

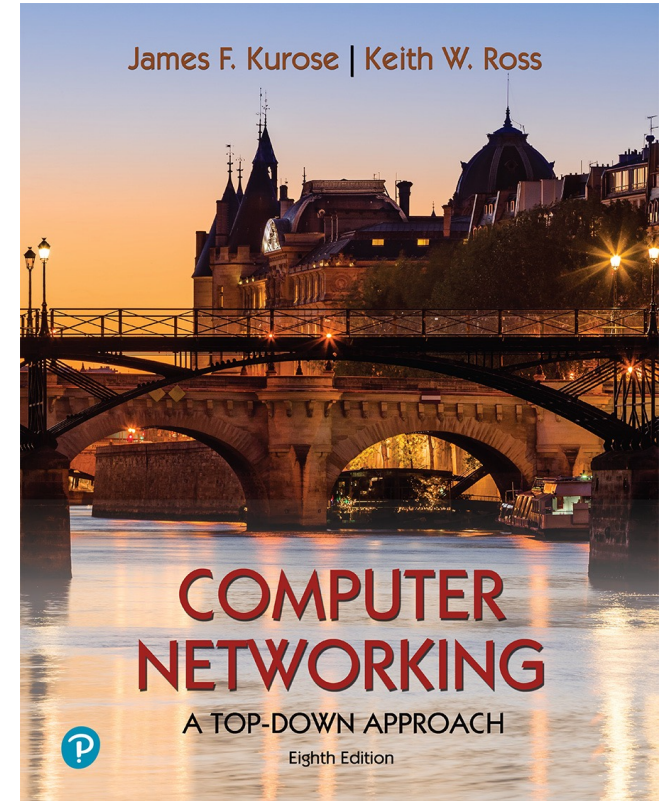
Chapter 4

Network Layer: Data Plane

Yaxiong Xie

Department of Computer Science and Engineering
University at Buffalo, SUNY

Adapted from the slides of the book's authors



*Computer Networking: A
Top-Down Approach*

8th edition

Jim Kurose, Keith Ross
Pearson, 2020

Network layer: our goals

- understand principles behind network layer services, focusing on data plane:
 - network layer service models
 - forwarding versus routing
 - how a router works
 - addressing
 - generalized forwarding
 - Internet architecture
- instantiation, implementation in the Internet
 - IP protocol
 - NAT, middleboxes

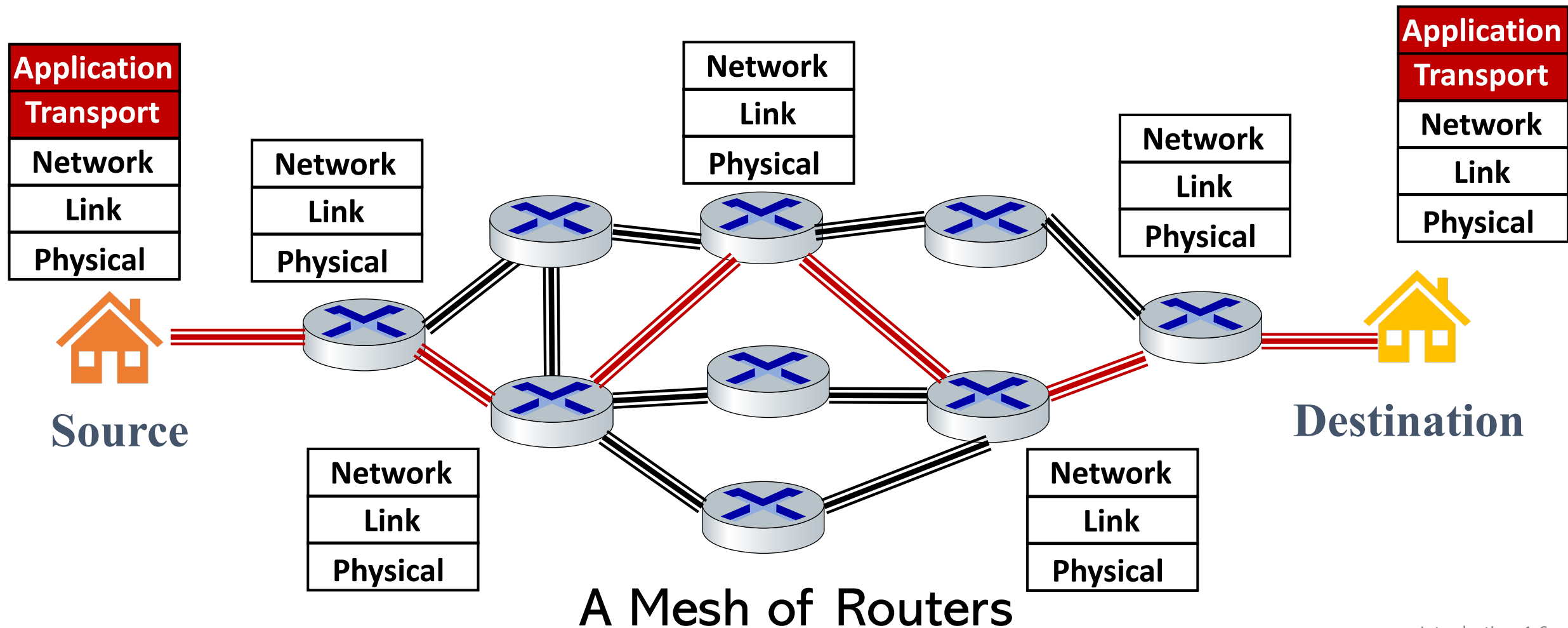
Network layer: “data plane” roadmap

- Network layer: overview
 - data plane
 - control plane
- What’s inside a router
 - input ports, switching, output ports
 - buffer management, scheduling
- IP: the Internet Protocol
 - datagram format
 - addressing
 - network address translation
 - IPv6
- Generalized Forwarding, SDN
 - Match+action
 - OpenFlow: match+action in action
- Middleboxes

