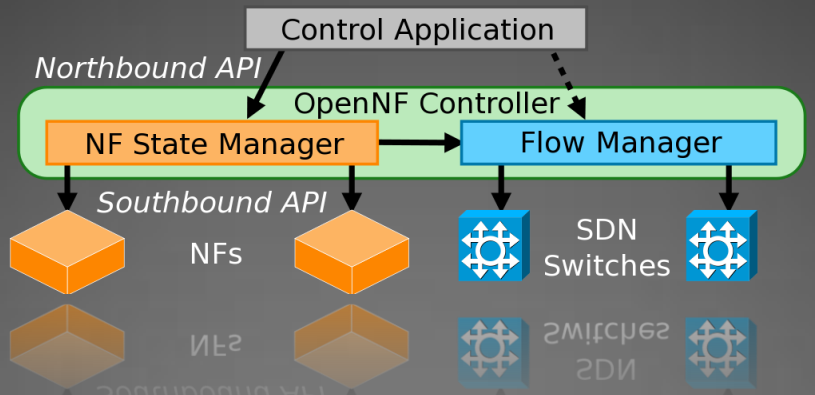


OpenNF: Enabling Innovation in Network Function Control

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

- Important goals:
  1. Satisfy SLAs
  2. Minimize costs
  3. Act correctly

NF trends

- Network Functions Virtualization (NFV)

WAN optimizer

Caching proxy

Intrusion detection system (IDS)

NF trends

- Network Functions Virtualization (NFV)

  → dynamically allocate NF instances

NF trends

- Network Functions Virtualization (NFV)

  → dynamically allocate NF instances

- Software-defined Networking

  → dynamically reroute flows

NF trends

- Network Functions Virtualization (NFV)

  → dynamically allocate NF instances

  ✚

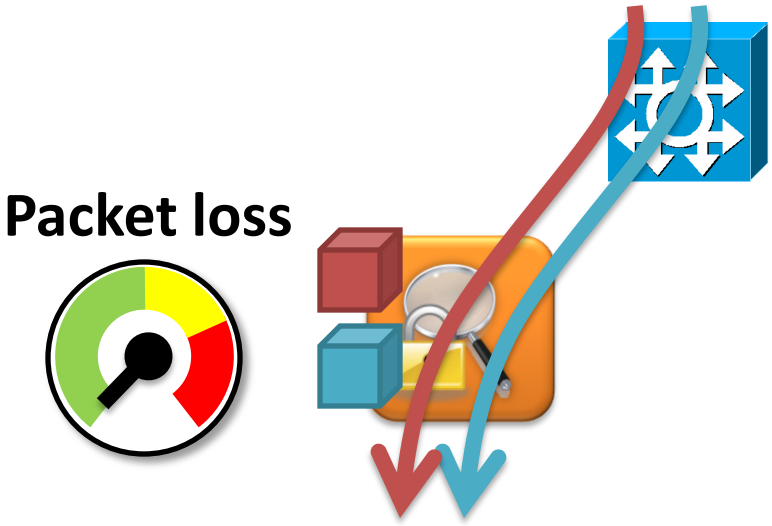
- Software-defined Networking

  → dynamically reroute flows

  ⬇

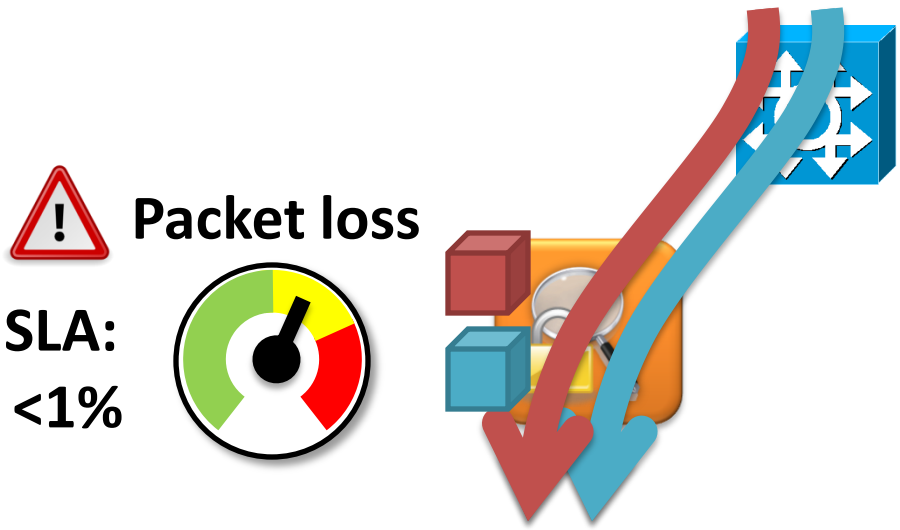
Dynamic reallocation
of packet processing
e.g., elastic NF scaling

