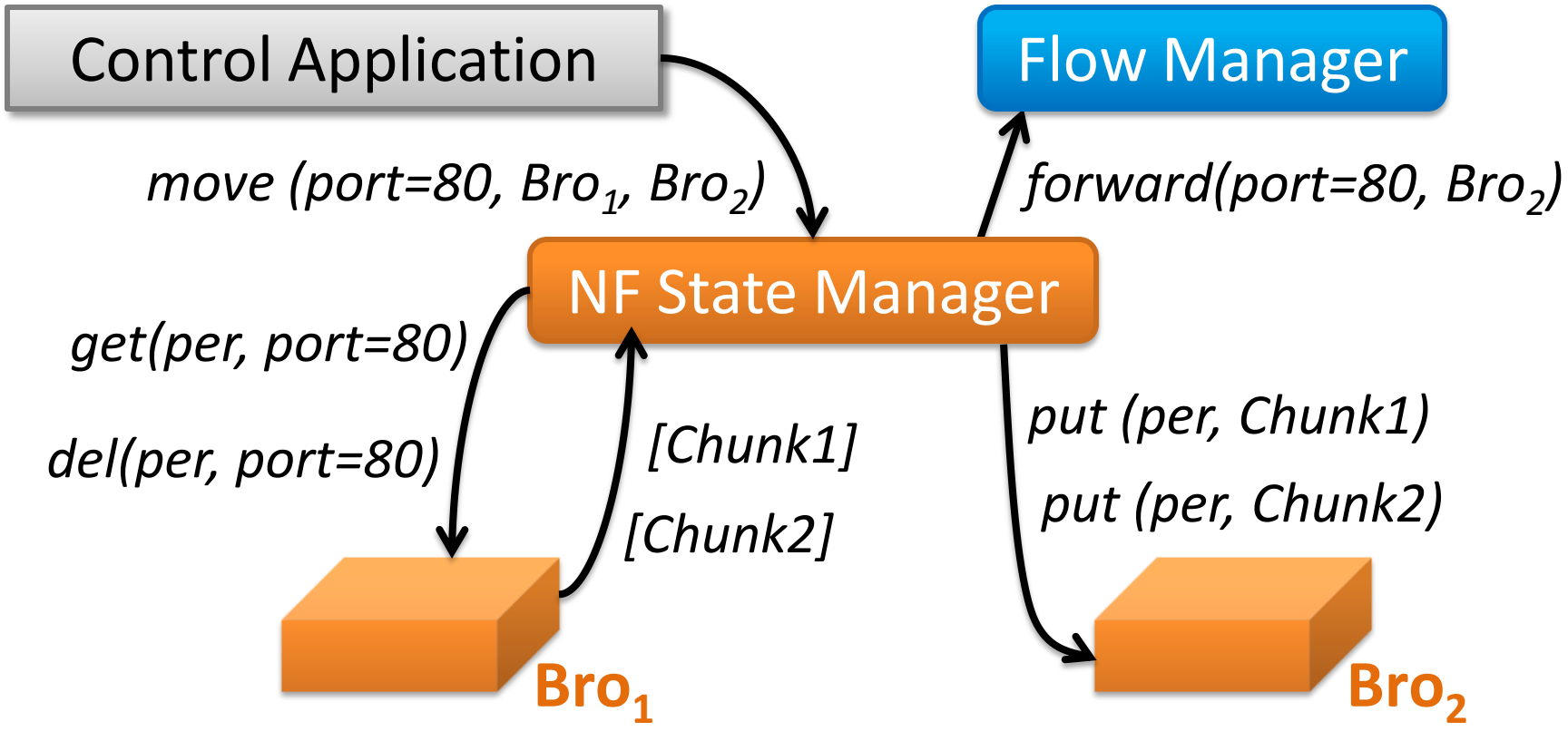
Control operations: move



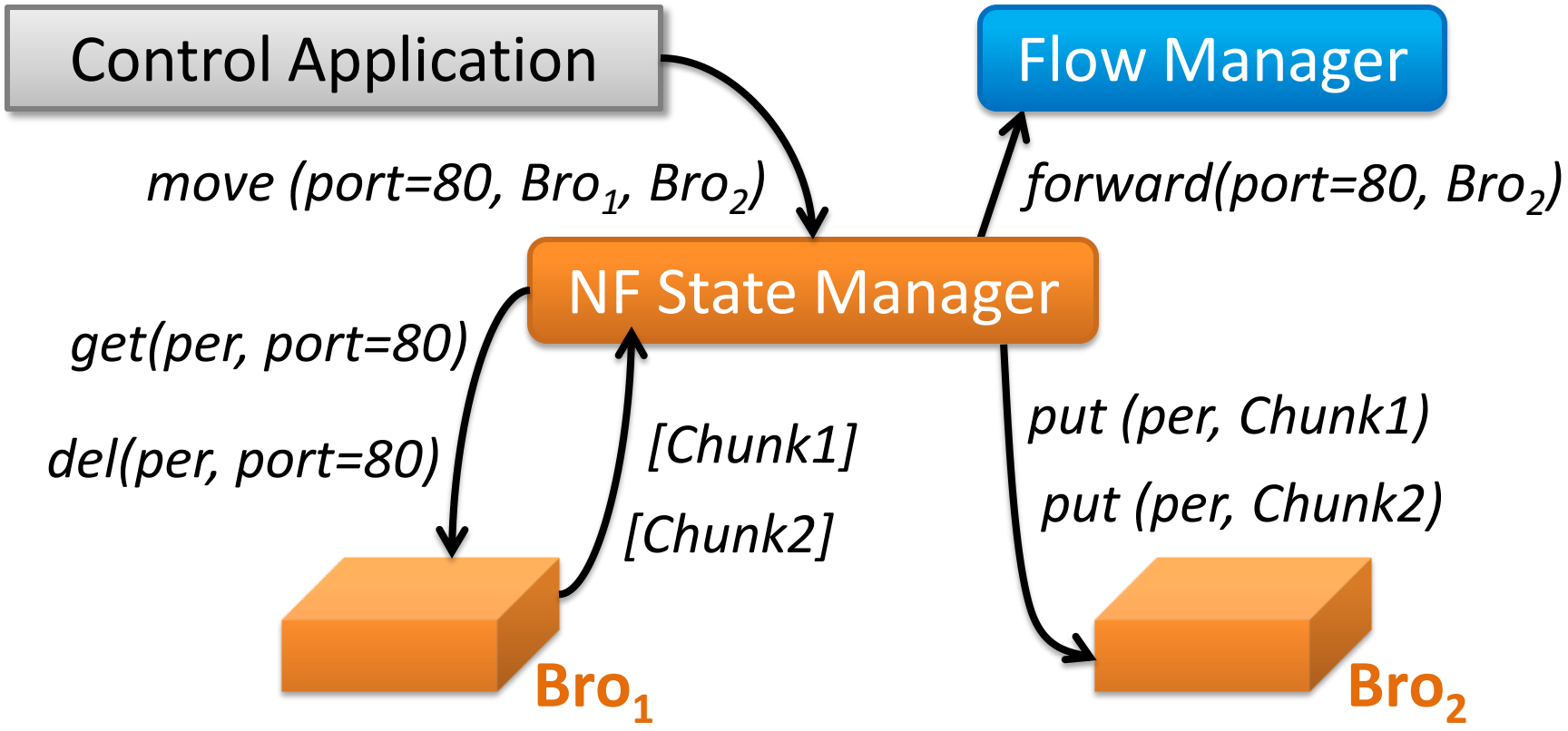
Control operations: move



Control operations: move

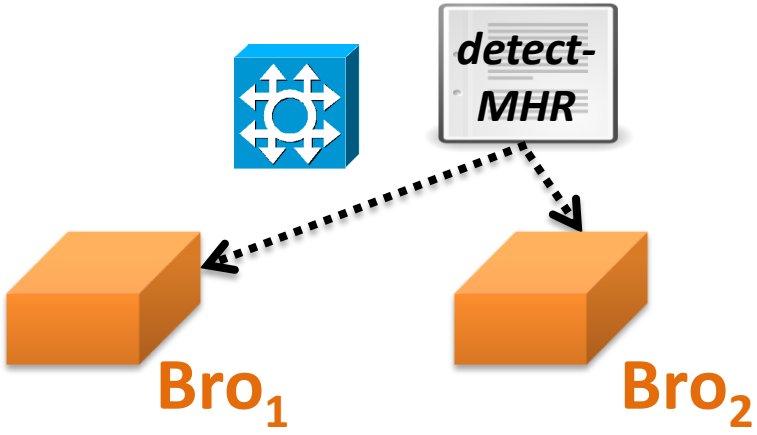


Control operations: move

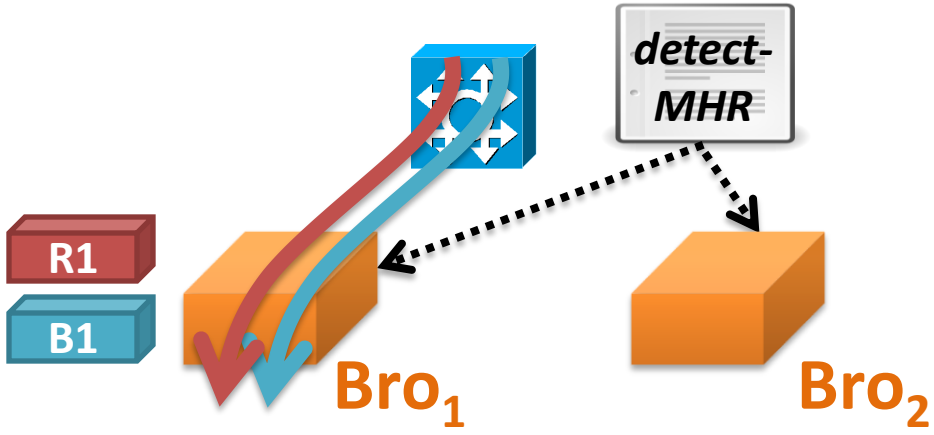


Also provide copy and share

