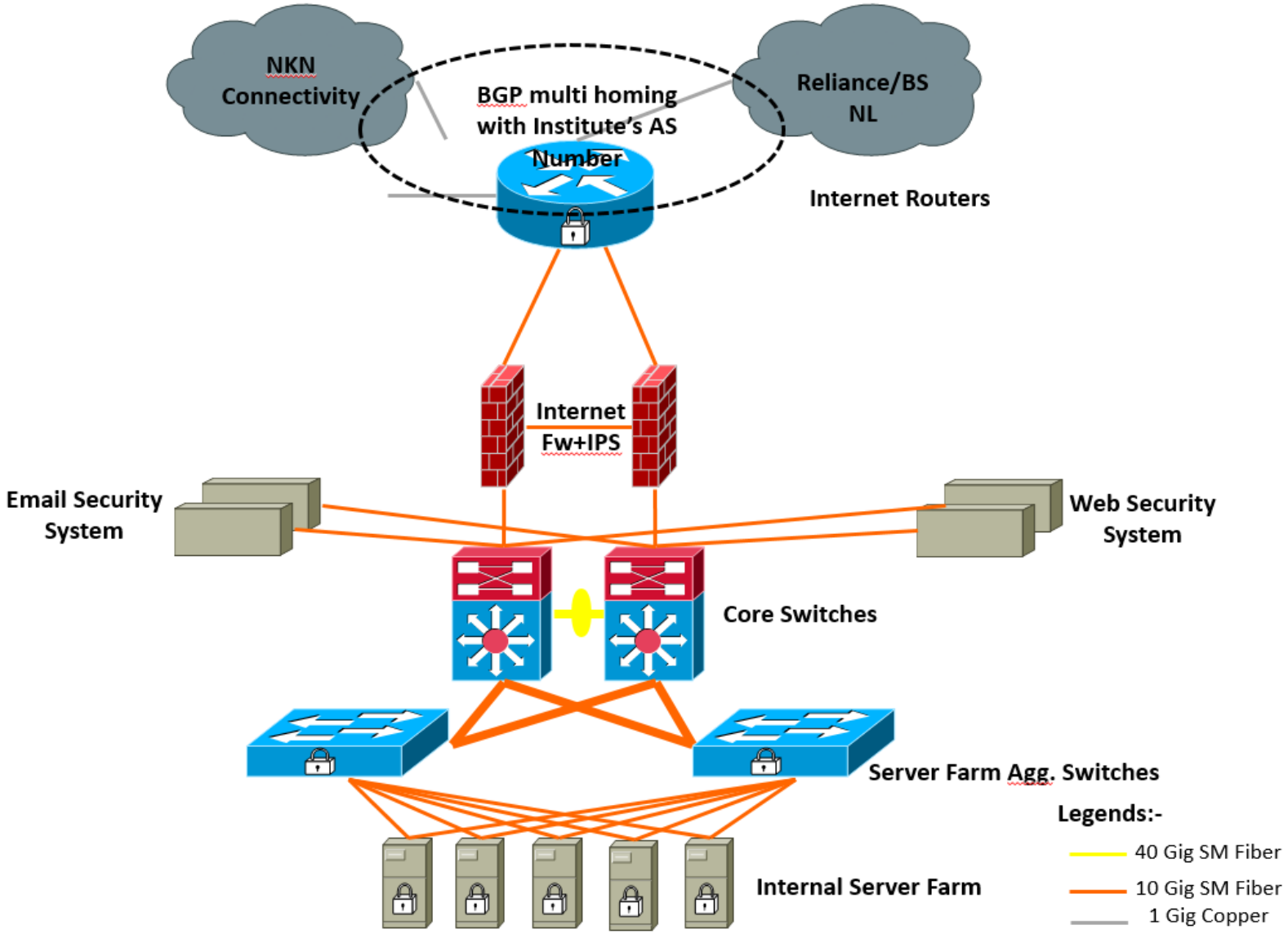
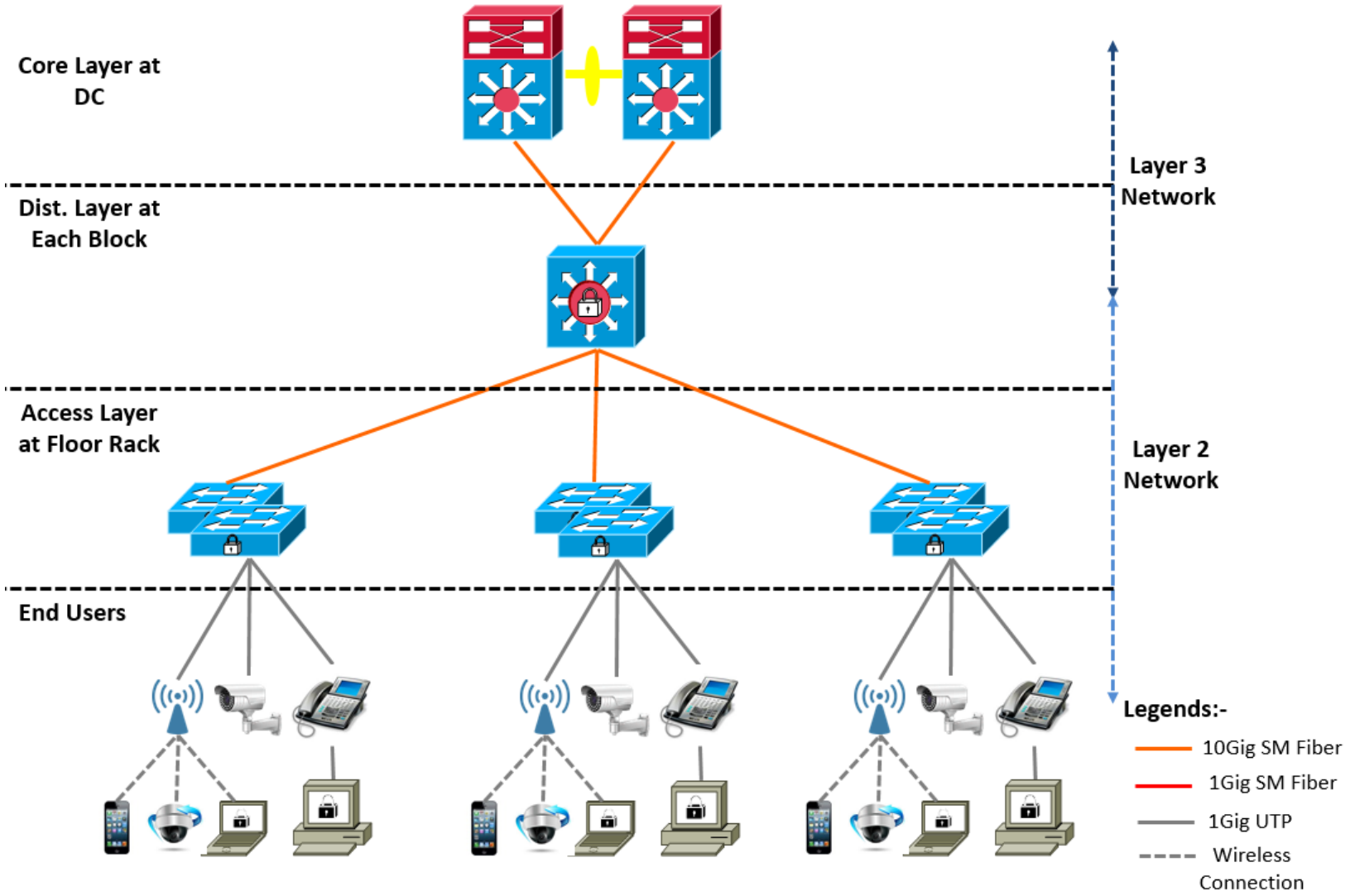
# Network Architecture at DC



# Access Network Architecture




# SDN

Anand Baswade

anand@iitbhilai.ac.in

# Source

- Software-Defined Networking: A Comprehensive Survey by Diego Kreutz and others
- CN: Top-down Approach by Ross and Kurose
- SDN and OpenFlow Slides from Saurav Das With contributions Nick McKeown, Guru Parulkar, Scott Shenker, Brandon Heller, Rob Sherwood, Guido Appenzeller, Martin Casado and many, many others…
- Talk by Prof. Scott Shenkar from UC berkeley : https://www.youtube.com/watch?v=WVs7Pc99S7w
- SDN slides @ Duke
- Prof. Raj Jain's Slides on SDN and OpenFlow

# Software defined networking (SDN)

- Internet network layer: historically implemented via distributed, per-router control approach:
  - *monolithic* router contains switching hardware, runs proprietary implementation of Internet standard protocols (IP, RIP, OSPF, BGP) in proprietary router OS (e.g., Cisco IOS)
  - different "middleboxes" for different network layer functions: firewalls, load balancers, NAT boxes, ..
- ~2005: renewed interest in rethinking network control plane

