Composition of SDN applications: Options/challenges for real implementations

Arne Schwabe

Pedro A. Aranda Gutiérrez

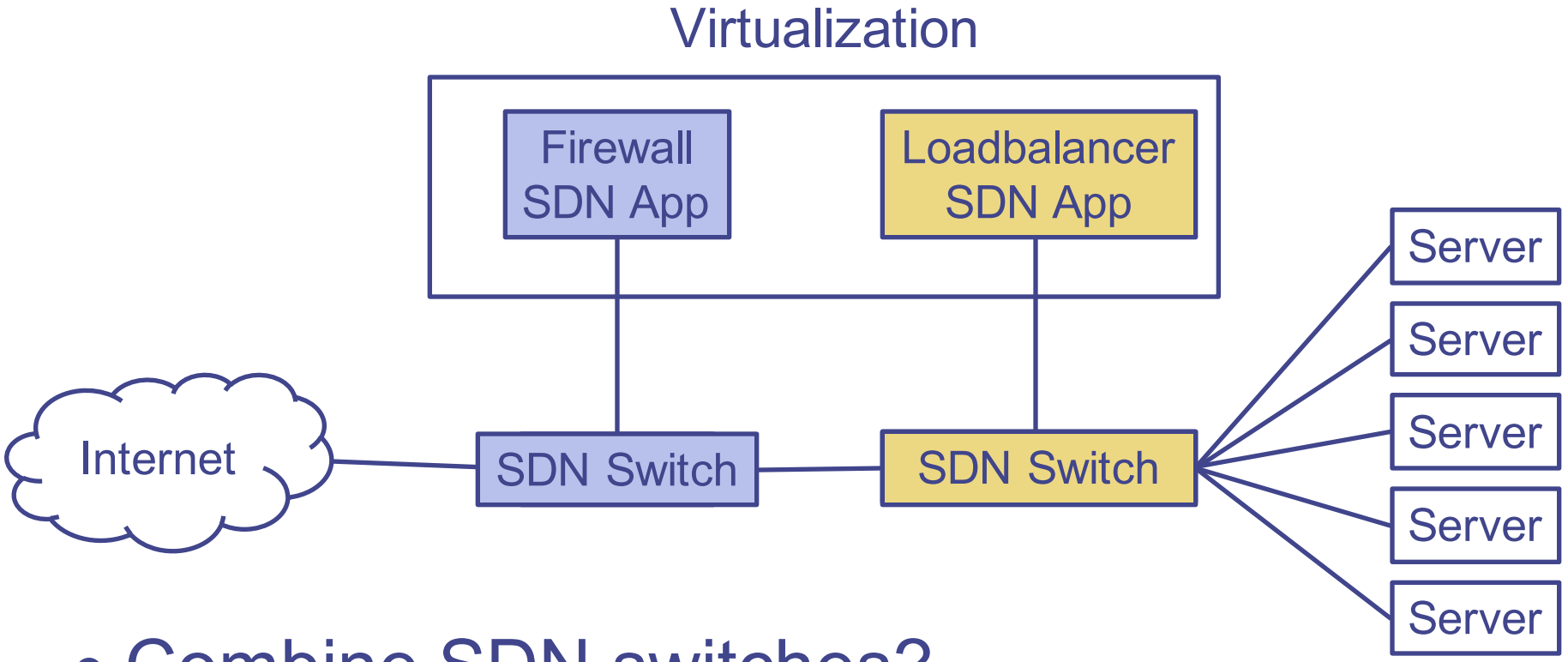
Holger Karl

Computer Networks Group

Universität Paderborn

Modernizing the setup

- Simple standard network setup