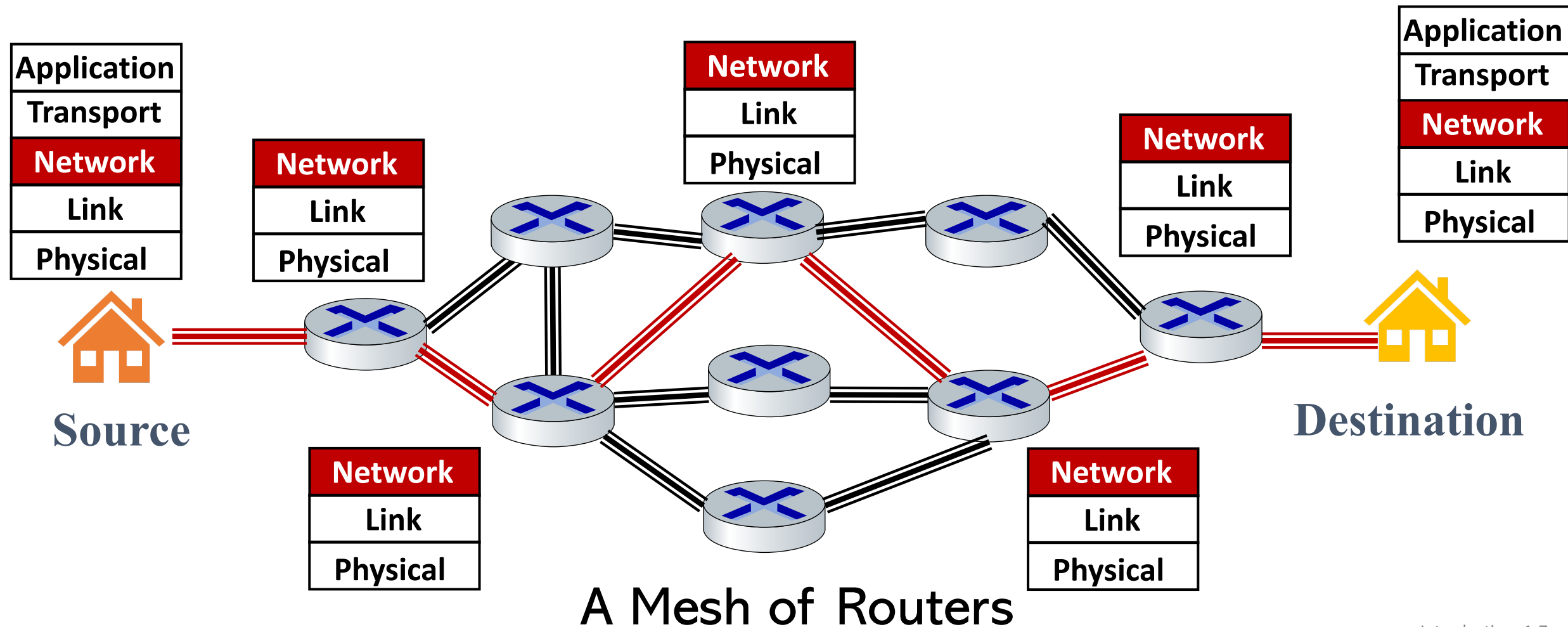


Where is network layer inside a Network?

Application and transport layer is end-to-end

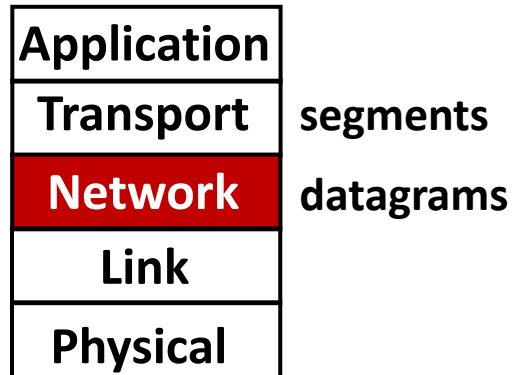


Network-layer is in every network device



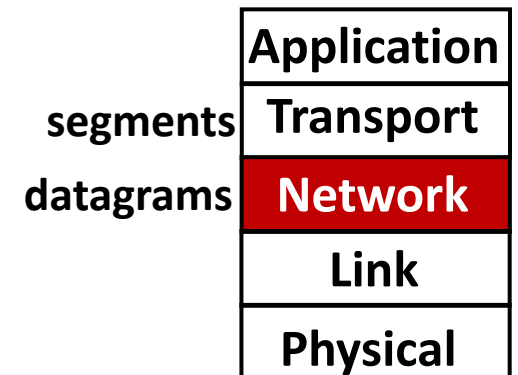
What does a network layer do?

Network-layer services and protocols



Source

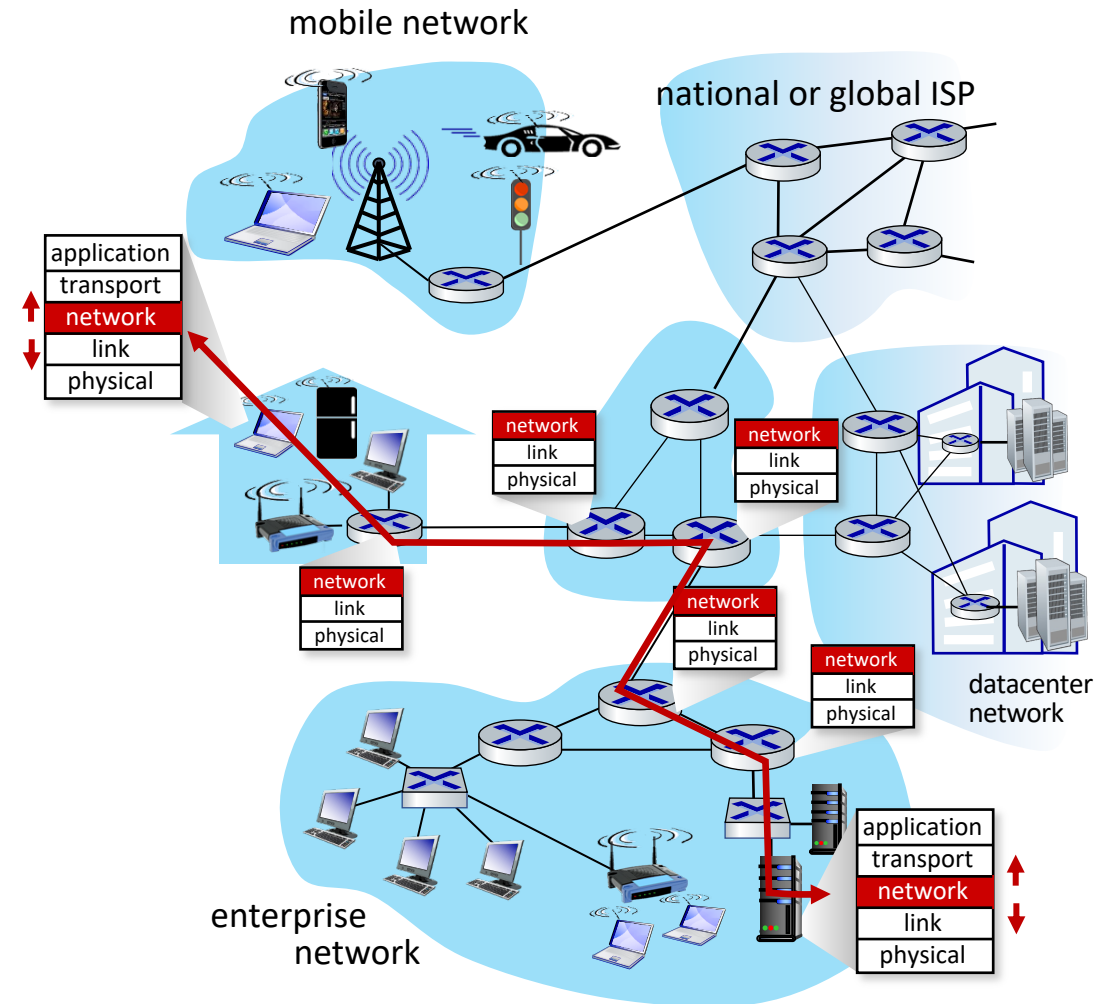
- transport segment from sending to receiving host
 - **sender**: encapsulates segments into datagrams, passes to link layer
 - **receiver**: delivers segments to transport layer protocol