# Network-layer functions

- forwarding: move packets from router's input to appropriate router output

*data plane*

- routing: determine route taken by packets from source to destination

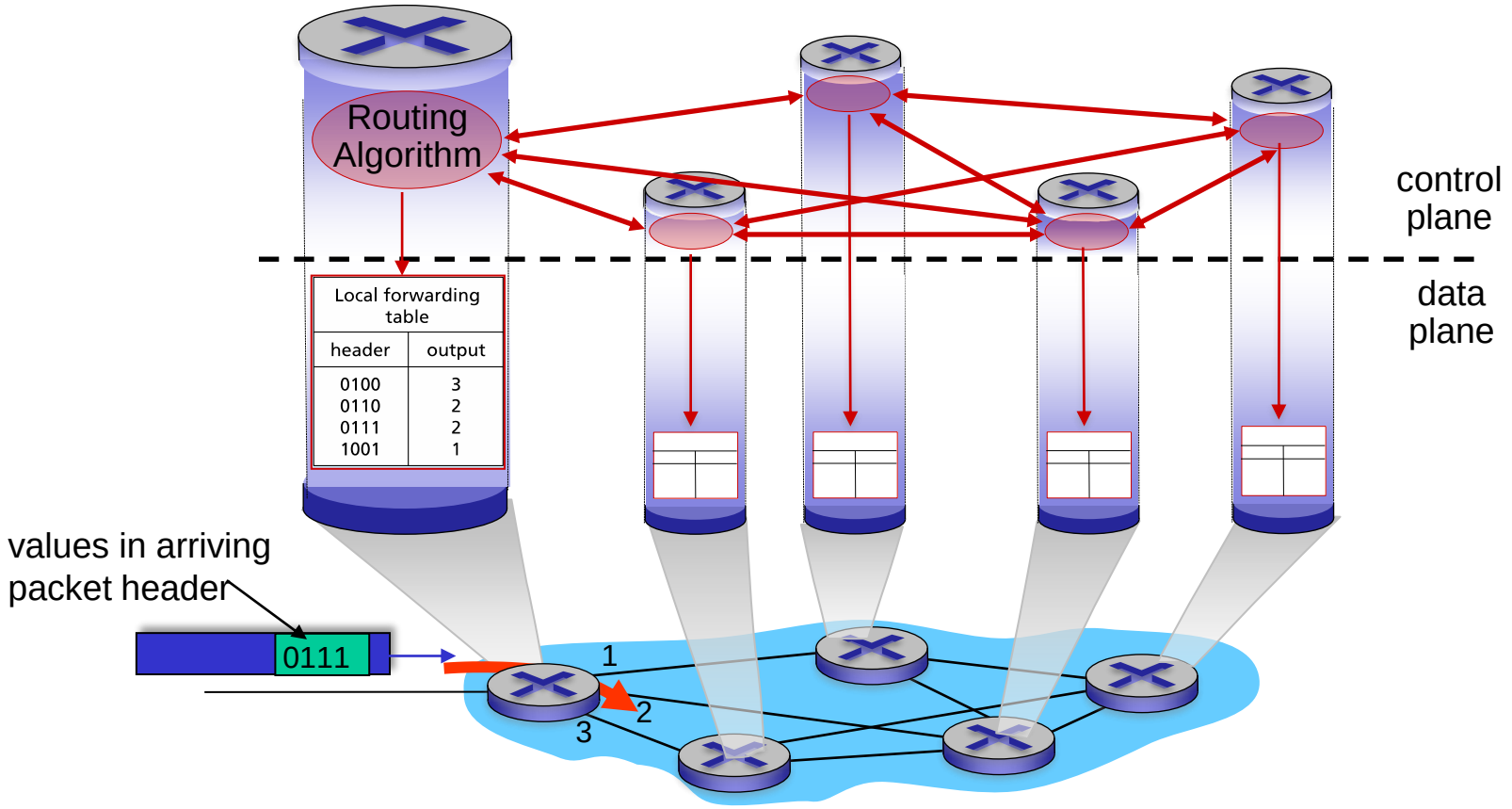
*control plane*

Two approaches to structuring network control plane:

- per-router control (traditional)
- logically centralized control (software defined networking)

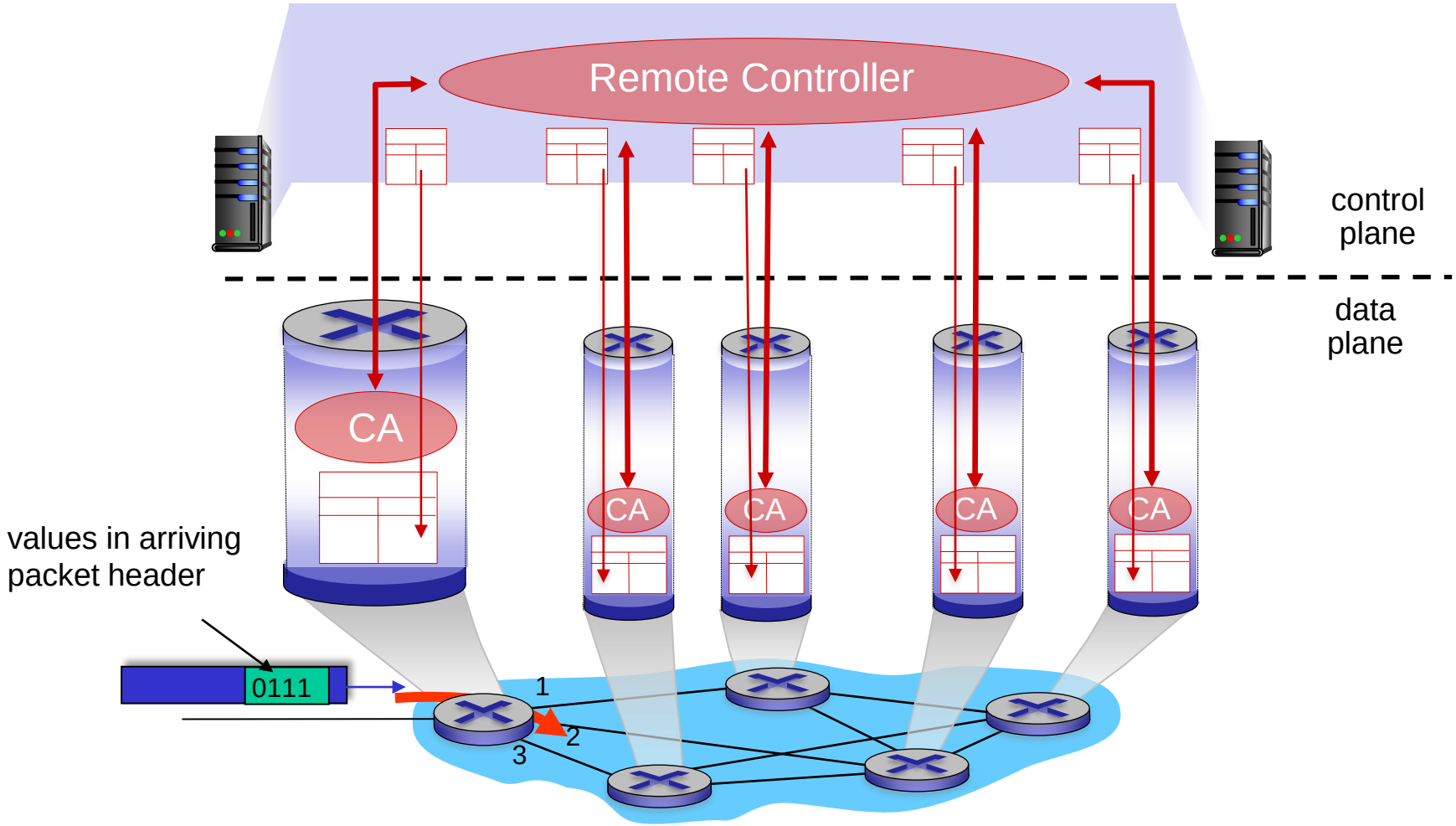
# Per-router control plane

Individual routing algorithm components *in each and every router* interact in the control plane



# Software-Defined Networking (SDN) control plane

## Remote controller computes, installs forwarding tables in routers

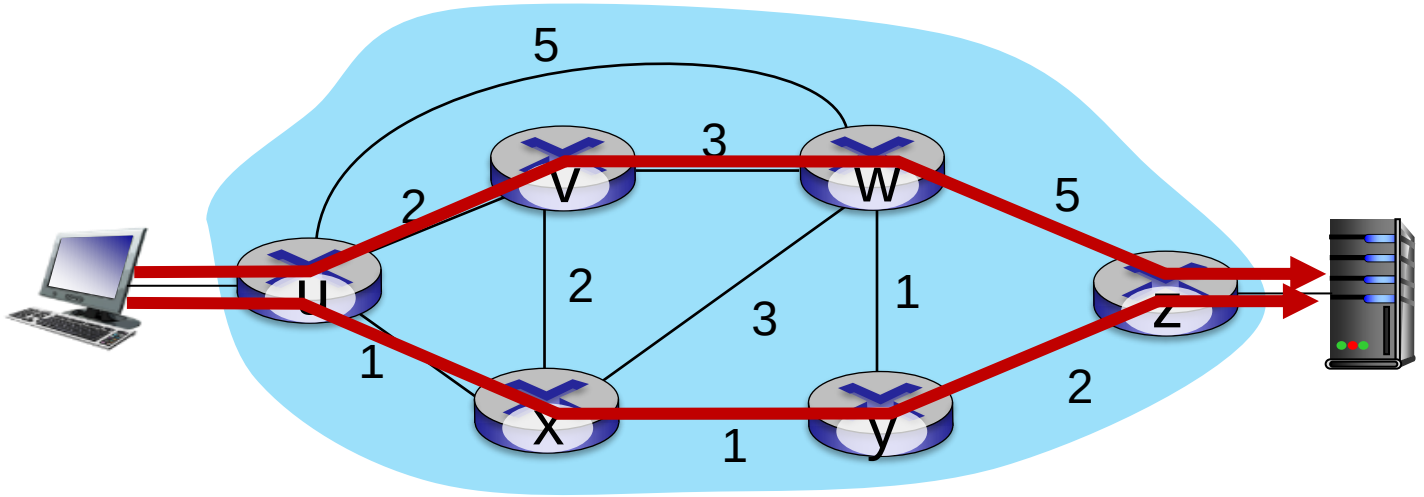


# Software defined networking (SDN)

*Why* a *logically centralized* control plane?

