CS 356: Computer Network Architectures

Lecture 26: Router hardware, Software defined networking, and programmable routers

[PD] chapter 3.4

Xiaowei Yang

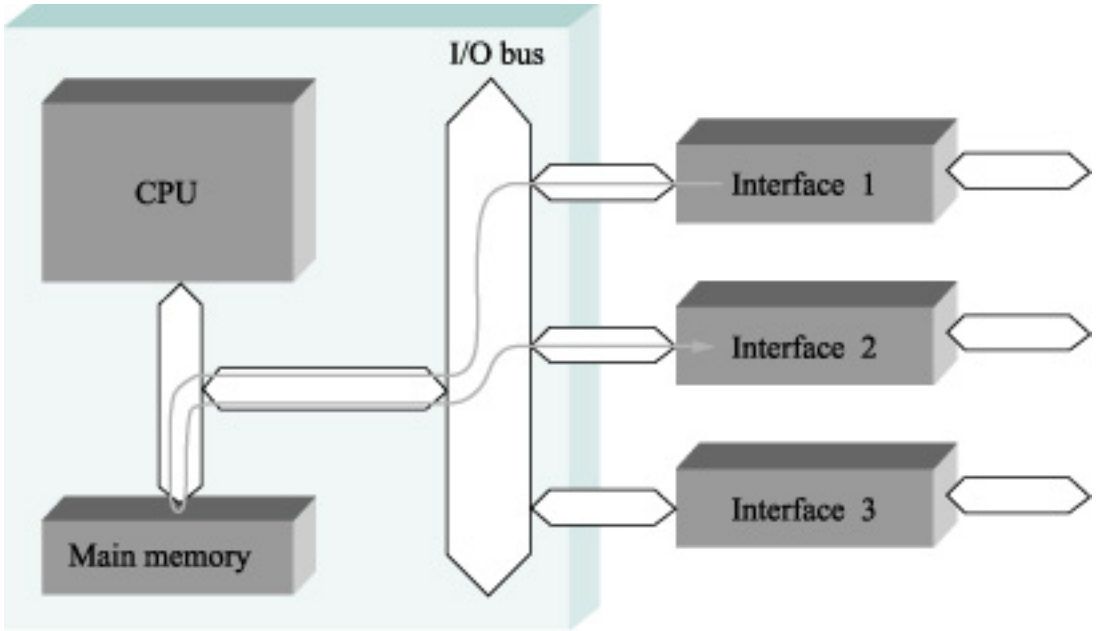
xwy@cs.duke.edu

Overview

- Switching hardware
- Software defined networking
- Programmable routers

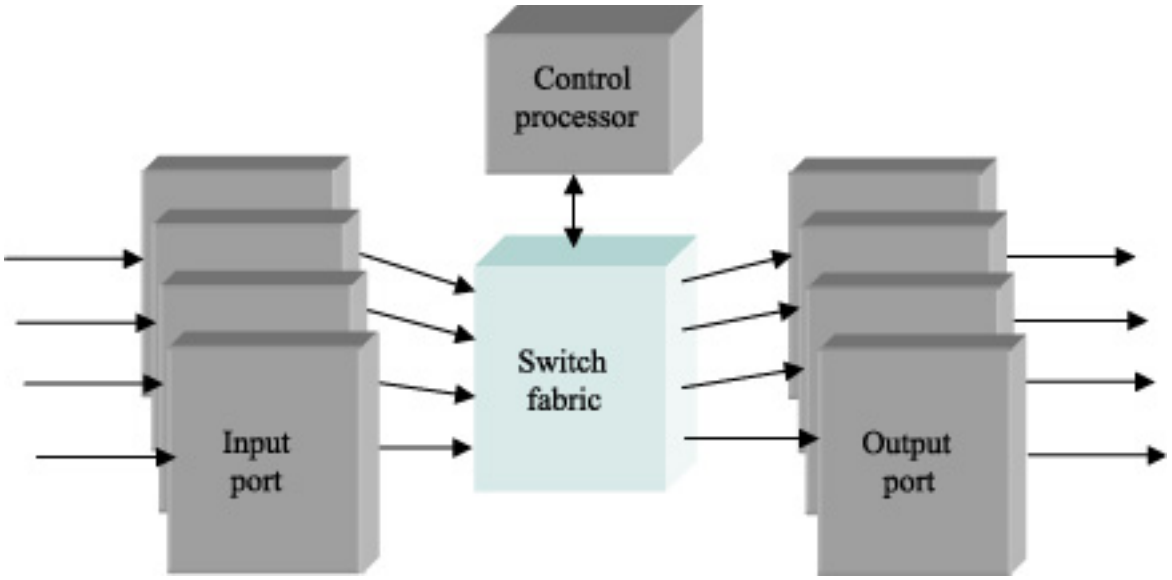
Switching hardware

Software switch



- Packets cross the bus twice
  - Half of the memory bus speed
    - 133Mhz, 64-bit wide I/O bus → 4Gpbs
- Short packets reduce throughput
  - 1Mpps, 64 bytes packet
  - Throughput = 512 Mbps
  - Shared by 10 ports: 51.2Mbps

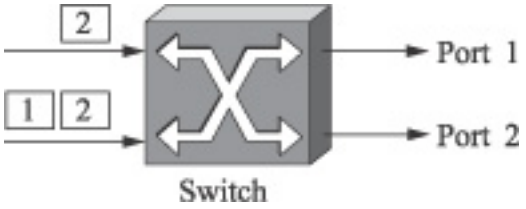
Hardware switches



- Ports communicate with the outside world
  - Eg, maintains VCI tables
- Switching fabric is simple and fast

Performance bottlenecks

- Input port
  - Line speed: 2.48 Gbps
    - $2.48 \times 10^9/(64 \times 8) = 4.83$ Mpps

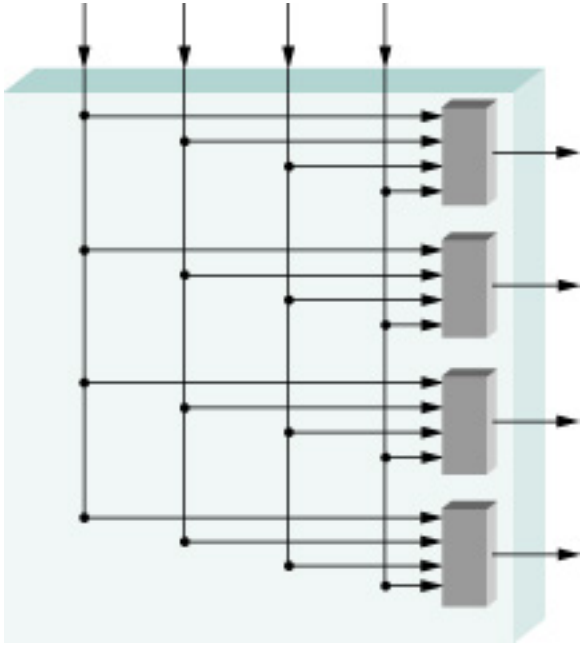


- Buffering
  - Head of line blocking
  - May limit throughput to only 59%
  - Use output buffers or sophisticated buffer management algorithms to improve performance

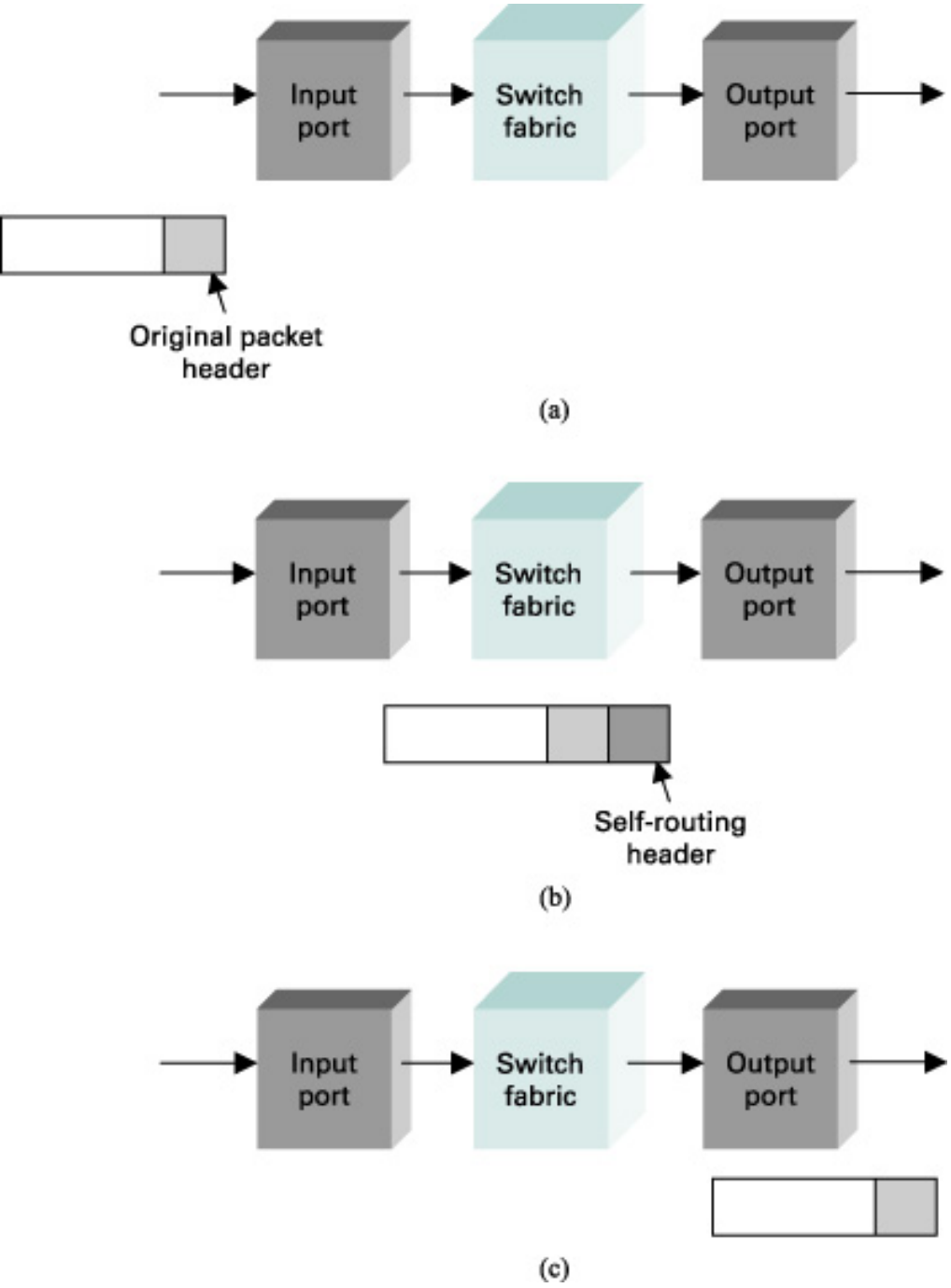
Fabrics

- Shared bus
  - The workstation switch
- Shared memory
  - Input ports read packets to shared memory
  - Output ports read them out to links

Fabrics



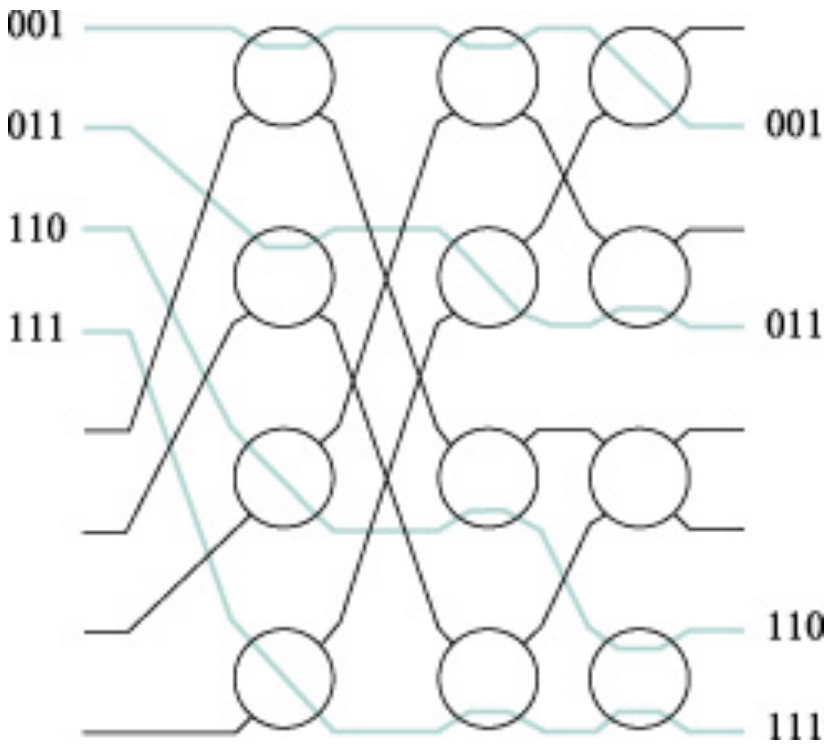
- Cross bar
  - A matrix of pathways that can be configured to accept packets from all inputs at once



Fabrics

- Self routing
  - a self-routing header added by the input port
  - Most scalable
  - Often built from 2x2 switching units

An example of self-routing



- 3-bit numbers are self-routing headers
- Multiple 2x2 switching elements
  - 0: upper output; 1: lower output

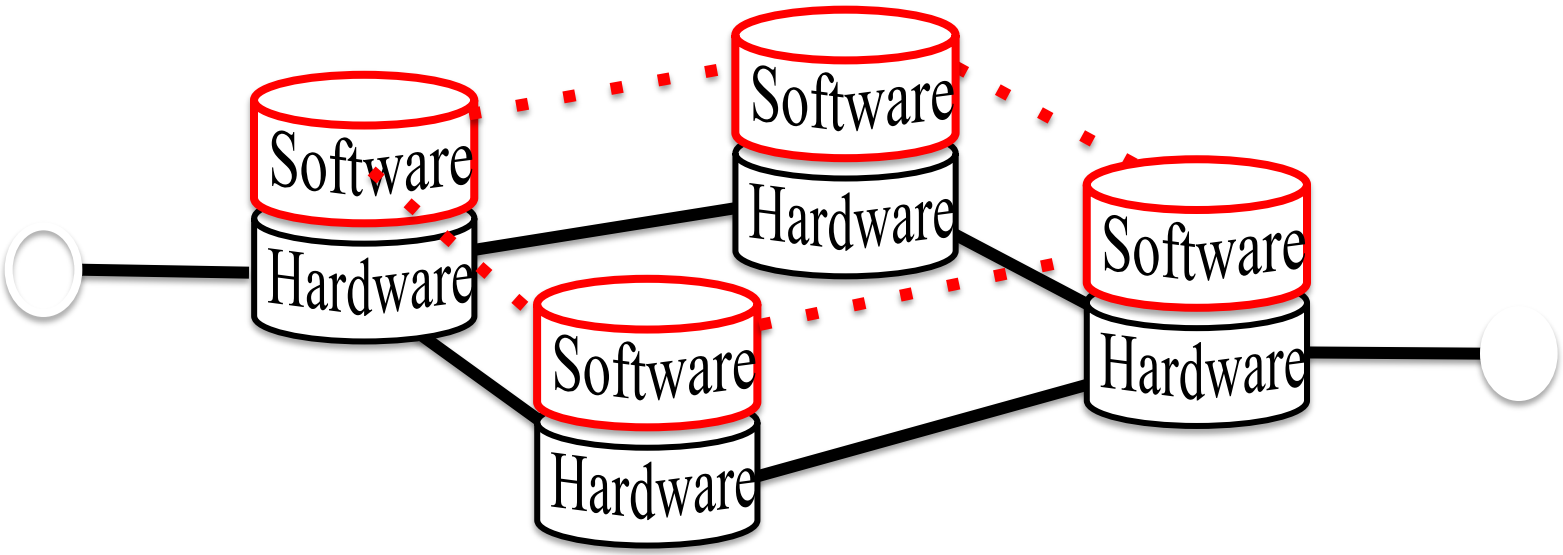
Software Defined Networking

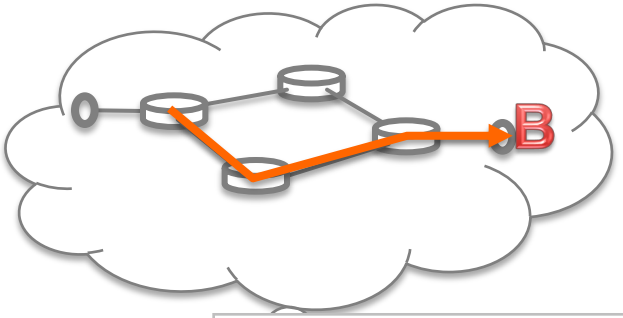
Slides adapted from Mohammad Alizadeh (MIT)'s SDN lecture