Why NFV + SDN falls short



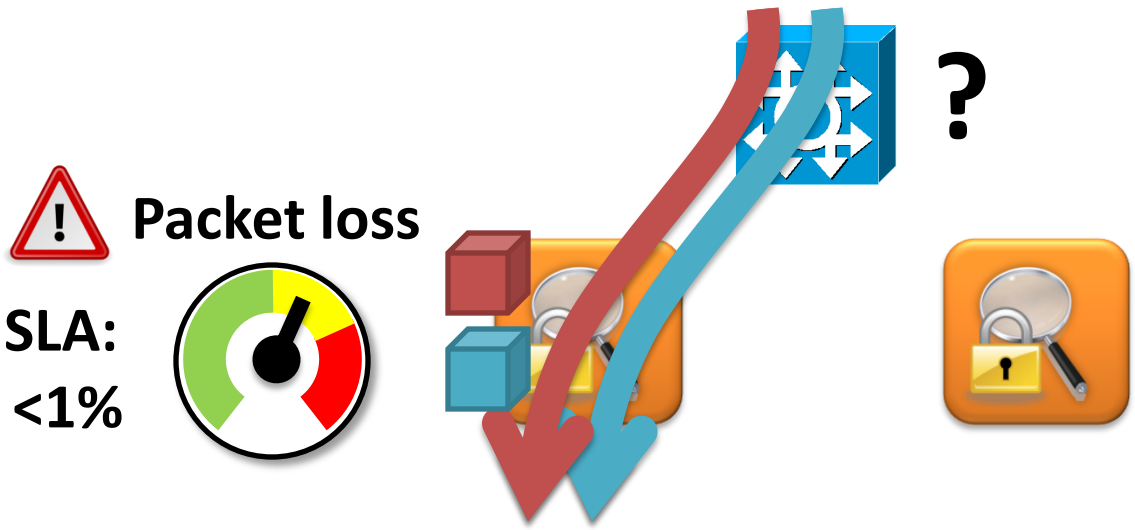
1. SLAs 2. Cost 3. Accuracy

Why NFV + SDN falls short



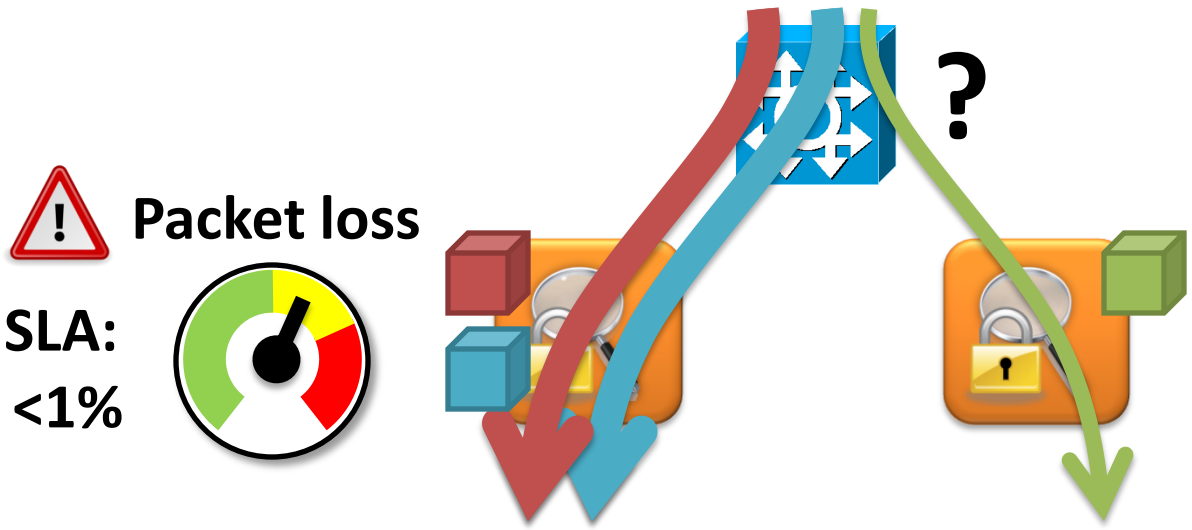
1. SLAs 2. Cost 3. Accuracy

Why NFV + SDN falls short



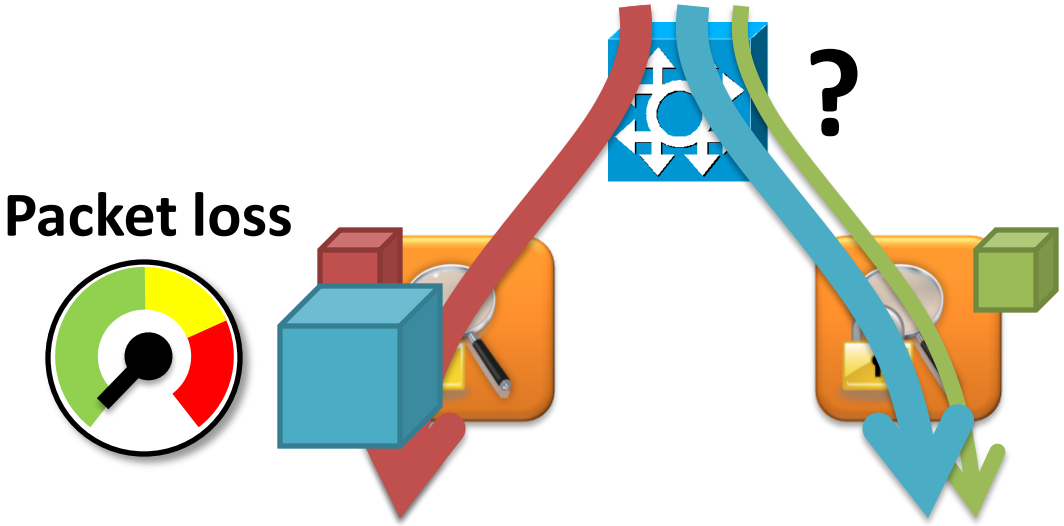
1. SLAs 2. Cost 3. Accuracy

Why NFV + SDN falls short



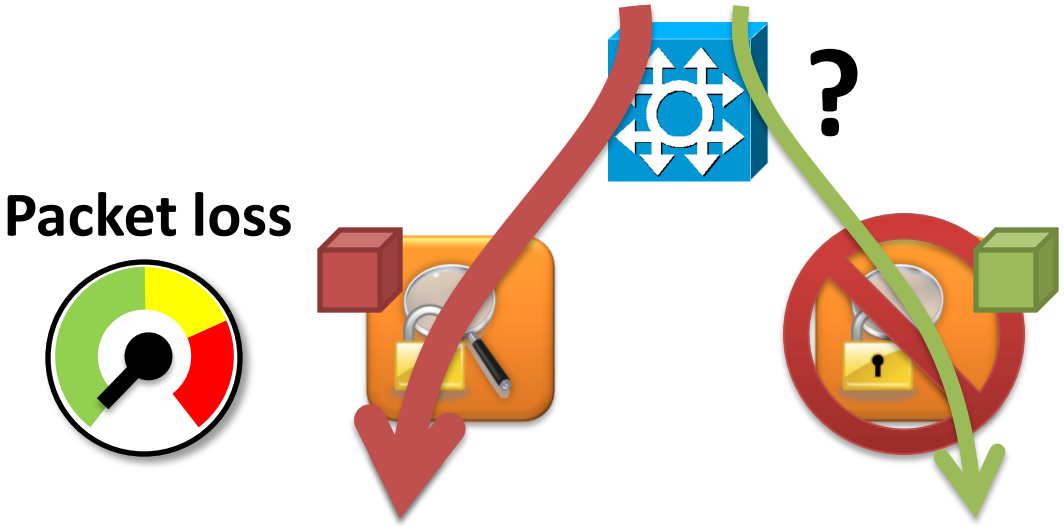
	1. SLAs	2. Cost	3. Accuracy
Reroute new flows	✗	✓	?

Why NFV + SDN falls short



	1. SLAs	2. Cost	3. Accuracy
Reroute new flows	✗	✓	?
Reroute existing flows	✓	✓	✗

Why NFV + SDN falls short



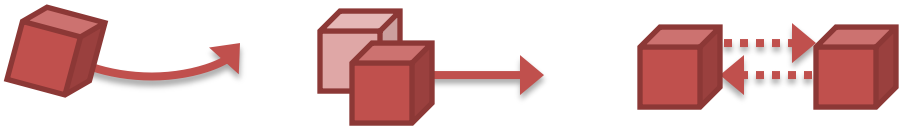
	1. SLAs	2. Cost	3. Accuracy
Reroute new flows	✗	✓	?
Reroute existing flows	✓	✓	✗