Destination

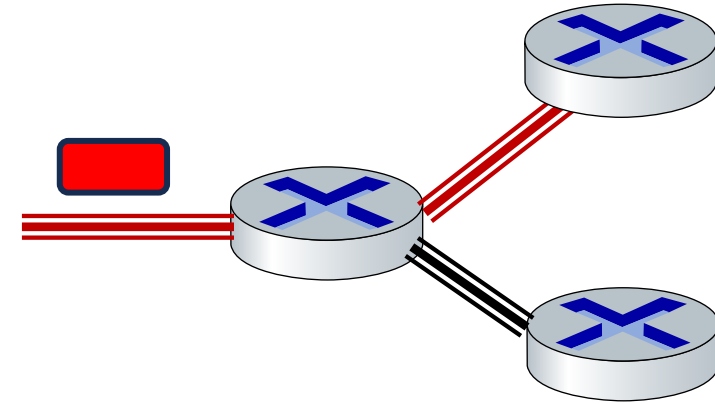
Network-layer services and protocols

- transport segment from sending to receiving host
 - **sender:** encapsulates segments into datagrams, passes to link layer
 - **receiver:** delivers segments to transport layer protocol
- network layer protocols in *every Internet device*: hosts, routers
- **routers:**
 - examines header fields in all IP datagrams passing through it
 - moves datagrams from input ports to output ports to transfer datagrams along end-end path



Network-layer services and protocols

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

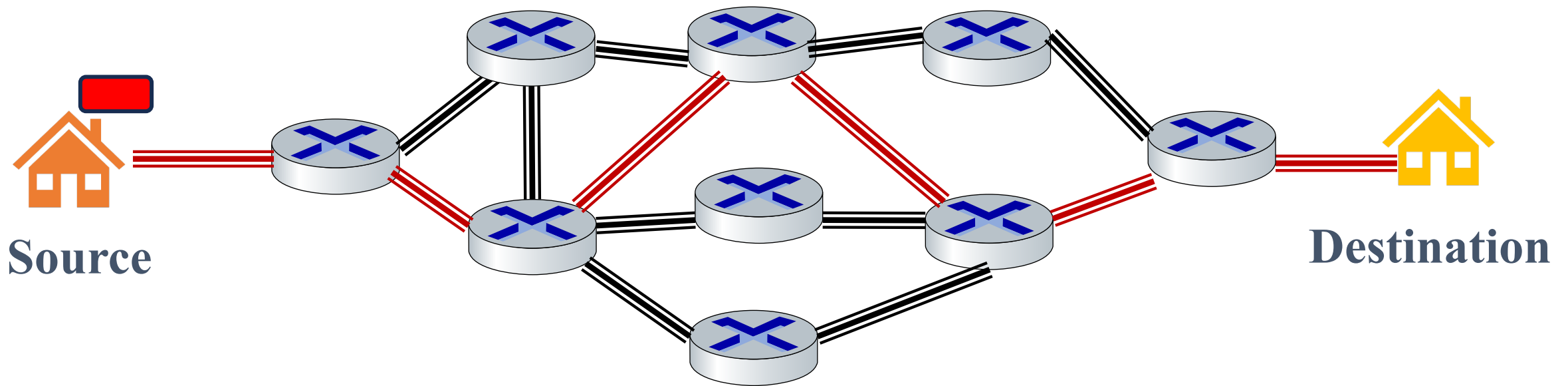


Network-layer services and protocols

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

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

- *routing algorithms*



A Mesh of Routers

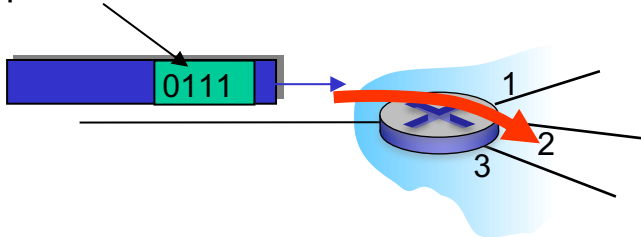
Data Plane VS. Control Plane

Network layer: data plane, control plane

Data plane:

- *local*, per-router function
- determines how datagram arriving on router input port is forwarded to router output port

values in arriving packet header



Broadcom Tomahawk 6

Cisco Silicon One G300

Vendor-published forwarding /
switching capacity

102.4 Tbps

102.4 Tbps

1.024×10^{14} bits/s

152.4 Billion Packets per Second

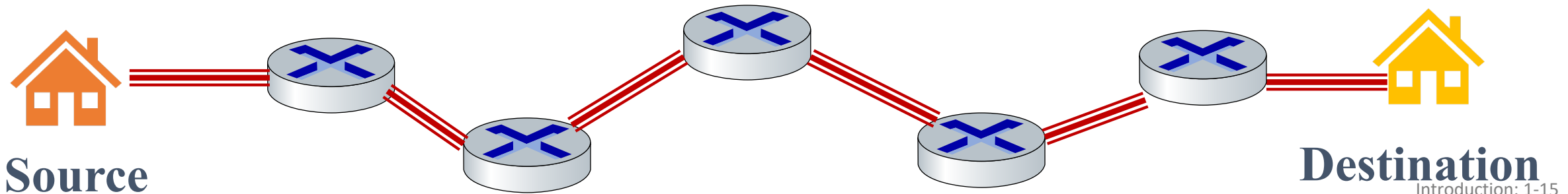
Network layer: data plane, control plane