- Replace boxes with SDN

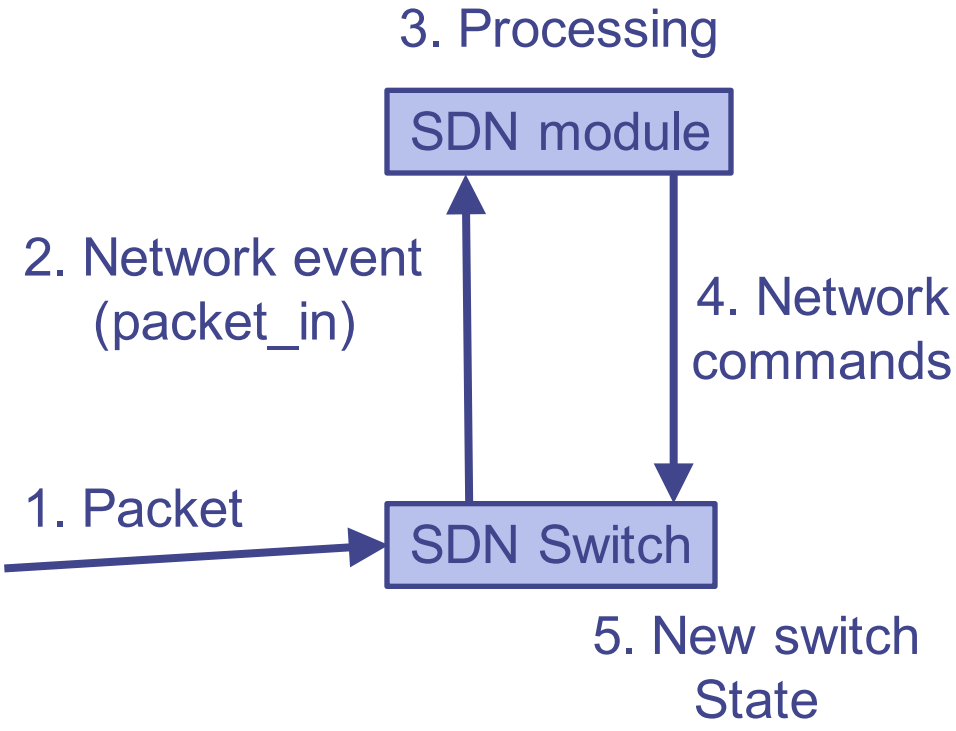


- Combine SDN switches?

Overview

- Motivation
- **Composing SDN apps in general**
- OpenFlow specific composition
- Conclusion

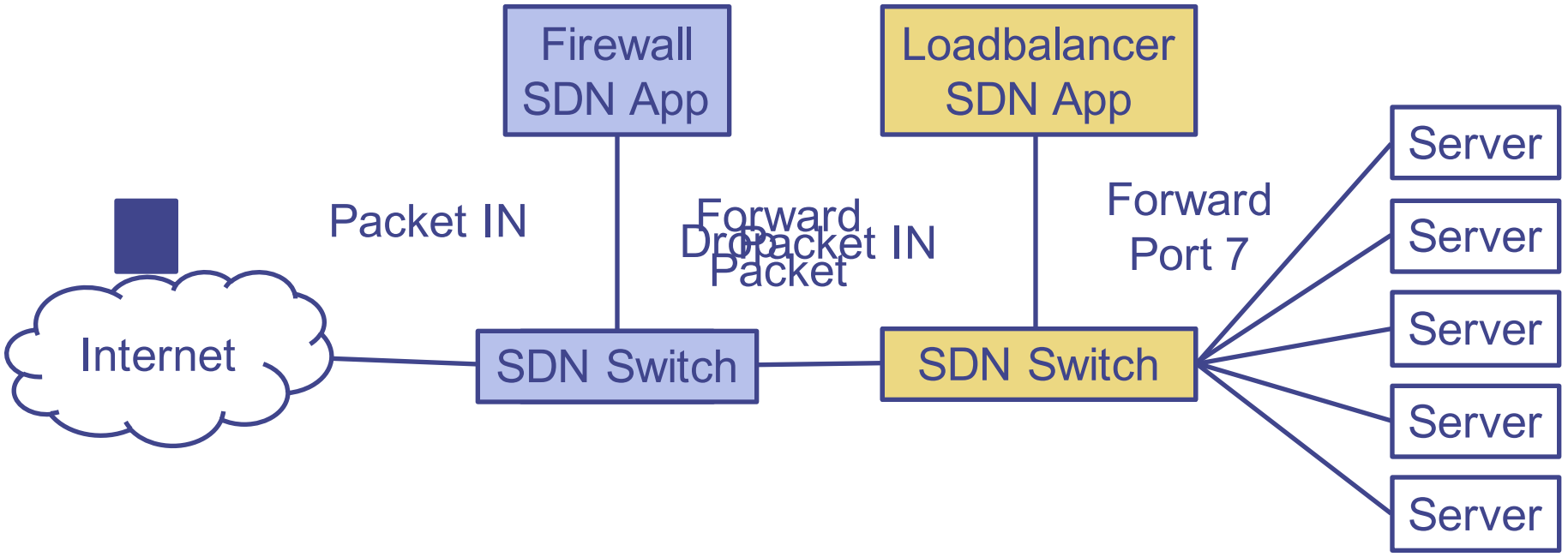
What does an SDN app do?