- easier network management: avoid router misconfigurations, greater flexibility of traffic flows
- table-based forwarding allows "programming" routers
  - centralized "programming" easier: compute tables centrally and distribute
  - distributed "programming" more difficult: compute tables as result of distributed algorithm (protocol) implemented in each-and-every router
- open (non-proprietary) implementation of control plane
  - faster innovation

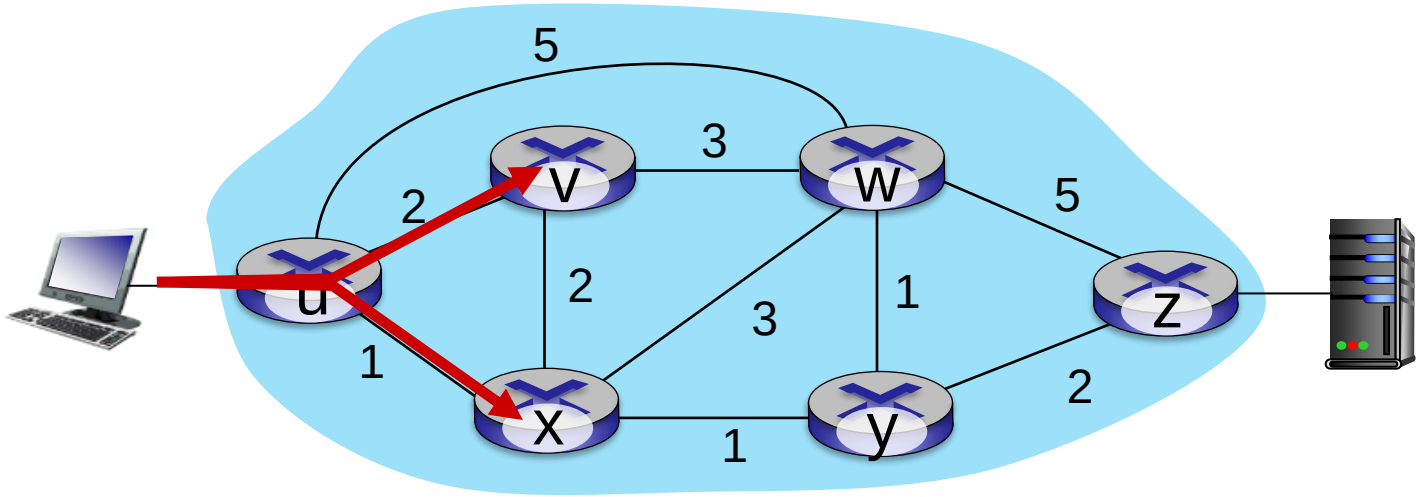
# Traffic engineering: difficult with traditional routing




*Q:* what if network operator wants u-to-z traffic to flow along *uvwz*, rather than *uxyz*?

*A:* need to re-define link weights so traffic routing algorithm computes routes accordingly (or need a new routing algorithm)!

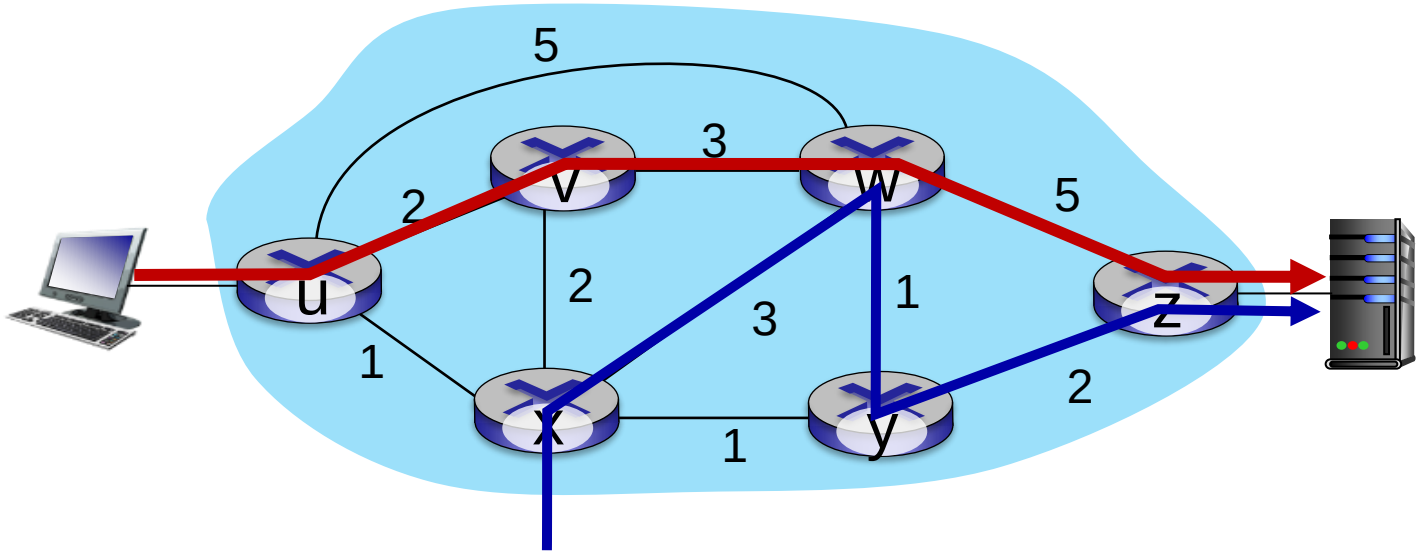
# Traffic engineering: difficult with traditional routing




*Q:* what if network operator wants to split u-to-z traffic along uvwz *and* uxyz (load balancing)?

*A:* can't do it (or need a new routing algorithm)

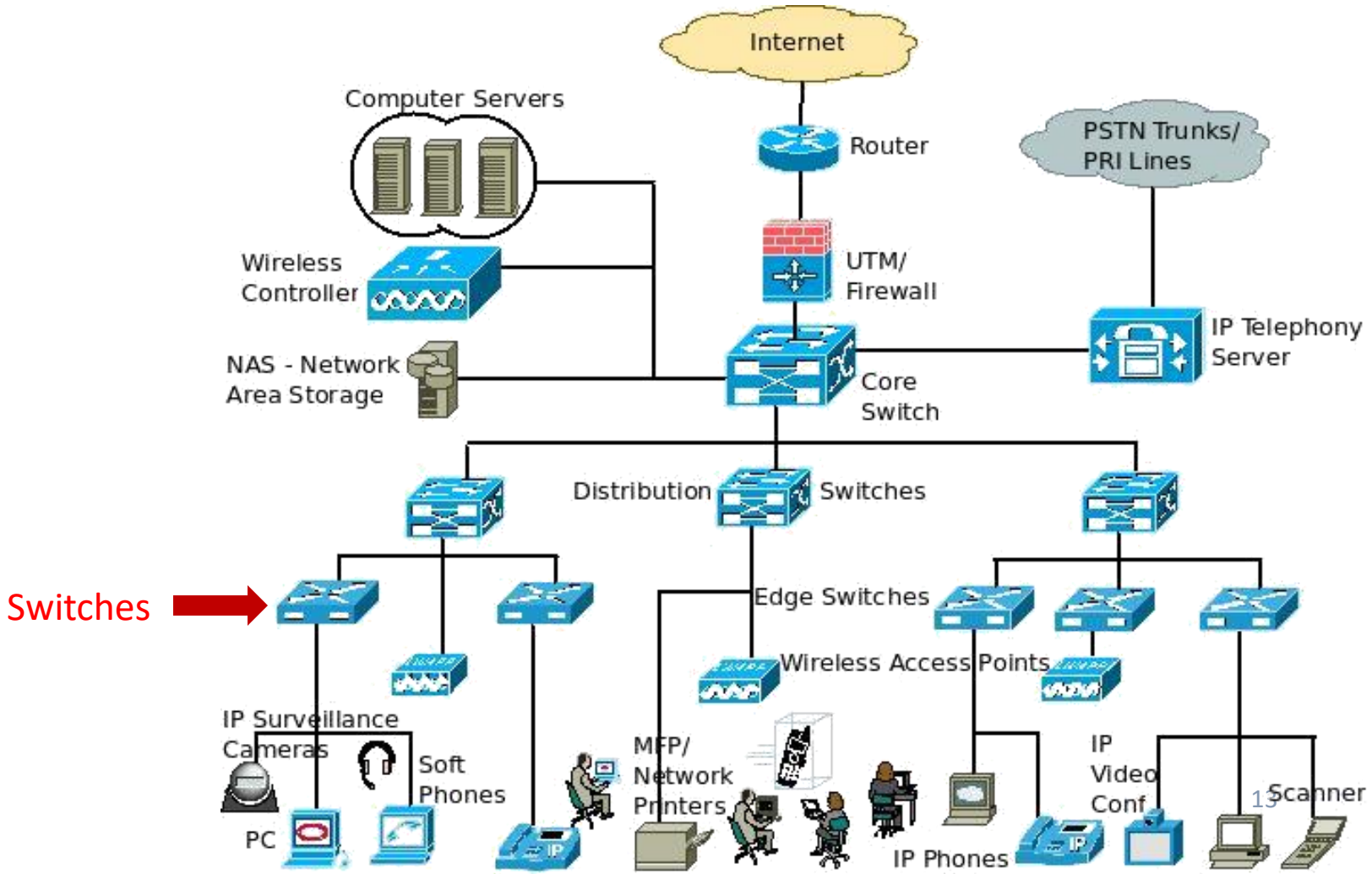
# Traffic engineering: difficult with traditional routing




*Q:* what if w wants to route blue and red traffic differently from w to z?