Outline

- Networking before SDN
- What is SDN?
- OpenFlow basics
- Why is SDN happening now? (a brief history)

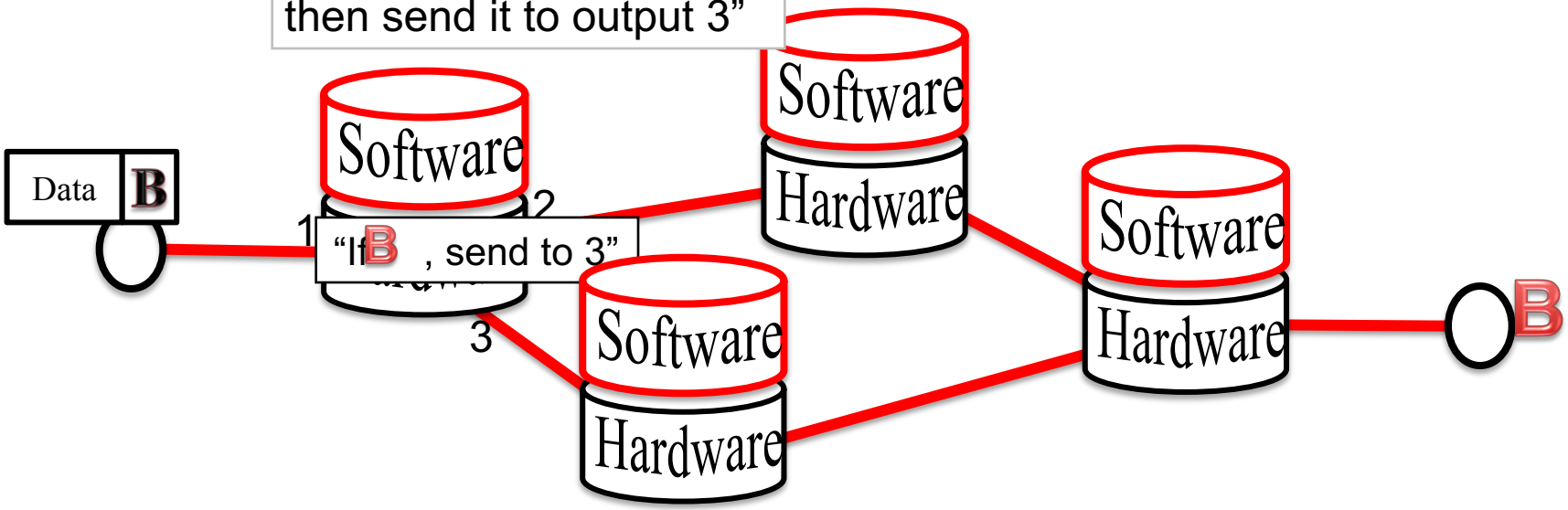
Networking before SDN





1. Figure out which routers and links are present.
2. Run Dijkstra's algorithm to find shortest paths.

"If a packet is going to B, then send it to output 3"



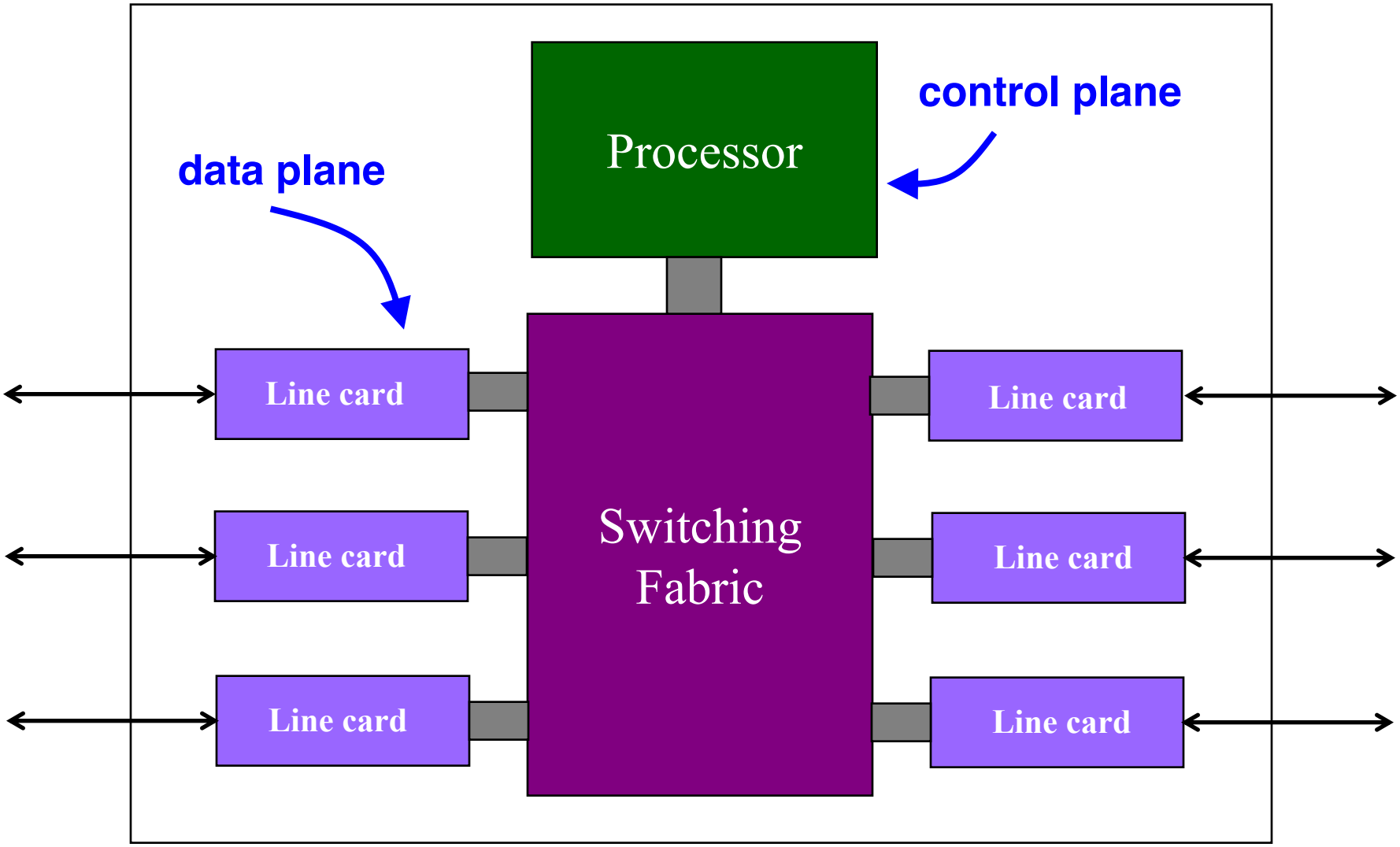
The Networking "Planes"

- **Data plane**: processing and delivery of packets with local forwarding state
  - Forwarding state + packet header → forwarding decision
  - Filtering, buffering, scheduling
- **Control plane**: computing the forwarding state in routers
  - Determines how and where packets are forwarded
  - Routing, traffic engineering, failure detection/recovery, …
- **Management plane**: configuring and tuning the network
  - Traffic engineering, ACL config, device provisioning, …

Timescales

	Data	Control	Management
Time-scale	Packet (nsec)	Event (10 msec to sec)	Human (min to hours)
Location	Linecard hardware	Router software	Humans or scripts

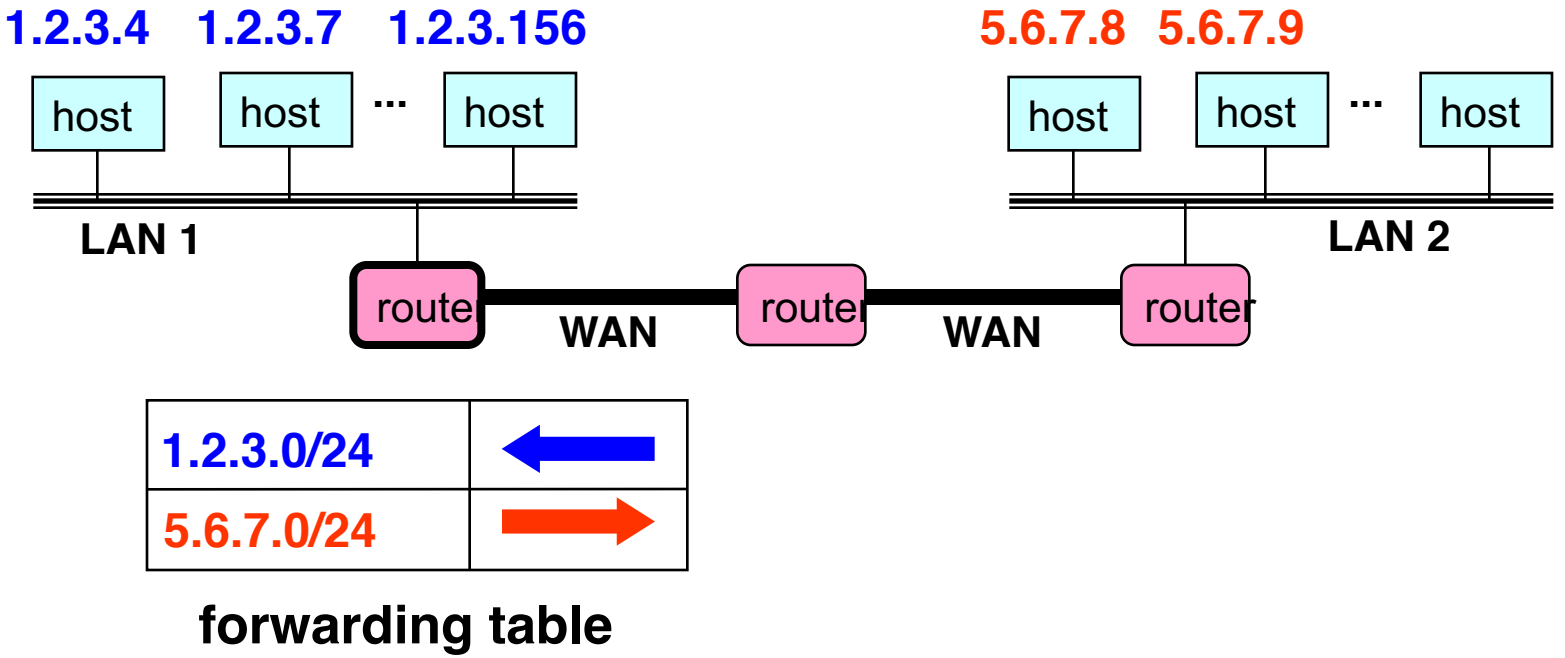
Data and Control Planes



Data Plane

- Streaming algorithms on packets
  - Matching on some header bits
  - Perform some actions

- Example: IP Forwarding



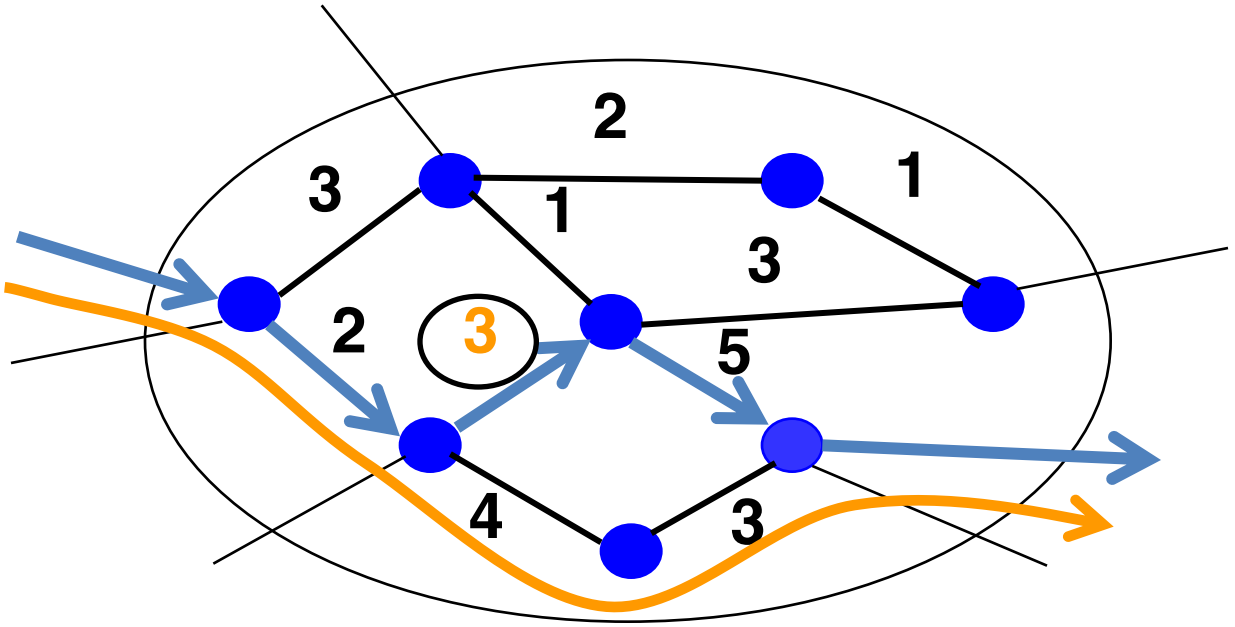
Control Plane

- Compute paths the packets will follow
  - Populate forwarding tables
  - Traditionally, a distributed protocol

- Example: Link-state routing (OSPF, IS-IS)
  - Flood the entire topology to all nodes
  - Each node computes shortest paths
  - Dijkstra's algorithm

Management Plane

- Traffic Engineering: setting the weights
  - Inversely proportional to link capacity?
  - Proportional to propagation delay?
  - Network-wide optimization based on traffic?



Challenges

(Too) many task-specific control mechanisms
- No modularity, limited functionality

Indirect
- Mu ou want
- Ex.