See modules as stateful function:

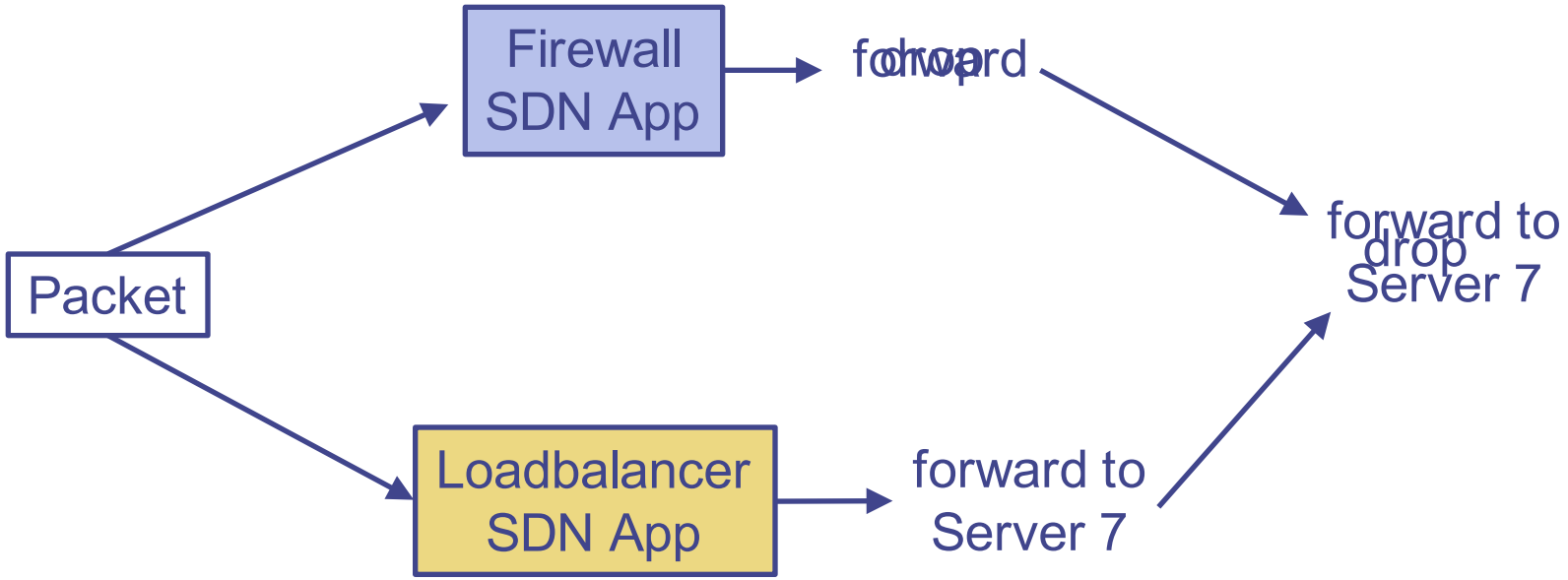
M: *event* → network command

Modernized setup in detail



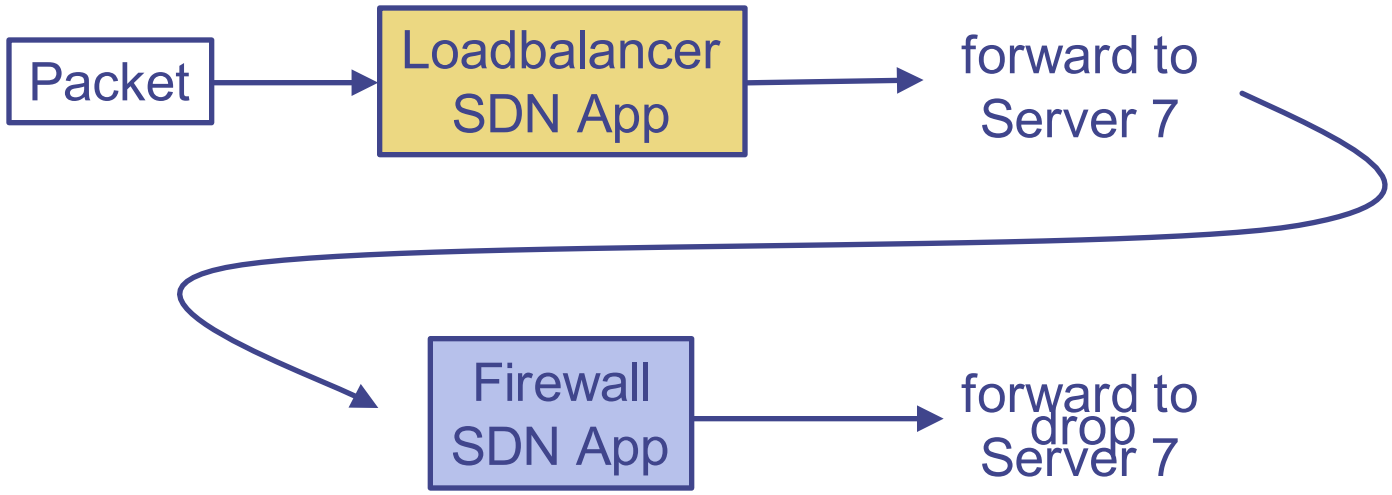
Idea: Parallel composition