Why NFV + SDN falls short

	1. SLAs	2. Cost	3. Accuracy
Reroute new flows	✗	✓	?
Reroute existing flows	✓	✓	✗
Wait for flows to die	✓	✗	✓

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

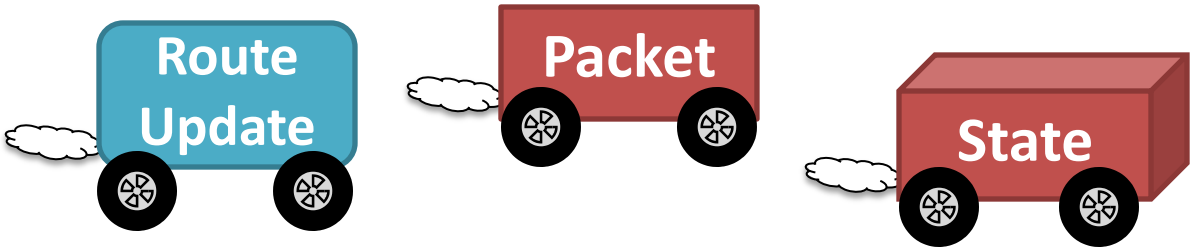
Outline

- Motivation and requirements
- Challenges
- OpenNF architecture
- Evaluation

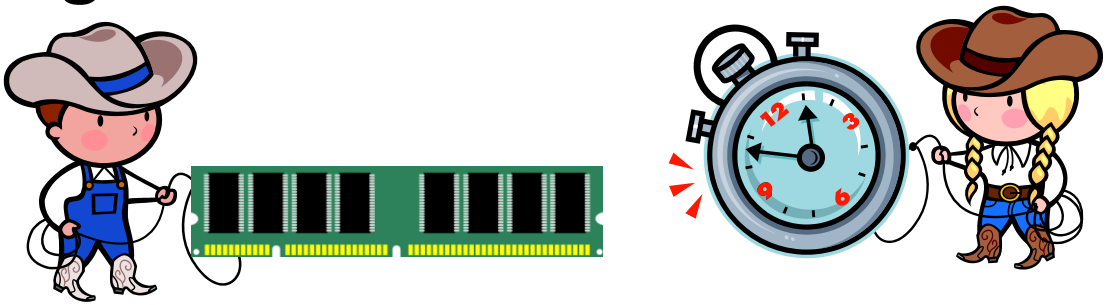
Challenges

1. Supporting many NFs with minimal changes

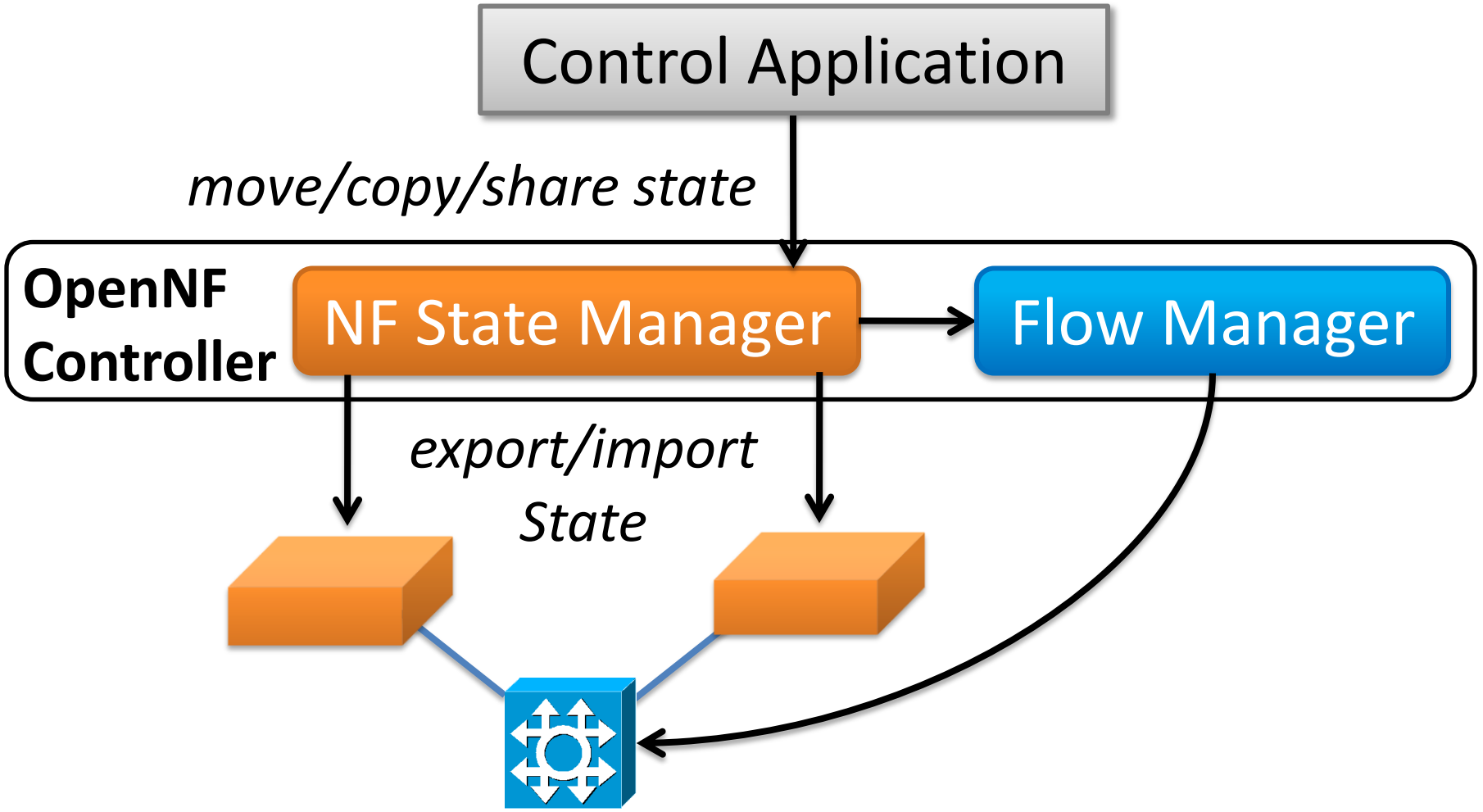