Uncoord
- Ca

Interacting protocols and mechanisms
- Routing, addressing, access control, QoS

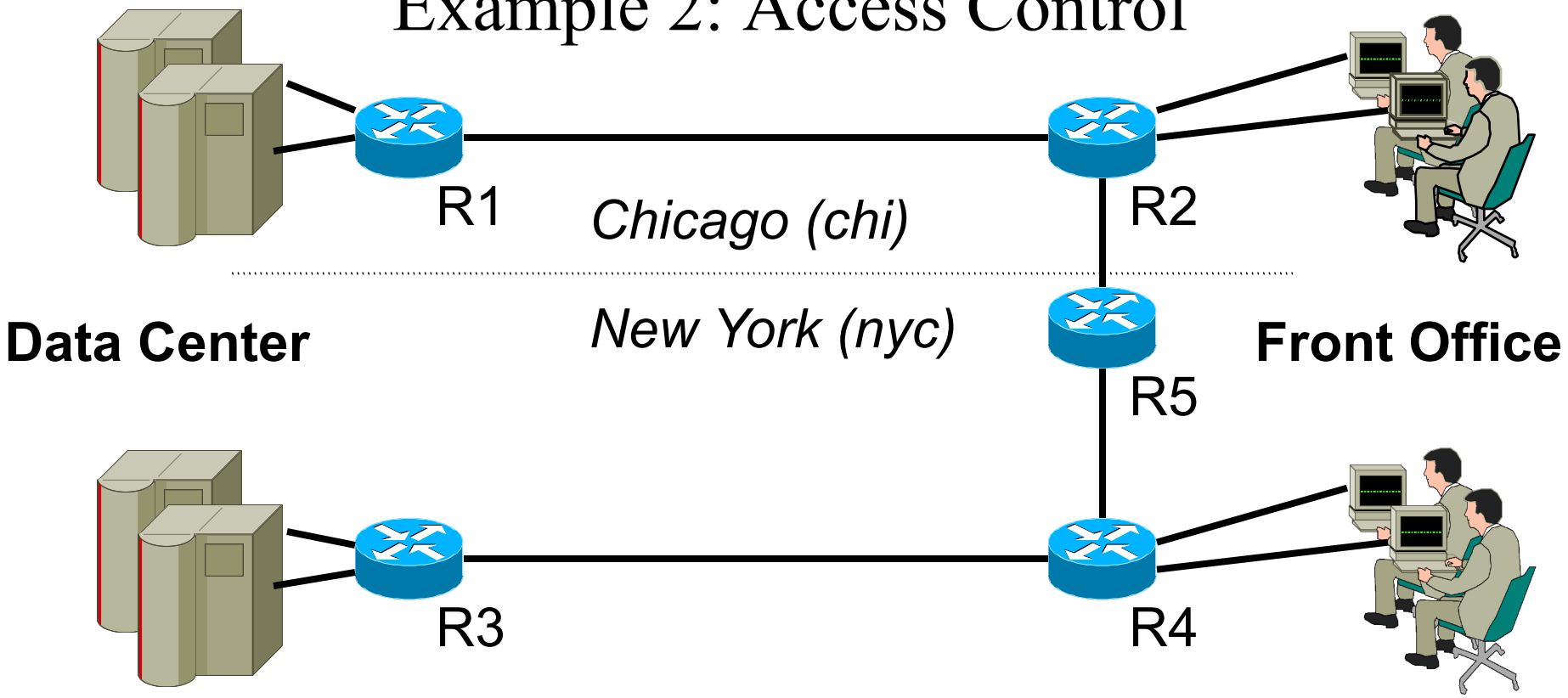
The network is
- Hard to reason about
- Hard to evolve
- Expensive

Example 1: Inter-domain Routing

- Today's inter-domain routing protocol, BGP, artificially constrains routes
  - Routing only on **destination IP address blocks**
  - Can only influence **immediate neighbors**
  - Very difficult to incorporate other information

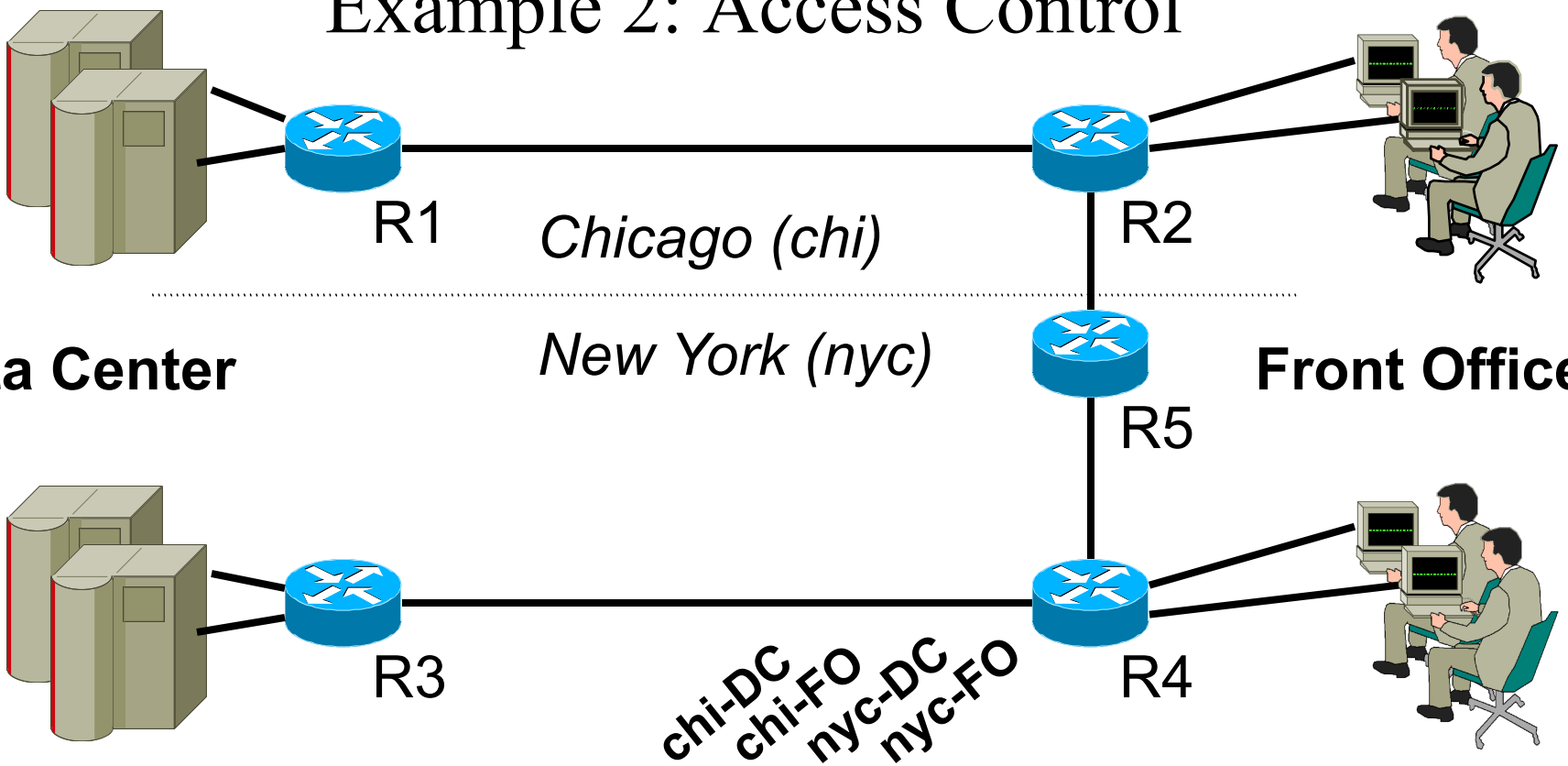
- Application-specific peering
  - Route video traffic one way, and non-video another
- Blocking denial-of-service traffic
  - Dropping unwanted traffic further upstream
- Inbound traffic engineering
  - Splitting incoming traffic over multiple peering links

Example 2: Access Control



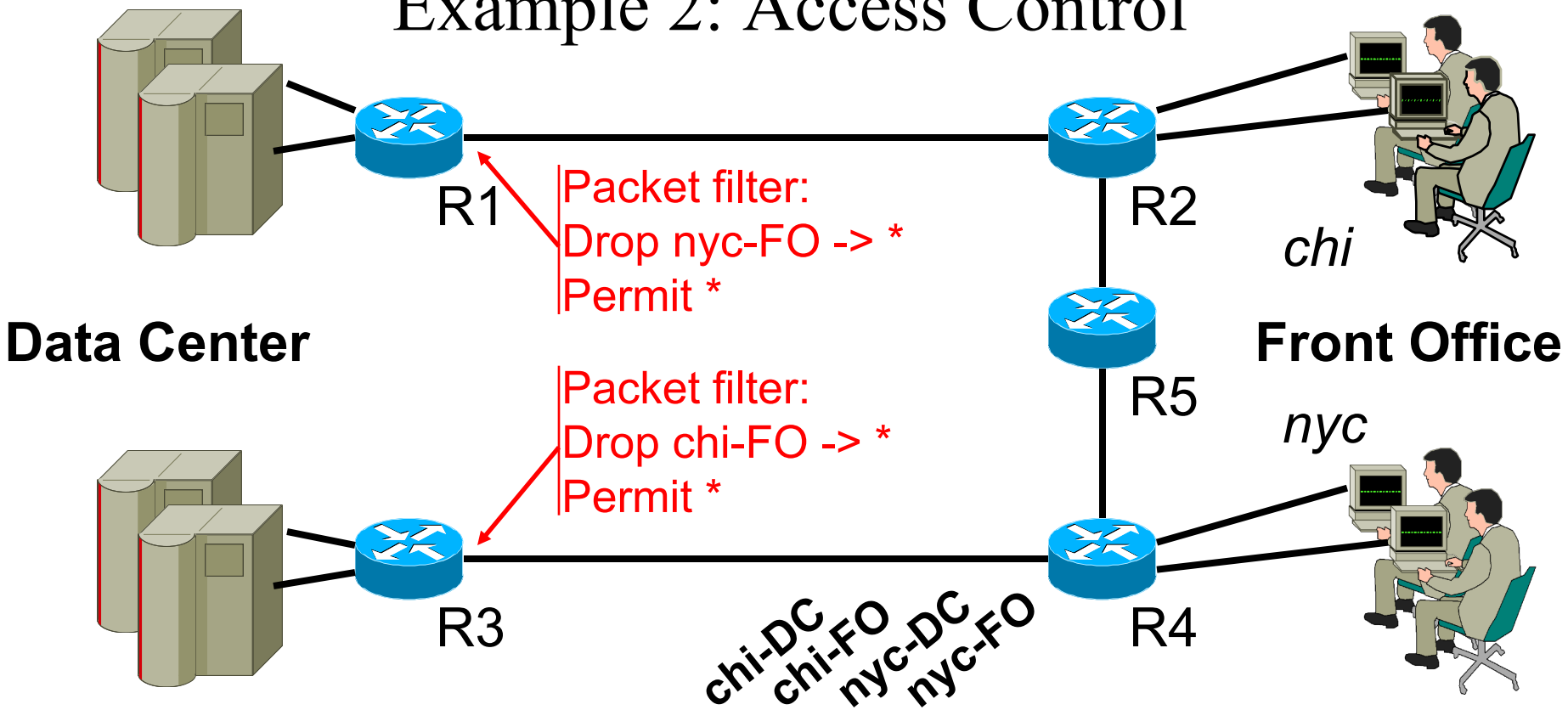
- Two locations, each with data center & front office
- All routers exchange routes over all links

Example 2: Access Control



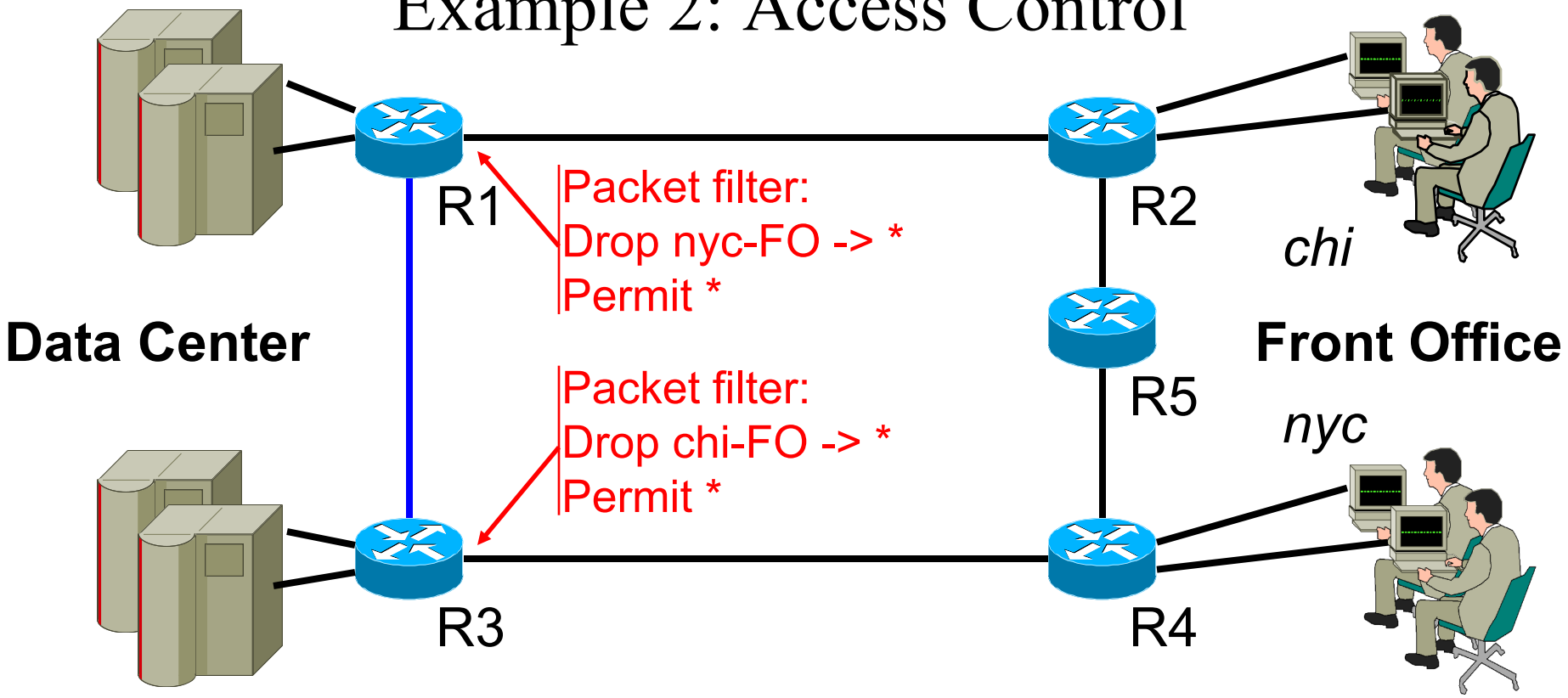
	chi-DC	chi-FO	nyc-DC	nyc-FO
chi-DC		●	●	⊘
chi-FO	●		⊘	●
nyc-DC	●	⊘		●
nyc-FO	⊘	●	●	

Example 2: Access Control



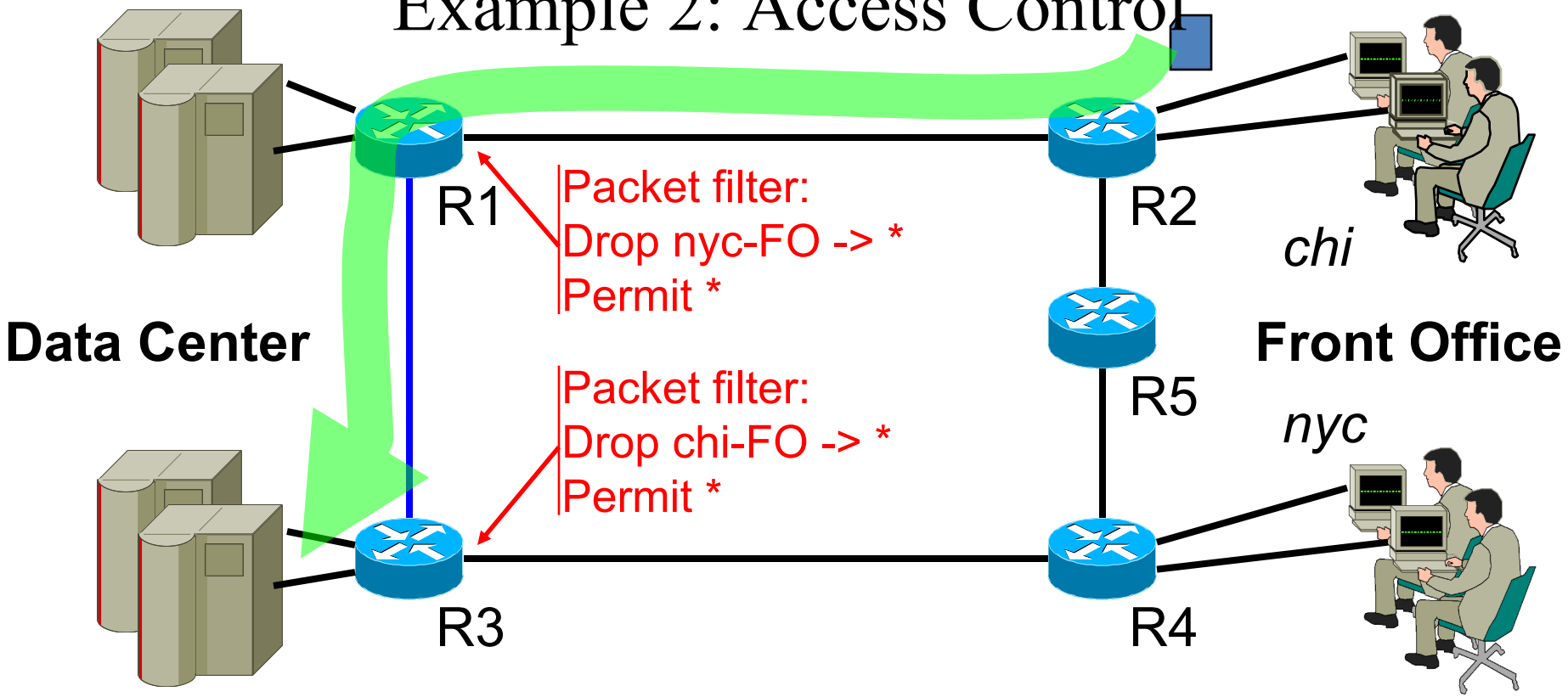
	chi-DC	chi-FO	nyc-DC	nyc-FO
chi-DC		●	●	⊘
chi-FO	●		⊘	●
nyc-DC	●	⊘		●
nyc-FO	⊘	●	●	