Data plane:

- *local*, per-router function
- determines how datagram arriving on router input port is forwarded to router output port

Control plane

- *network-wide* logic
- determines how datagram is routed among routers along end-end path from source host to destination host



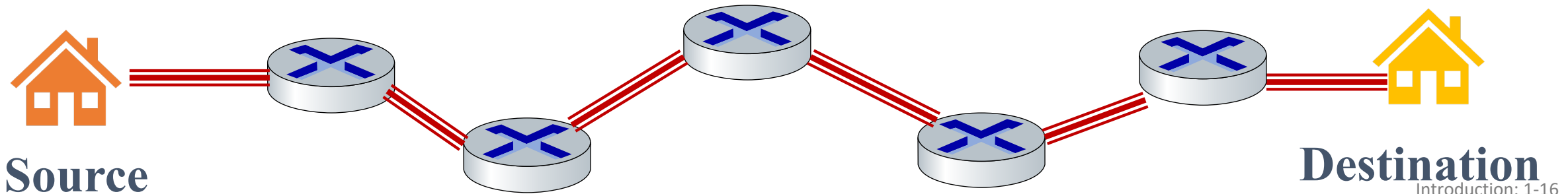
Network layer: data plane, control plane

Data plane:

- *local*, per-router function
- determines how datagram arriving on router input port is forwarded to router output port

Control plane

- *network-wide* logic
- two control-plane approaches:
 - *traditional routing algorithms*: implemented in routers
 - *software-defined networking (SDN)*: implemented in (remote) servers



Control Plane is full of challenges

Scale of the Network

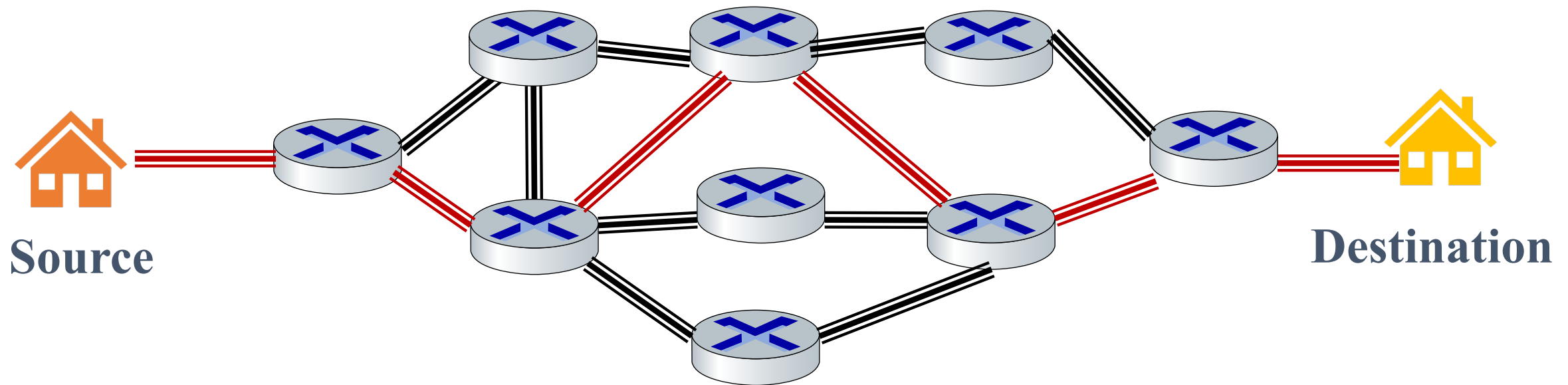
- Billions of devices connected globally
- Path for any arbitrary source and destinations

Dynamics of the Network

- New device/networks joins anytime anywhere
- Devices fails, crashes, gets removed frequently

Policy and Conflicting Objectives

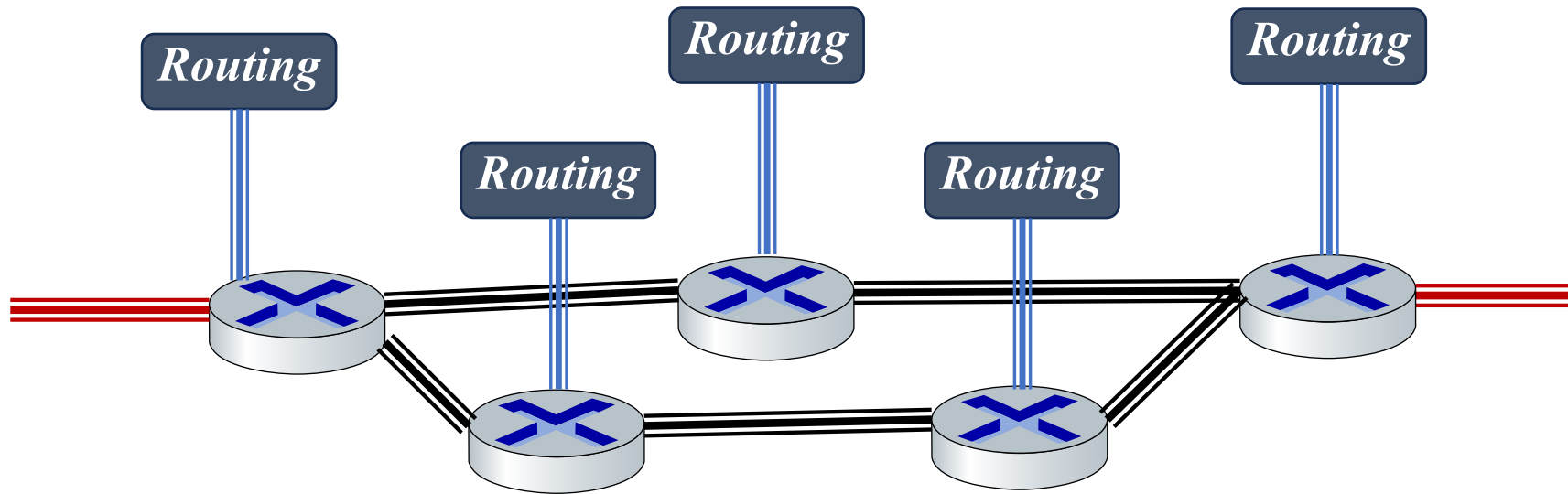
- Networks of networks
- Different country and organizations having different policy and regulations



Where should we implement
the control plane?