Lost updates during move

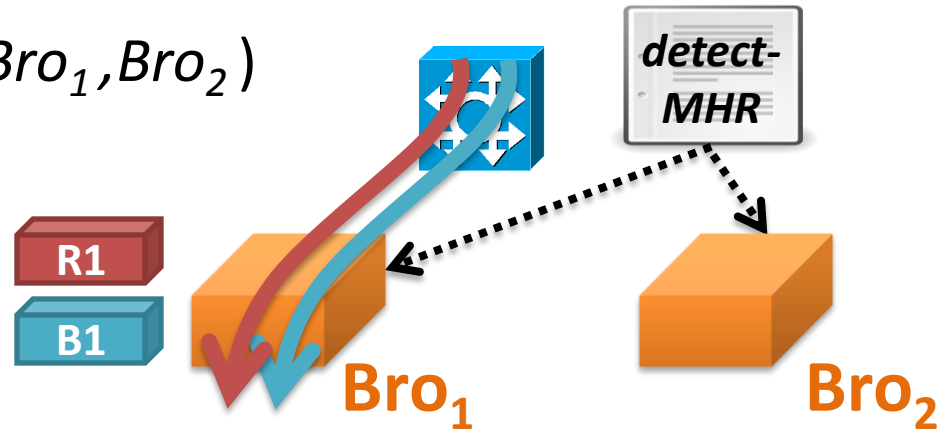


Lost updates during move



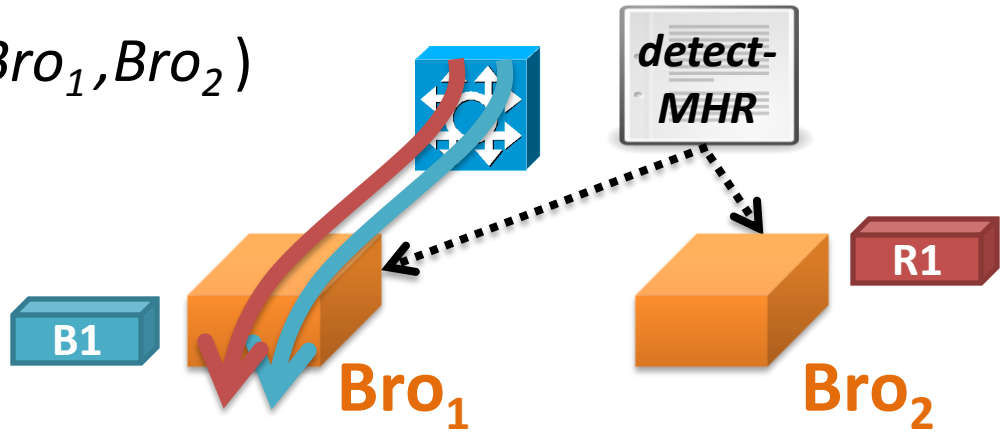
Lost updates during move

move(red,Bro$_1$,Bro$_2$)

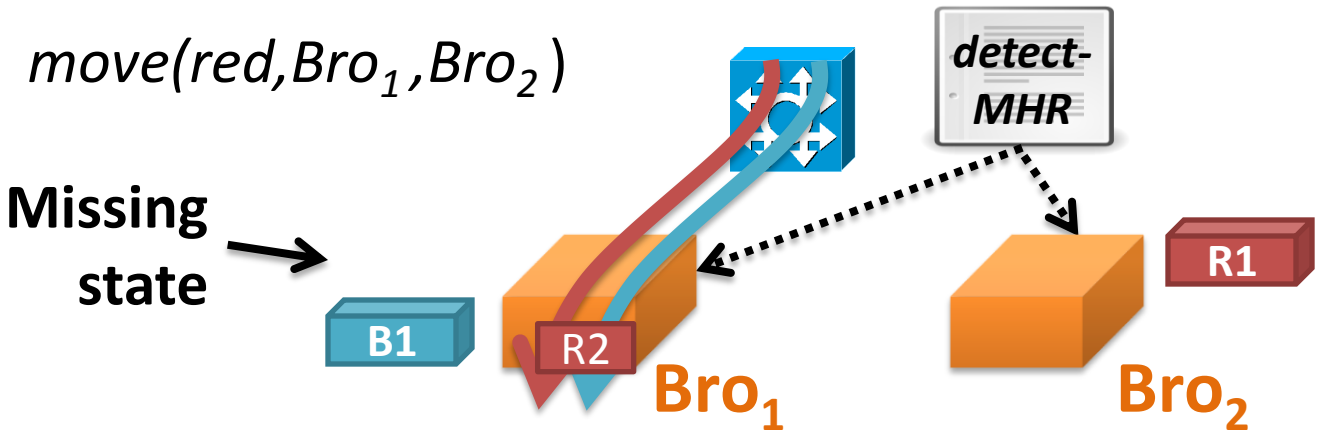


Lost updates during move

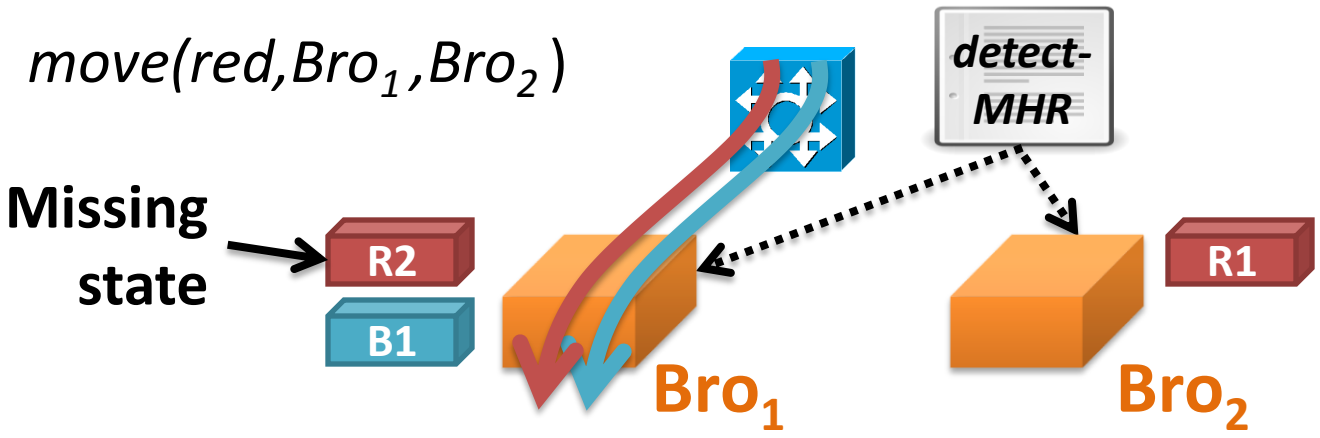
move(red,Bro$_1$,Bro$_2$)