*A:* can't do it (with destination-based forwarding, and LS, DV routing)

# Limitations of Current Networks

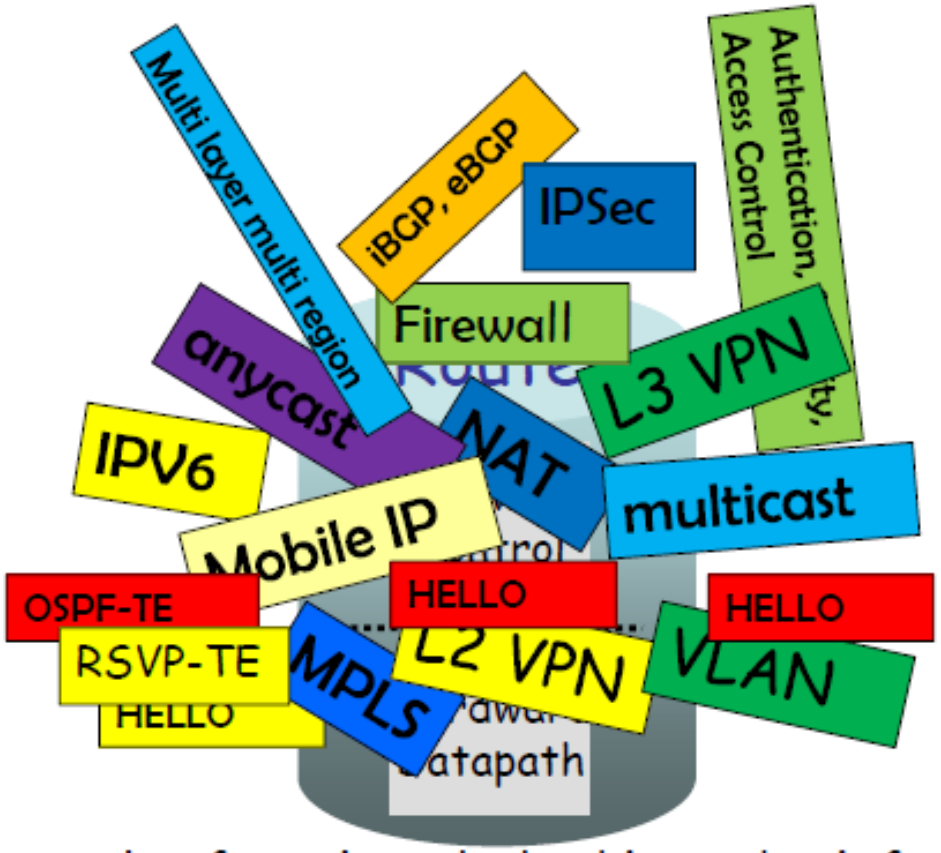



http://www.excitingip.net/27/a-basic-enterprise-lan-network-architecture-block-diagram-and-components/

# Networks are getting Complex

- Networks used to be simple
  - Basic Ethernet/IP straightforward, easy to manage
- New control requirements have led to complexity
  - Access Control Lists, VLANs, Traffic Engineering, Middleboxes, DPI, ….
- The infrastructure still works … (Why?)
  - Only because of your great ability to master complexity
- Extreme complexity often signifies weak foundation

# Cont..

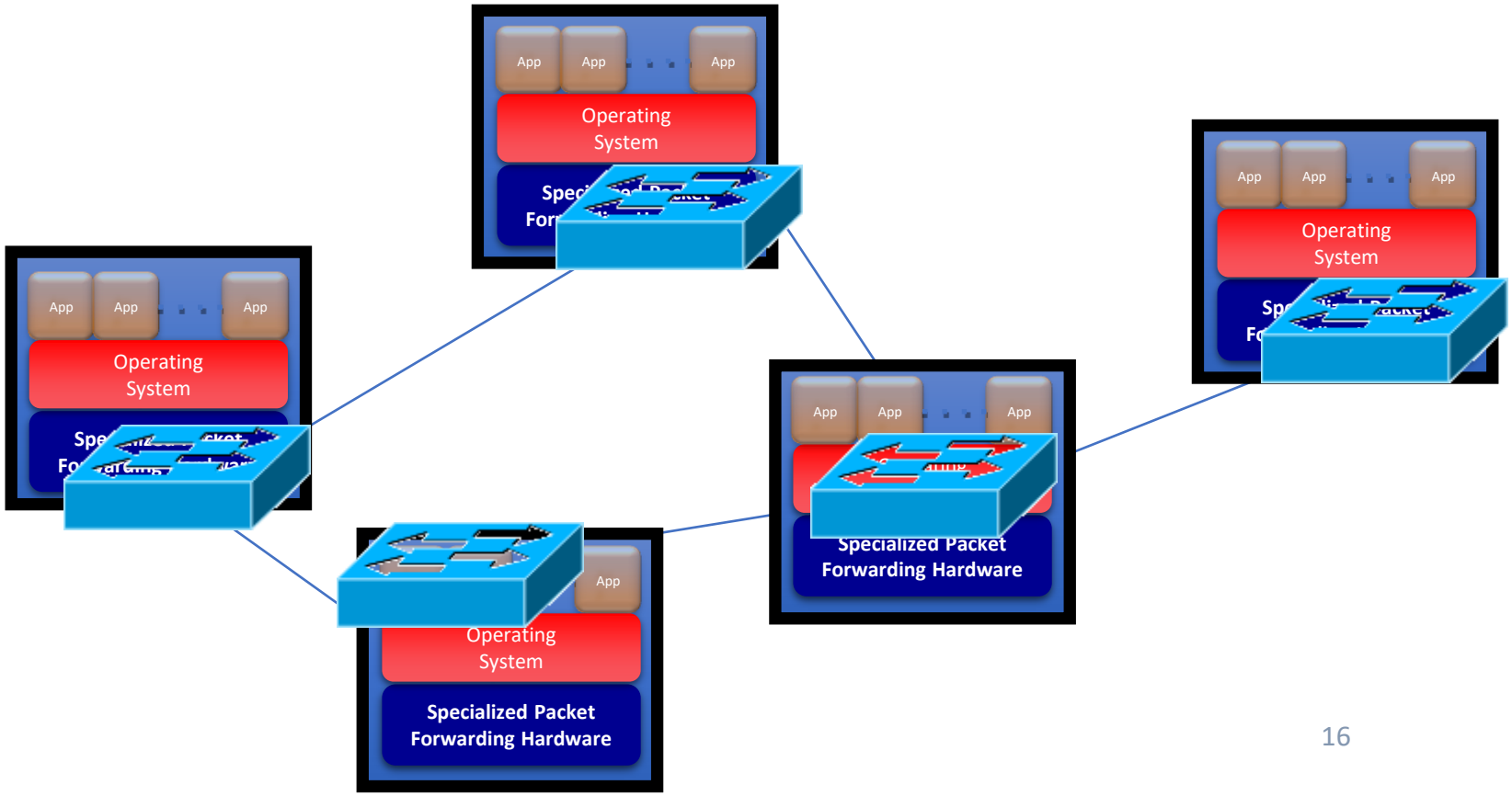



Many complex functions baked into the infrastructure

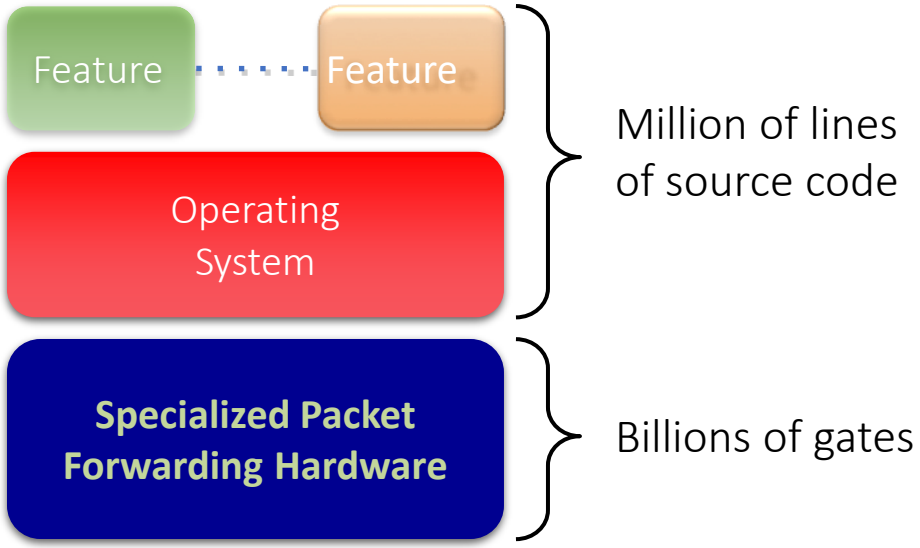
*OSPF, BGP, multicast, differentiated services,*
*Traffic Engineering, NAT, firewalls, MPLS, redundant layers, …*

# Limitations of Current Networks

- **Old ways to configure a network**



# Limitations of Current Networks




Many complex functions baked into infrastructure

*OSPF, BGP, multicast, differentiated services, Traffic Engineering, NAT, firewalls, …*