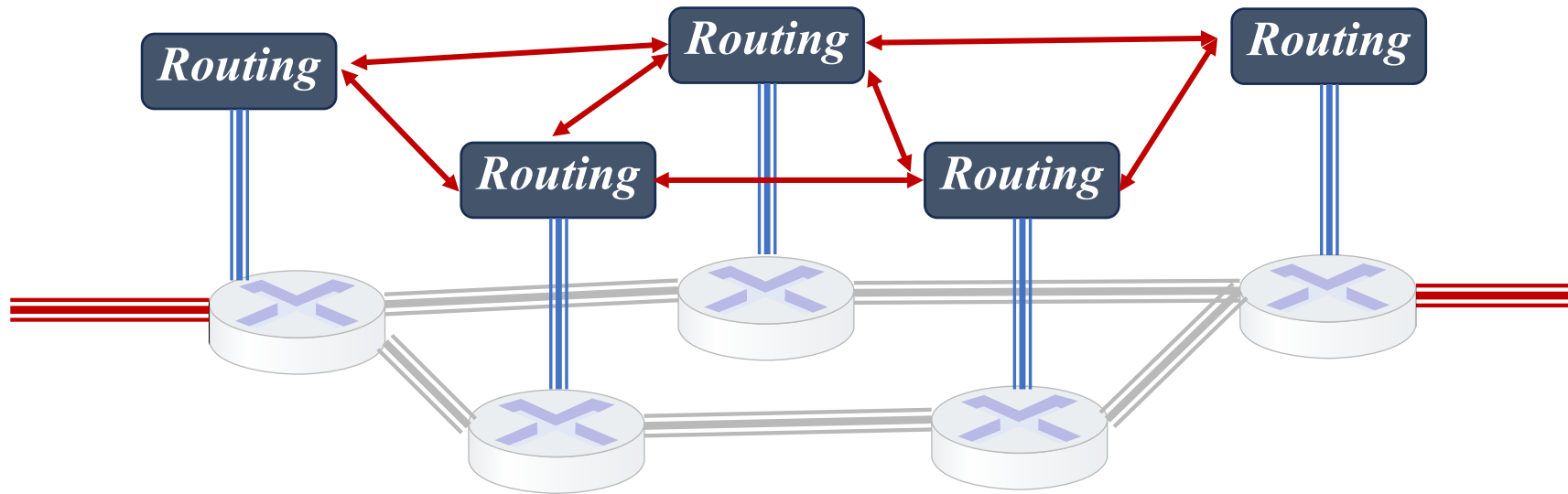
Control plane: Distributed Control Plane

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



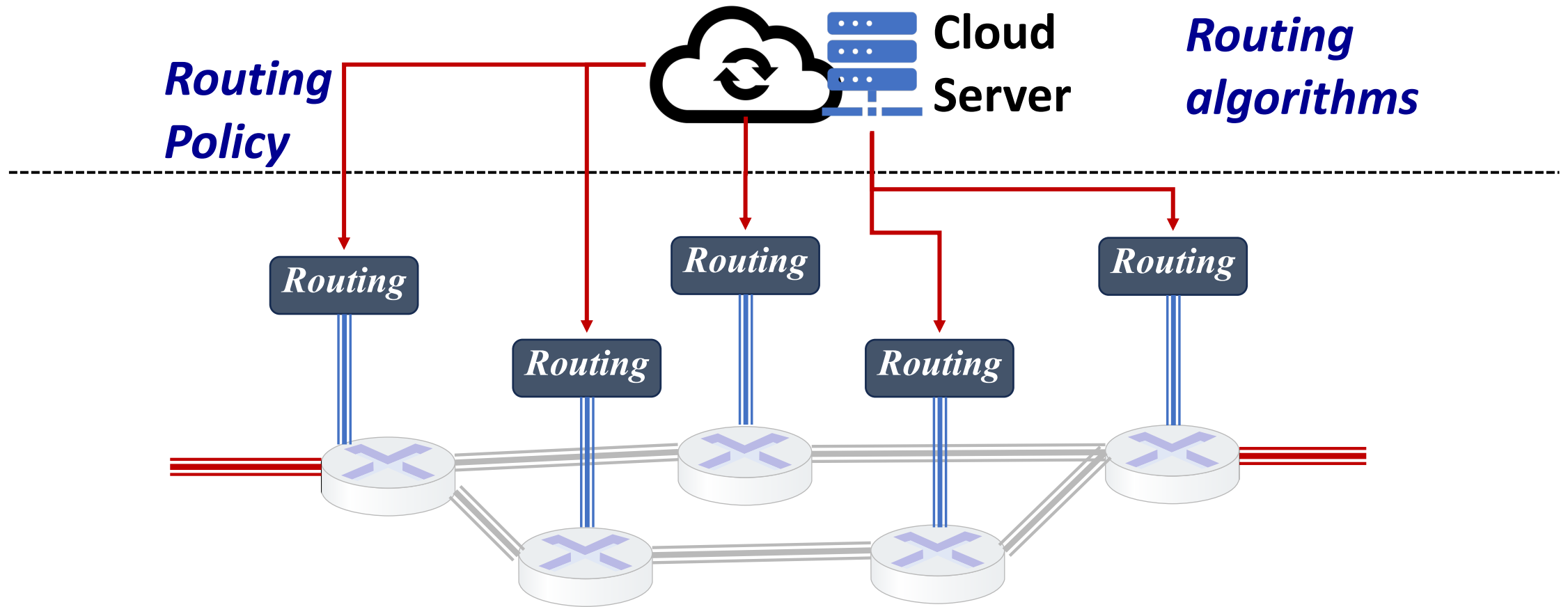
Control plane: Distributed Control Plane