2. Dealing with race conditions



3. Bounding overhead

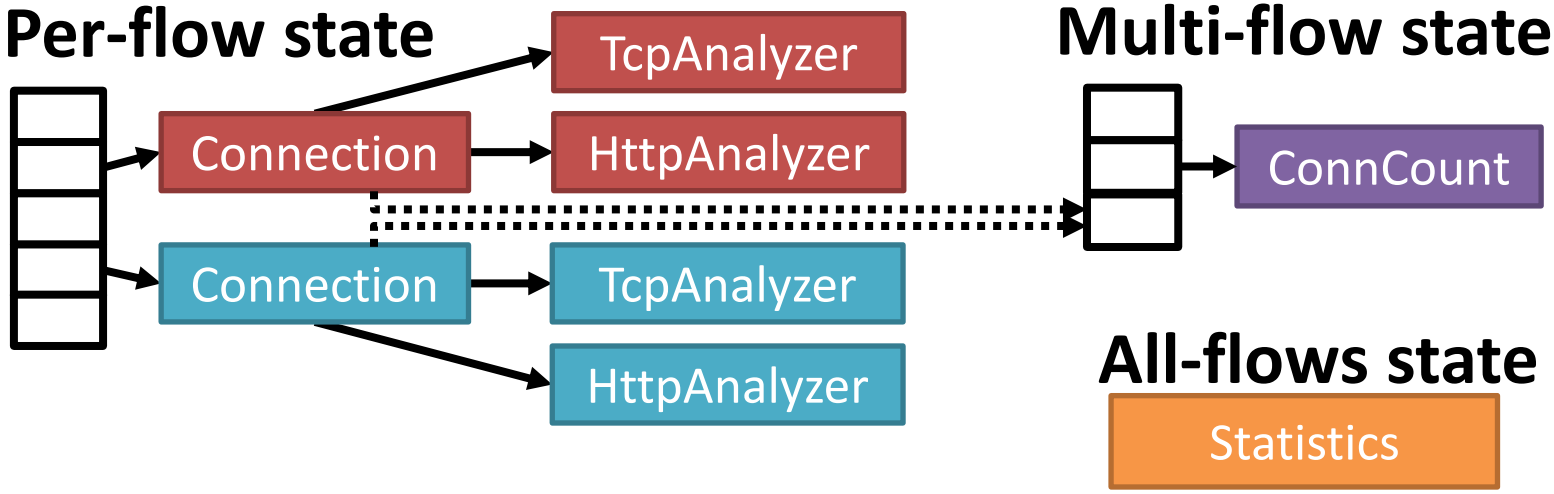


OpenNF overview



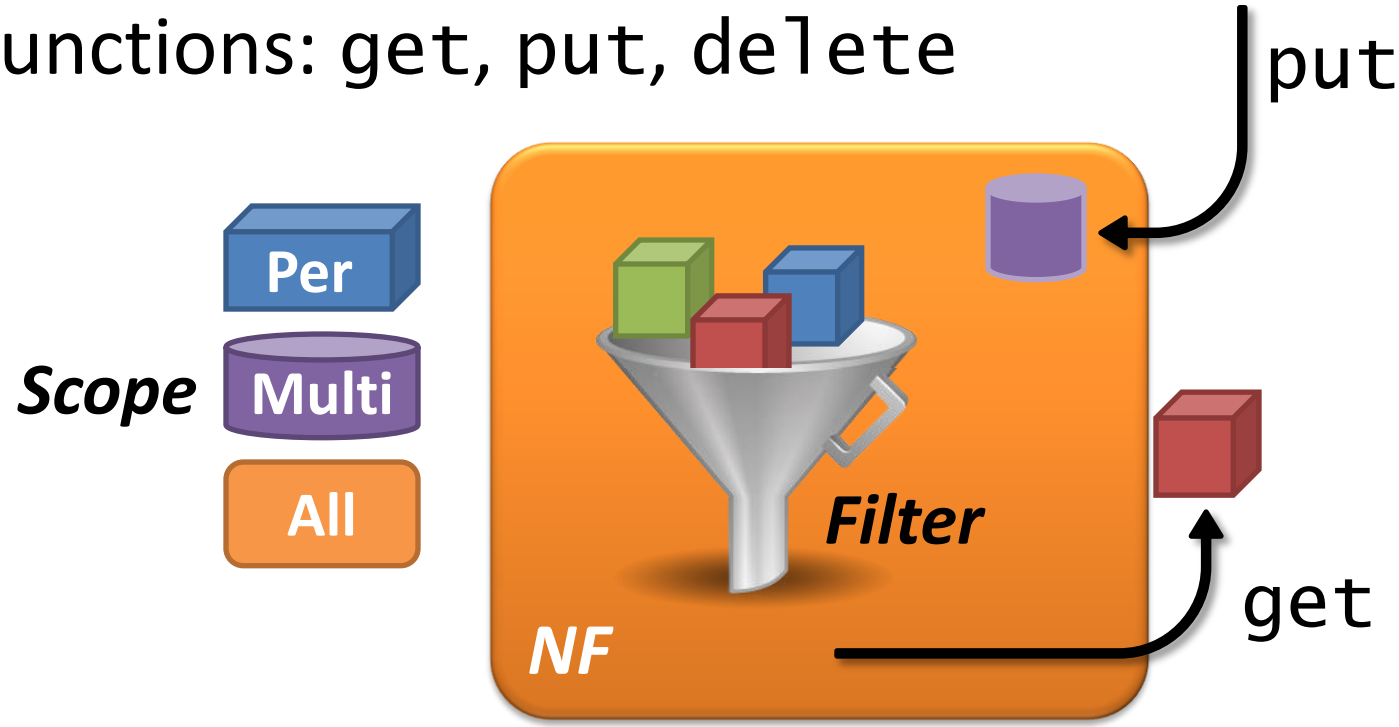
NF state taxonomy

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



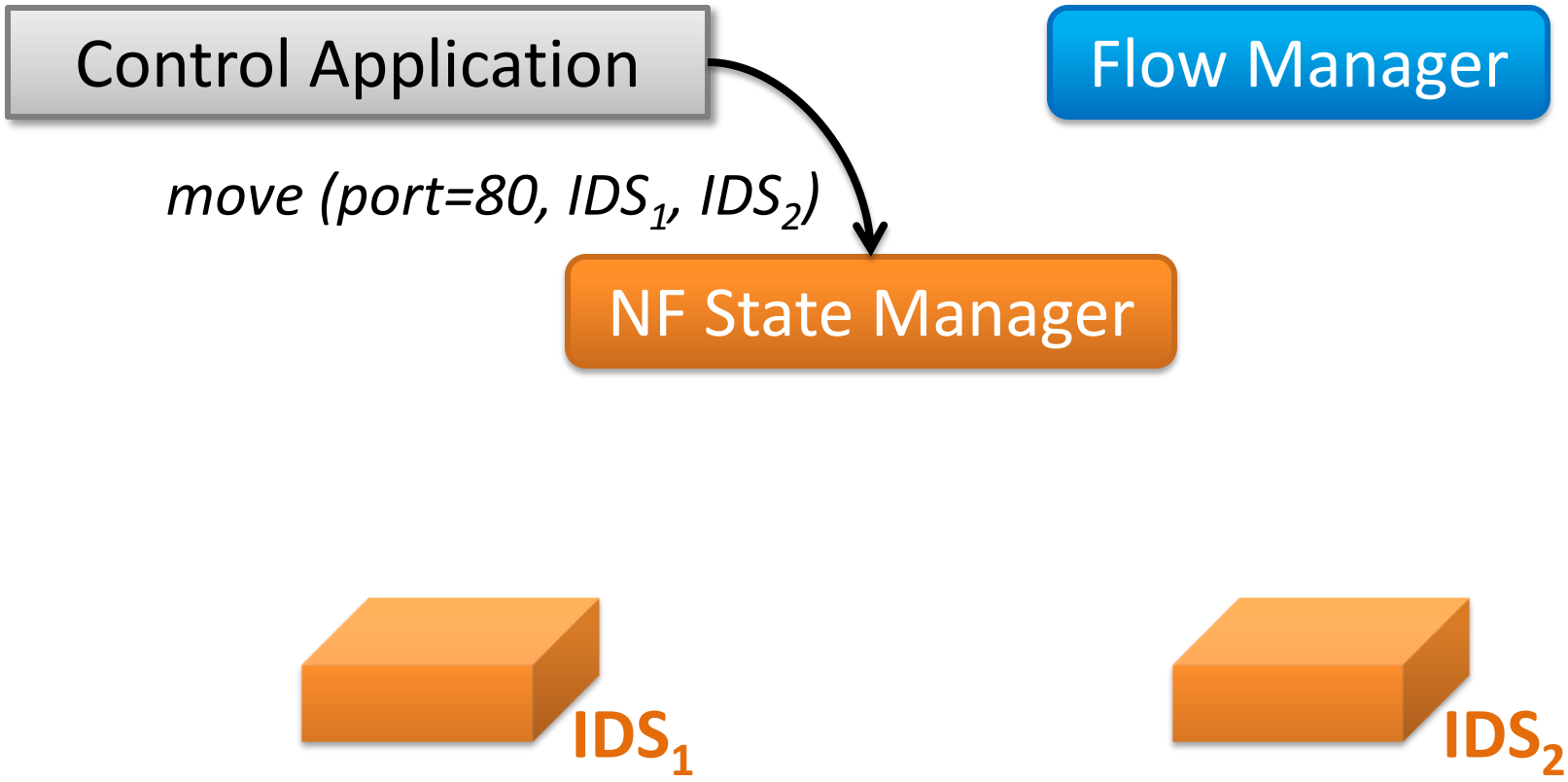
NF API: export/import state

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

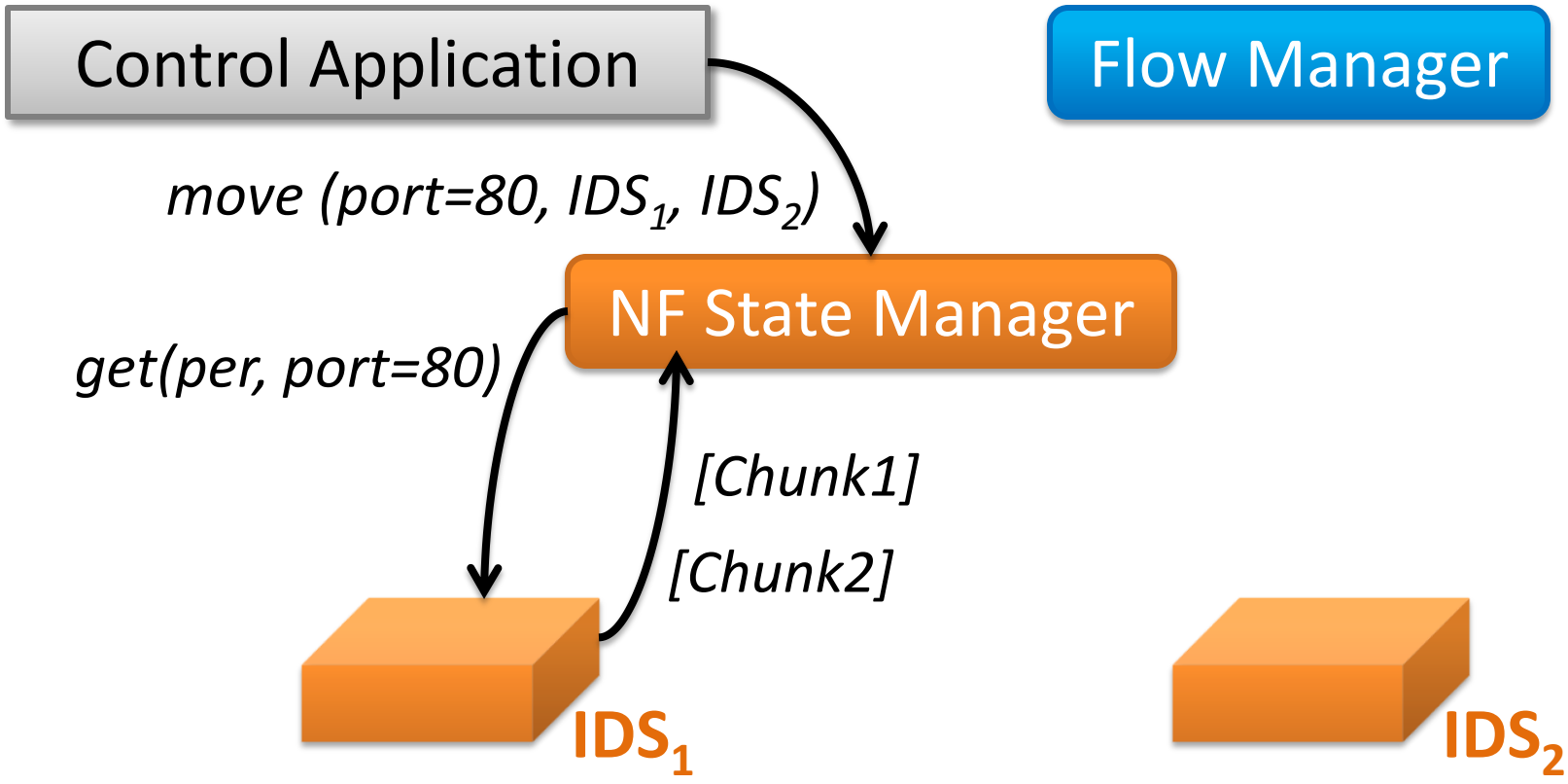


No need to expose/change internal state organization!