Cannot dynamically change according to network conditions

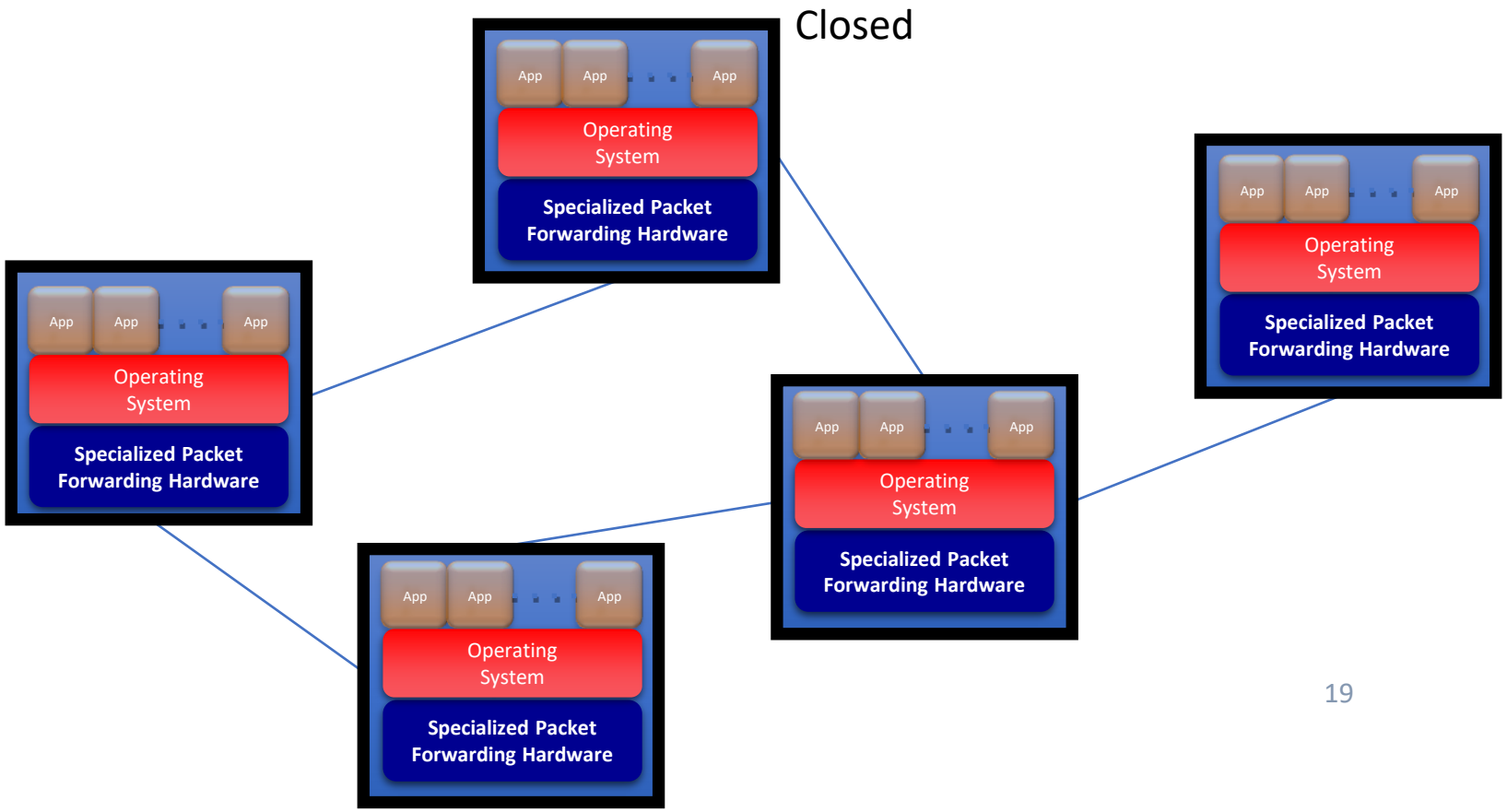
# Limitations of Current Networks

- **No control plane abstraction for the whole network!**

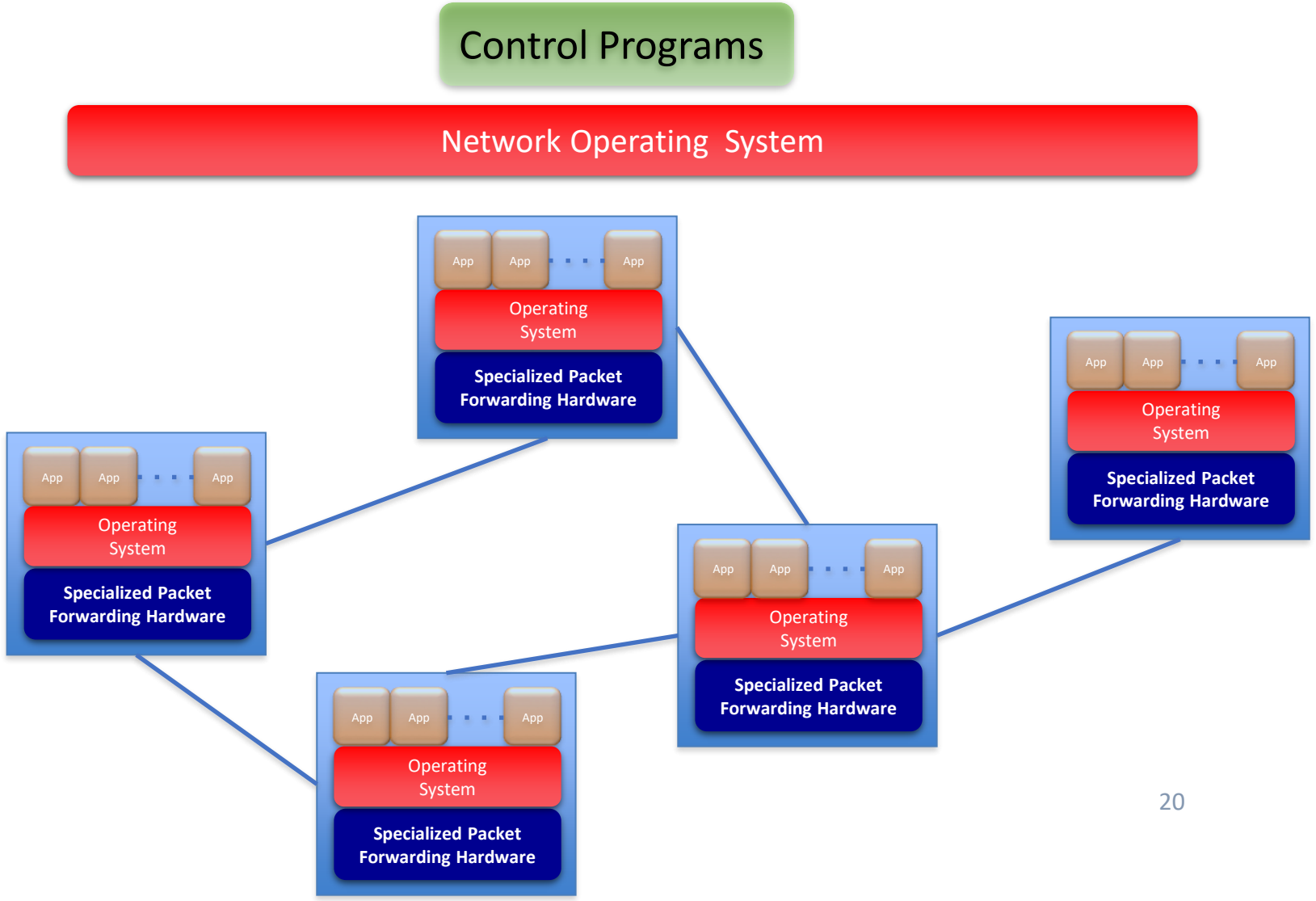
- **It's like old times – when there was no OS…**