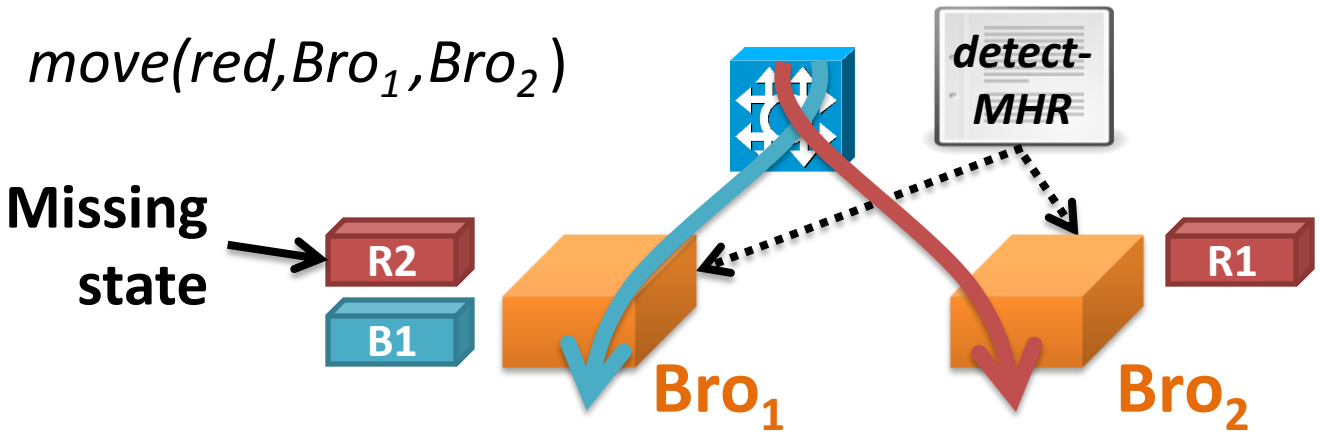
Lost updates during move



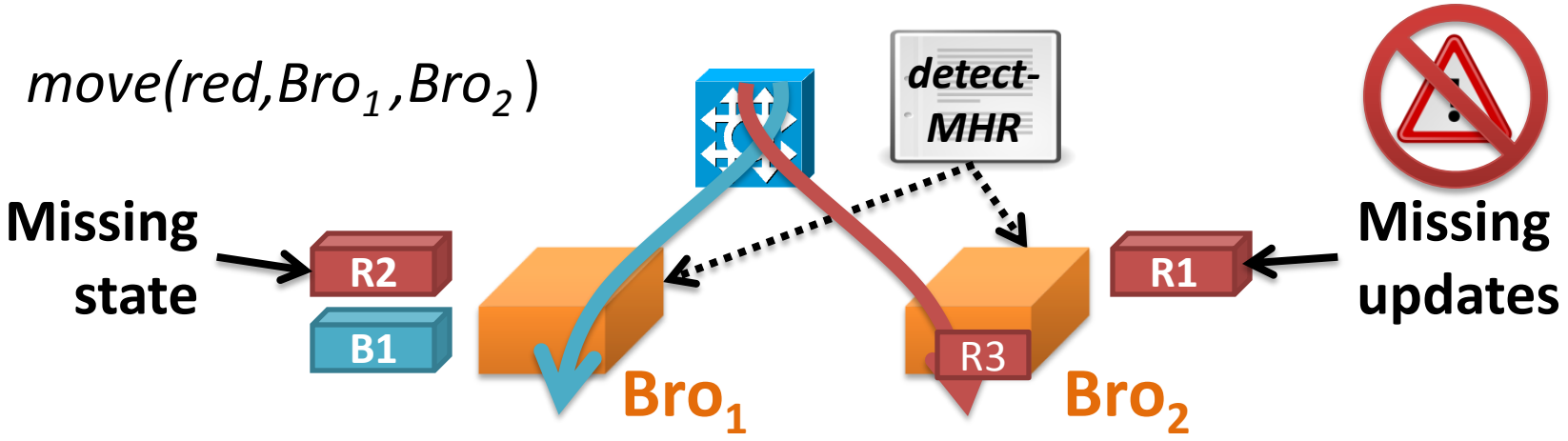
Lost updates during move



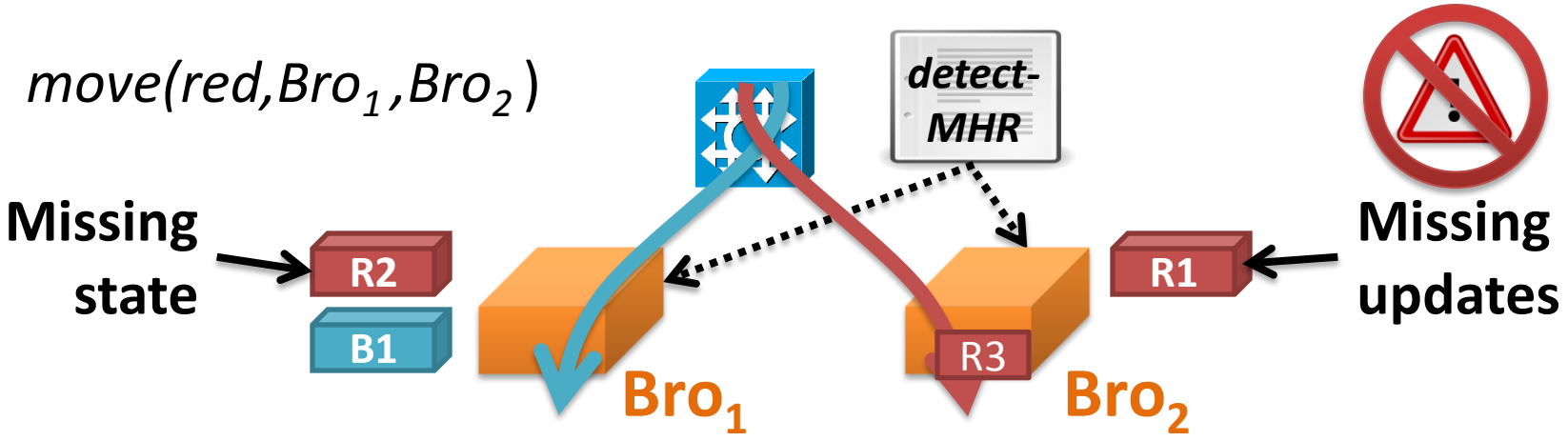
Lost updates during move



Lost updates during move



Lost updates during move

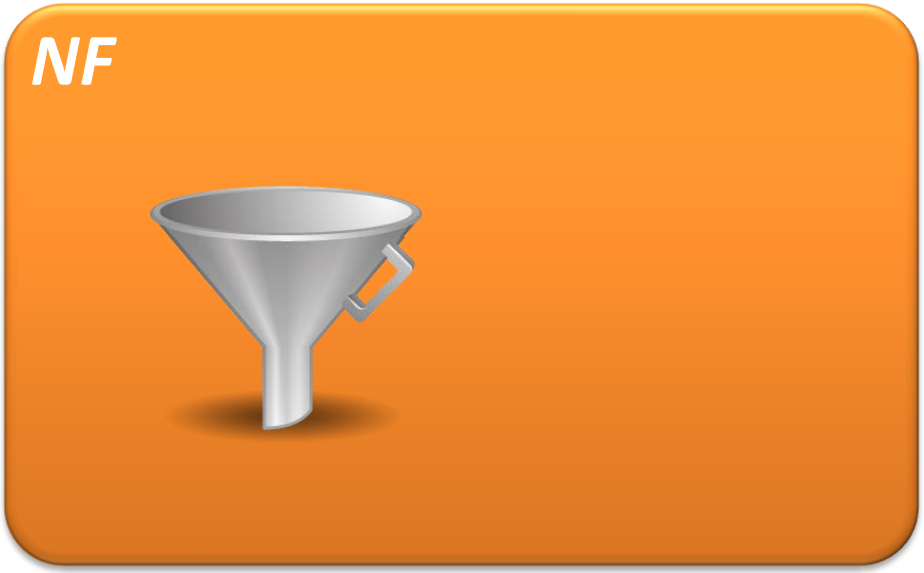


Loss-free: All state updates should be reflected in the transferred state, and all packets should be processed

✗ Split/Merge *[NSDI '13]*: pause traffic, buffer packets

– Packets in-transit when buffering starts are dropped

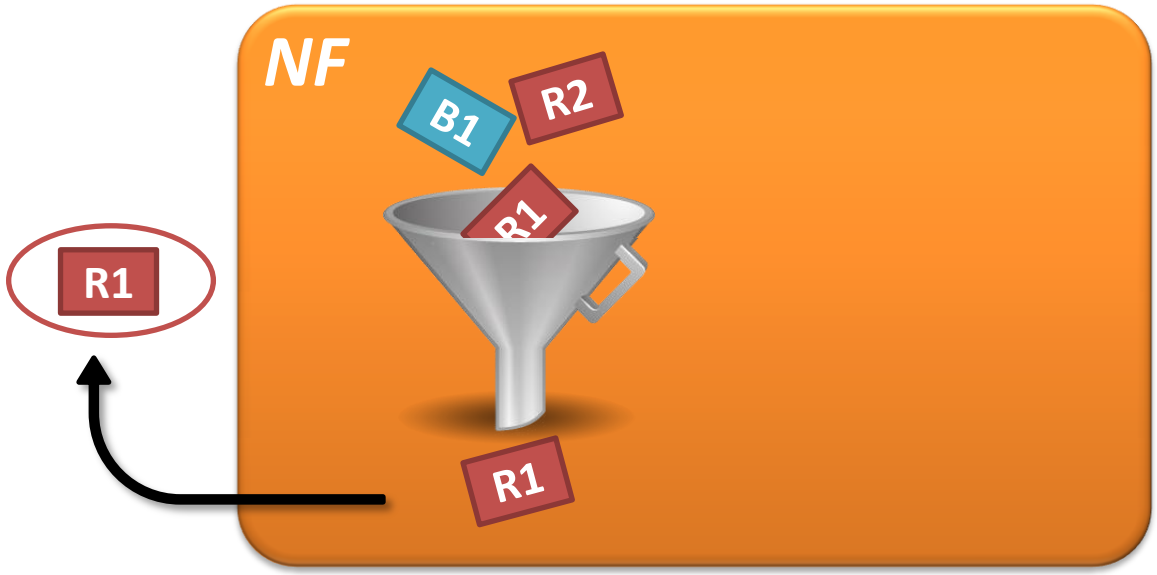
NF API: observe/prevent updates using events