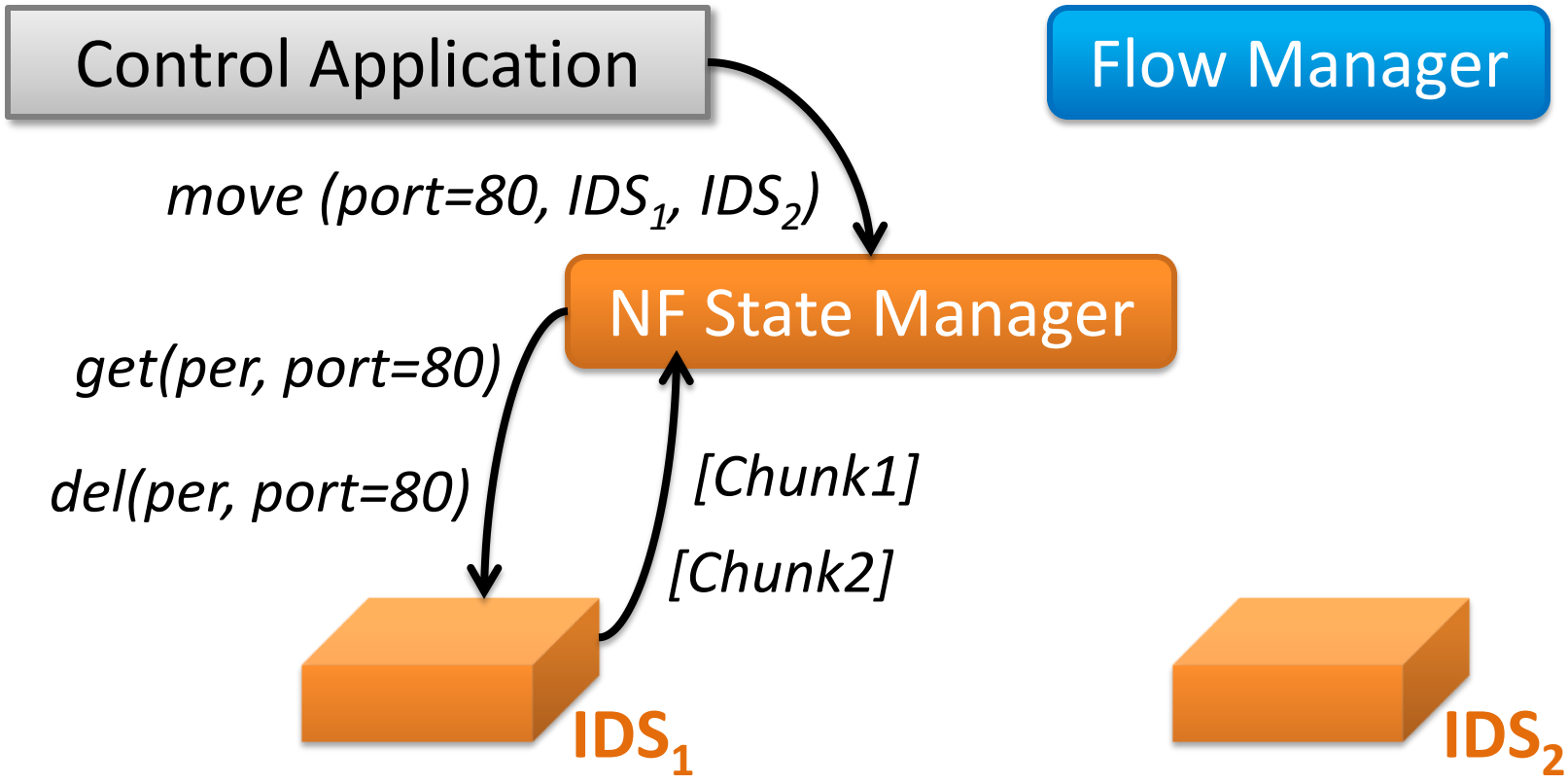
Control operations: move



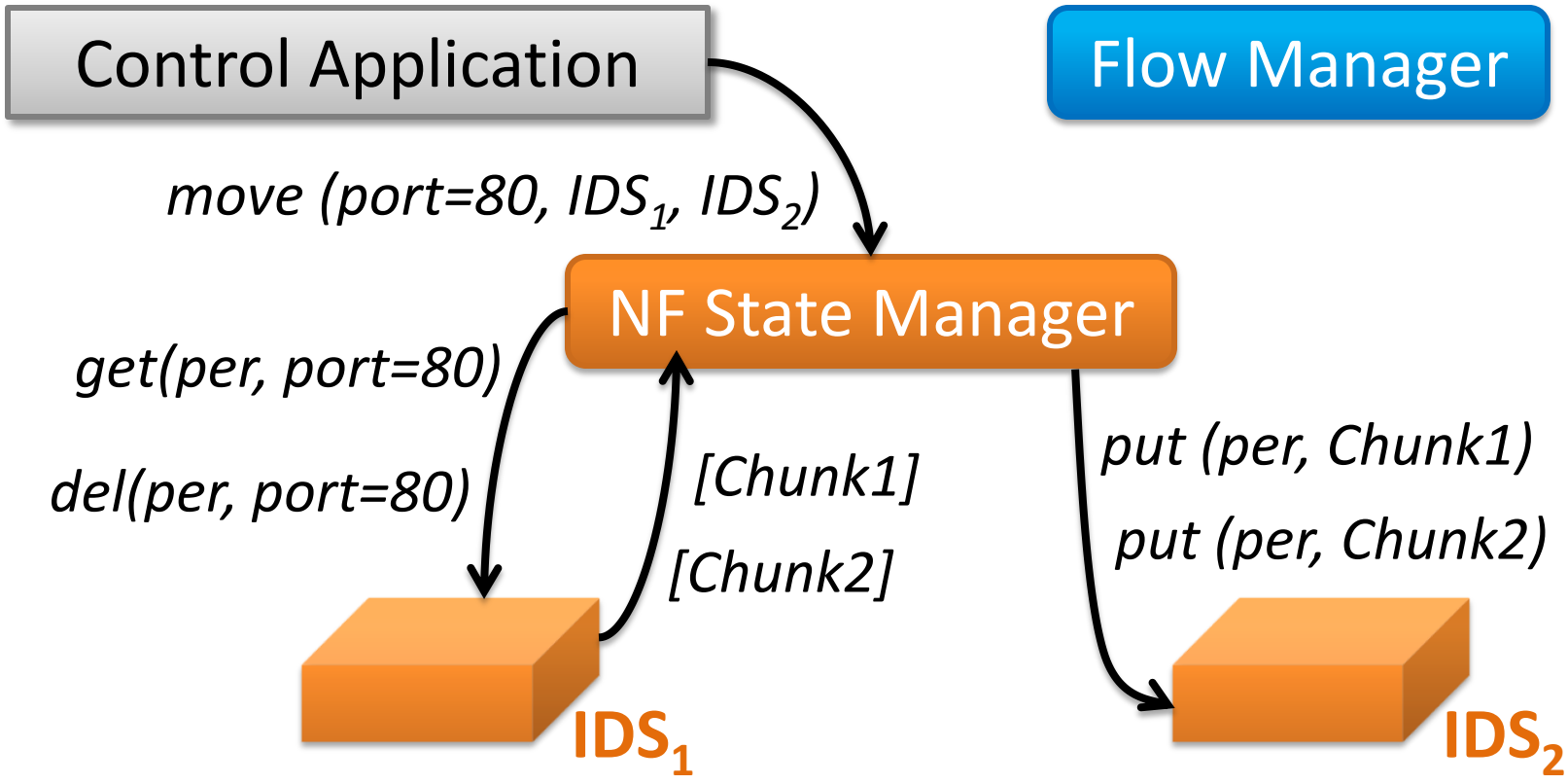
Control operations: move



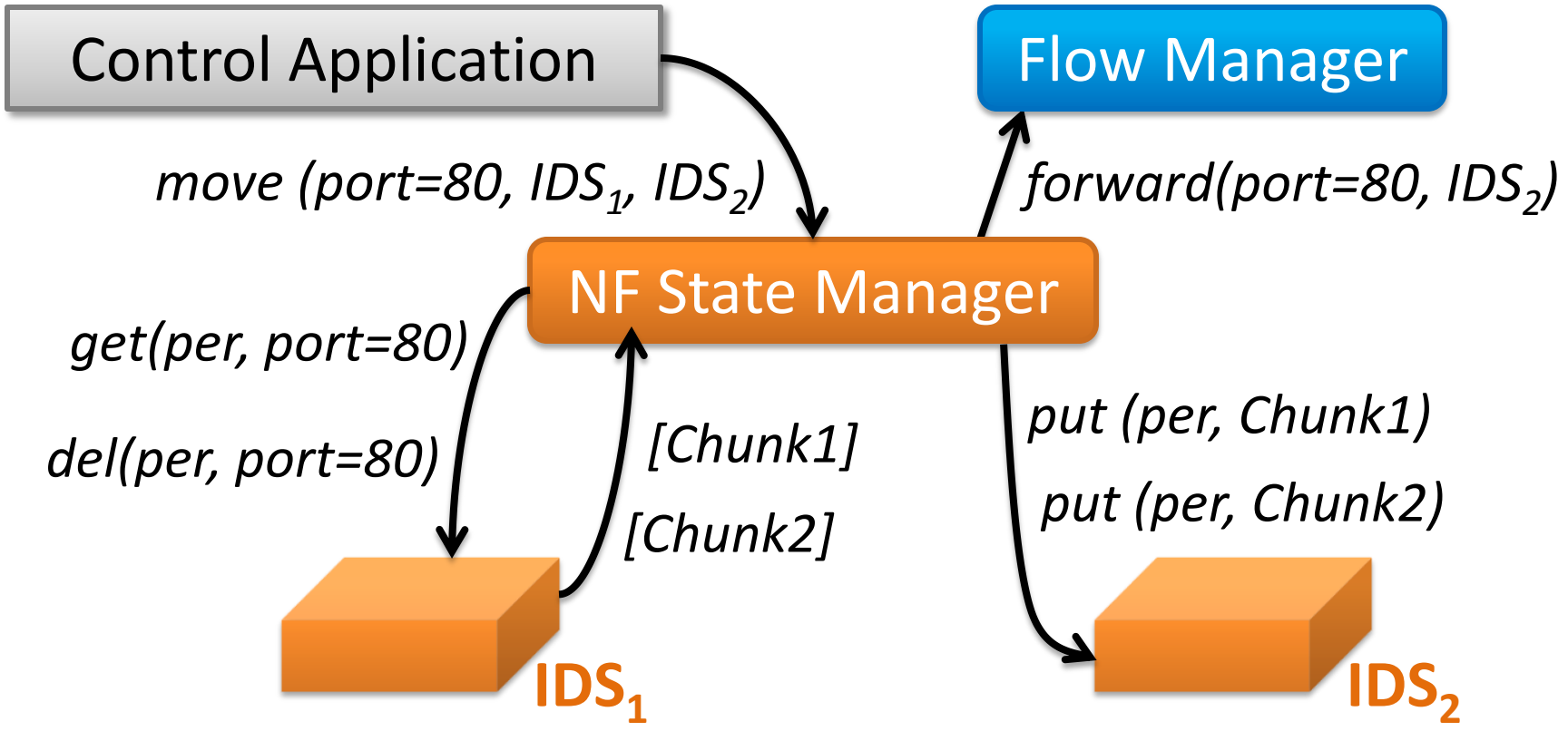
Control operations: move



Control operations: move



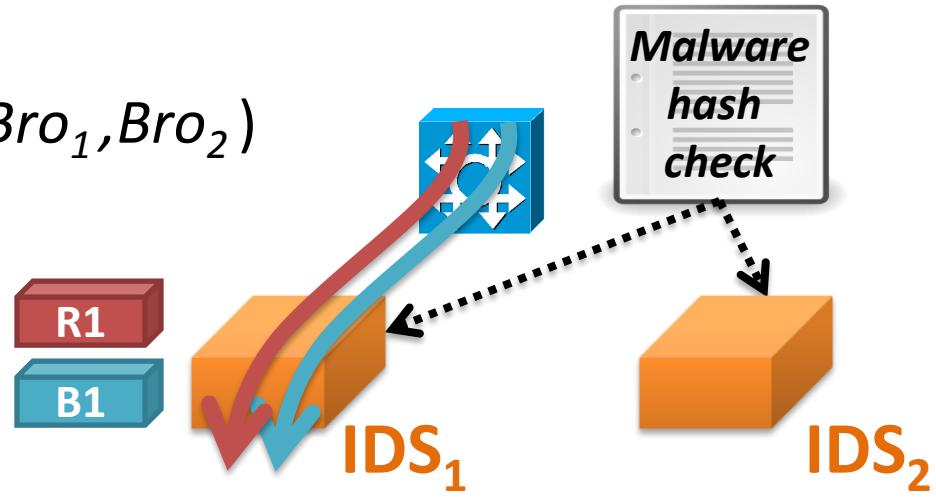
Control operations: move