Wilkes with the EDSAC, 1949

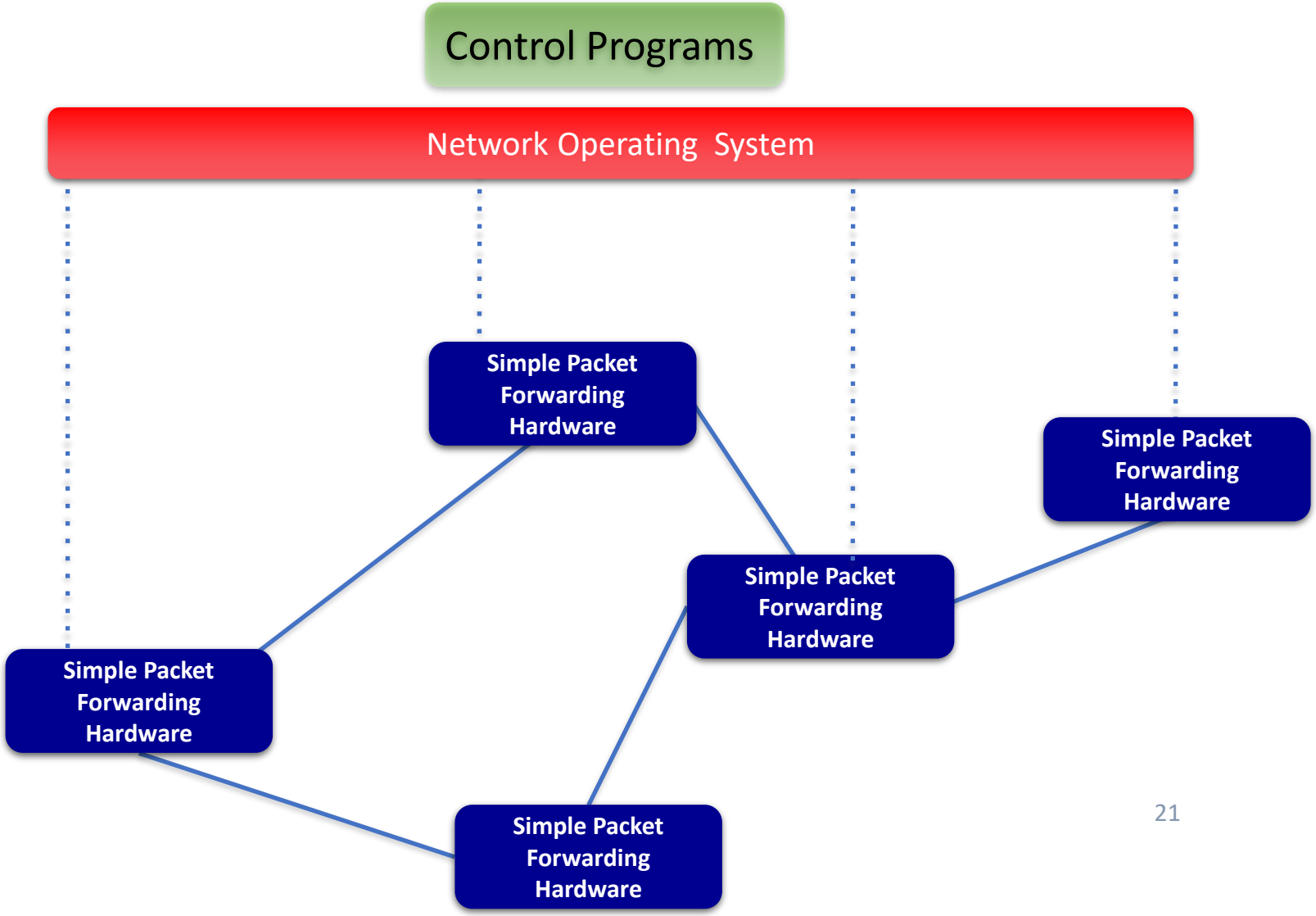
# Idea: An OS for Networks



# Idea: An OS for Networks



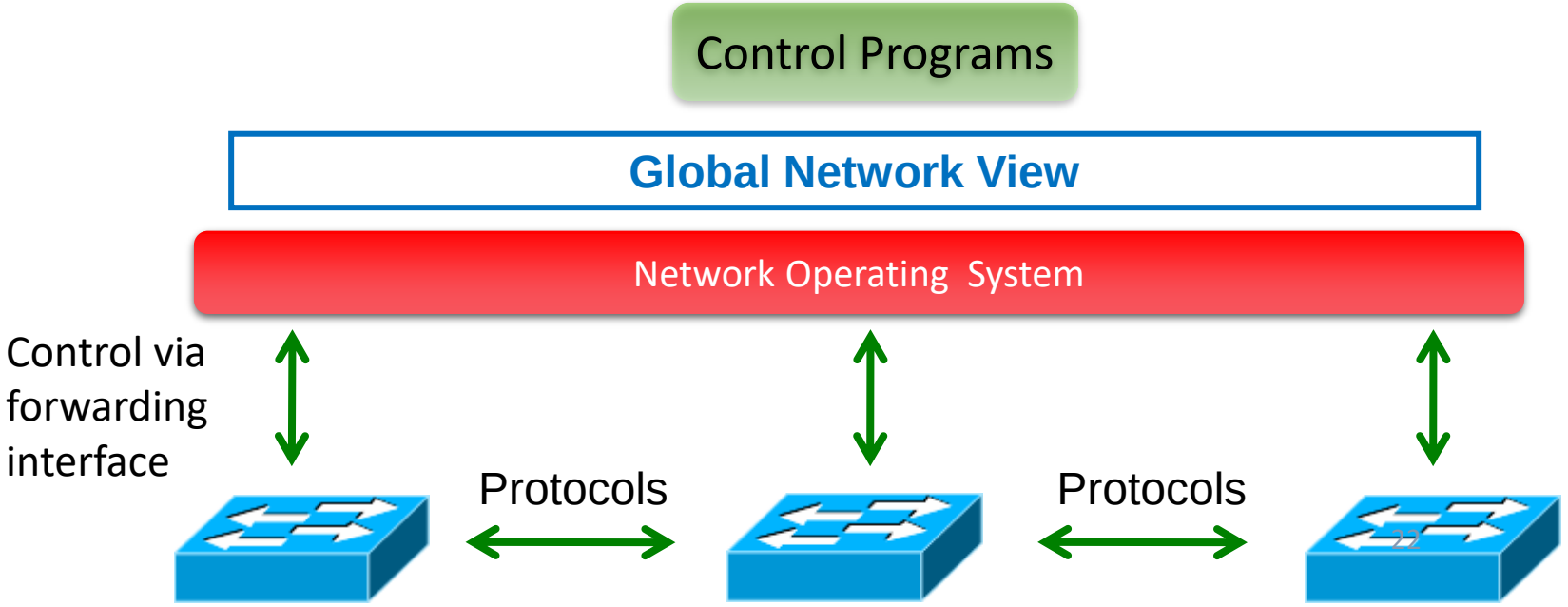
# Idea: An OS for Networks



# Idea: An OS for Networks

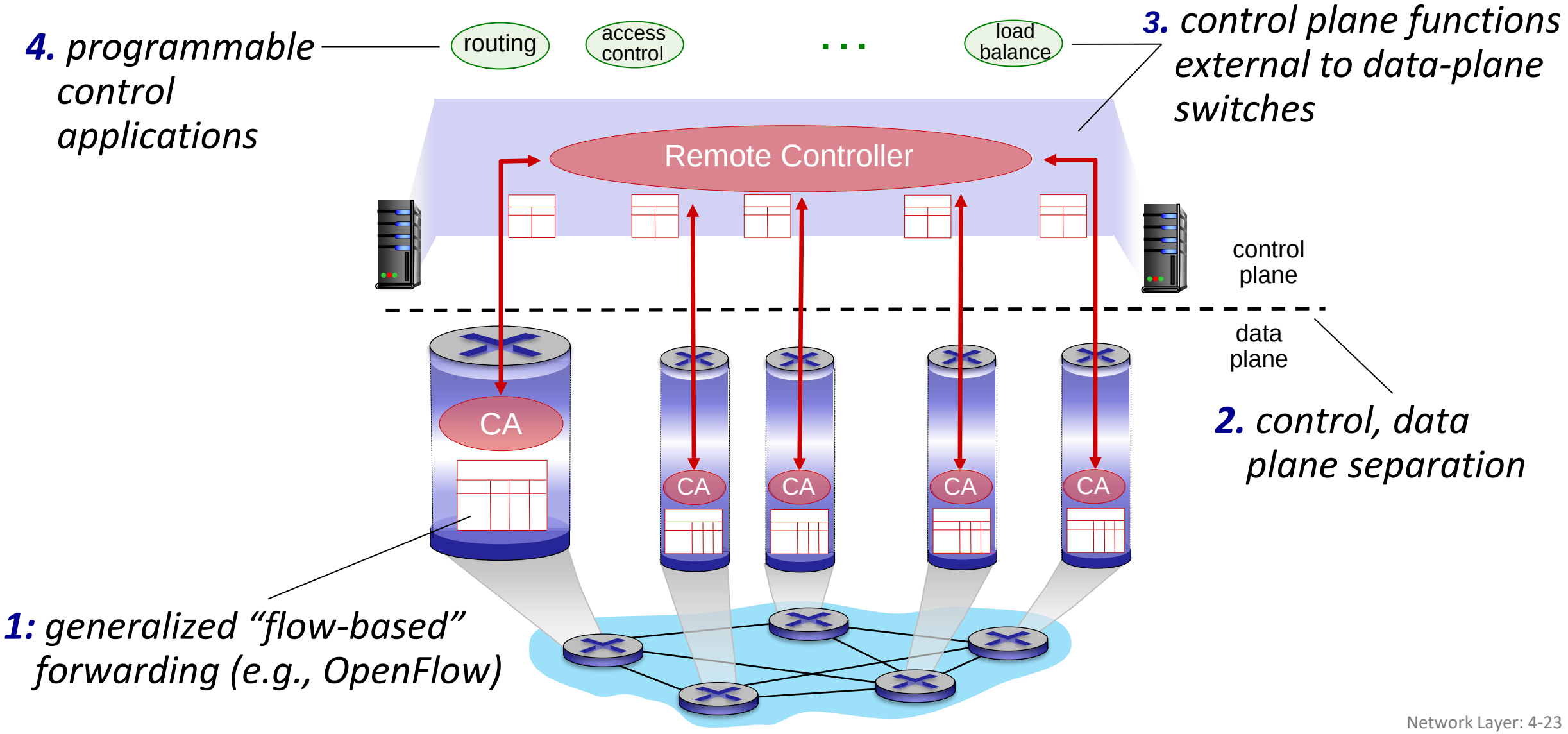
- **Towards an Operating System for Networks”**

**Software-Defined Networking (SDN)**

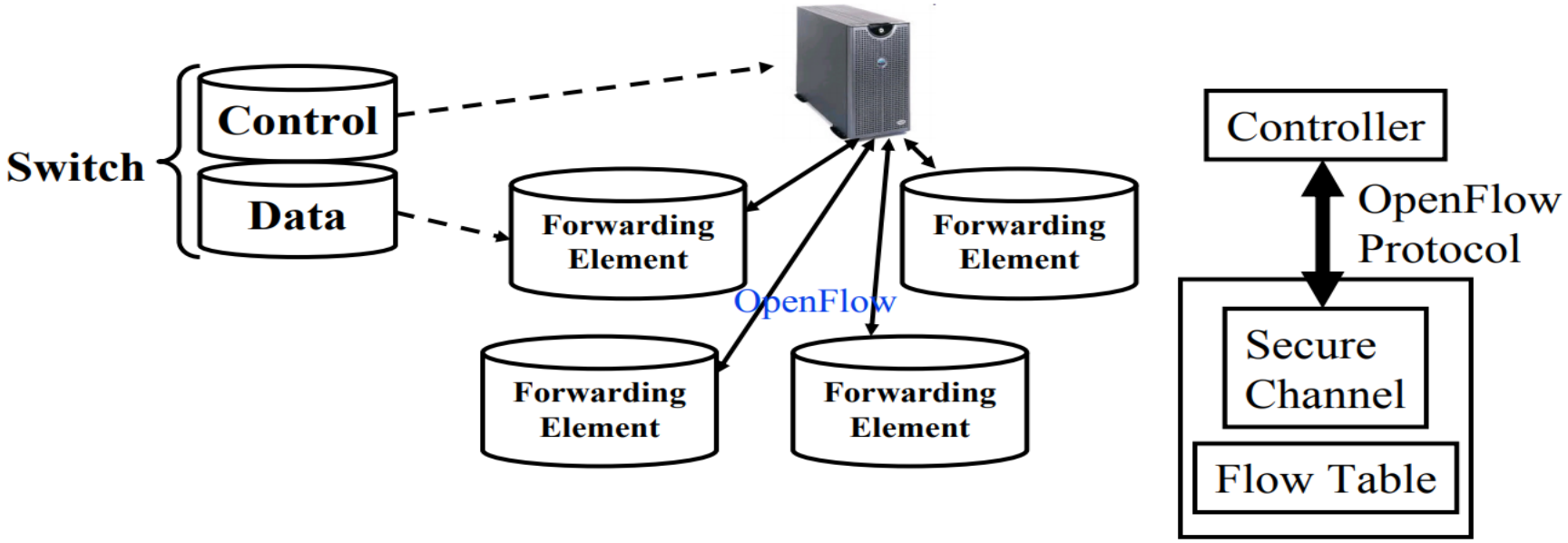



The Future of Networking, and the Past of Protocols, Scott Shenker, *with Martin Casado, Teemu Koponen, Nick McKeown*