Example 2: Access Control



- A new short-cut link added between data centers
- Intended for backup traffic between centers

Example 2: Access Control



- Oops – new link lets packets violate ***access control policy***!
- Routing changed, but
- Packet filters don't update automatically

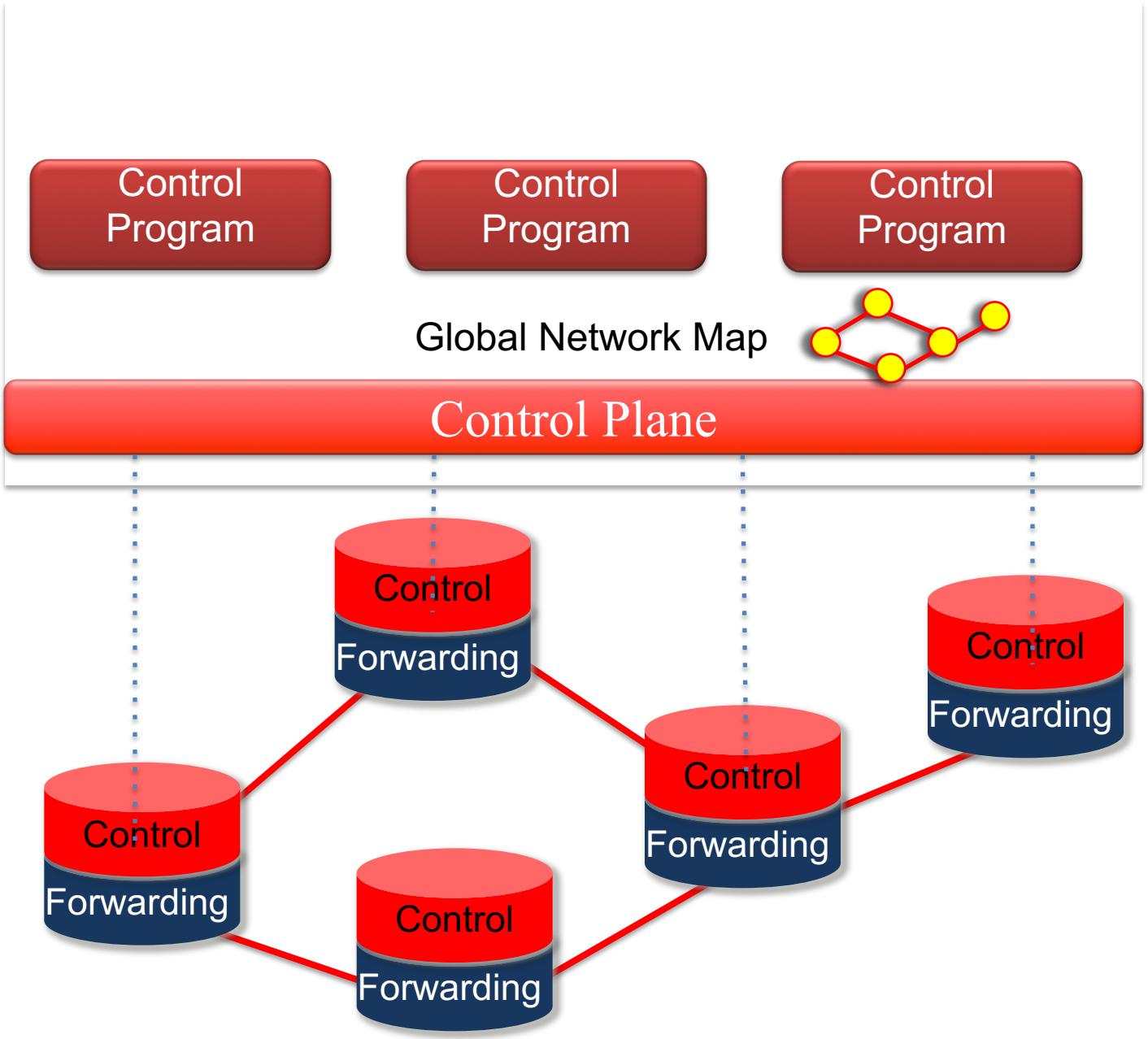
Software Defined Network

A network in which the control plane is physically separate from the data plane.

and

A single (logically centralized) control plane controls several forwarding devices.

Software Defined Network (SDN)



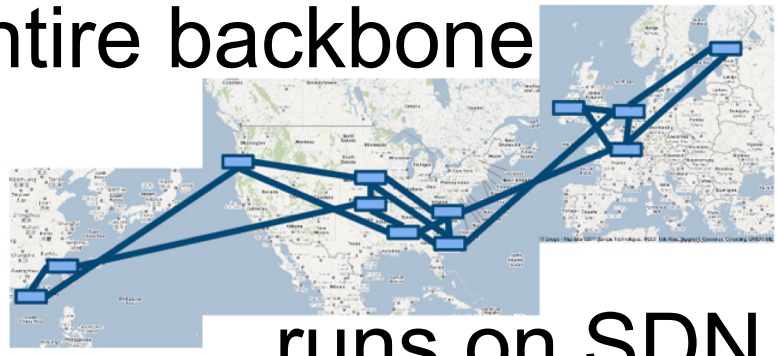
A Major Trend in Networking

OPEN NETWORKING FOUNDATION

Deutsche Telekom · facebook · Goldman Sachs · Google · Microsoft · NTT Communications · verizon · YAHOO!

Google

Entire backbone

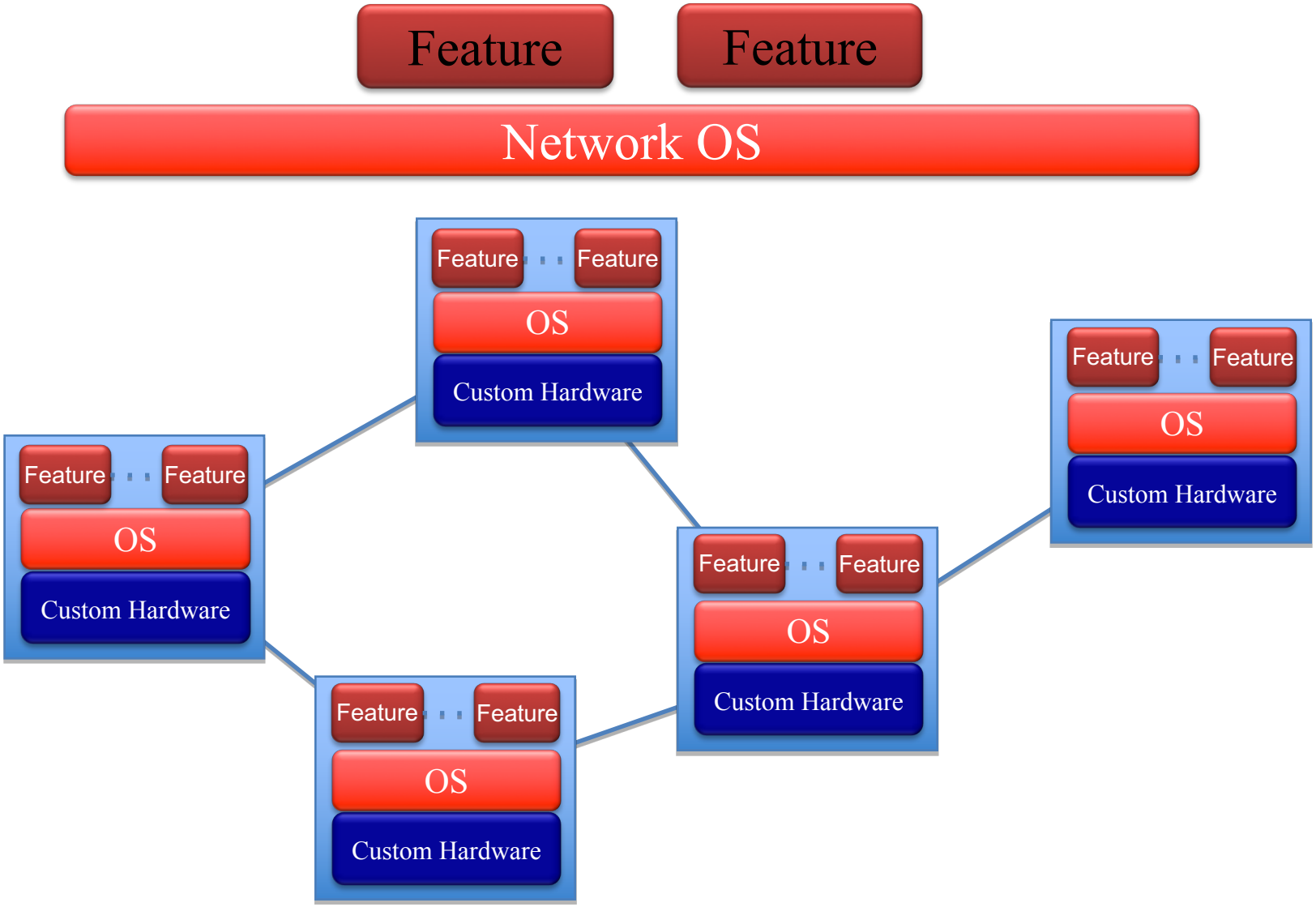


runs on SDN

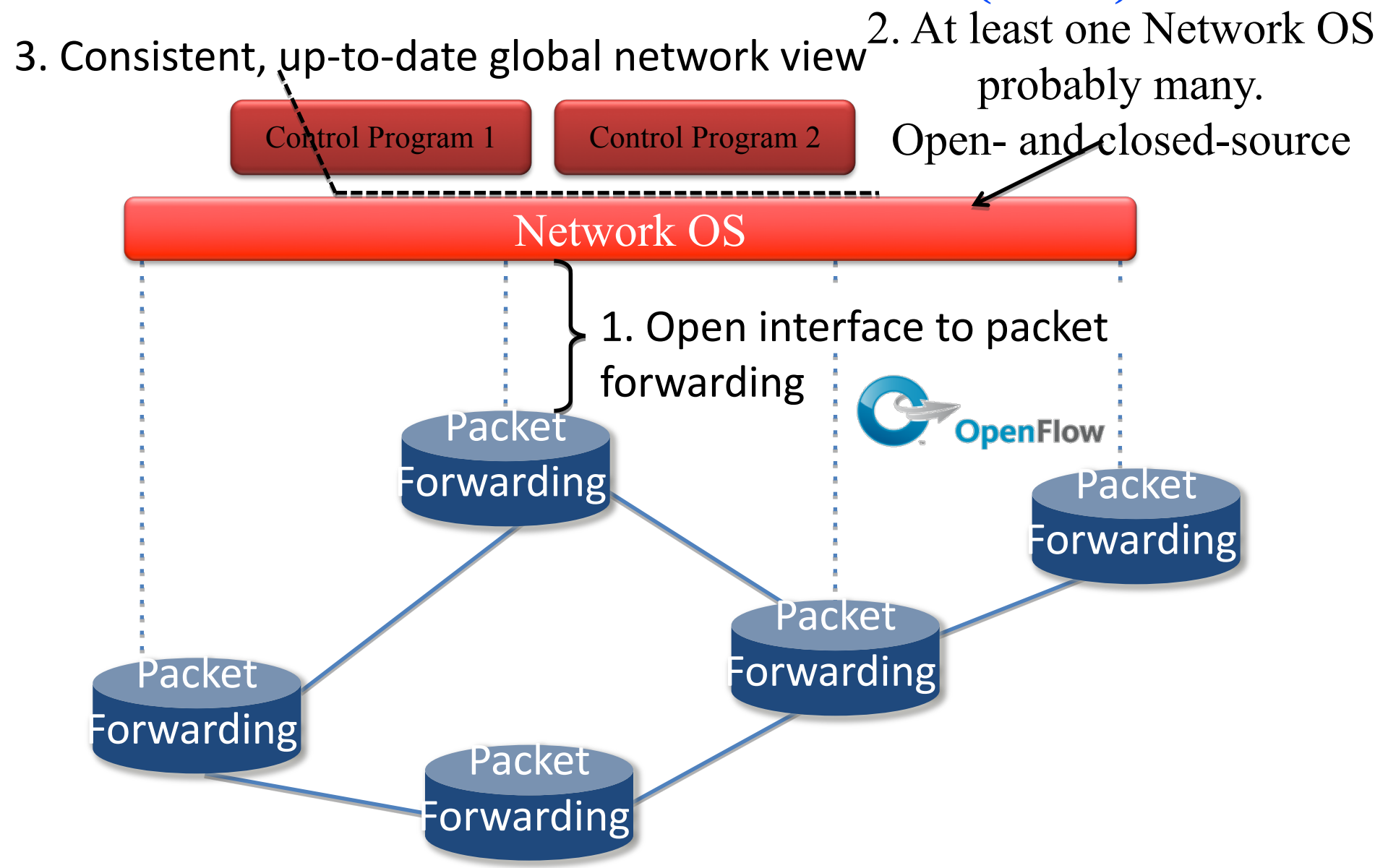
nicira

Bought for **$1.2 billion**
(mostly cash)

How SDN Changes the Network



Software Defined Network (SDN)



Network OS

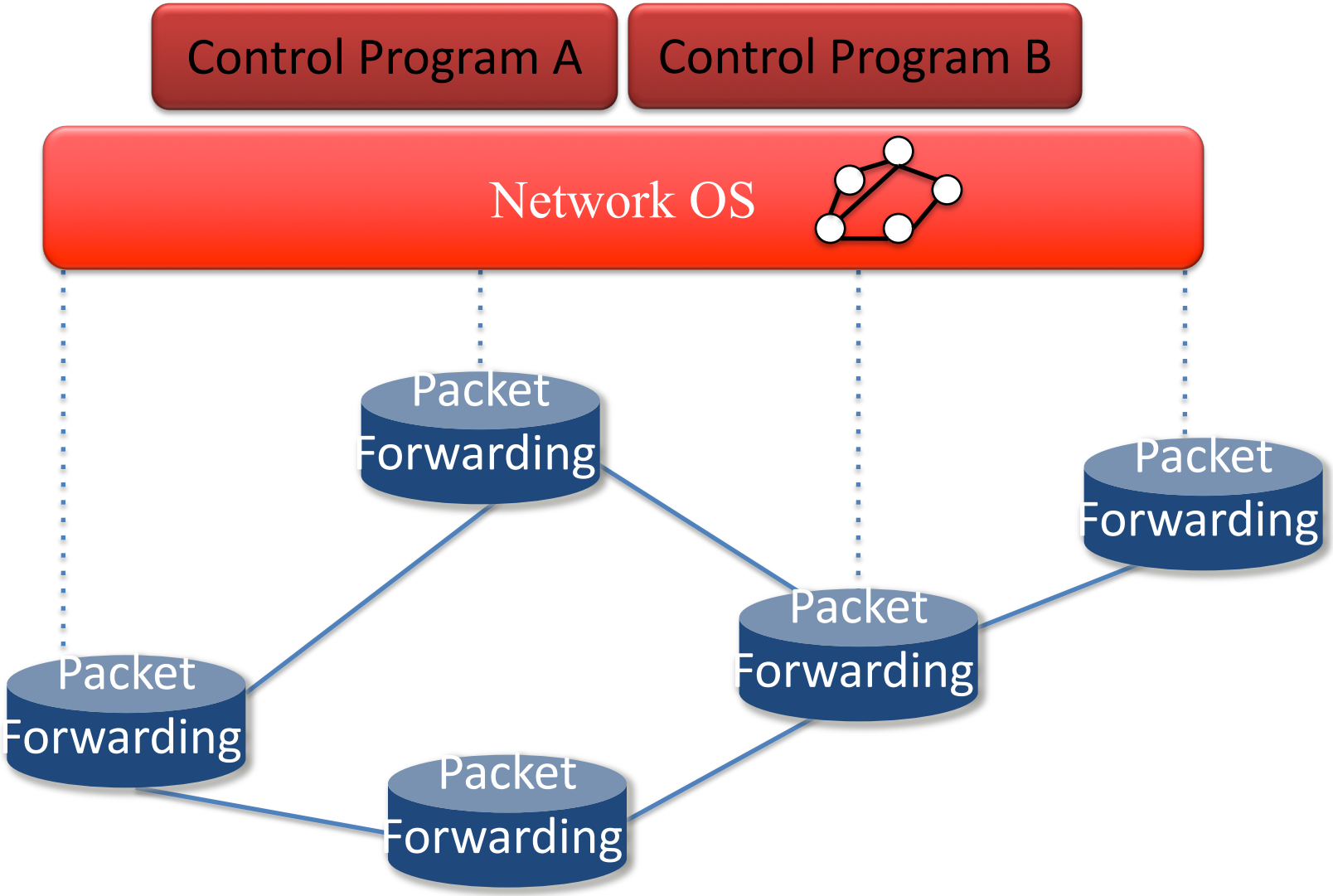
Network OS: distributed system that creates a consistent, up-to-date network view