Also provide copy and share

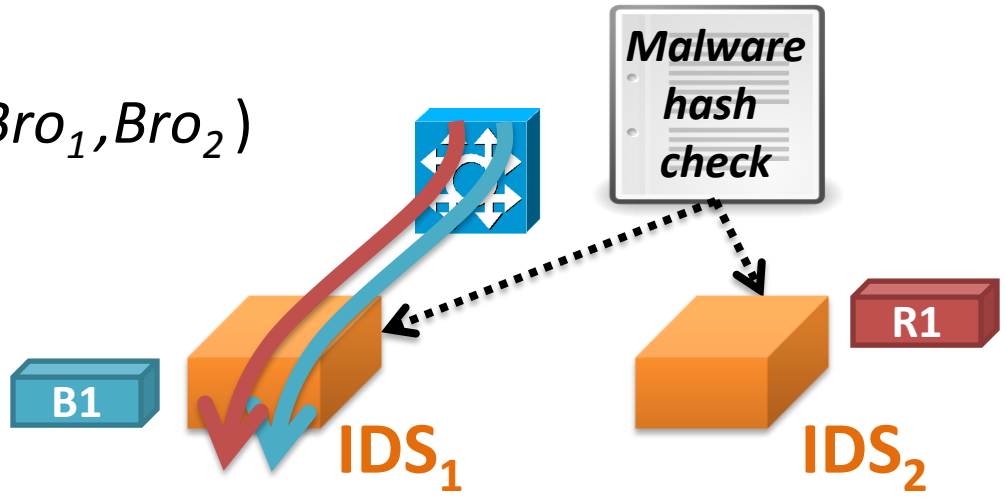
Lost updates during move

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

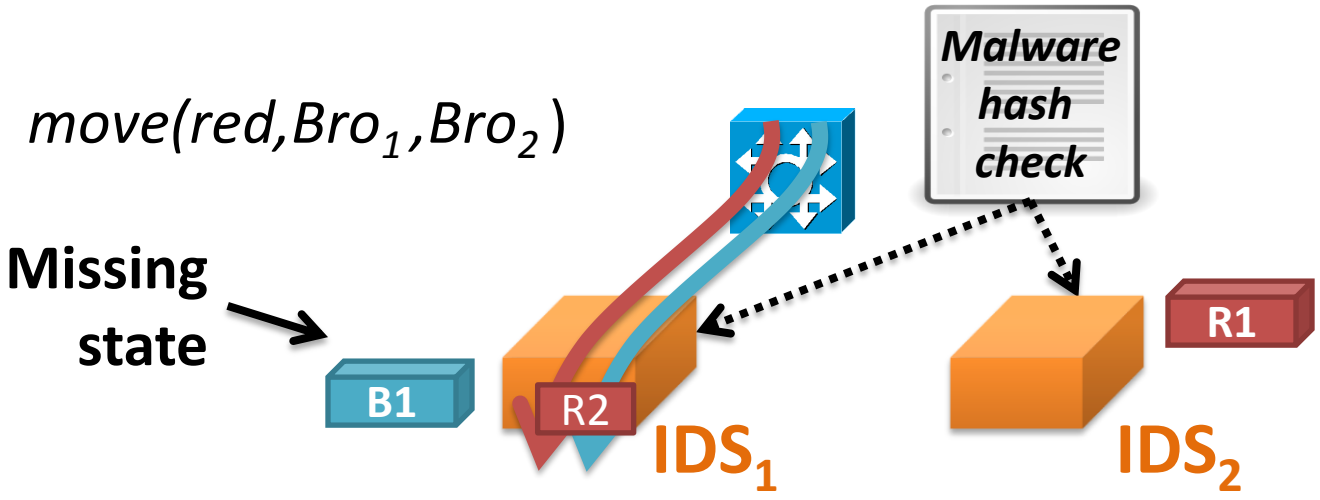


Lost updates during move

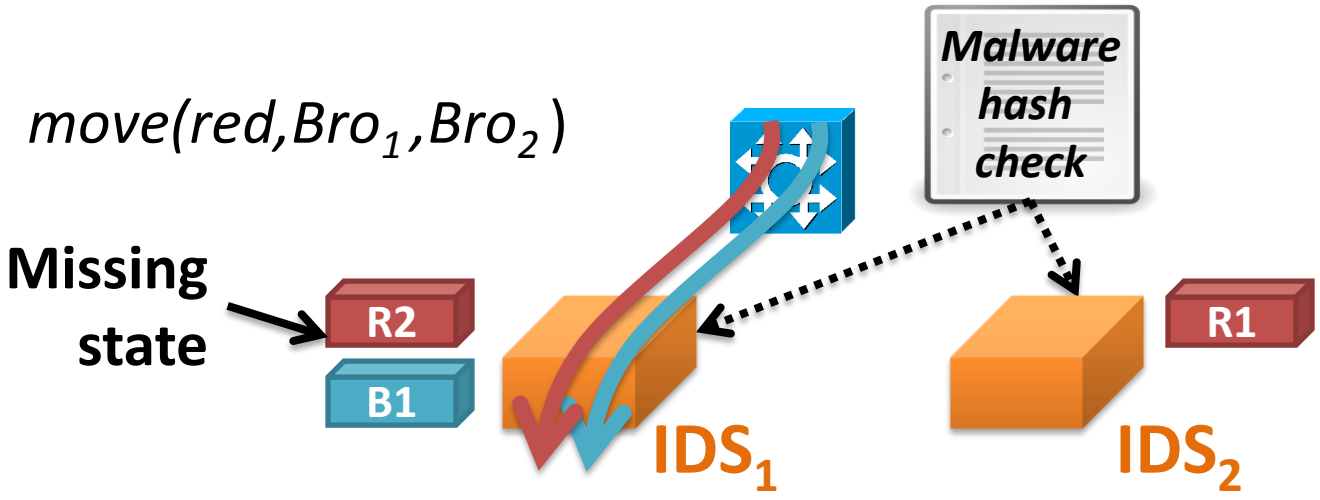
move(red,Bro$_1$,Bro$_2$)