# Software defined networking (SDN)



# Separation of Control and Data Plane



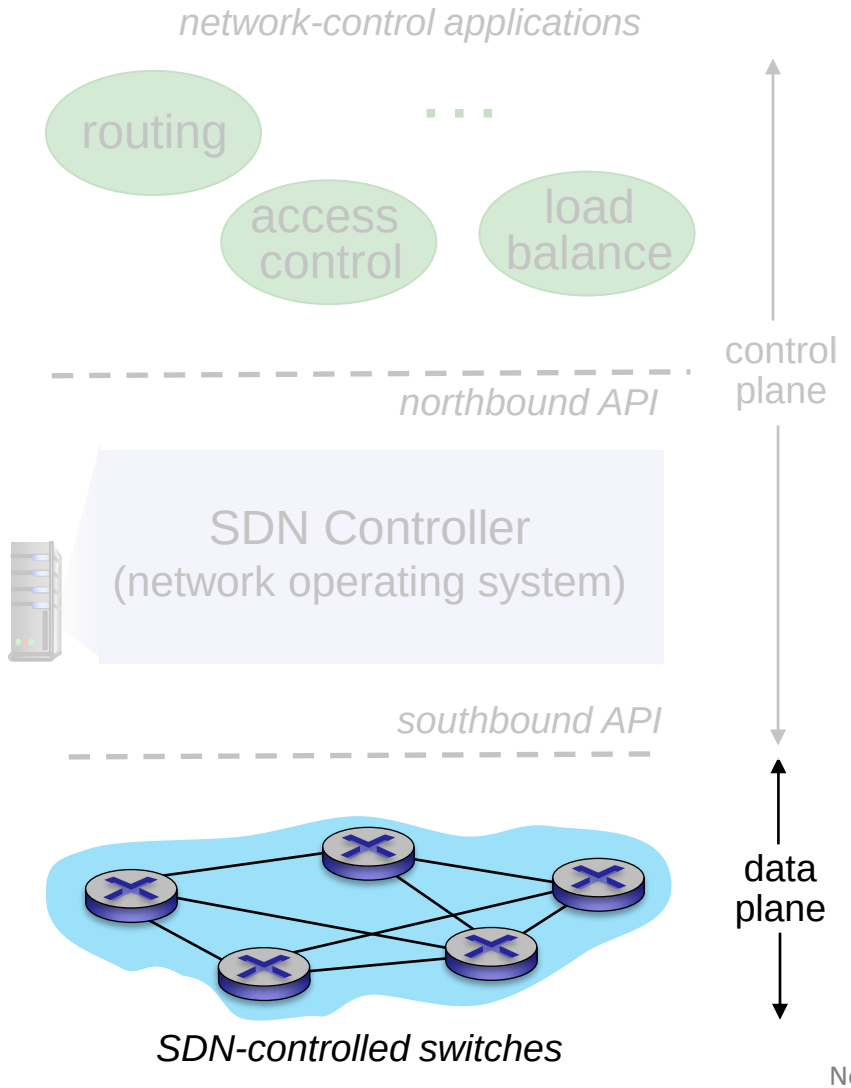

- Control logic is moved to a controller
- Switches only have forwarding elements
- One expensive controller with a lot of cheap switches
- OpenFlow is the protocol to send/receive forwarding rules from controller to switches

Source: Prof. Raj Jain's OpenFlow slides

# Software defined networking (SDN)

## Data-plane switches:

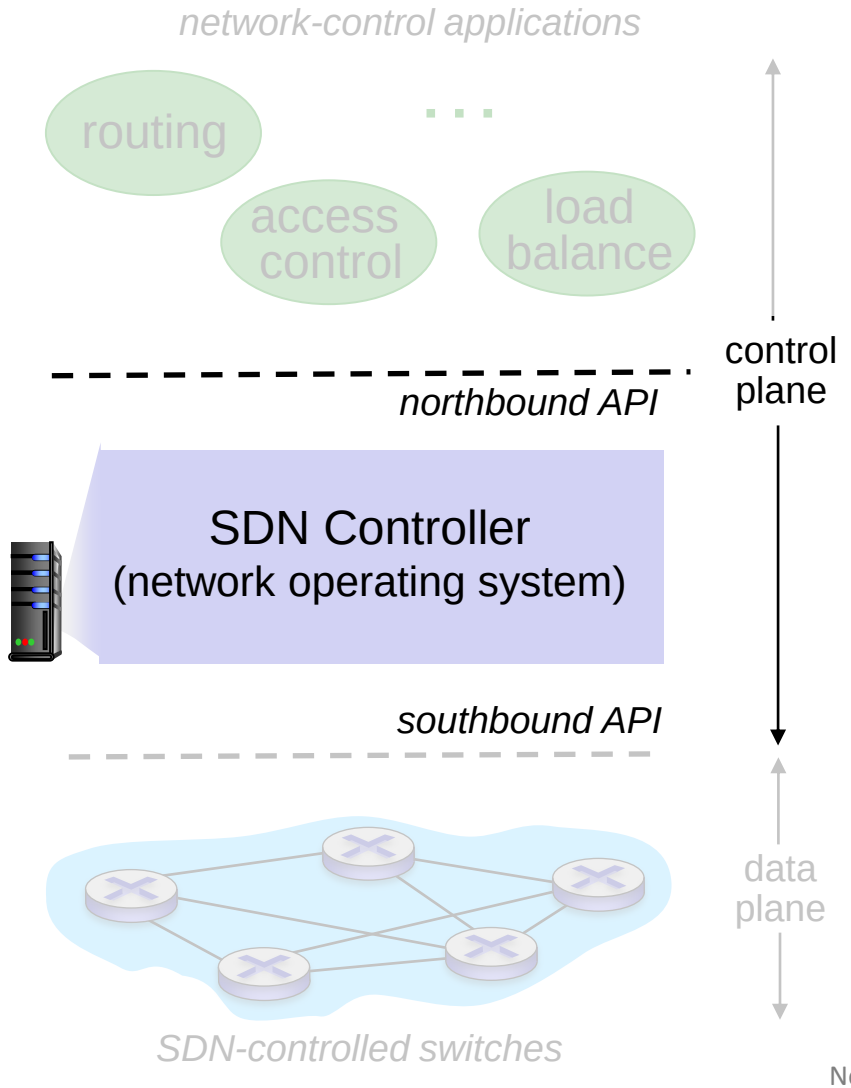
- fast, simple, commodity switches implementing generalized data-plane forwarding in hardware
- flow (forwarding) table computed, installed under controller supervision
- API for table-based switch control (e.g., OpenFlow)
  - defines what is controllable, what is not
- protocol for communicating with controller (e.g., OpenFlow)



# Software defined networking (SDN)

## SDN controller (network OS):

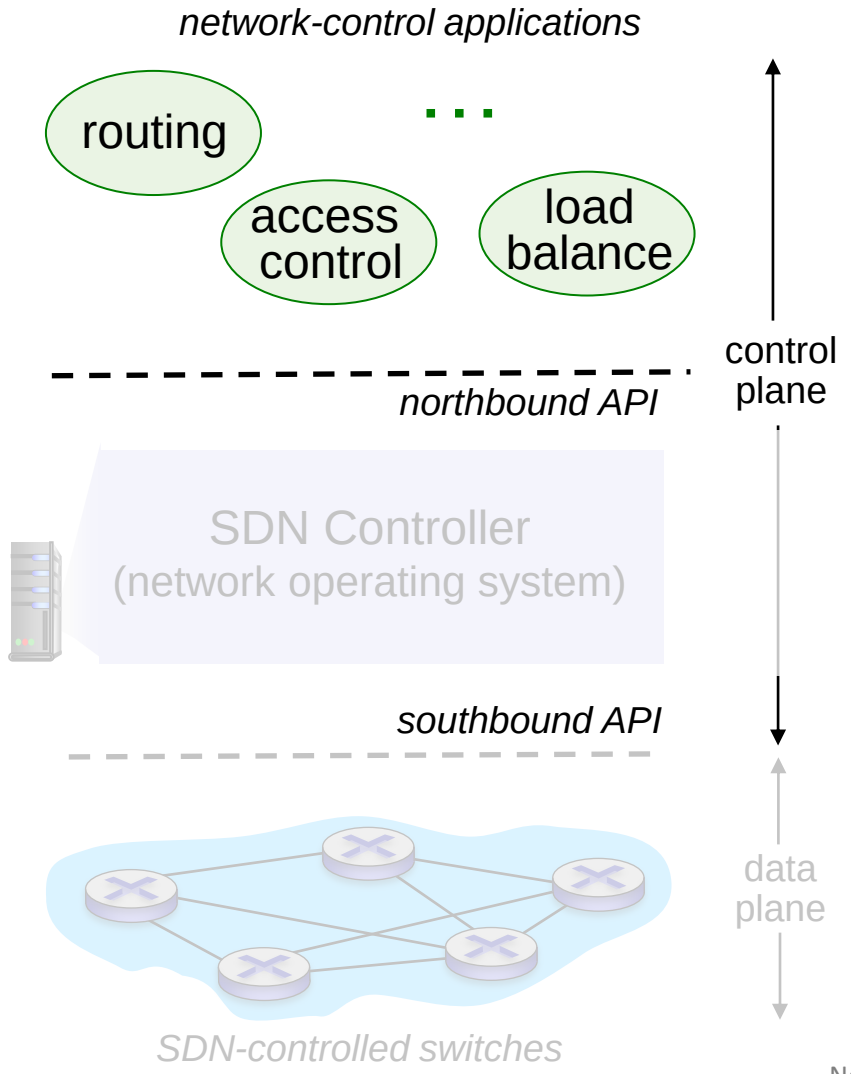
- maintain network state information
- interacts with network control applications “above” via northbound API
- interacts with network switches “below” via southbound API
- implemented as distributed system for performance, scalability, fault-tolerance, robustness



# Software defined networking (SDN)

## network-control apps:

- "brains" of control: implement control functions using lower-level services, API provided by SDN controller
- *unbundled:* can be provided by 3rd party: distinct from routing vendor, or SDN controller