- Reuse existing SDN Apps
- Combine results of the Apps: Parallel



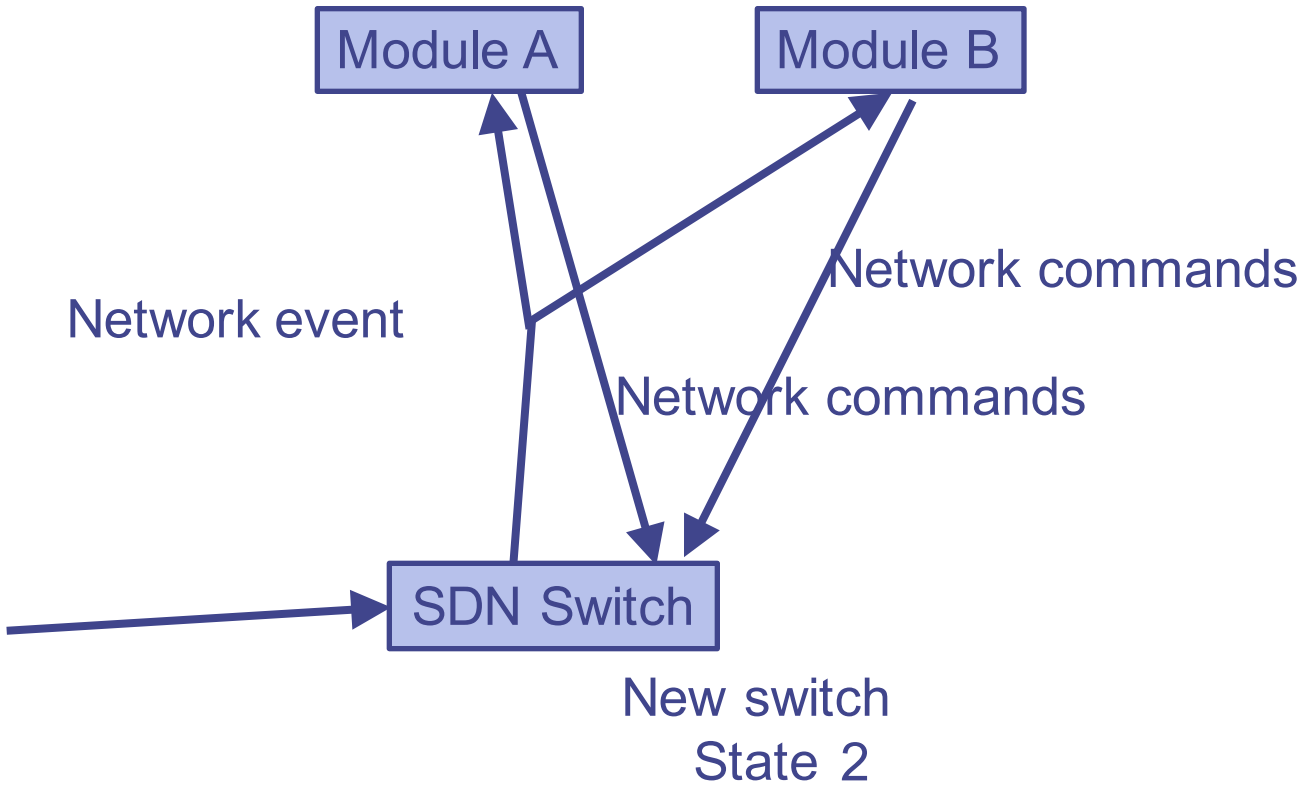
Idea: Serial composition

- Explicitly let the firewall have the final decision
- Combine results of the Apps: Serial