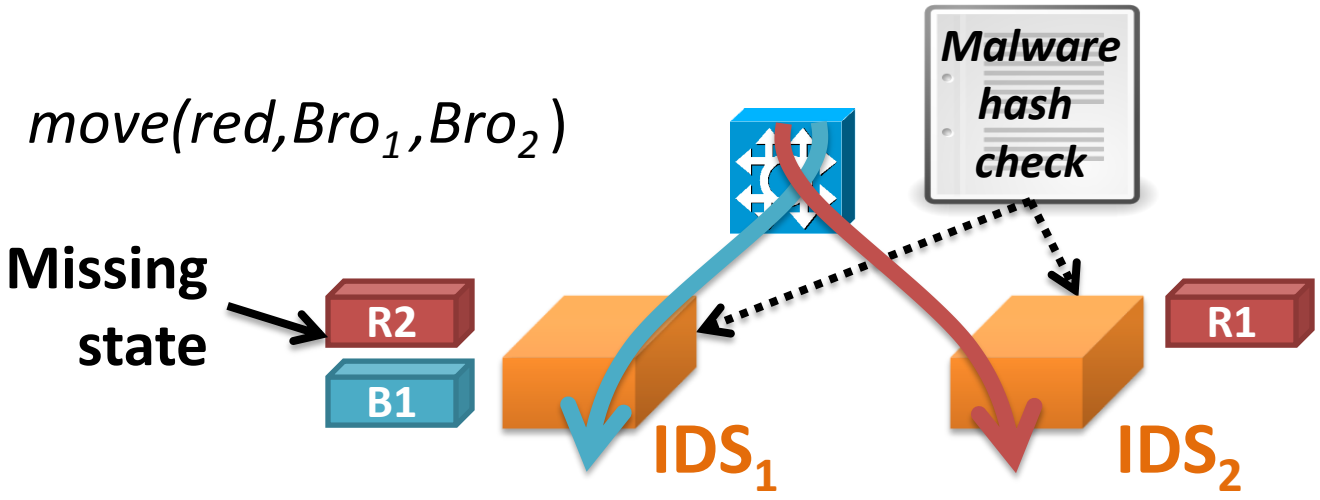
Lost updates during move



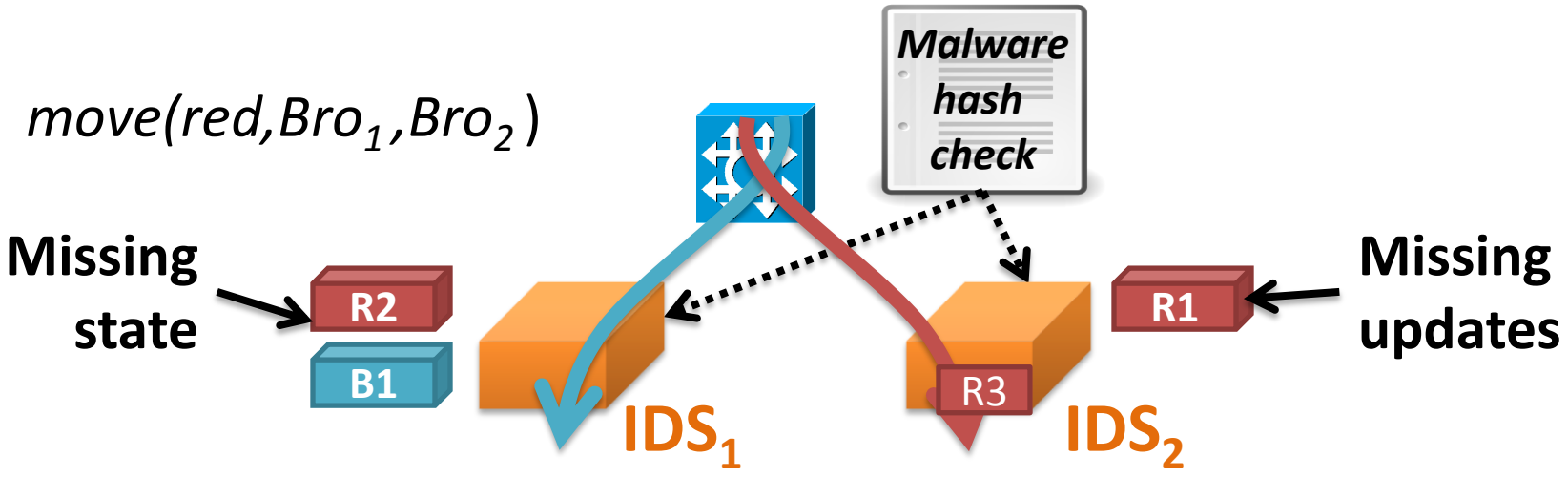
Lost updates during move



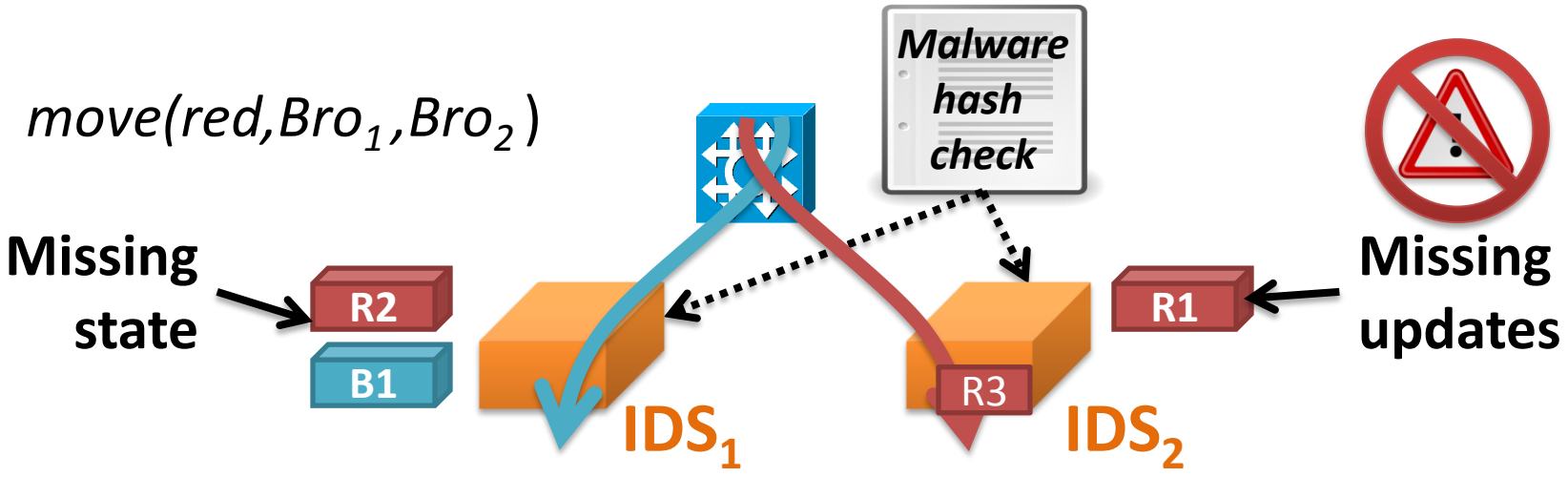
Lost updates during move



Lost updates during move

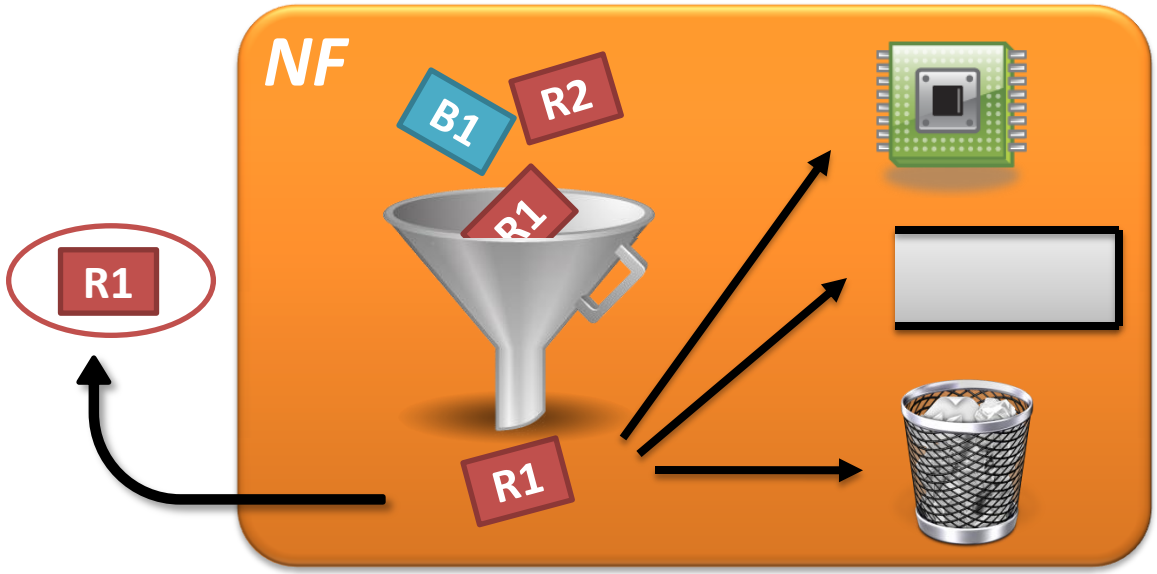


Lost updates during move



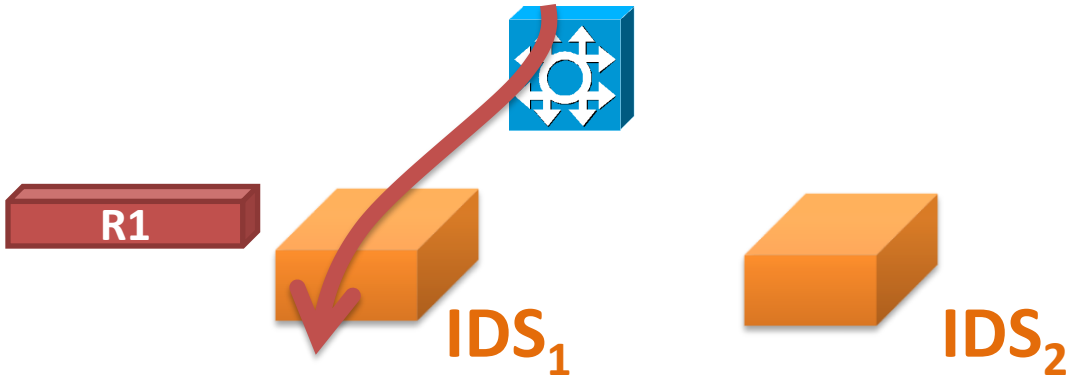
Loss-free: All state updates should be reflected in the transferred state, and all packets should be processed

NF API: observe/prevent updates using events



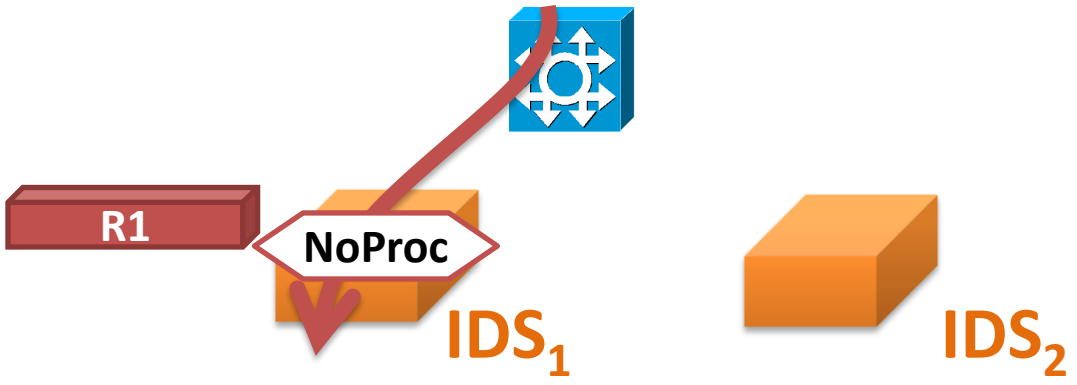
Only need to change an NF's receive packet function!