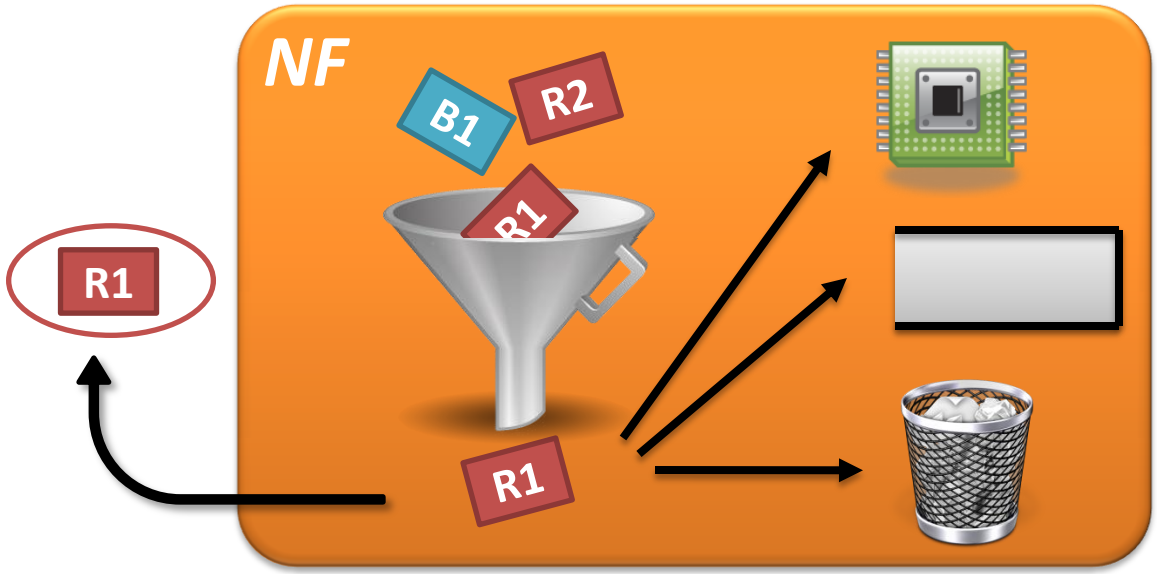
NF API: observe/prevent updates using events

NF API: observe/prevent updates using events

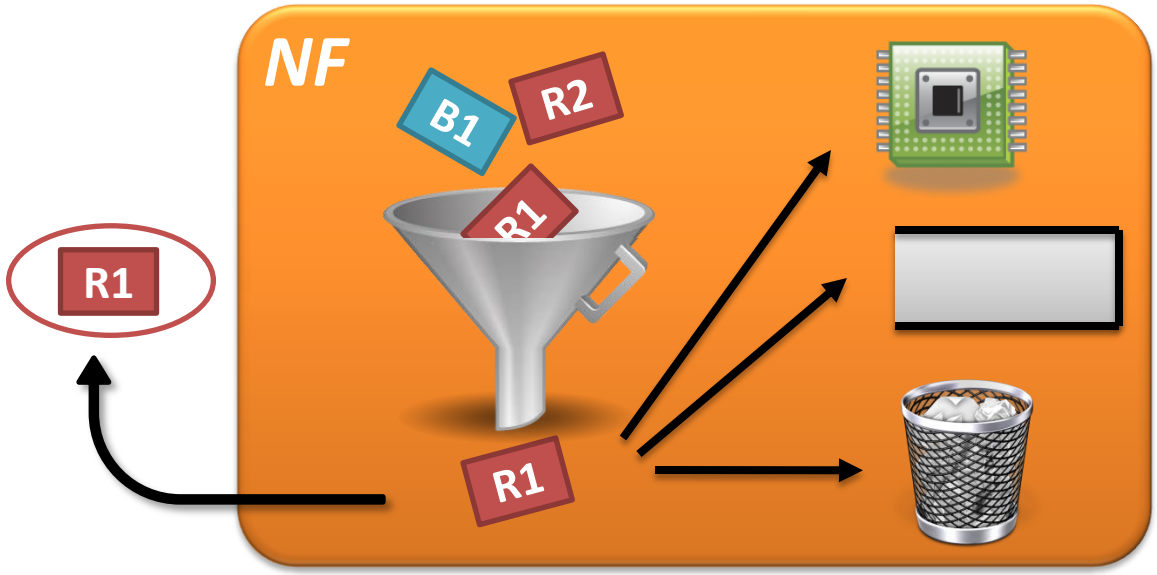
NF API: observe/prevent updates using events



NF API: observe/prevent updates using events

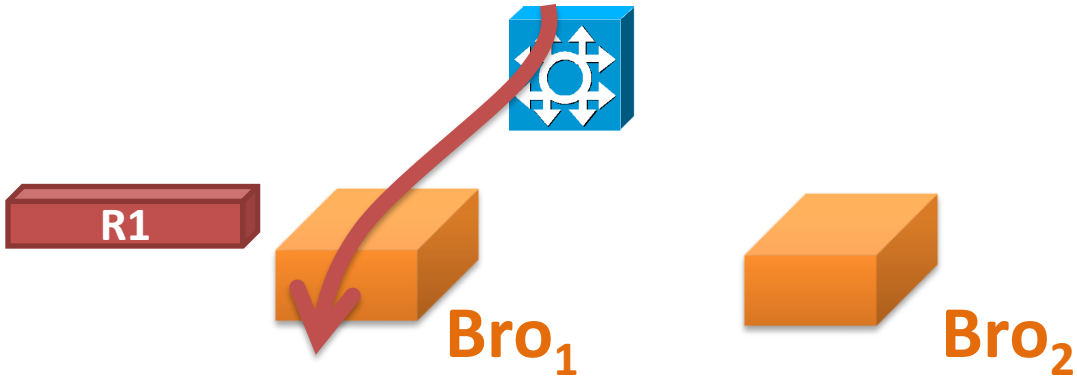


NF API: observe/prevent updates using events



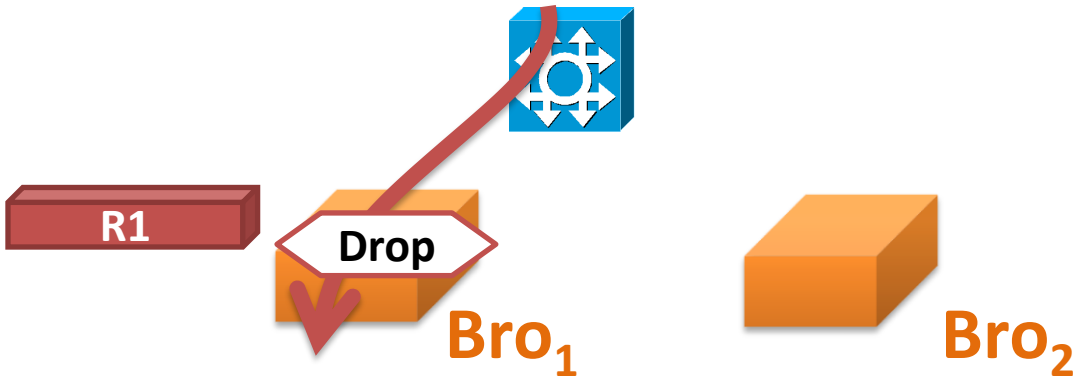
Only need to change an NF's receive packet function!