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



Control plane: Centralized Control Plane

Remote controller computes, installs forwarding tables in routers



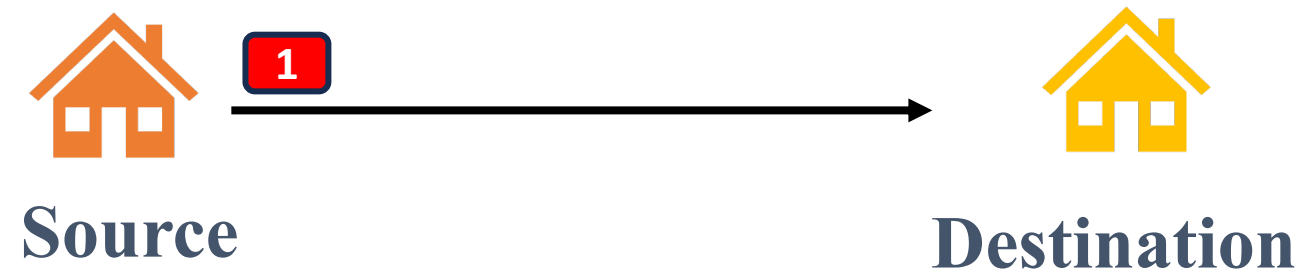
Software-Defined Networking (SDN)

Network service model

Q: What *service model* for “channel” transporting datagrams from sender to receiver?

example services for
individual datagrams:

- guaranteed delivery
- guaranteed delivery with less than 40 msec delay

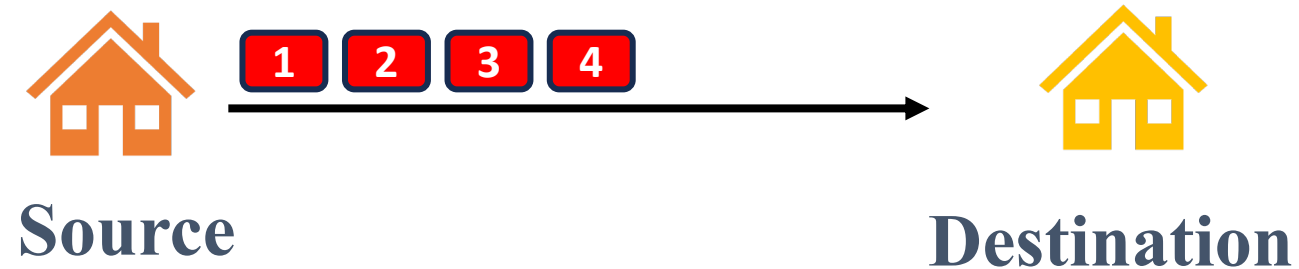


Network service model

Q: What *service model* for “channel” transporting datagrams from sender to receiver?

example services for a *flow* of datagrams:

- in-order datagram delivery
- guaranteed minimum bandwidth to flow
- restrictions on changes in inter-packet spacing



Network-layer service model

Network Architecture	Service Model	Quality of Service (QoS) Guarantees ?			
		Bandwidth	Loss	Order	Timing
Internet	best effort	none	no	no	no

Internet “best effort” service model

No guarantees on:

- i. successful datagram delivery to destination
- ii. timing or order of delivery
- iii. bandwidth available to end-end flow

Reflections on best-effort service:

- **simplicity of mechanism** has allowed Internet to be widely deployed adopted
- sufficient **provisioning of bandwidth** allows performance of real-time applications (e.g., interactive voice, video) to be “good enough” for “most of the time”
- **replicated, application-layer distributed services** (datacenters, content distribution networks) connecting close to clients’ networks, allow services to be provided from multiple locations
- congestion control of “elastic” services helps

It's hard to argue with success of best-effort service model

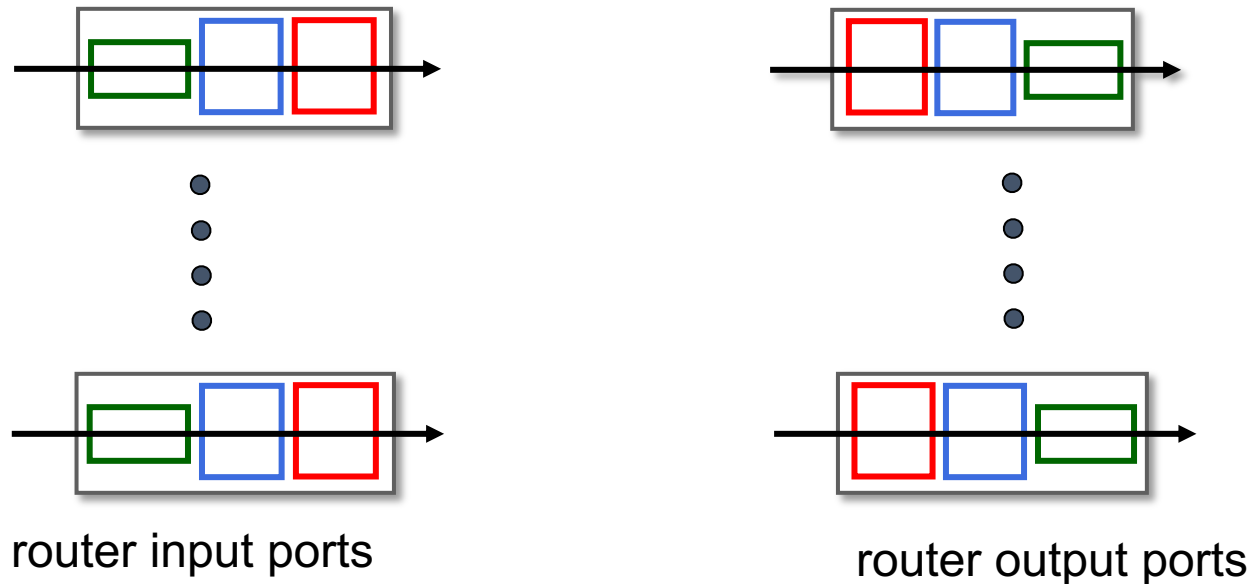
Network layer: “data plane” roadmap

- Network layer: overview
 - data plane
 - control plane
- What’s inside a router
 - input ports, switching, output ports
 - buffer management, scheduling
- IP: the Internet Protocol
 - datagram format
 - addressing
 - network address translation
 - IPv6
- Generalized Forwarding, SDN
 - Match+action
 - OpenFlow: match+action in action
- Middleboxes