- Runs on servers (controllers) in the network
- NOX, ONIX, Floodlight, Trema, OpenDaylight, HyperFlow, Kandoo, Beehive, Beacon, Maestro, … + more

Uses **forwarding abstraction** to:

- Get state information **from** forwarding elements
- Give control directives **to** forwarding elements

Software Defined Network (SDN)



Control Program

Control program operates on view of network

- **Input**: global network view (graph/database)
- **Output**: configuration of each network device

Control program is not a distributed system

- Abstraction hides details of distributed state

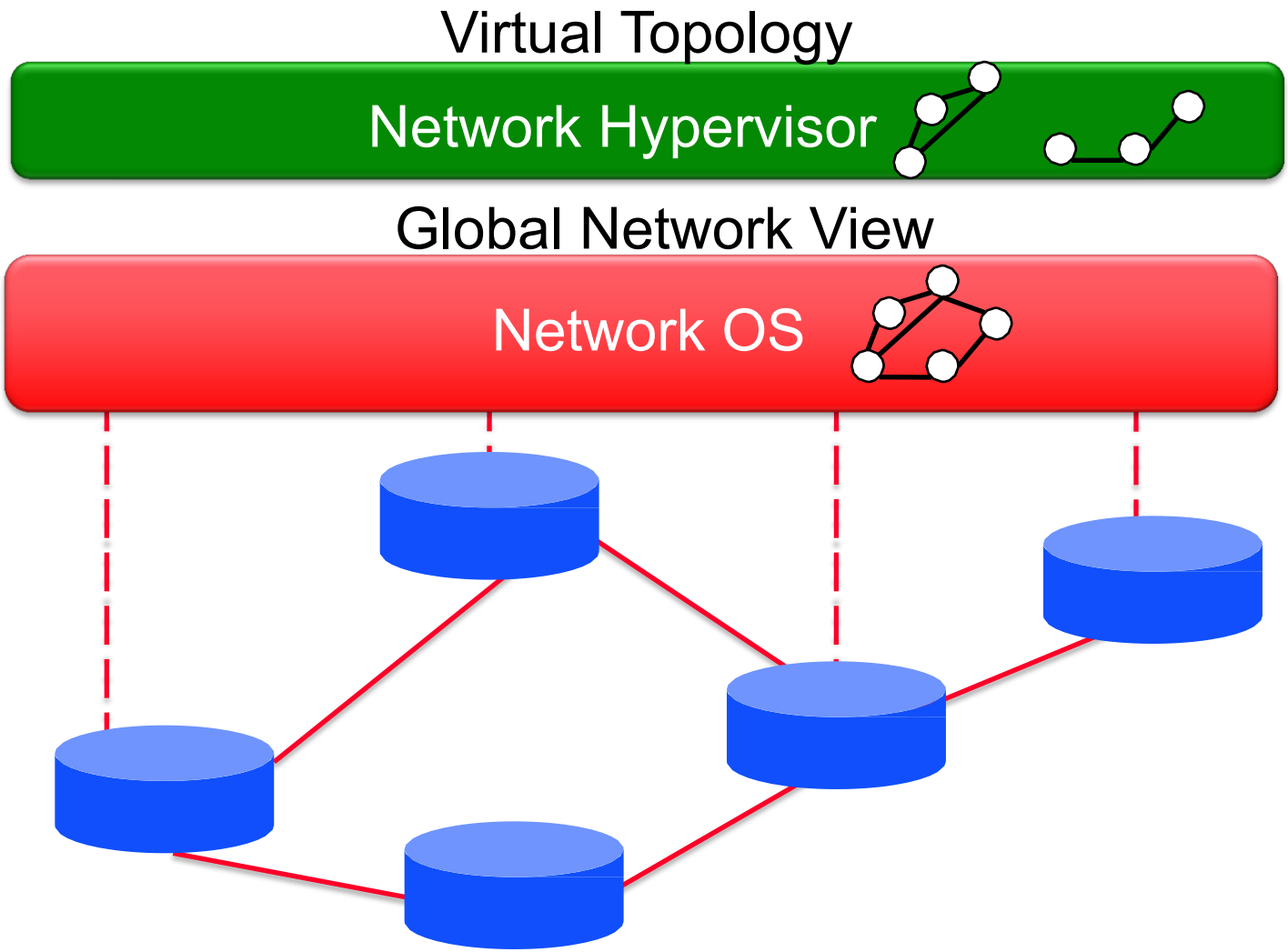
Forwarding Abstraction

Purpose: Standard way of defining forwarding state

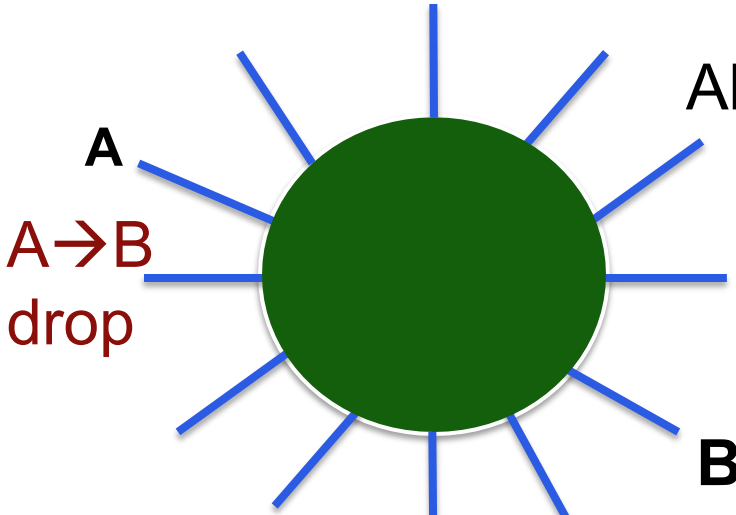
- Flexible
  - Behavior specified by control plane
  - Built from basic set of forwarding primitives
- Minimal
  - Streamlined for speed and low-power
  - Control program not vendor-specific

- OpenFlow is an example of such an abstraction

Software Defined Network

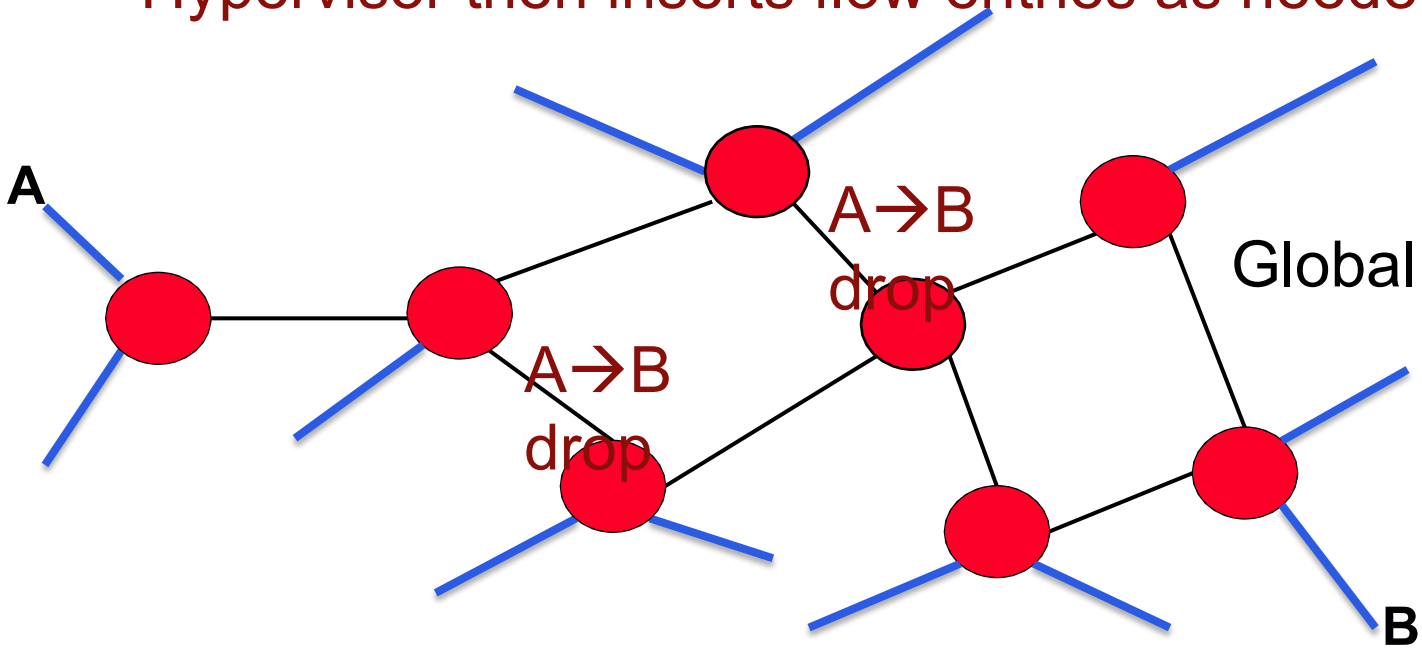


Virtualization Simplifies Control Program



Abstract Network View

Hypervisor then inserts flow entries as needed



Global Network View

Does SDN Simplify the Network?

Does SDN Simplify the Network?

Abstraction doesn't eliminate complexity

- NOS, Hypervisor are still complicated pieces of code

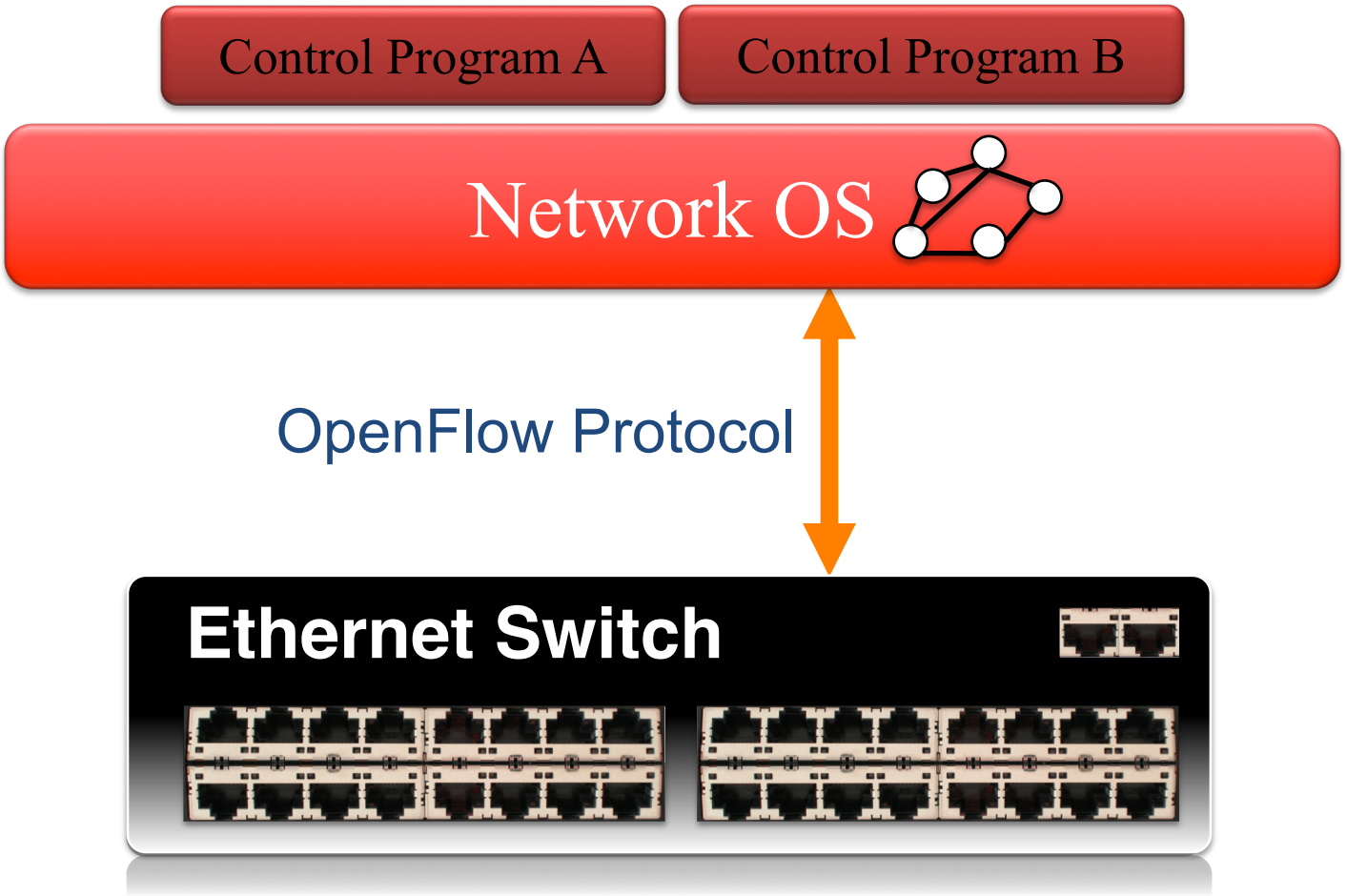
SDN main achievements

- Simplifies interface for control program (user-specific)
- Pushes complexity into reusable code (SDN platform)

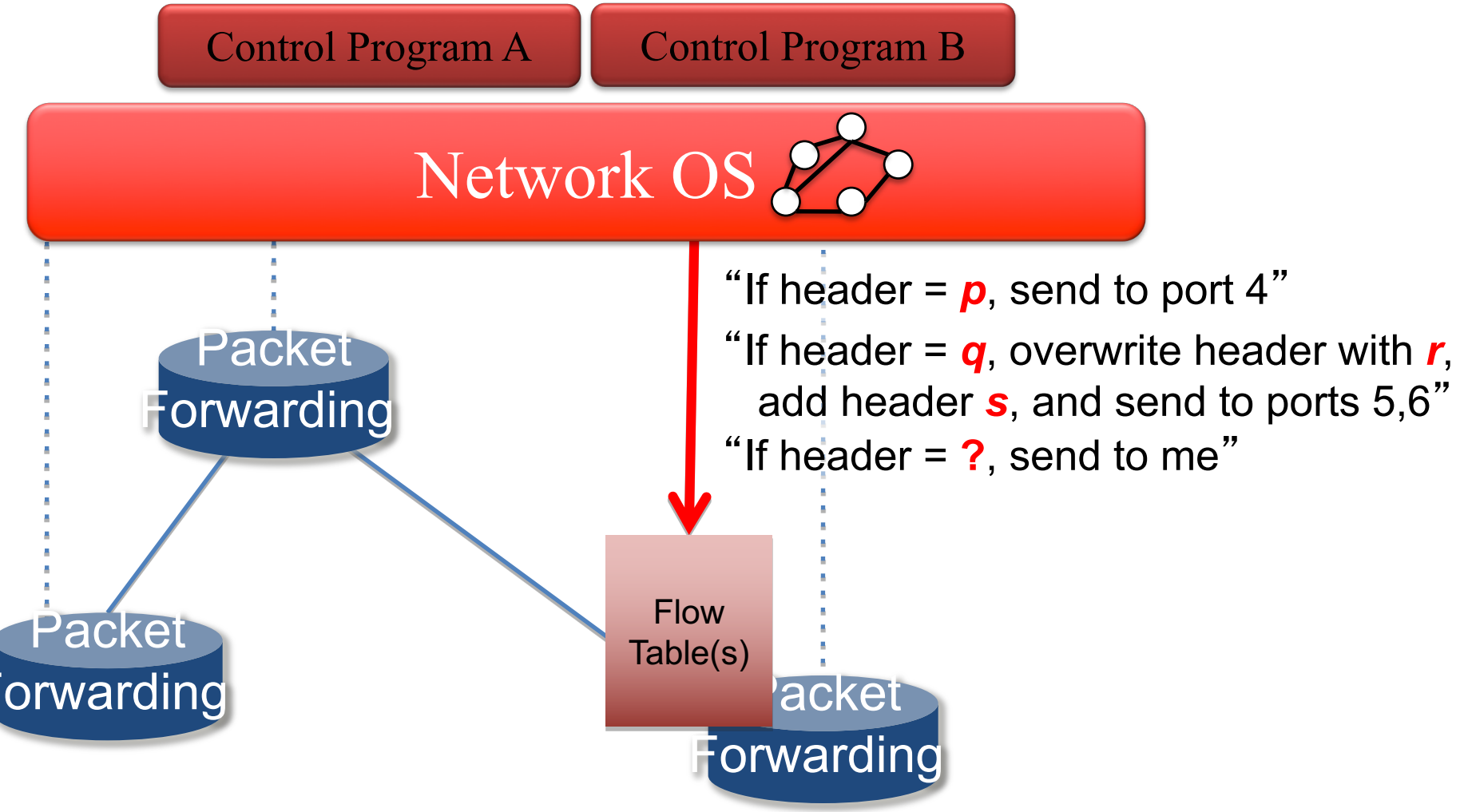
Just like compilers….