Use events for loss-free move



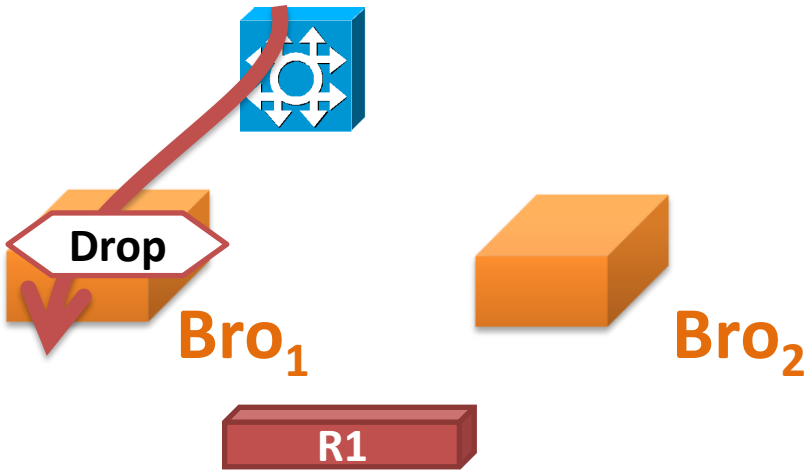
Use events for loss-free move

1. `enableEvents(red,drop)` on Bro_1



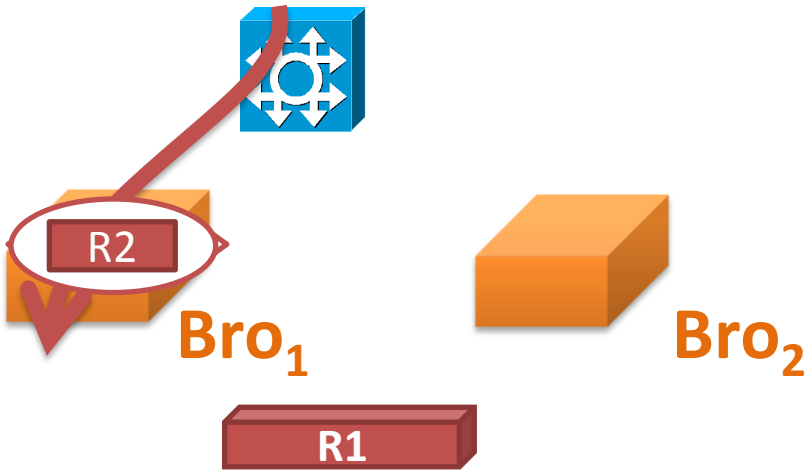
Use events for loss-free move

1. `enableEvents(red,drop)` on Bro_1
2. `get/delete` on Bro_1



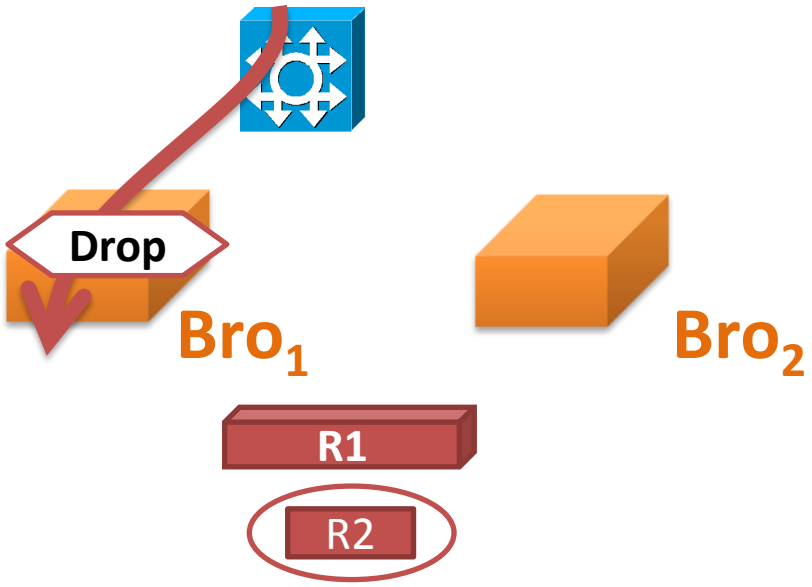
Use events for loss-free move

1. `enableEvents(red,drop)` on Bro_1
2. `get/delete` on Bro_1



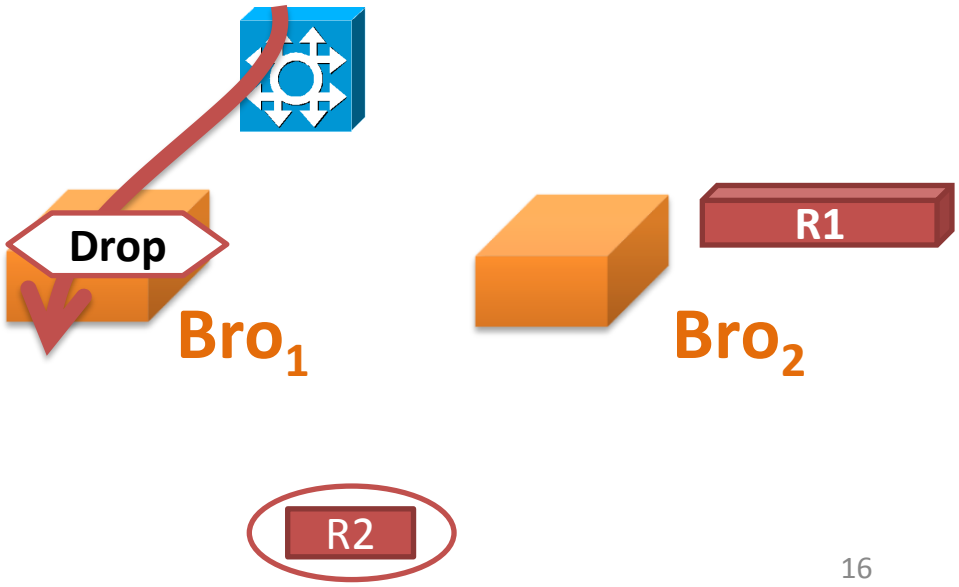
Use events for loss-free move

1. `enableEvents(red,drop)` on Bro_1
2. `get/delete` on Bro_1
3. Buffer events at controller



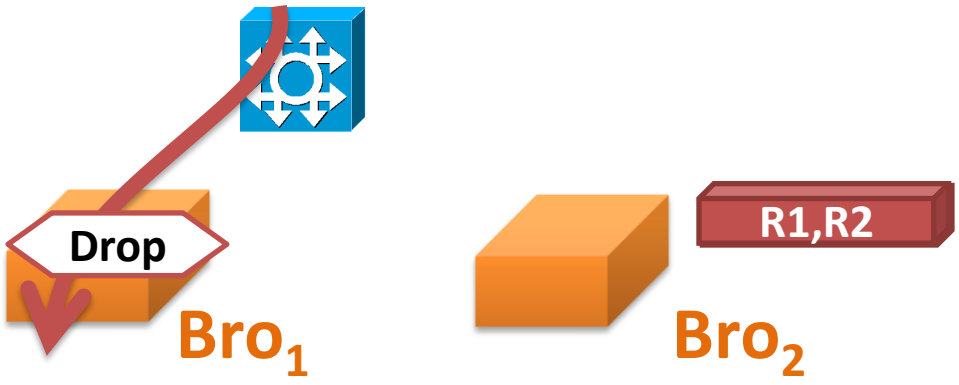
Use events for loss-free move

1. `enableEvents(red,drop)` on Bro_1
2. `get/delete` on Bro_1
3. Buffer events at controller
4. `put` on Bro_2



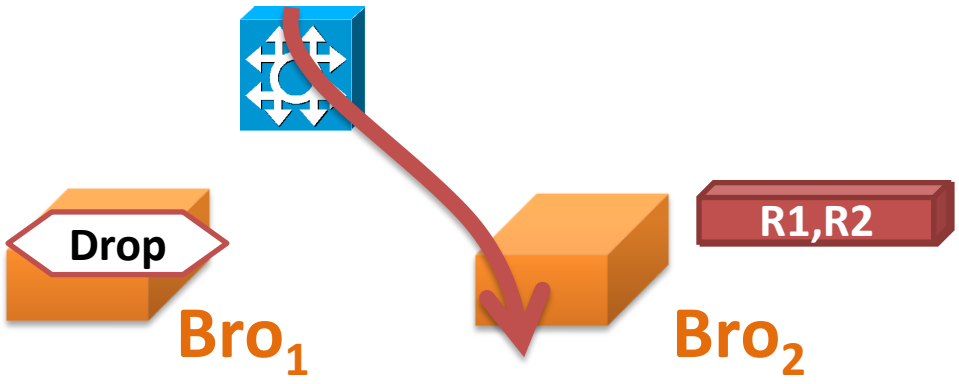
Use events for loss-free move

1. `enableEvents(red,drop)` on Bro_1
2. `get/delete` on Bro_1
3. Buffer events at controller
4. `put` on Bro_2
5. Flush packets in events to Bro_2



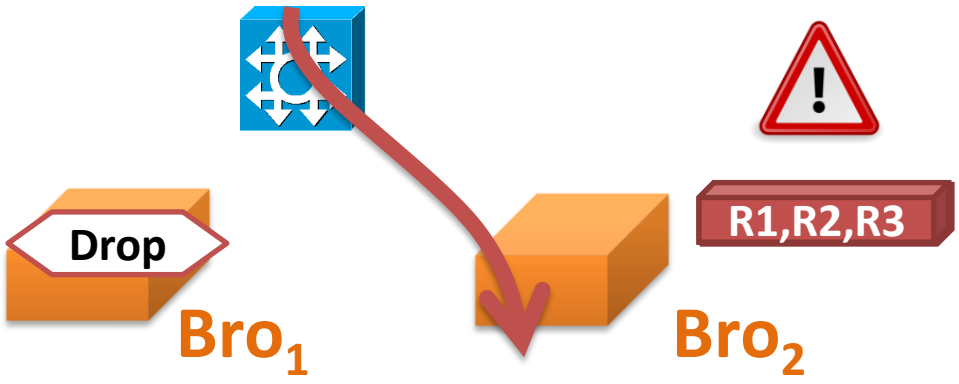
Use events for loss-free move

1. `enableEvents(red,drop)` on Bro_1
2. `get/delete` on Bro_1
3. Buffer events at controller
4. `put` on Bro_2
5. Flush packets in events to Bro_2
6. Update forwarding



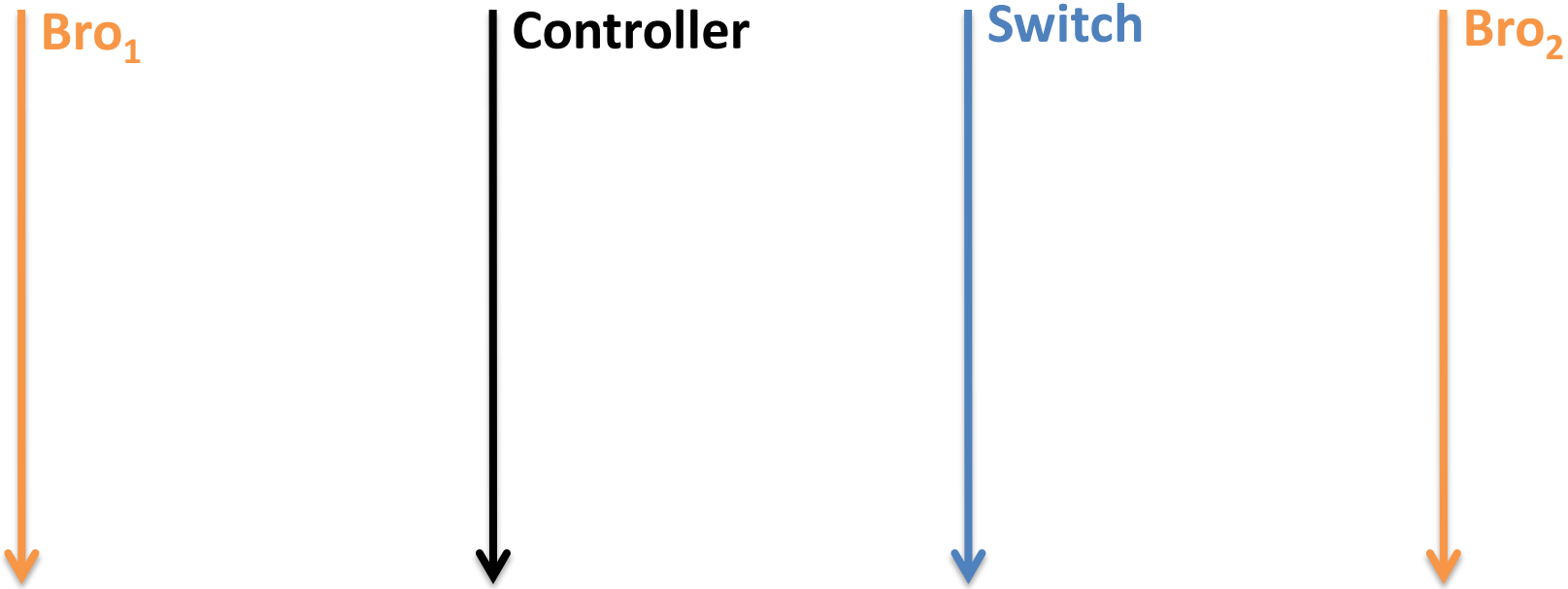
Use events for loss-free move

1. `enableEvents(red,drop)` on Bro_1
2. `get/delete` on Bro_1
3. Buffer events at controller
4. `put` on Bro_2
5. Flush packets in events to Bro_2
6. Update forwarding