Two more or more SDN apps

- SDN controller scenario



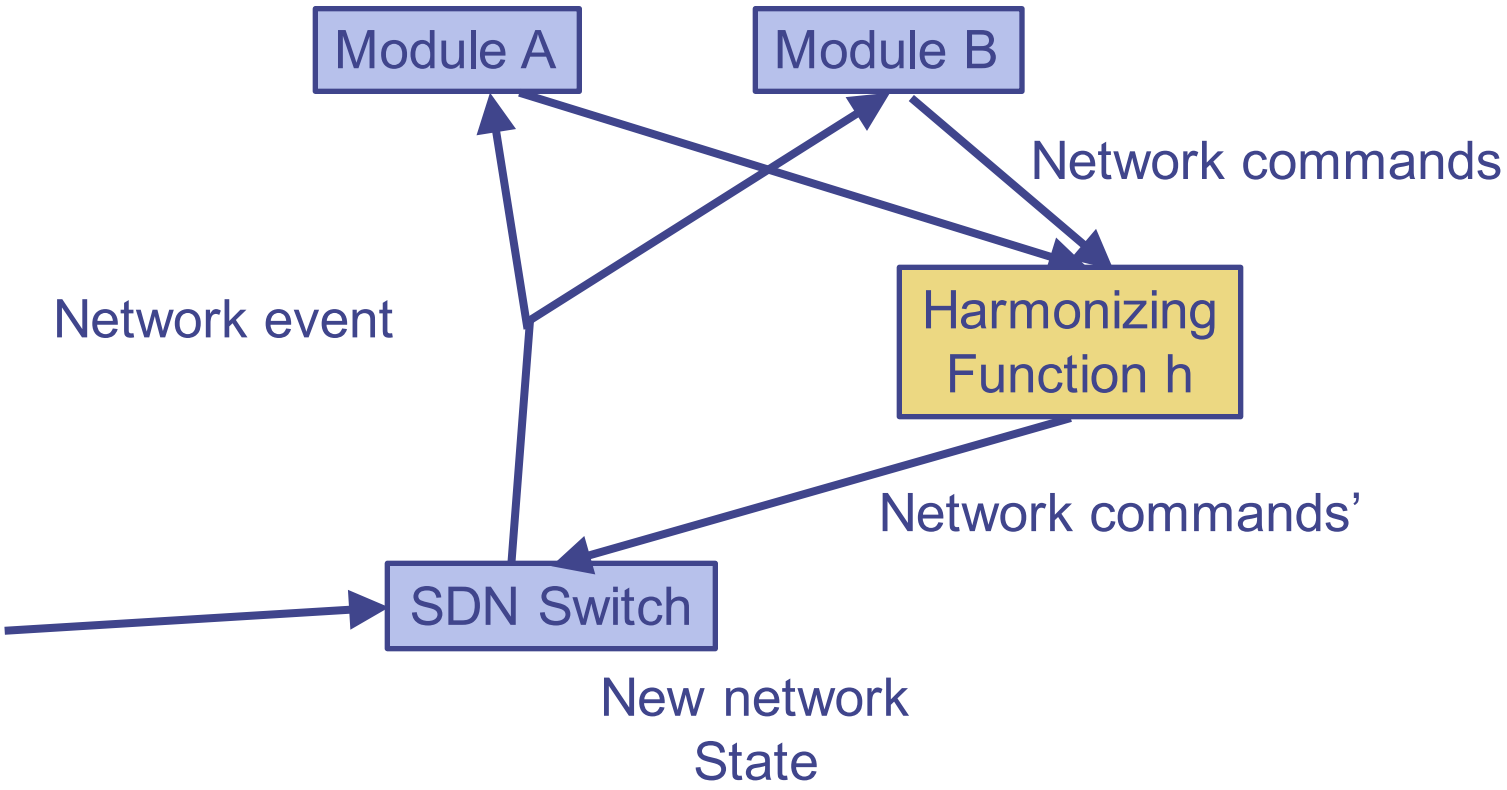
Challenges

- Order of network commands not predictable
  - Transitional states
- Network commands might conflict