Use events for loss-free move



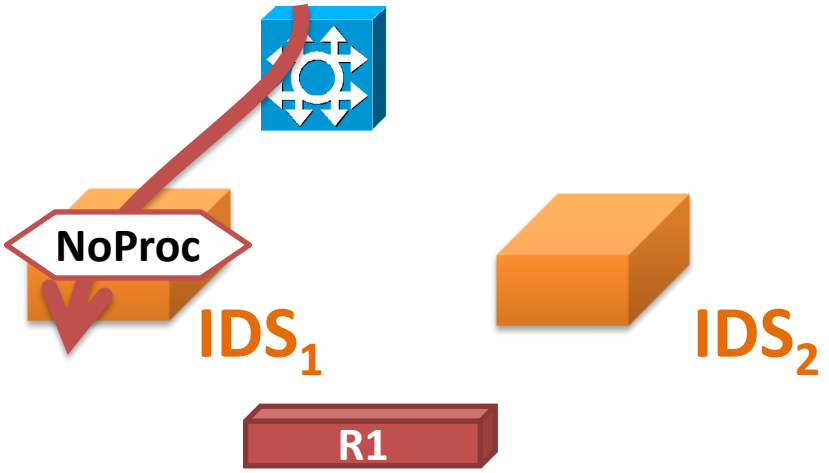
Use events for loss-free move

1. `enableEvents(red,noproc)` on IDS_1



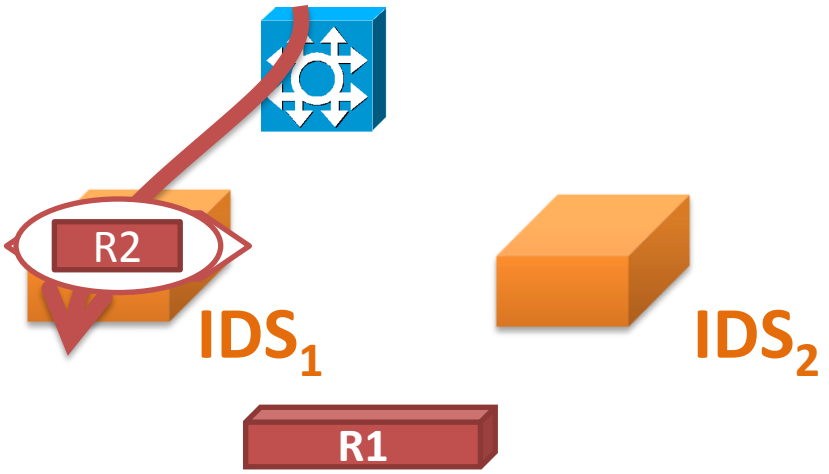
Use events for loss-free move

1. `enableEvents(red,noproc)` on IDS_1
2. `get/delete` on IDS_1



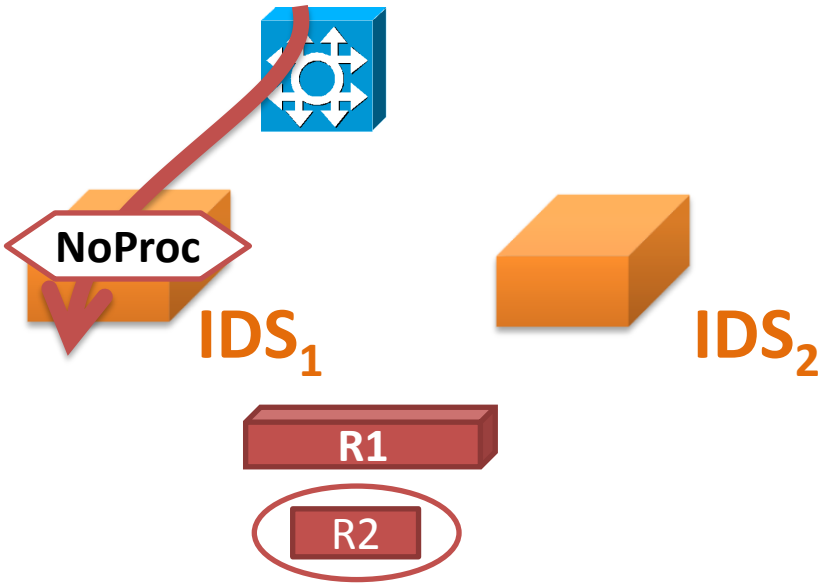
Use events for loss-free move

1. `enableEvents(red,noproc)` on IDS_1
2. `get/delete` on IDS_1



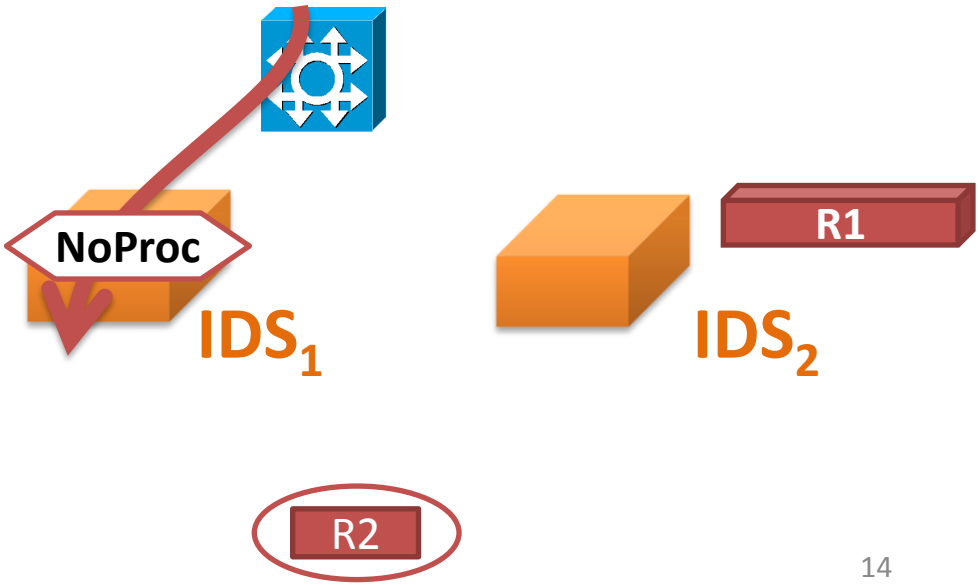
Use events for loss-free move

1. `enableEvents(red,noproc)` on IDS_1
2. `get/delete` on IDS_1
3. Buffer events at controller



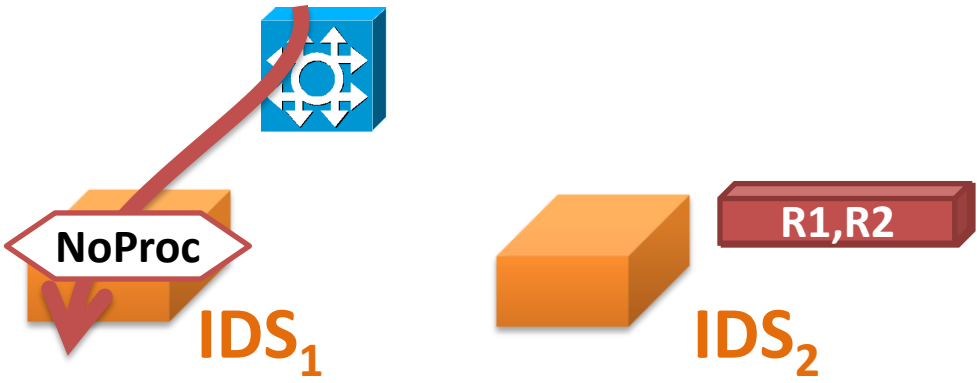
Use events for loss-free move

1. `enableEvents(red,noproc)` on IDS_1
2. `get/delete` on IDS_1
3. Buffer events at controller
4. `put` on IDS_2



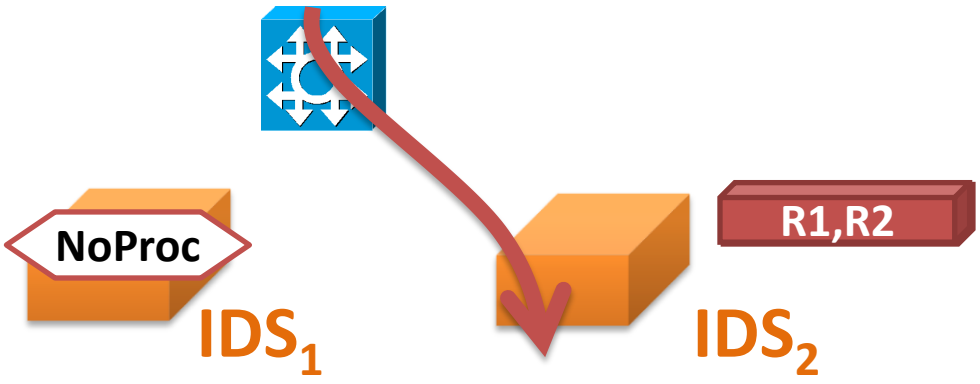
Use events for loss-free move

1. `enableEvents(red,noproc)` on IDS_1
2. `get/delete` on IDS_1
3. Buffer events at controller
4. `put` on IDS_2
5. Flush packets in events to IDS_2



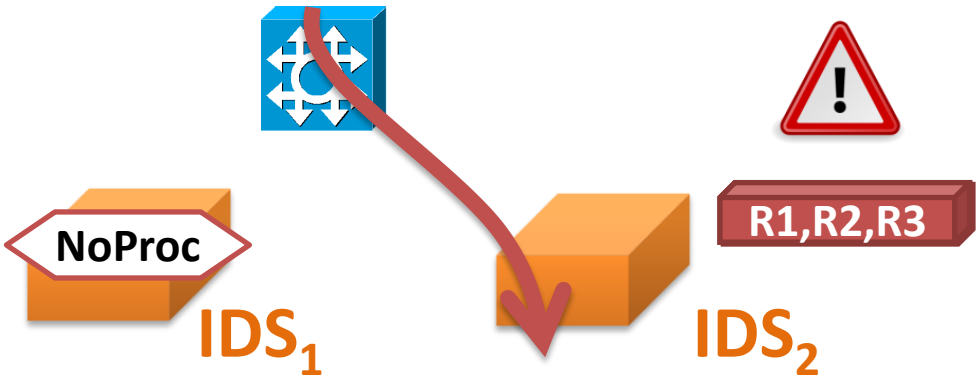
Use events for loss-free move

1. `enableEvents(red,noproc)` on IDS_1
2. `get/delete` on IDS_1
3. Buffer events at controller
4. `put` on IDS_2
5. Flush packets in events to IDS_2
6. Update forwarding



Use events for loss-free move

1. `enableEvents(red,noproc)` on IDS_1
2. `get/delete` on IDS_1
3. Buffer events at controller
4. `put` on IDS_2
5. Flush packets in events to IDS_2
6. Update forwarding



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

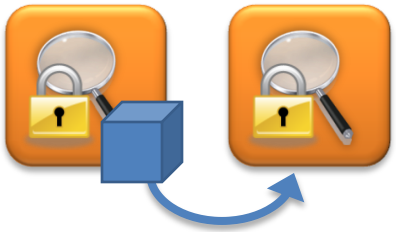
Evaluation: benefits for elastic scaling

- 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 in after state is moved
  - Wait for flows to die: scale in delayed 25+ minutes

Conclusion

- Realizing SLAs + cost + accuracy requires quick, safe control of internal network function state



- OpenNF provides flexible and efficient control with few modifications to NFs

Learn more and try it!
http://opennf.cs.wisc.edu