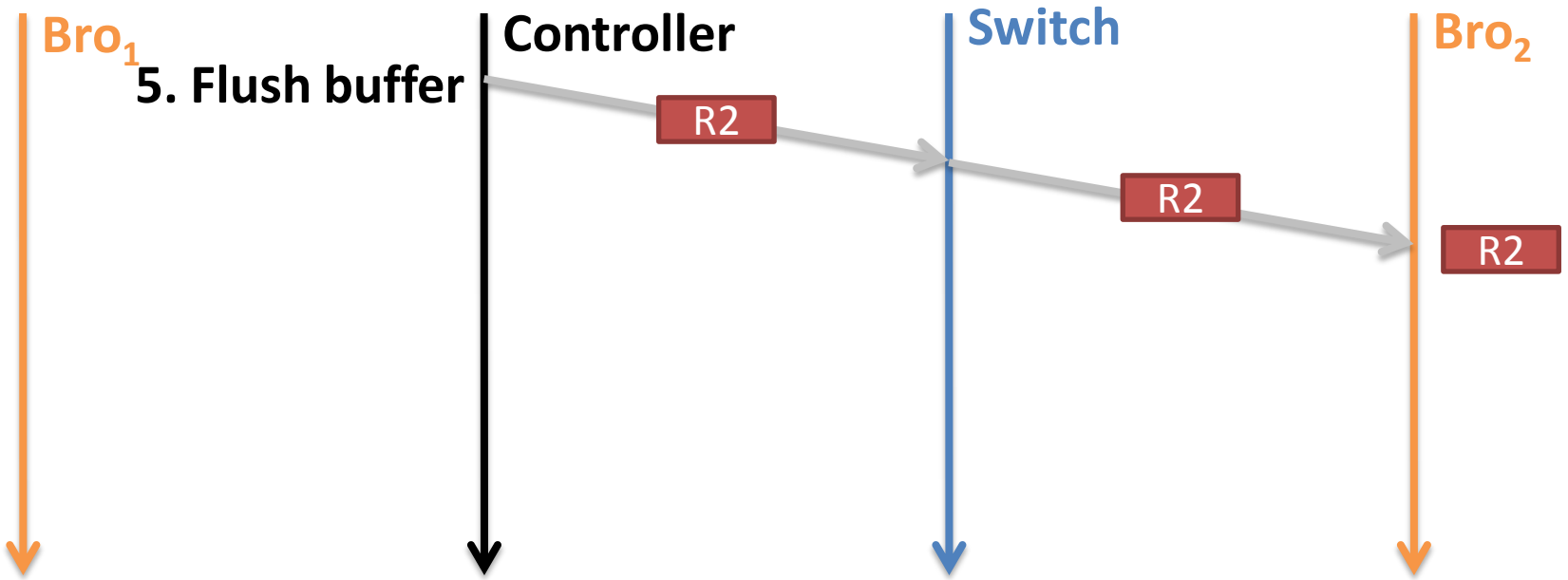
Re-ordering of packets

- False positives from Bro's `weird` script



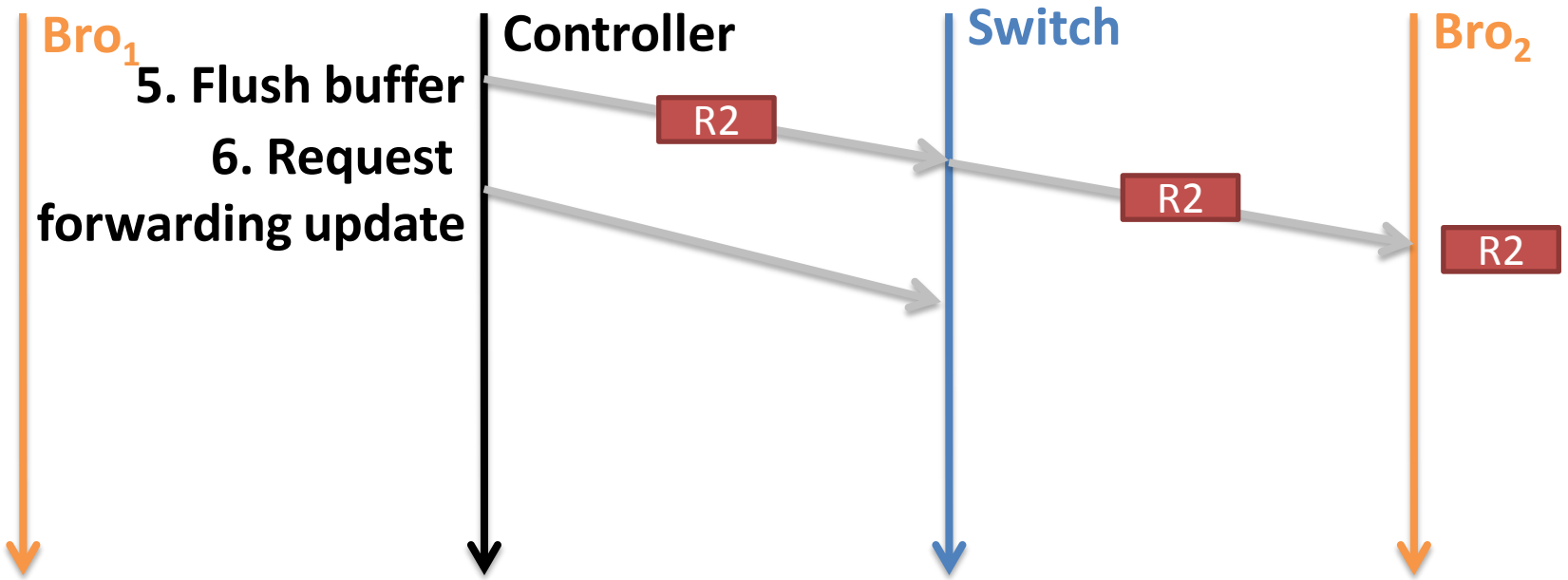
Re-ordering of packets

- False positives from Bro's `weird` script



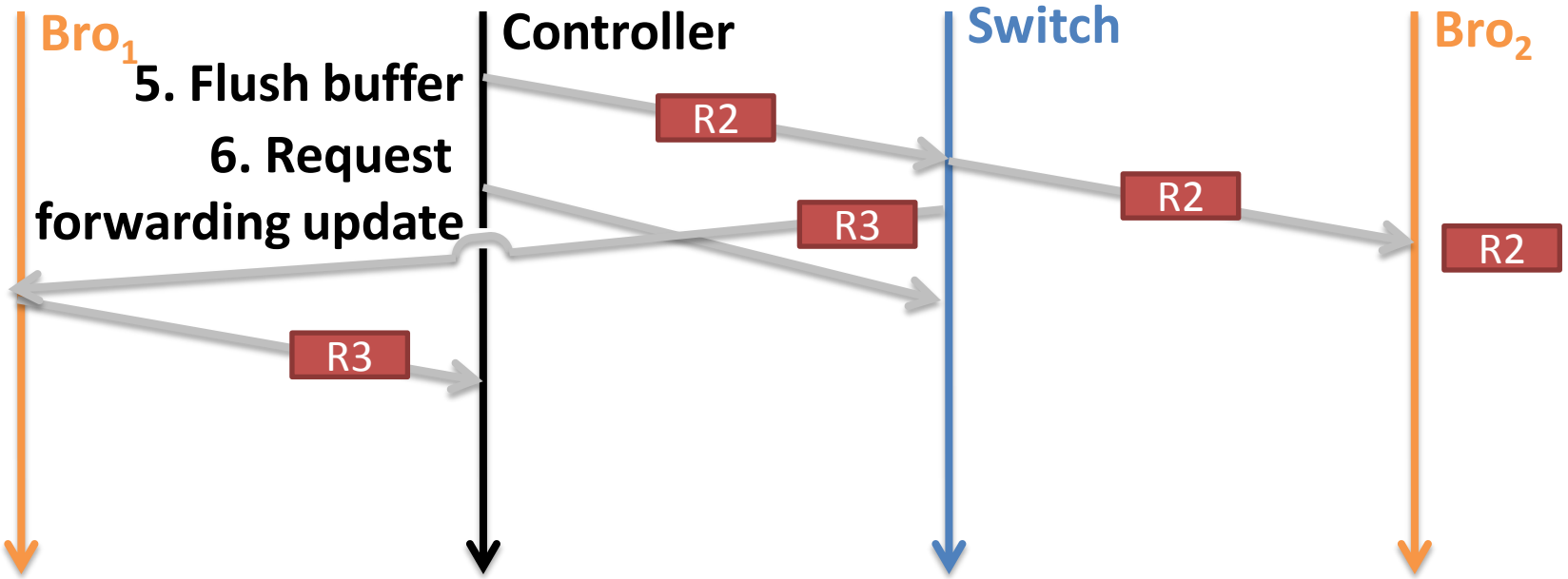
Re-ordering of packets

- False positives from Bro's `weird` script



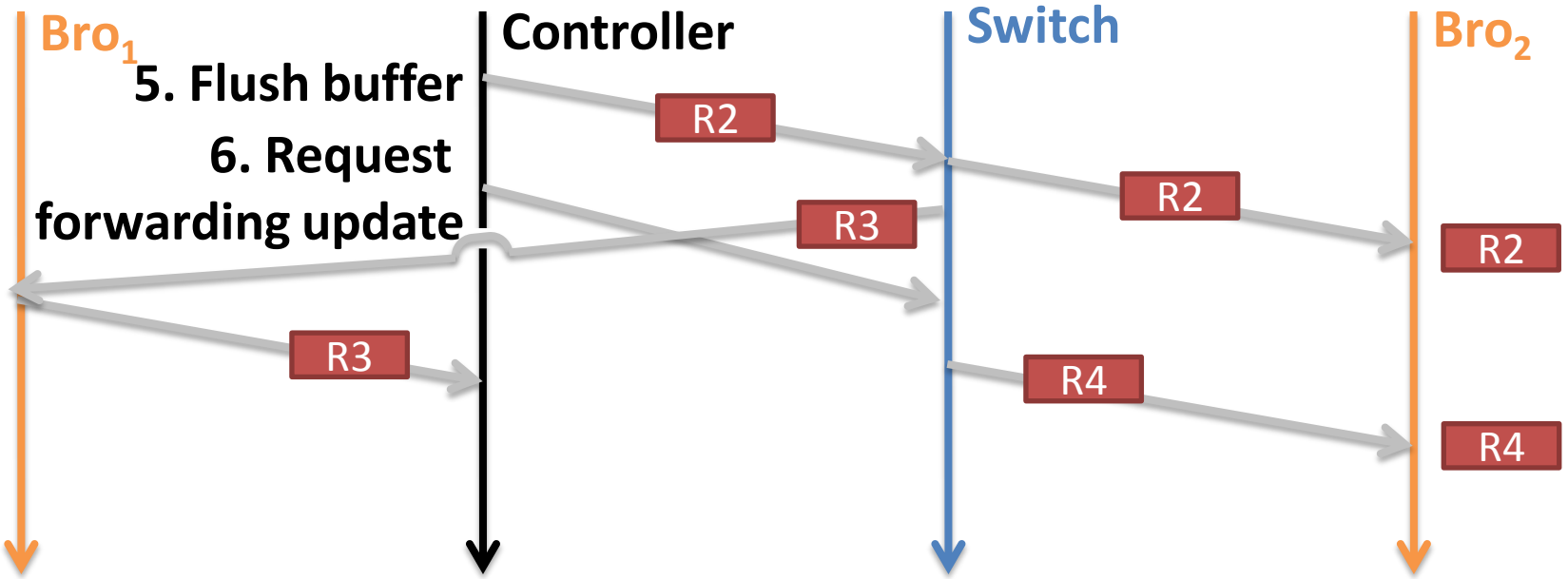
Re-ordering of packets