OpenFlow Basics

OpenFlow Basics



OpenFlow Basics



Primitives <Match, Action>

Match arbitrary bits in headers:

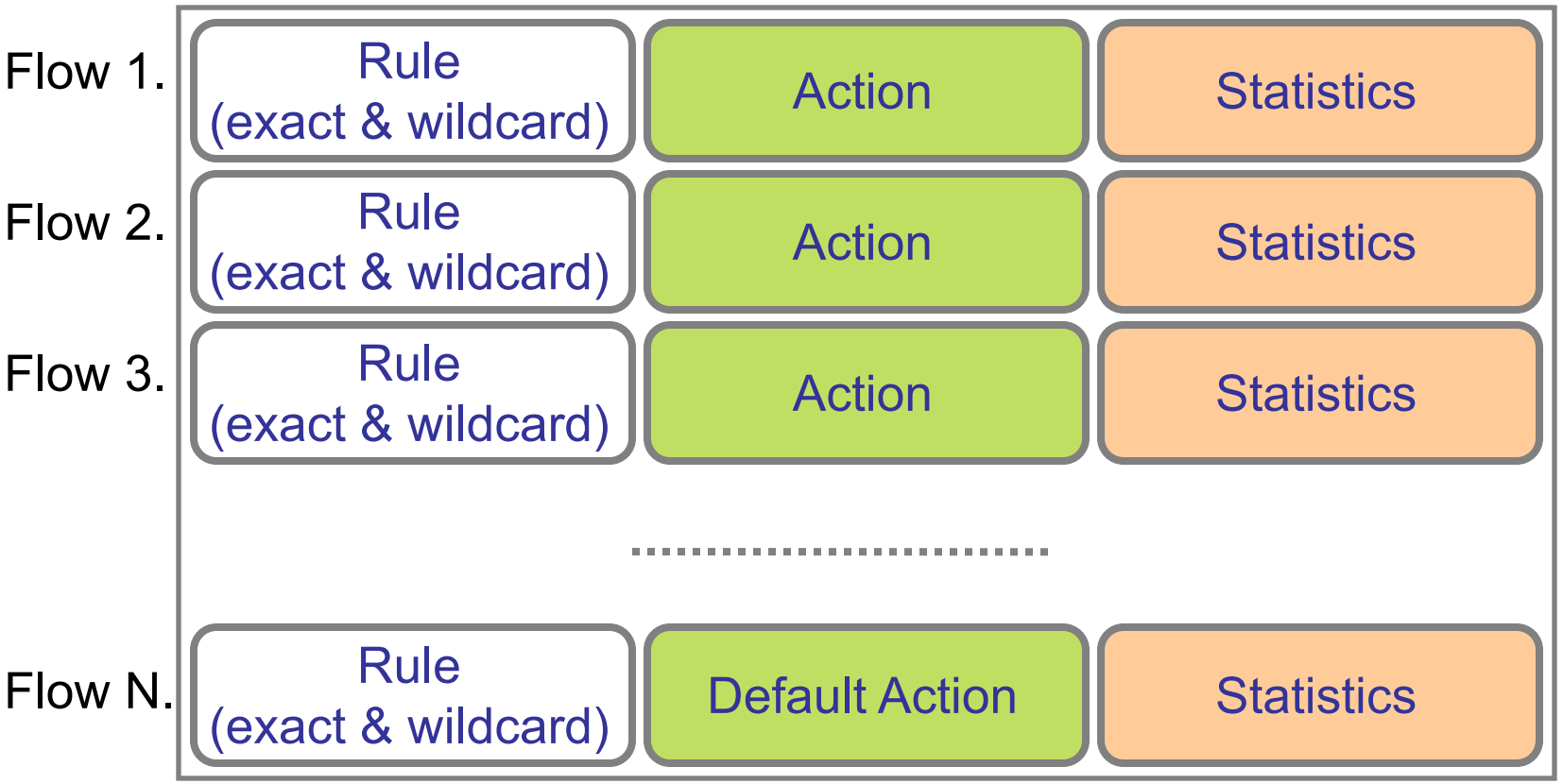
Match: 1000x01xx0101001x

- Match on any header, or new header
- Allows any flow granularity

Action

- Forward to port(s), drop, send to controller
- Overwrite header with mask, push or pop
- Forward at specific bit-rate

OpenFlow Rules



Exploit the flow table in switches, routers, and chipsets

Why is SDN happening now?

The Road to SDN

- Active Networking: 1990s
  - First attempt make networks programmable
  - Demultiplexing packets to software programs, network virtualization, …
- Control/Dataplane Separation: 2003-2007
  - ForCes [IETF], RCP, 4D [Princeton, CMU], SANE/Ethane [Stanford/Berkeley]
  - Open interfaces between data and control plane, logically centralized control
- OpenFlow API & Network Oses: 2008
  - OpenFlow switch interface [Stanford]
  - NOX Network OS [Nicira]

N. Feamster et al., “The Road to SDN: An Intellectual History of Programmable Networks”, ACM SIGCOMM CCR 2014.

SDN Drivers

- Rise of merchant switching silicon
  - Democratized switching
  - Vendors eager to unseat incumbents
- Cloud / Data centers
  - Operators face real network management problems
  - Extremely cost conscious; desire a lot of control
- The right balance between vision & pragmatism
  - OpenFlow compatible with existing hardware
- A “killer app”: Network virtualization

Virtualization is Killer App for SDN

Consider a multi-tenant datacenter

- Want to allow each tenant to specify virtual topology
- This defines their individual policies and requirements

Datacenter's network hypervisor compiles these virtual topologies into set of switch configurations

- Takes 1000s of individual tenant virtual topologies
- Computes configurations to implement all simultaneously

This is what people are paying money for….

- ***Enabled by SDN's ability to virtualize the network***

Overview

- The Trio of modern networking
  - SDN
  - NFV
  - Programmable switches

Network Functions Virtualisation

Classical Network Appliance Approach


Message Router


CDN


Session Border Controller


WAN Acceleration


DPI


Firewall


Carrier Grade NAT


Tester/QoE monitor

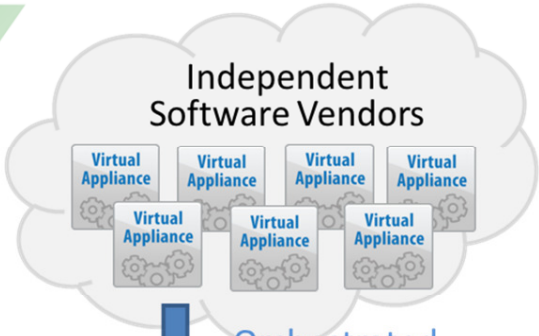

SGSN/GGSN


PE Router


BRAS


Radio Access Network Nodes

- Fragmented non-commodity hardware.
- Physical install per appliance per site.
- Hardware development large barrier to entry for new vendors, constraining innovation & competition.



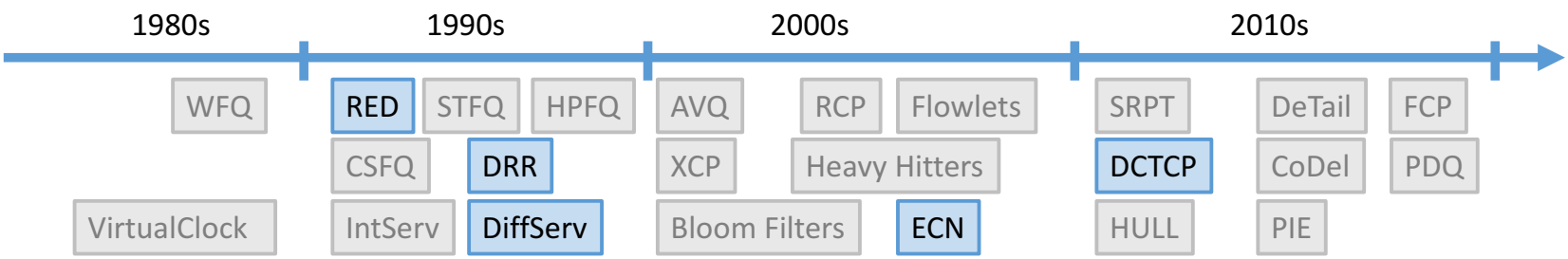

Standard High Volume Servers


Standard High Volume Storage


Standard High Volume Ethernet Switches

Network Virtualisation Approach

Motivation for programmable routers

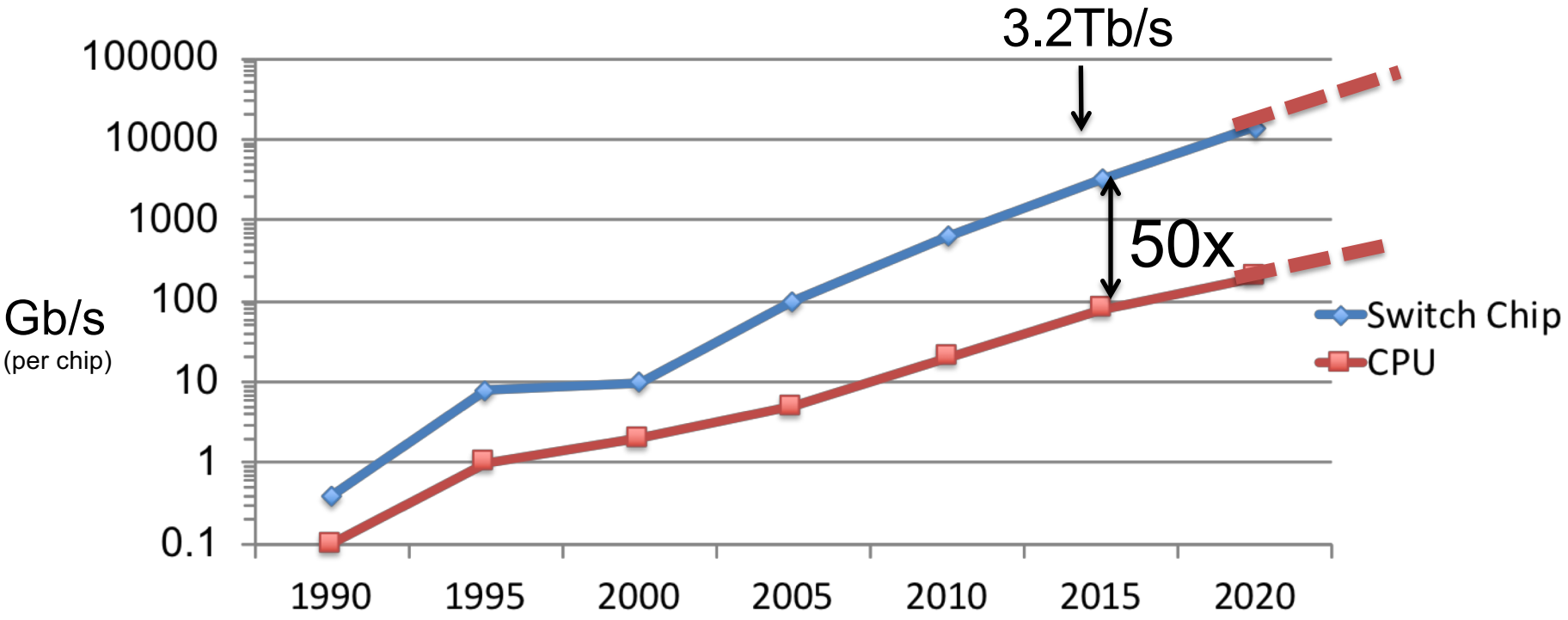


- Network changes fast
- Need to extend the forwarding plane

History of Programmable Routers

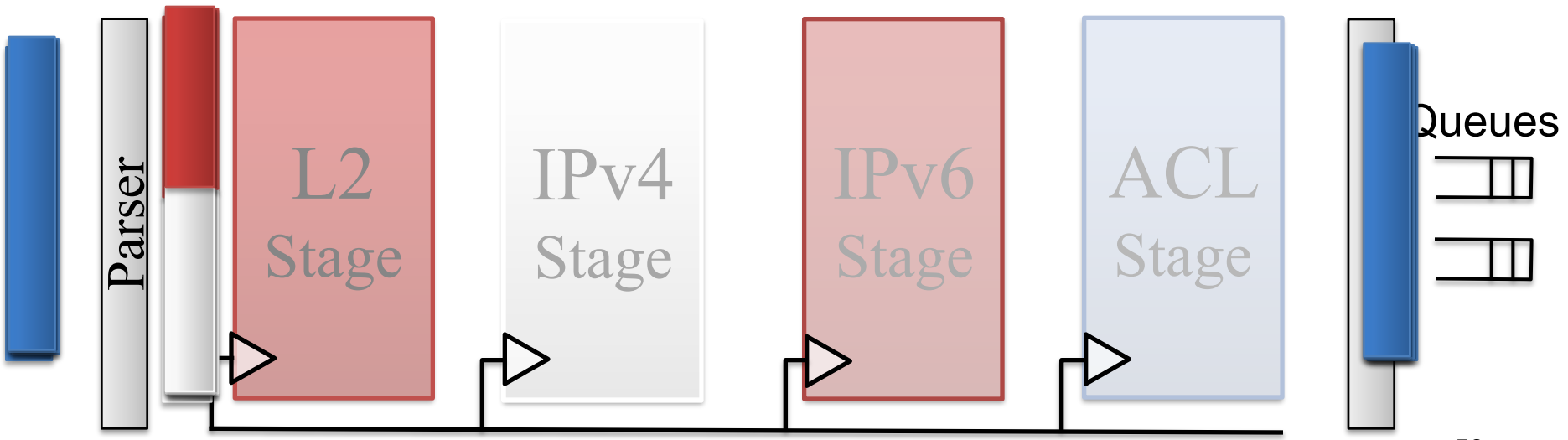
- Mini-computer based routers (1969-1990)
- Active networks (Mid 1990)
- Software routers (1999 – present)
  - Click, RouteBricks, PacketShader
- Software Defined Networking (2004– present)

Packet Forwarding Speeds



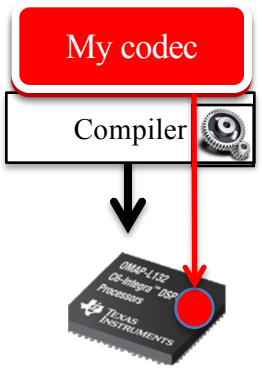
Conventional Wisdom:
"Programmable devices are 10-100x slower.
They consume much more power and area."