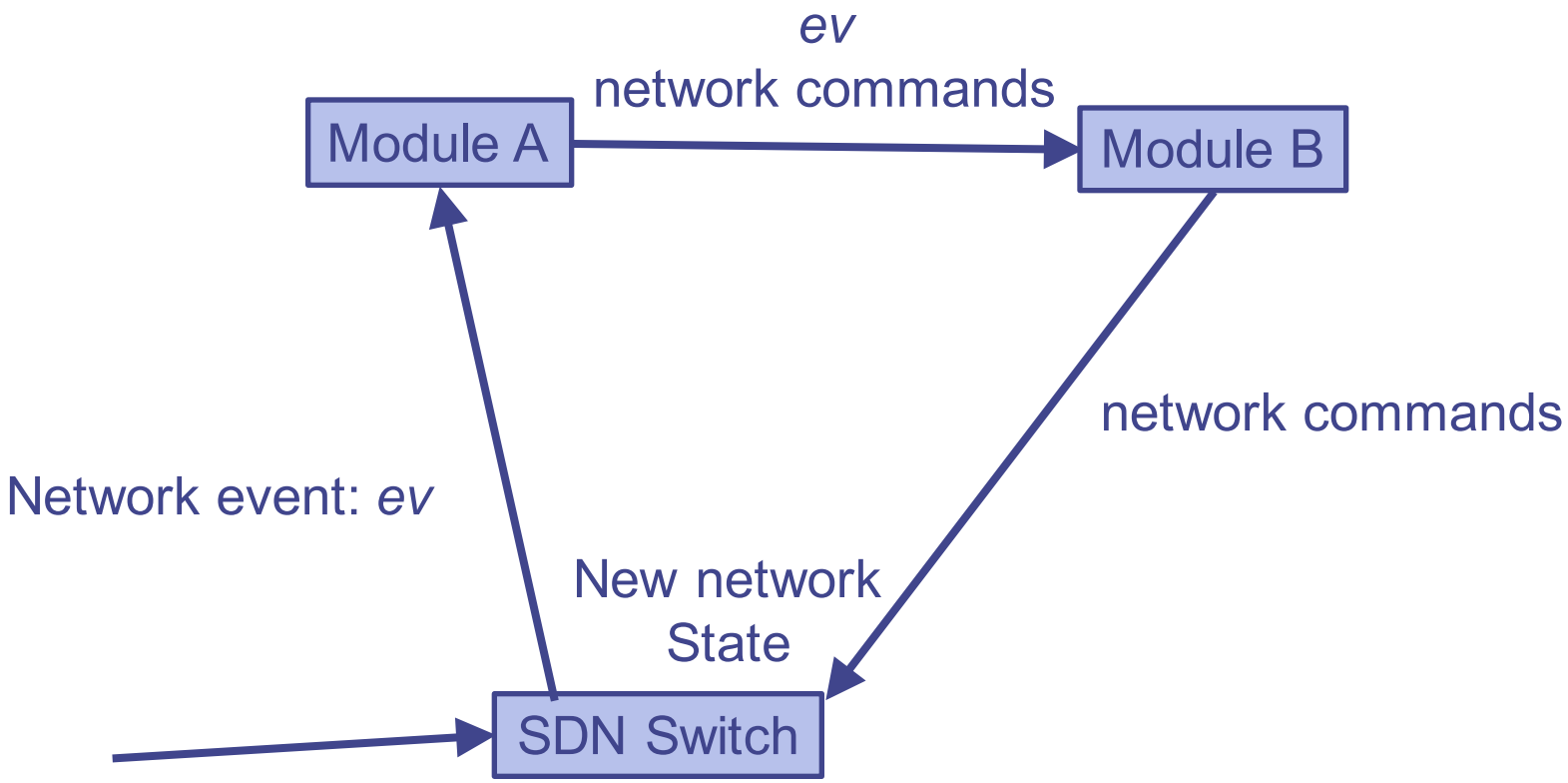
- Introduce harmonizing function
  - Network hypervisor/SDN frameworks
  - Example: Partition network by modules