- False positives from Bro's `weird` script



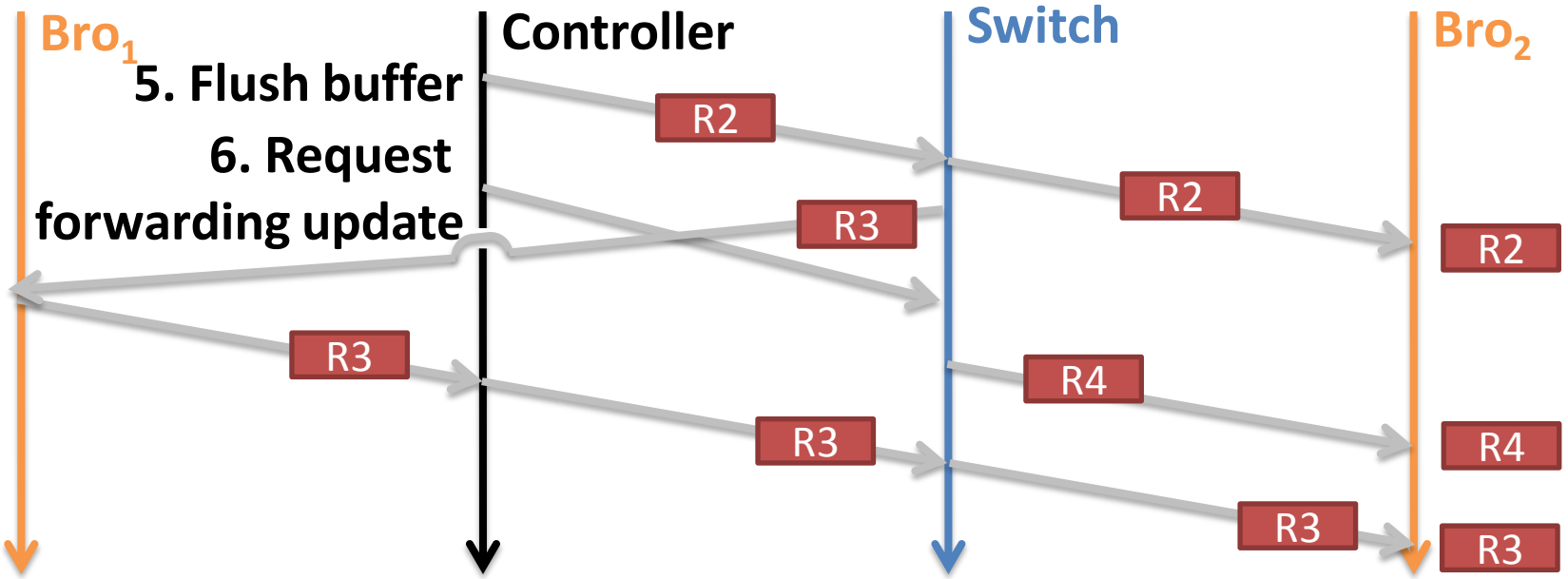
Re-ordering of packets

- False positives from Bro's `weird` script



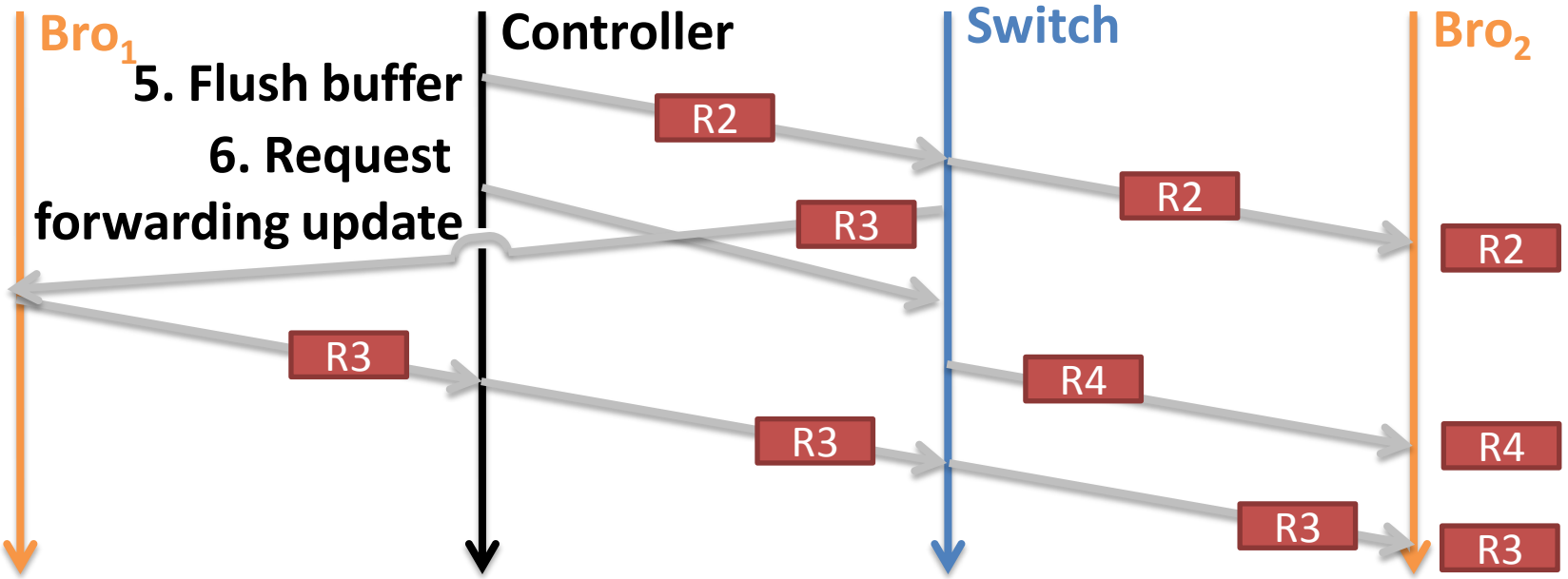
Re-ordering of packets

- False positives from Bro's `weird` script



Re-ordering of packets

- False positives from Bro's `weird` script



Order-preserving: All packets should be processed in the order they were forwarded by the switch