Fixed-Function Switch Chips



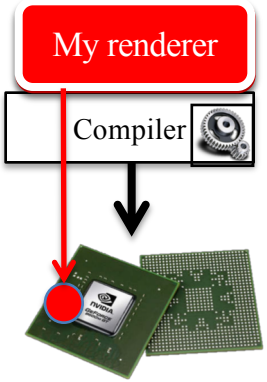
Domain Specific Processors

Signal Processing



DSP

Graphics



GPU

Conventional wisdom said: programmability too expensive

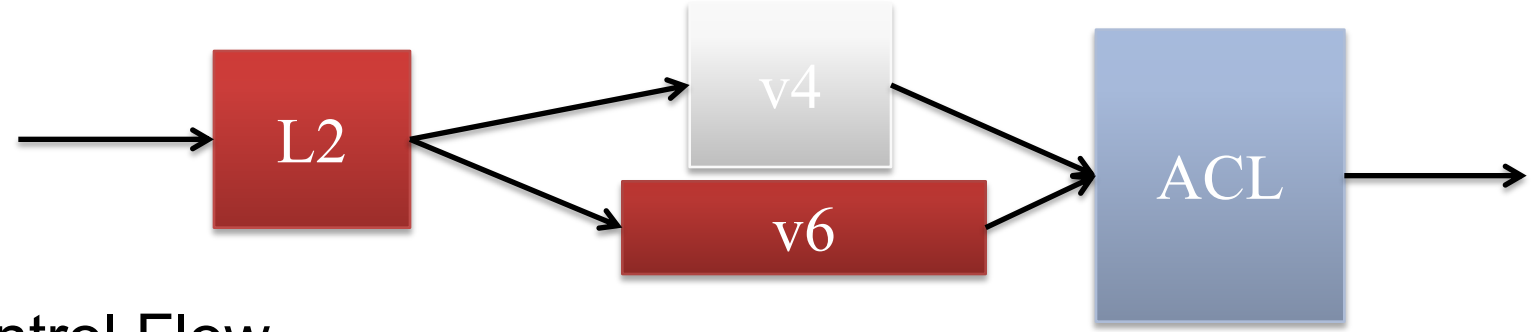
Then, someone identified:

1. The right model for data-parallelism
2. Basic underlying processing primitives

Domain-specific processors were built

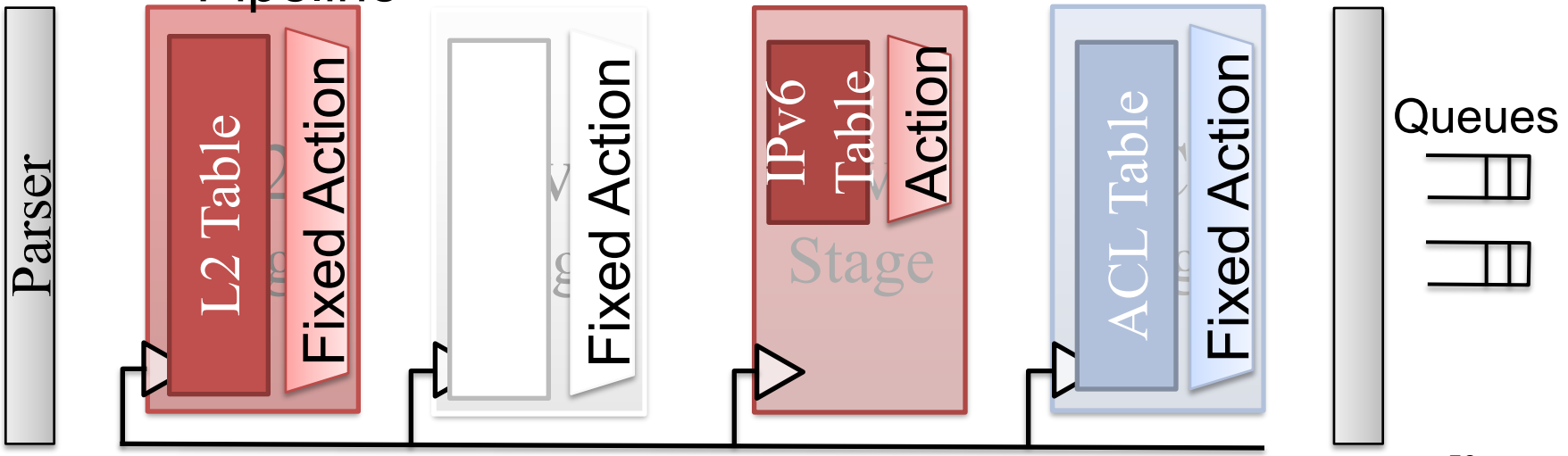
Domain-specific languages, compilers and tool-chains

Control Flow Graph



Control Flow Graph

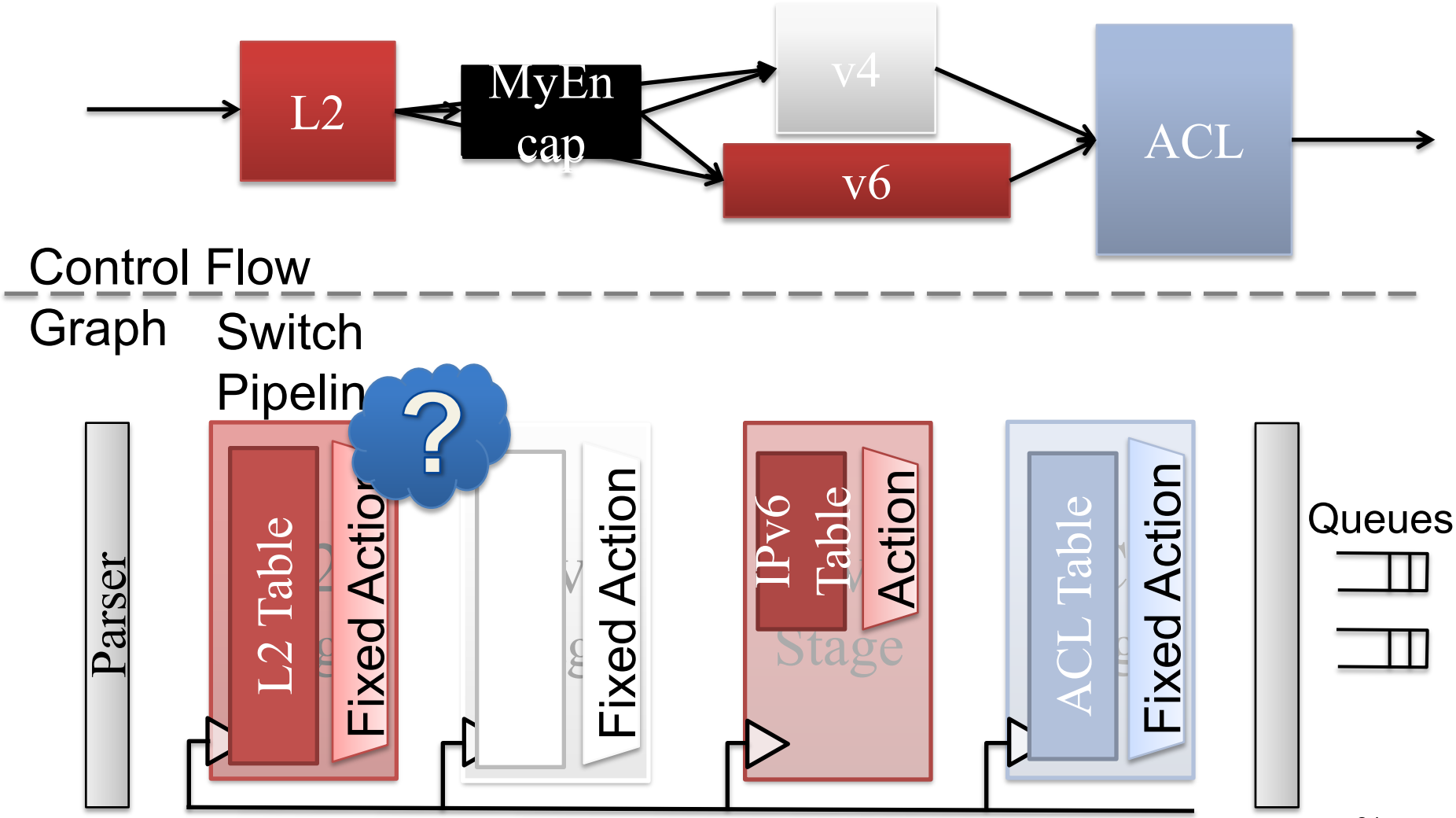
Switch Pipeline



Fixed-Function Switch Chips Are **Limited**

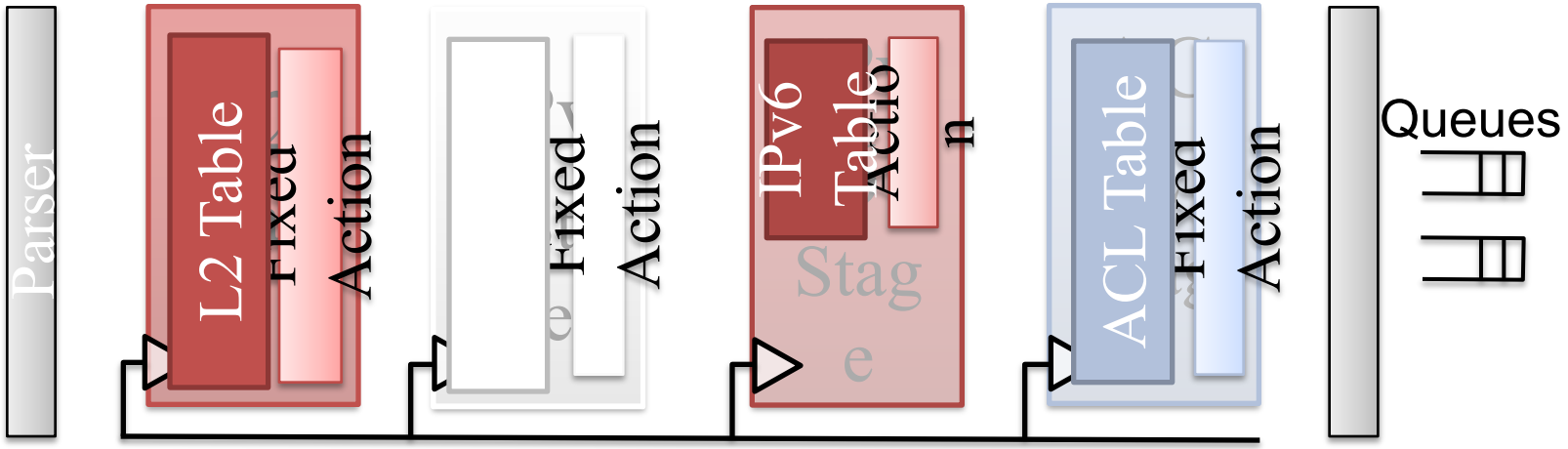
1. Can't add new forwarding functionality