Harmonizing output

h: network commands → network commands



Serial composition

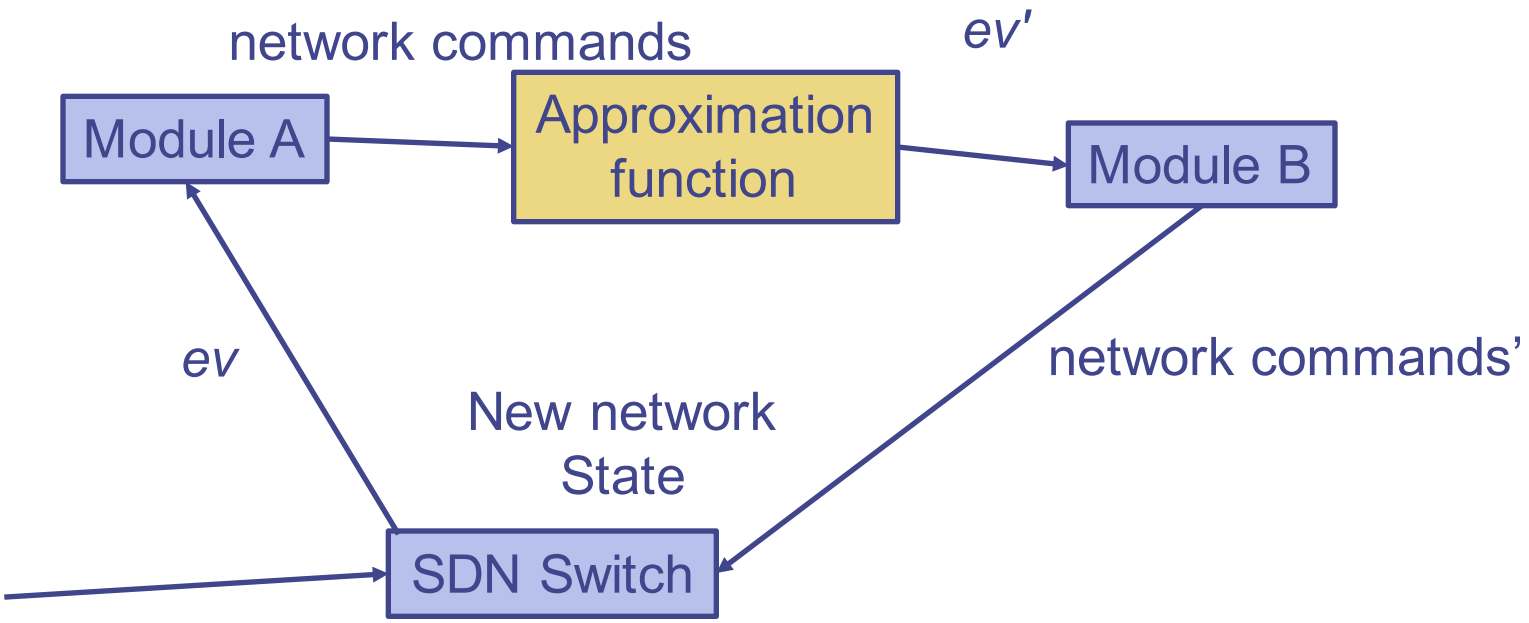


Signature of Module B changes:

M: event × command → command

Approximate Serial composition

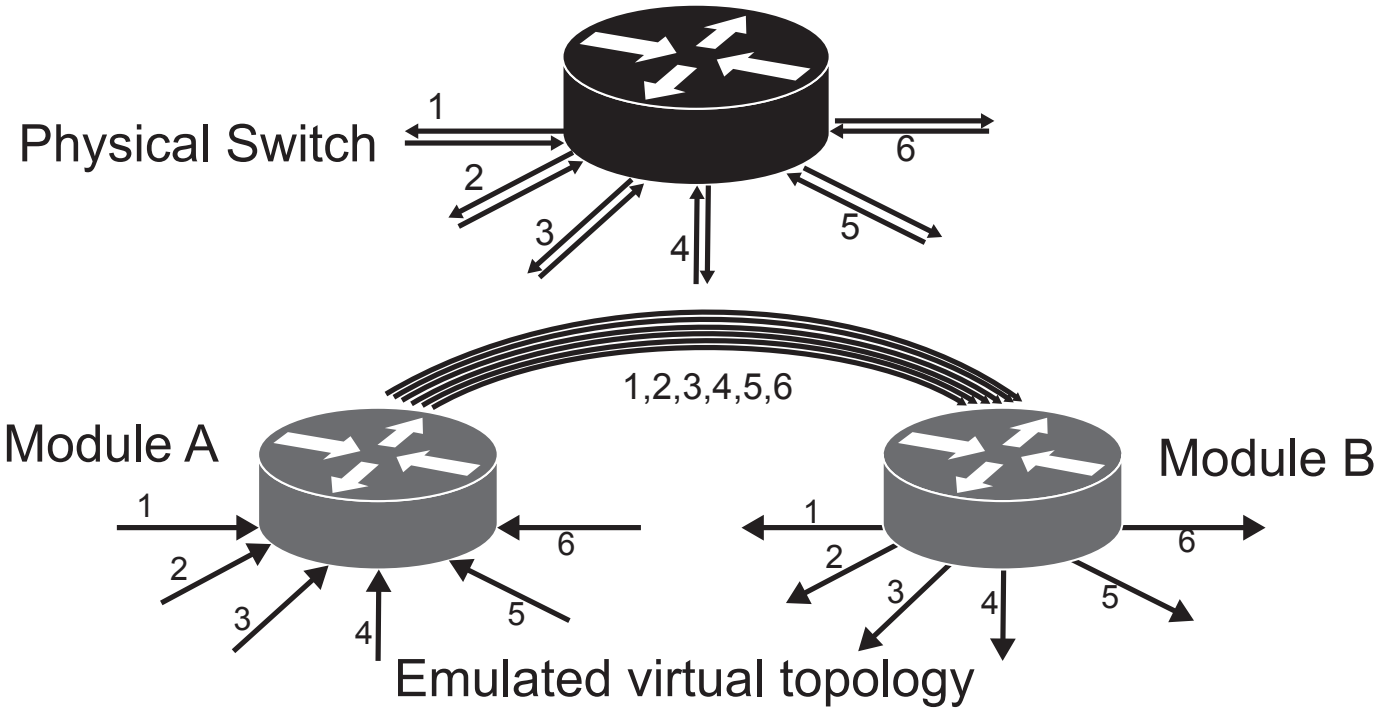
Approximate event for B:
α: command → event