# Software Defined Networking

- **No longer designing distributed control protocols**

- **Much easier to write, verify, maintain, …**
  - An interface for programming

- **NOS serves as fundamental control block**
  - With a global view of network

# SDN architecture and its fundamental abstractions

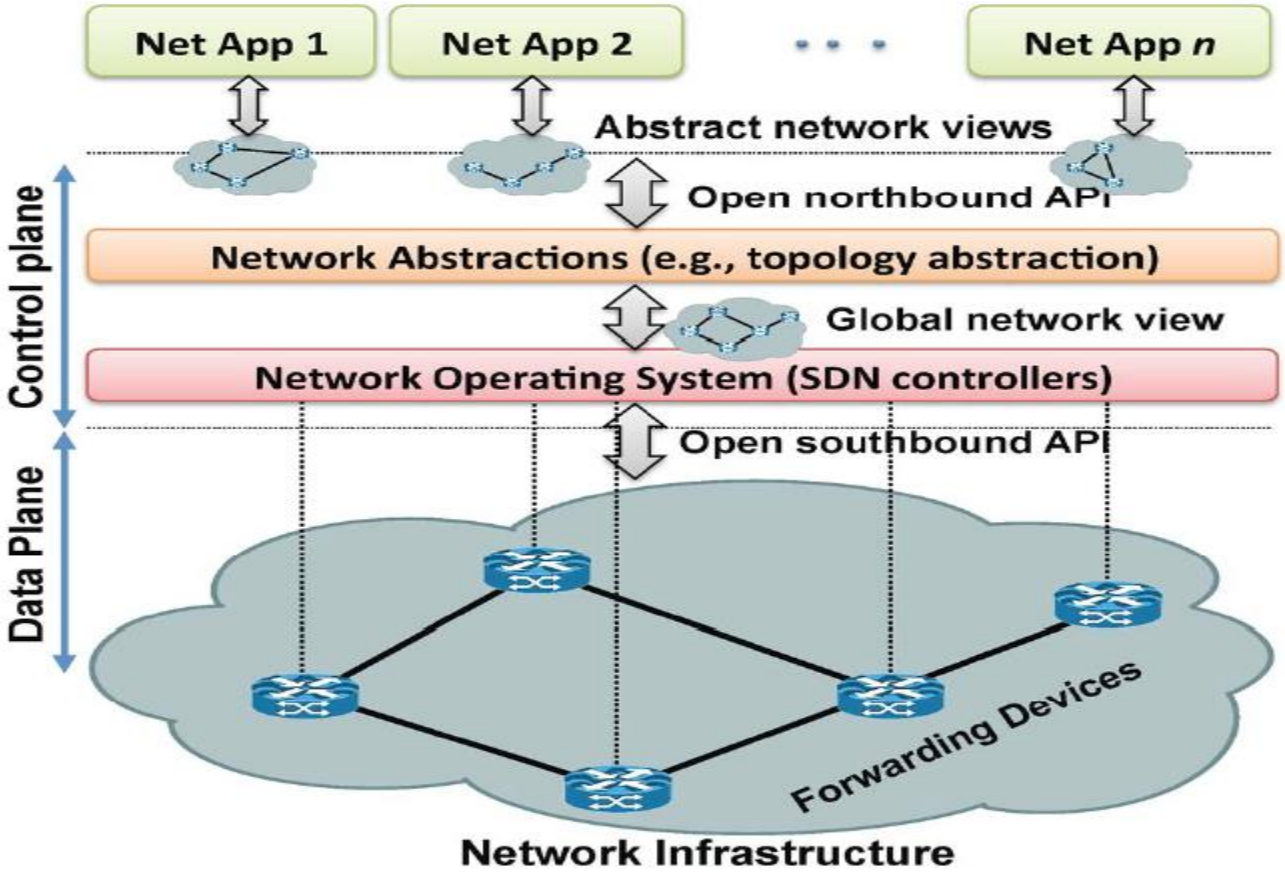
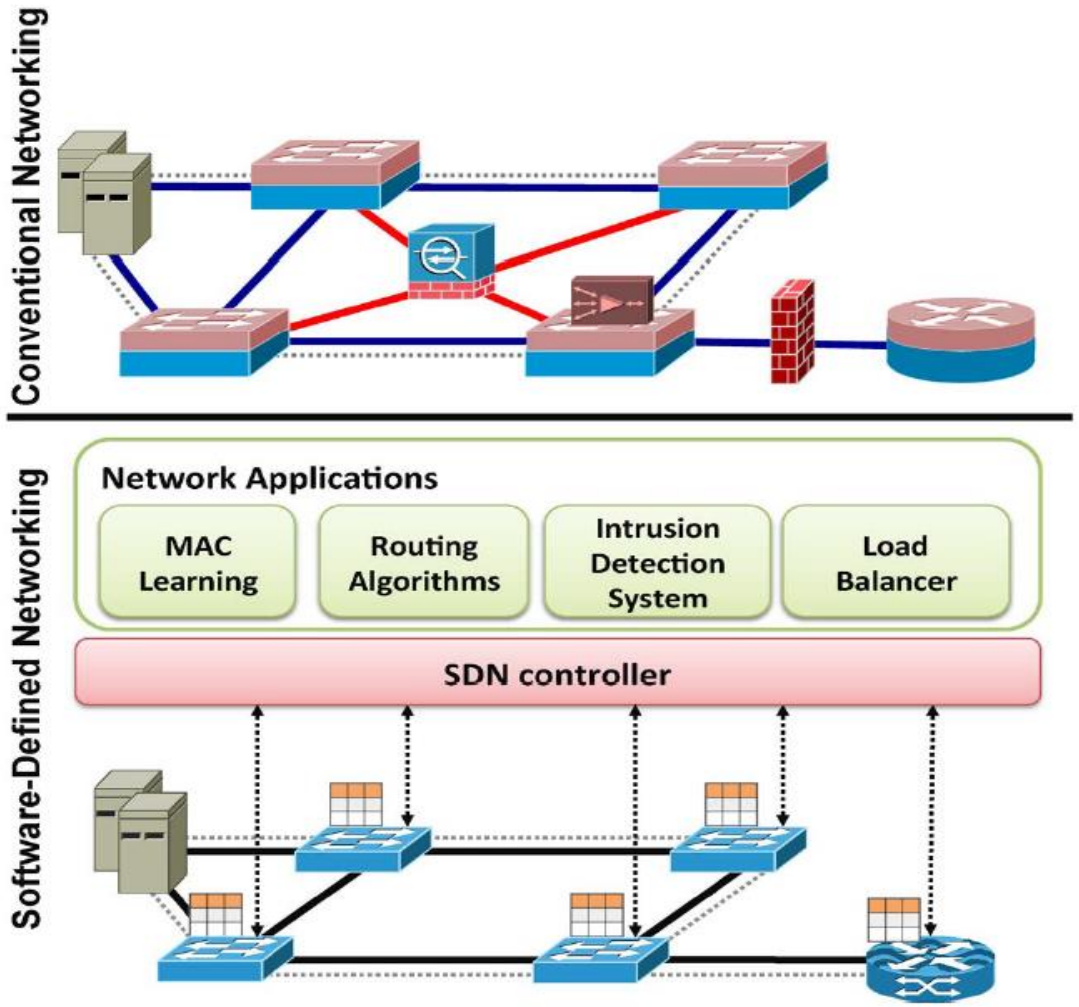



**Fig. 4.** *SDN architecture and its fundamental abstractions.*

# Traditional networking versus SDN



# Distributed controllers: east/westbound APIs

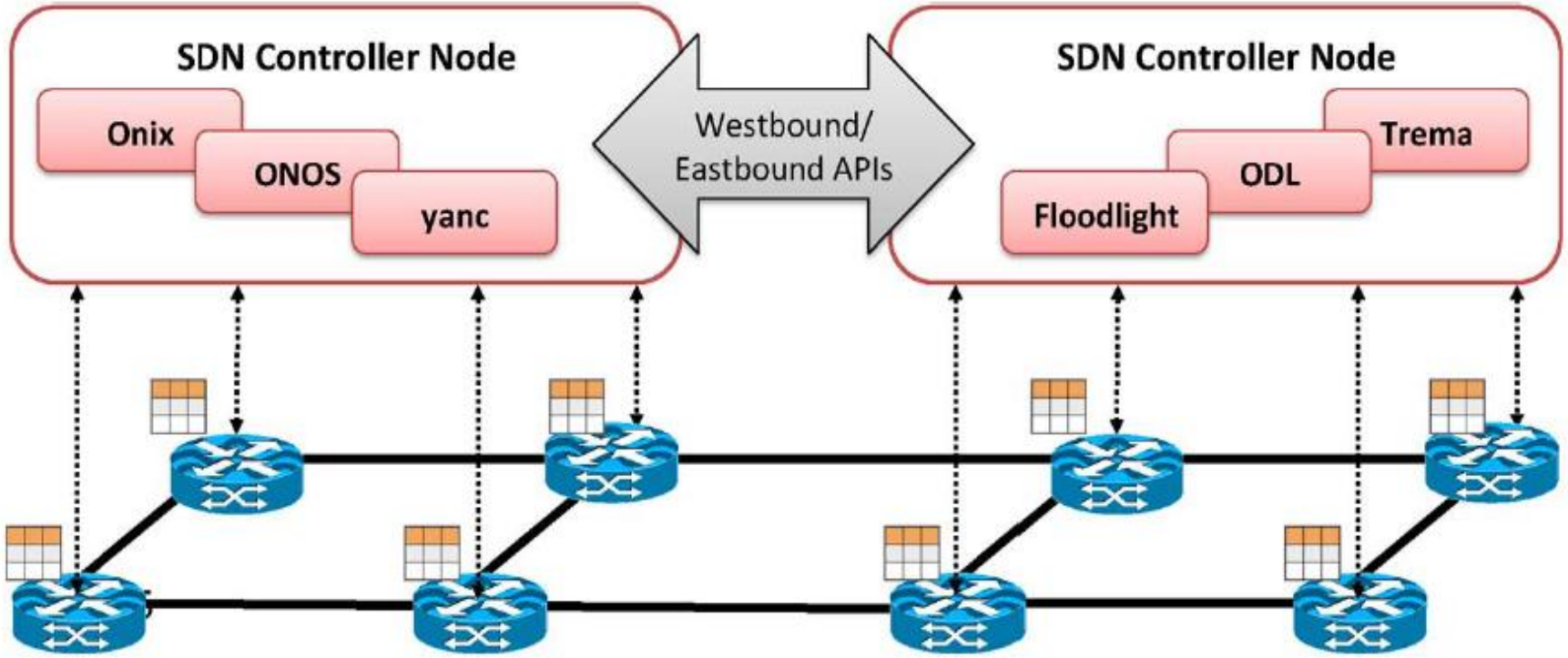



**Fig. 9.** ***Distributed controllers: east/westbound APIs.***