Fixed-Function Switch Chips

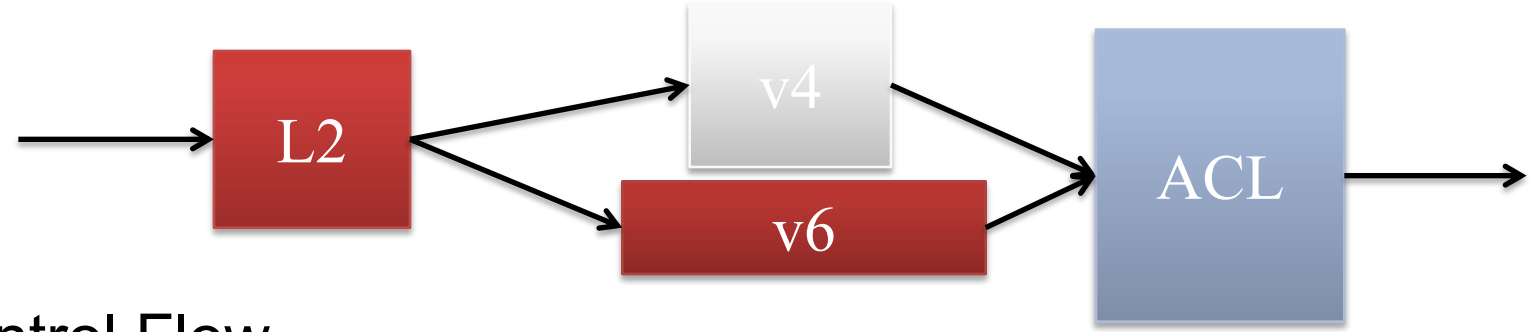


Fixed-Function Switch Chips Are **Limited**

1. Can't add new forwarding functionality
2. Can't move resources between functions

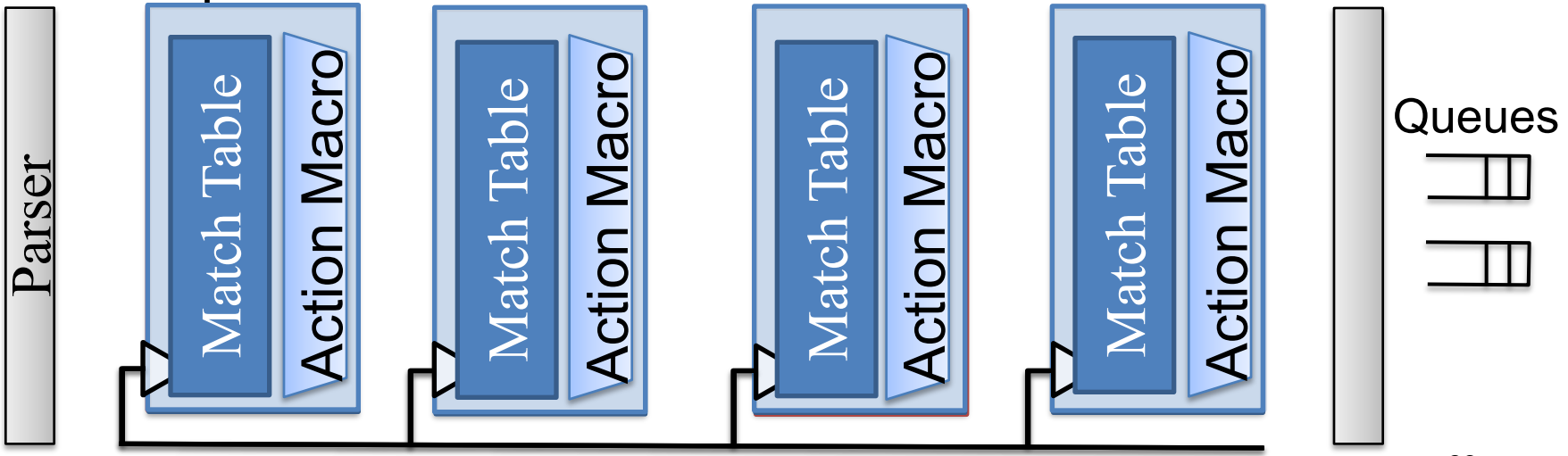


Programmable Switch Chips

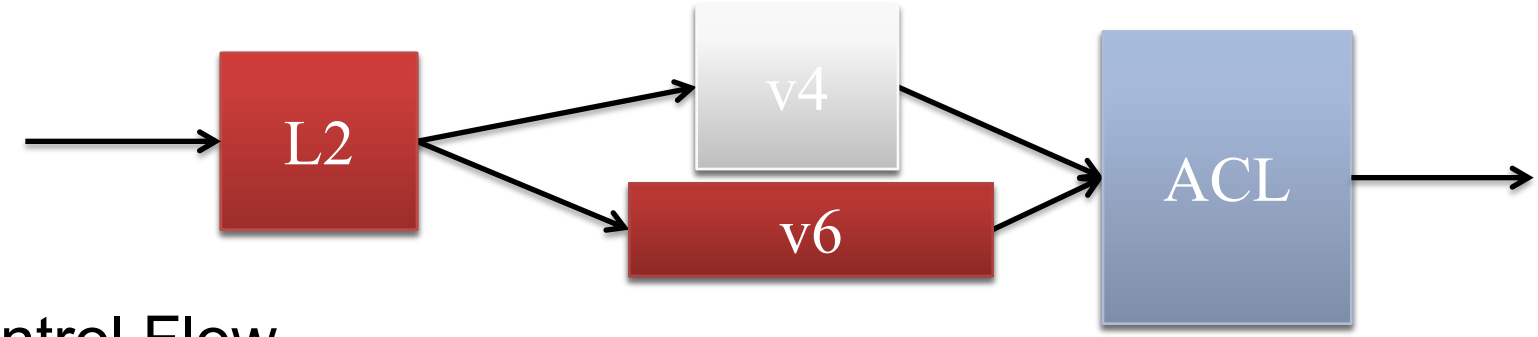


Control Flow Graph

Switch Pipeline

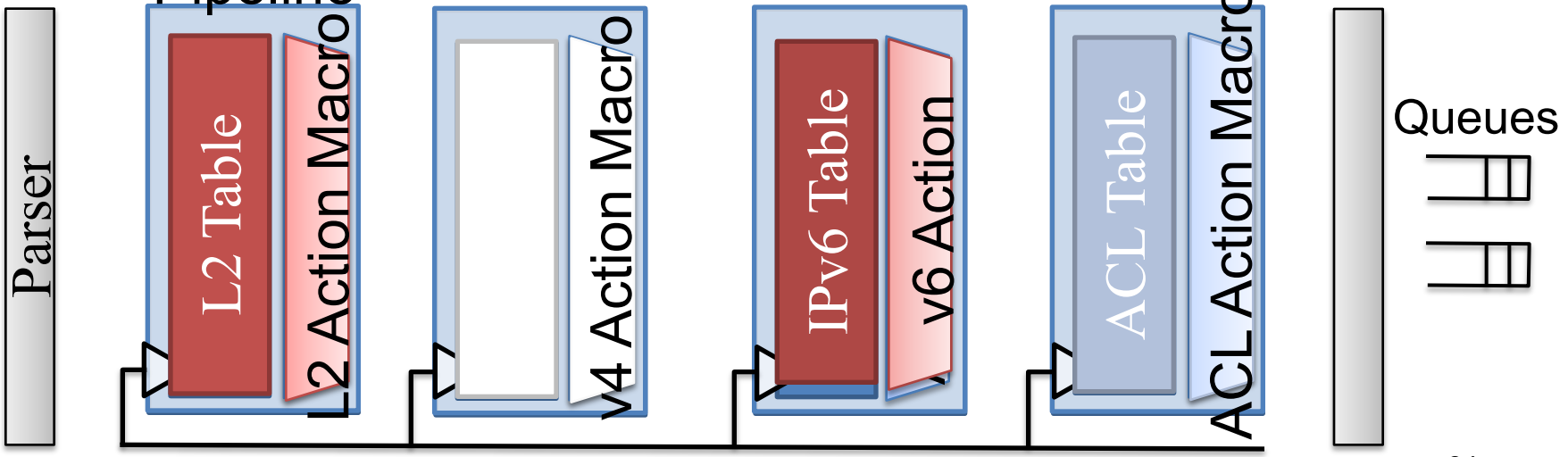


Mapping Control Flow to Programmable Switch Chip.



Control Flow Graph

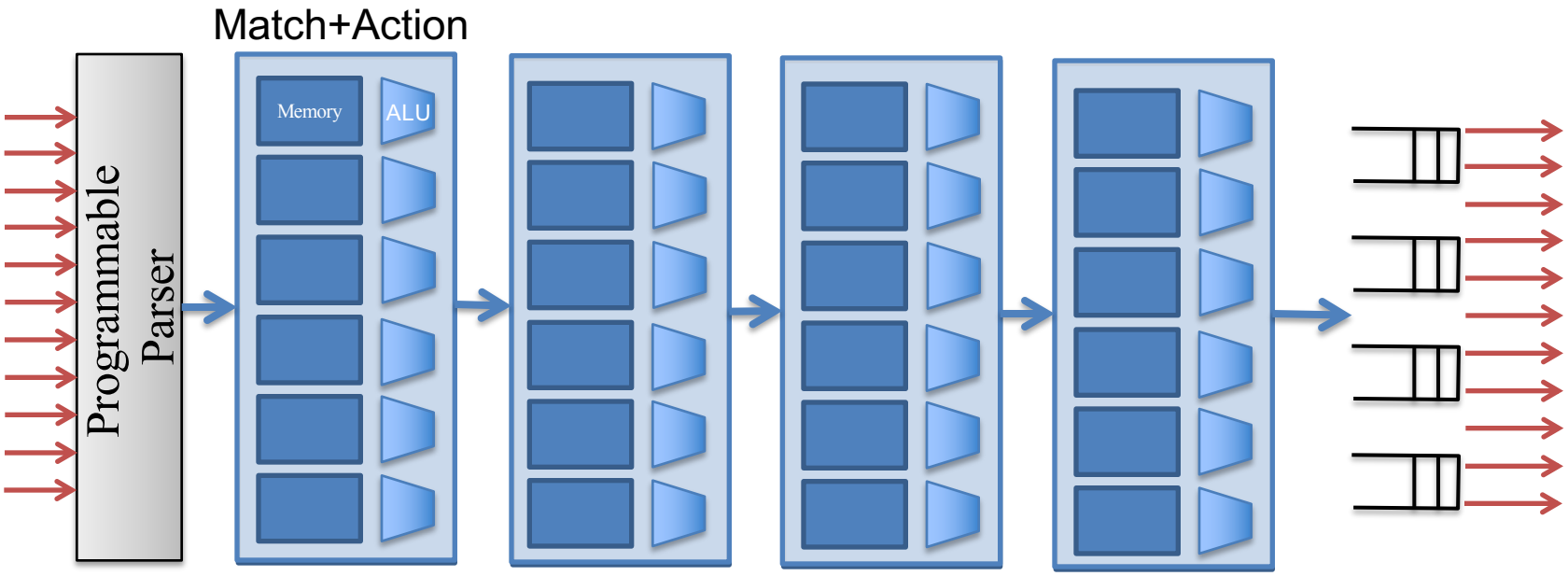
Switch Pipeline

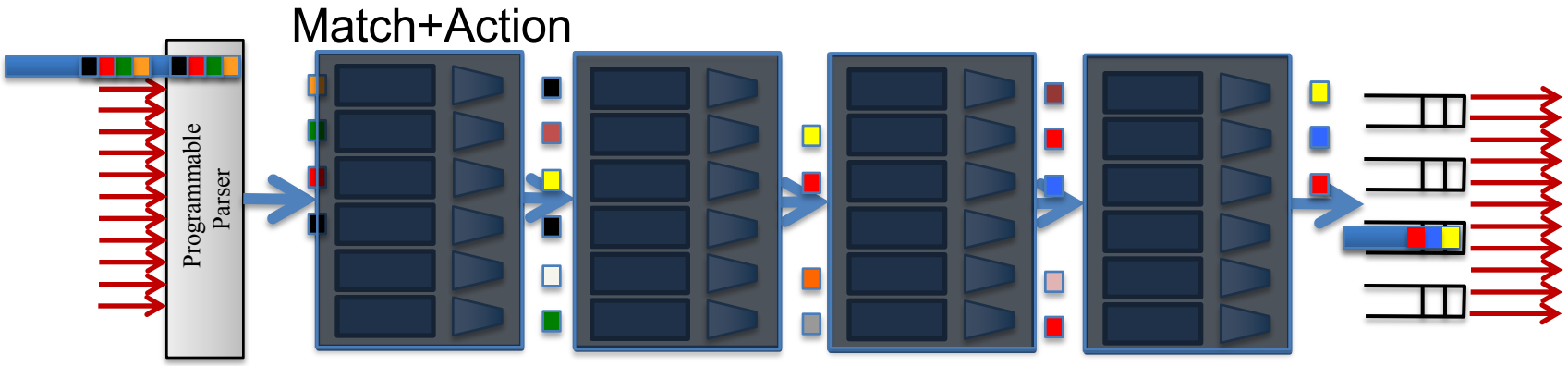


RMT: Reconfigurable Match + Action

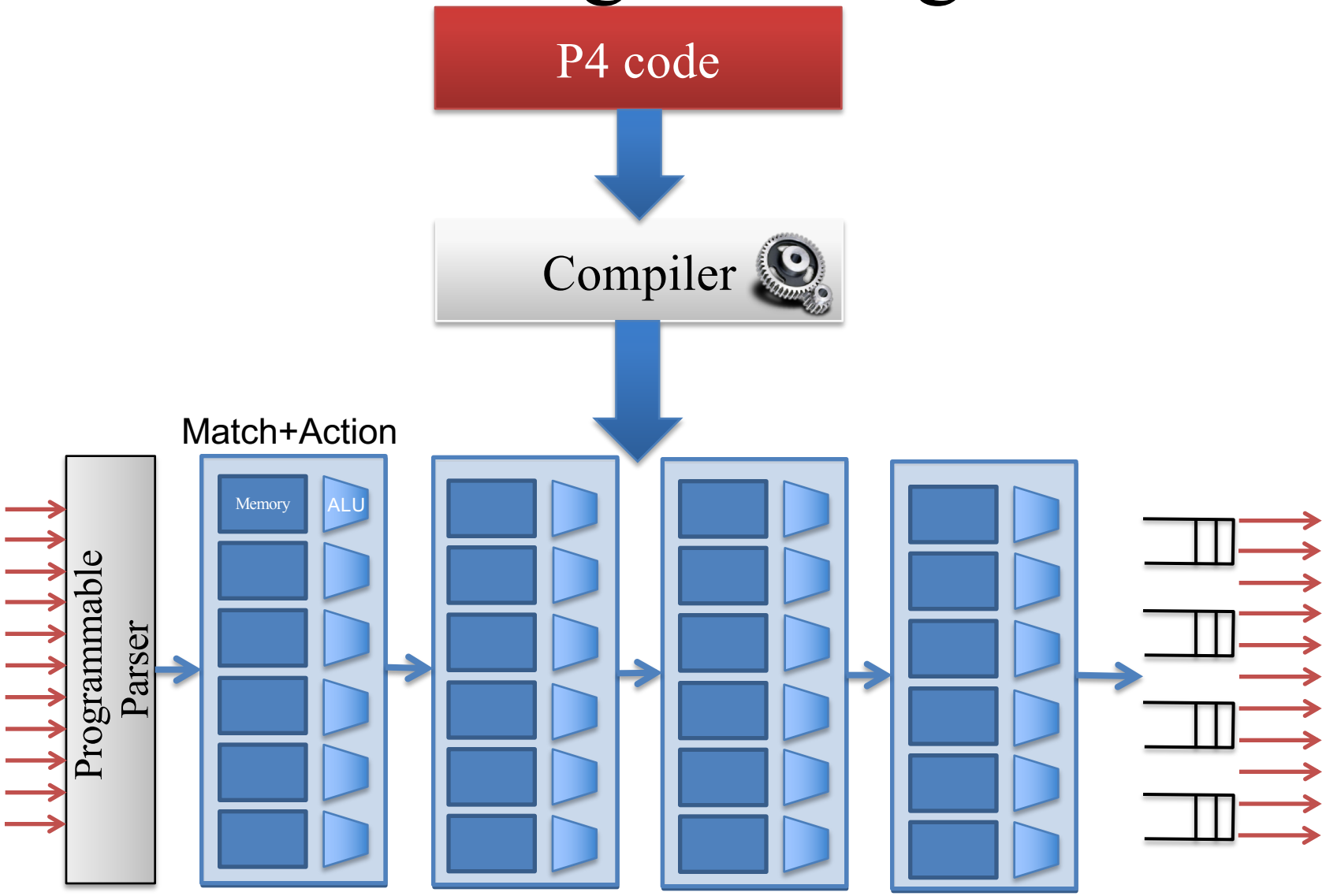
(Now more commonly called “PISA”)

PISA: Protocol Independent Switch Architecture

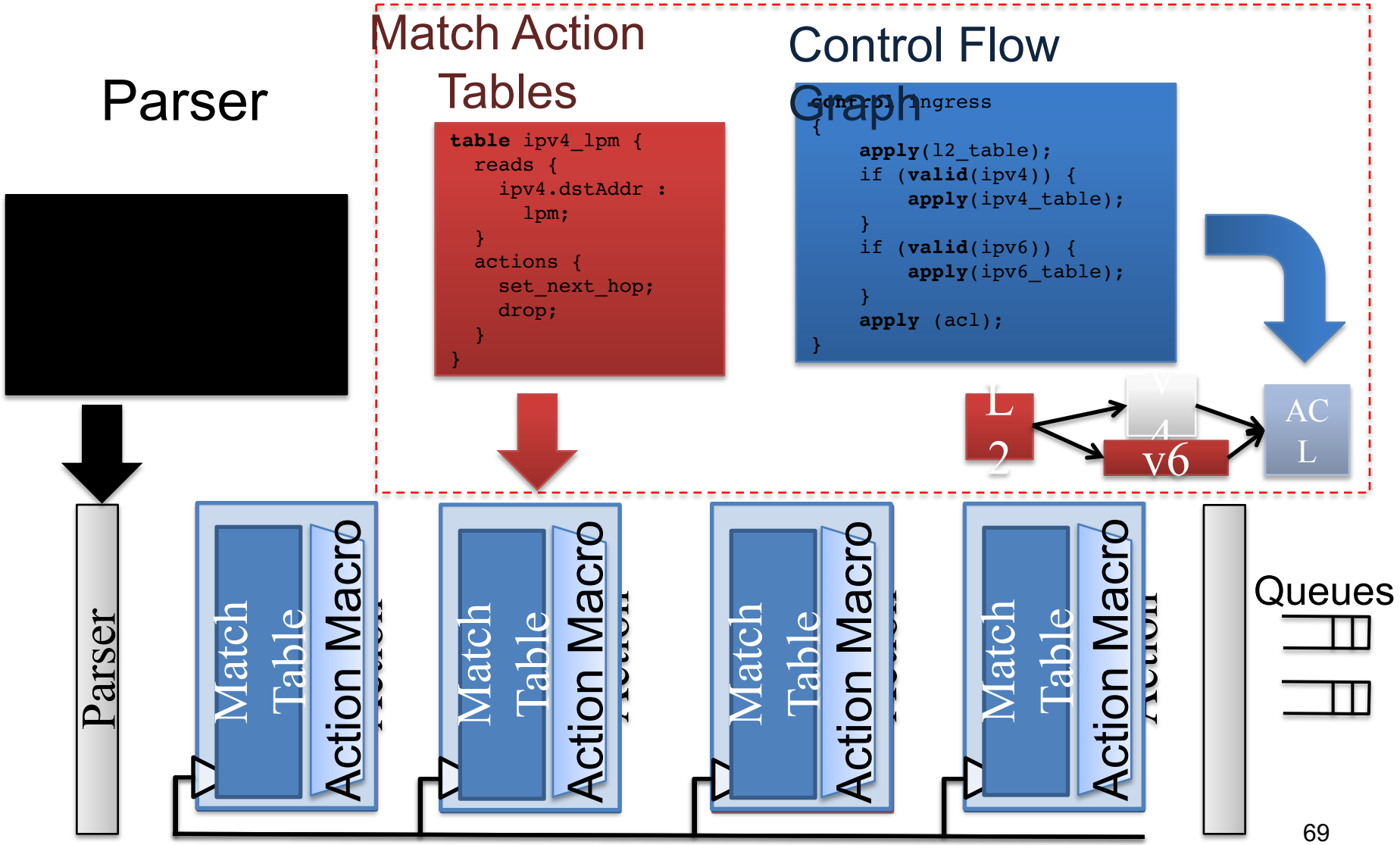



Match+Action
Programmable
Parser

P4 Programming



P4 (http://p4.org/)



Summary

- Router architecture
- Software defined networking
- Programmable routers