Not everything representable in ev':
Example: input port

Approximate serial example

- Emulate topology to generate new packet In



- Output port a becomes input port of B

Overview

- Motivation
- Composing SDN apps in general
- **OpenFlow specific composition**
- Conclusion

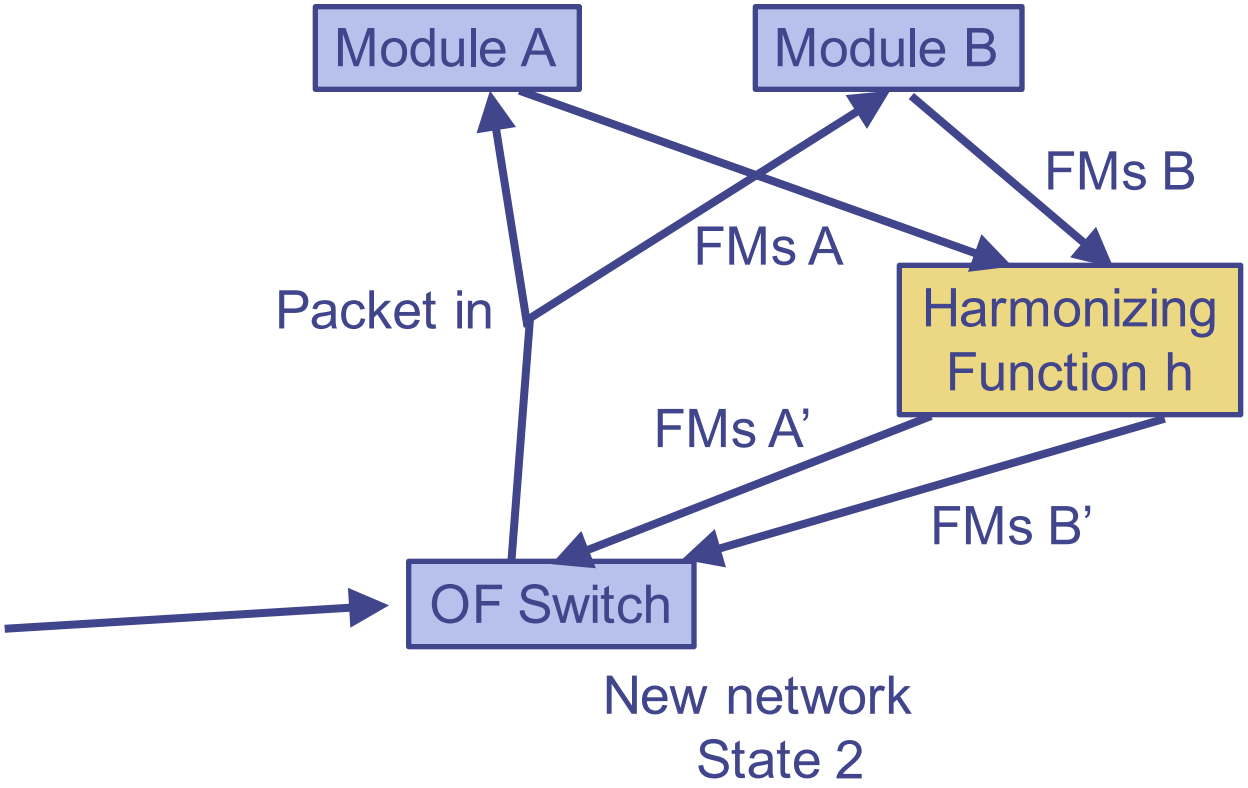
OpenFlow

- De facto standard
- Desire to reuse for composition
- Question: Does it work?

- Network Event: Packet in
- Network commands
  - Flow mod
  - Packet Out

Harmonizing output (OpenFlow Version)

Again with h: command → command



Parallel composition

- No relation between Packet in and network commands
  - No ”take all inputs, combine”
  - Makes harmonizing more difficult/less useful
- Transient state even with harmonizing
  - Not always a problem (Partitioning)
- Network commands without event

Serial composition (OpenFlow)

- Packet_IN has only in_port and packet
- Workaround: Apply actions (e.g. port, IP) to packet contents
- Other properties lost: E.g. flowmods

Making OpenFlow work (NetIDE)

- Add custom header
- Assign transaction id to network event
- application signal end of transaction
- Restrict allowed behavior
- Concentrate on parallel composition

Implementation/Approaches

- Composition friendly frameworks
  - Pyretic
- Network Hypervisors
  - OpenVirtex
  - FlowVisor
- CoVisor: Full composition
  - Paper does not discuss problems mentioned here

Conclusion

- Composition can work in real world
- API behavior is important

- OpenFlow works poorly
- Modifying OpenFlow for composition:
  - Custom protocol
  - Still much left to desire