OpenNF: SLAs + cost + accuracy

1. Dealing with diversity

Export/import state based
on its association with flows

2. Dealing with race conditions

Events + Lock-step forwarding updates

Implementation

- Controller (*3.8K lines of Java*)
- Communication library (2.6K lines of C)
- Modified NFs (3-8% increase in code)

Bro IDS

iptables

Squid Cache

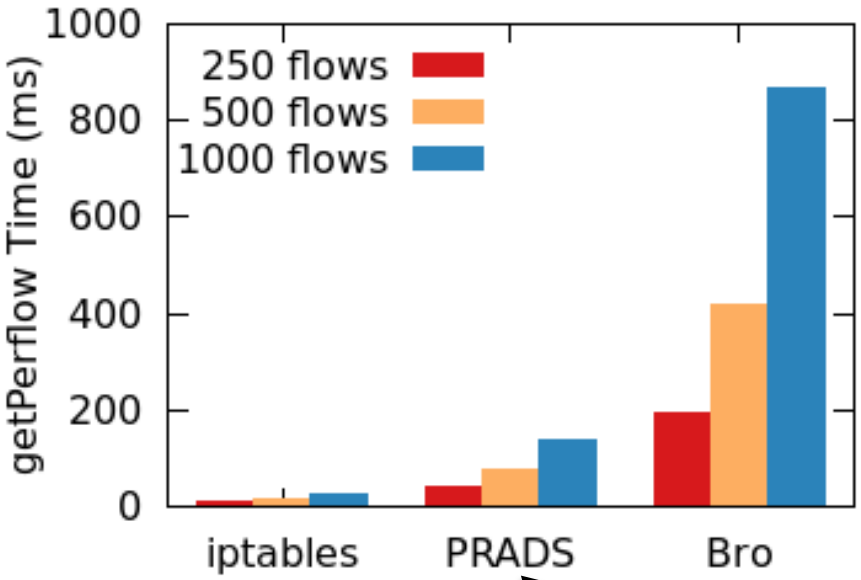
PRADS

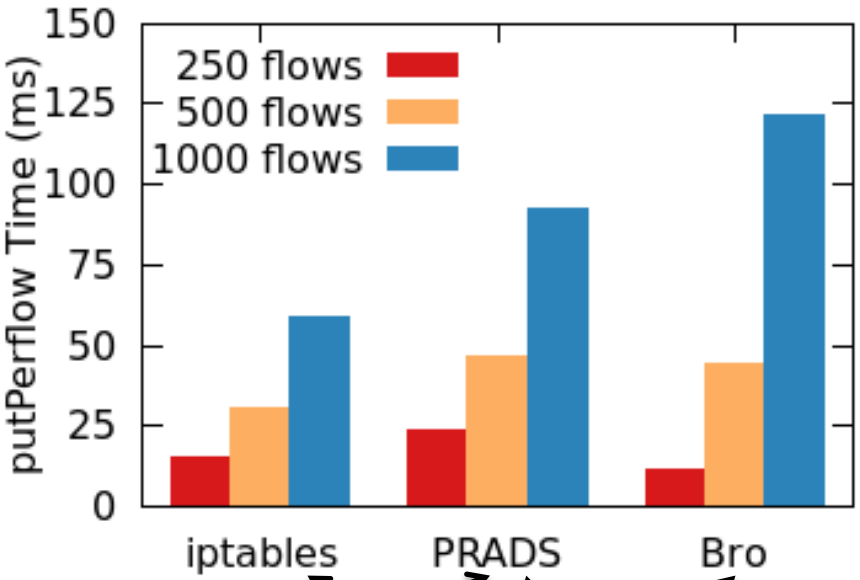
Overall benefits for elastic scaling

- Bro IDS processing 10K pkts/sec
  - *At 180 sec:* move HTTP flows (489) to new IDS
  - *At 360 sec:* move back to old IDS
- SLAs: 260ms to move (loss-free)

- Accuracy: same log entries as using one IDS
  - VM replication: incorrect log entries
- Cost: scale down after state is moved
  - Stratos: scale down delayed 25+ minutes
    [arXiv:1305.0209]

Evaluation: state export/import



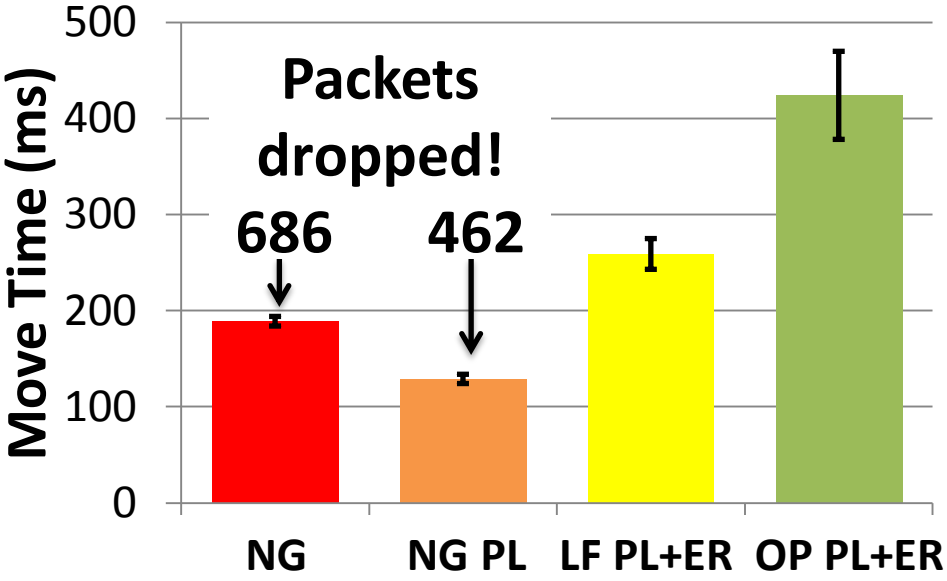


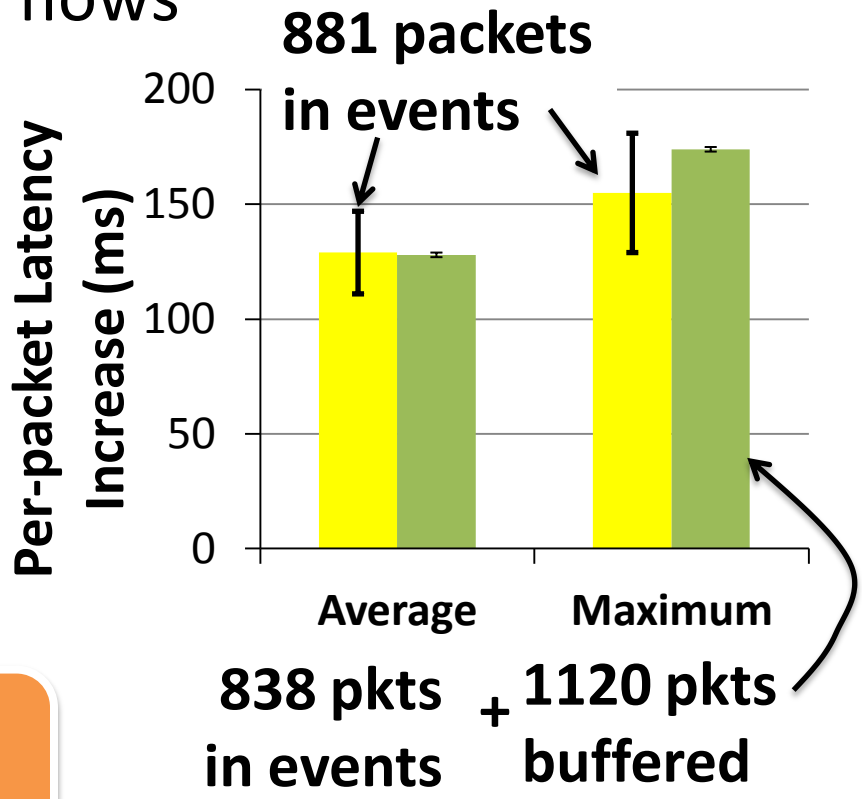
Serialization/deserialization costs dominate

Cost grows with state complexity

Evaluation: operations

- PRADS asset detector processing 5K pkts/sec
- Move per-flow state for 500 flows

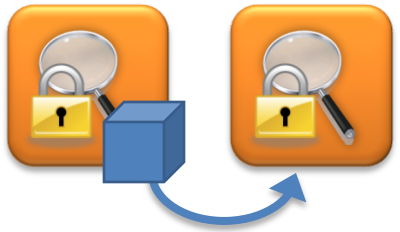




Operations are efficient, but guarantees come at a cost!

Conclusion

- Dynamic reallocation of packet processing enables new services



- Realizing SLAs + cost + accuracy requires quick, safe control of internal NF state
- OpenNF provides flexible and efficient control with few NF modifications

Points

- NFV provides virtualization, orchestration, scaling, automation, hardware independence, but not zero cost!
  - **Network performance of virtualized framework**
  - **Placement of virtual instances**
  - **Fine-grained trouble shooting and fault recovering**