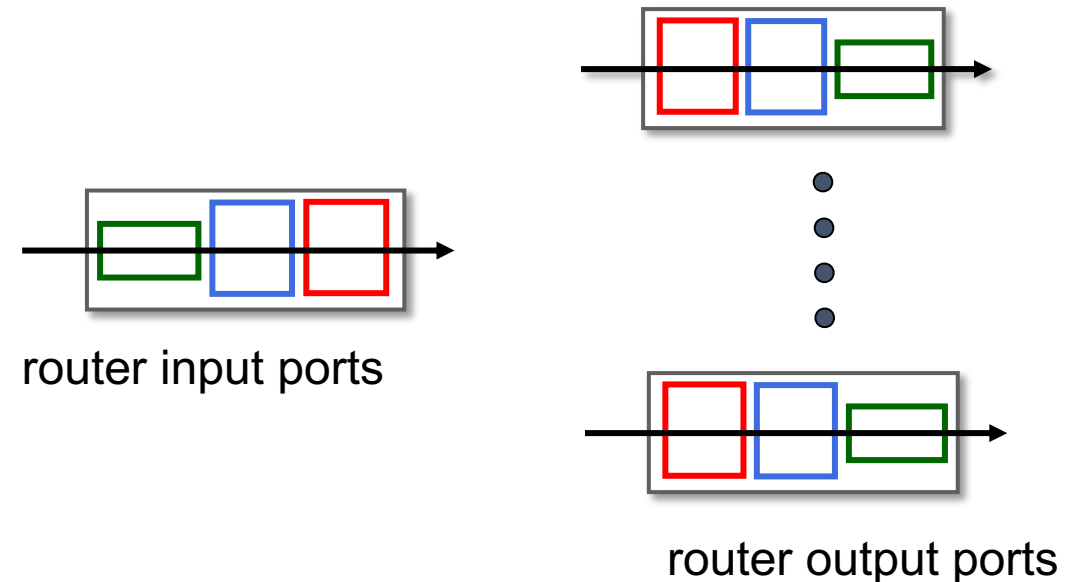
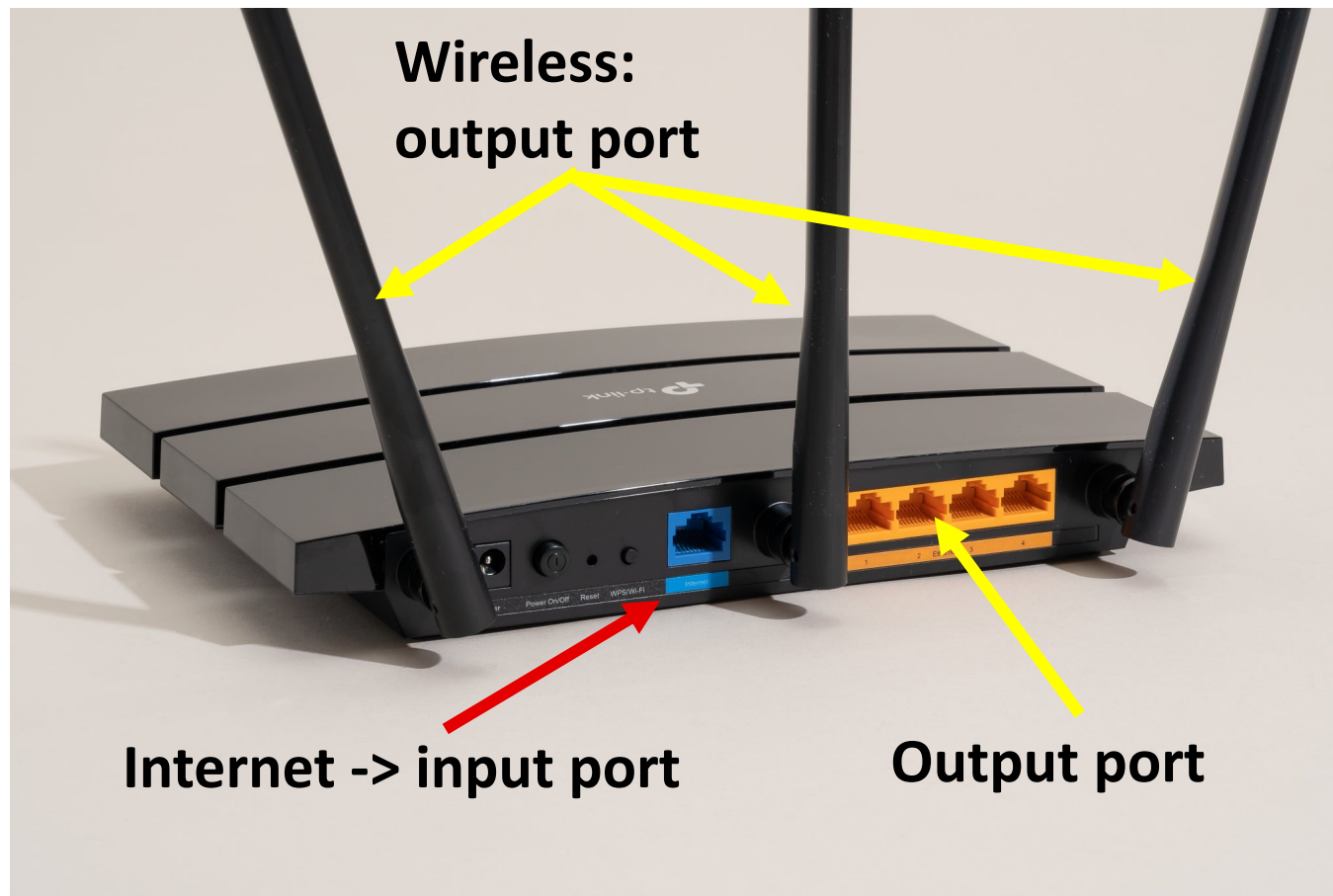
Router architecture overview

high-level view of generic router architecture:



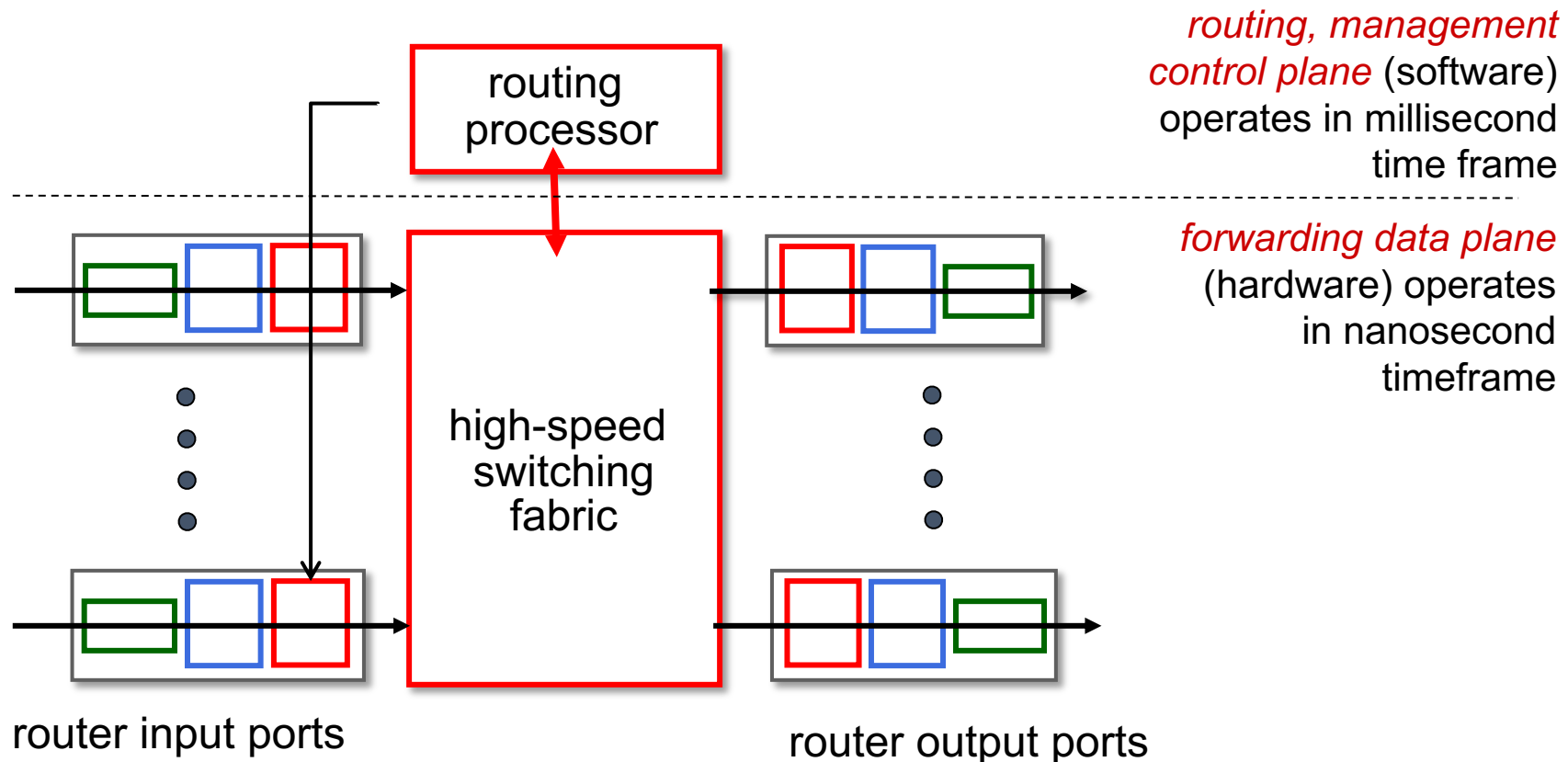
Router architecture overview

high-level view of generic router architecture:



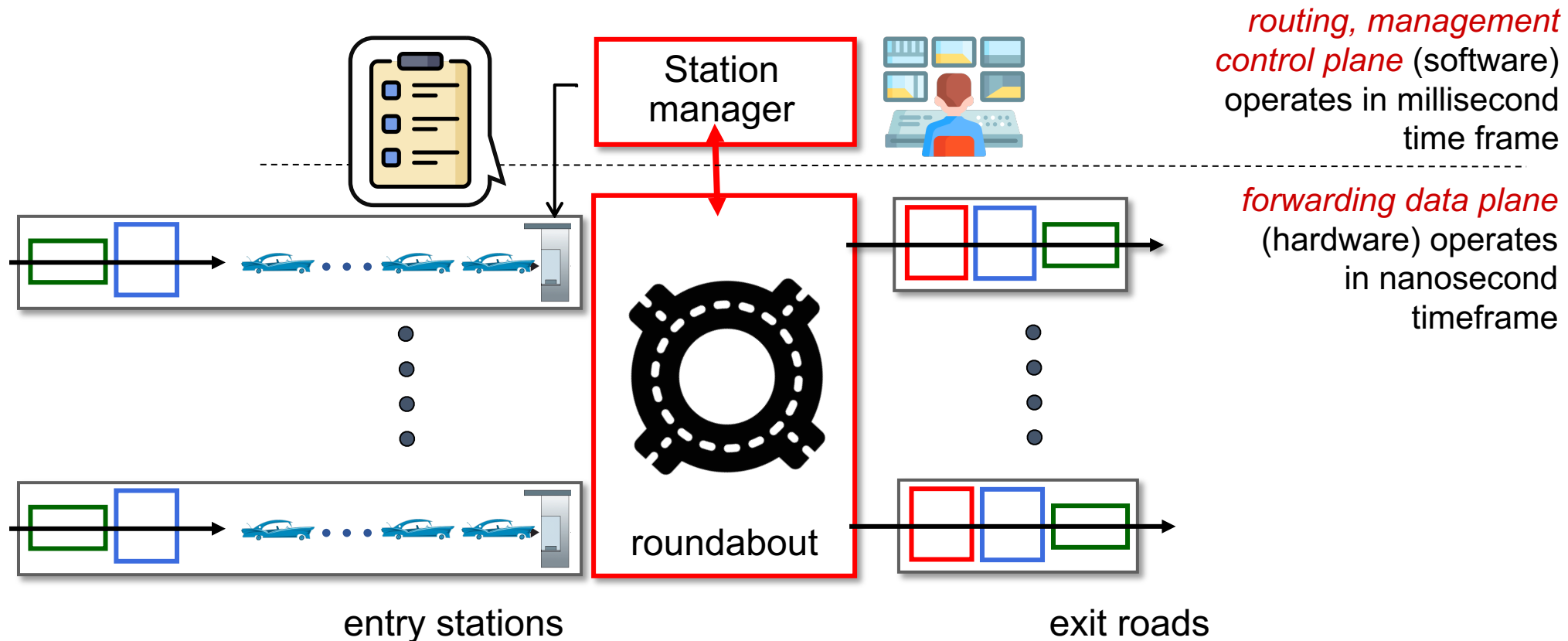
Router architecture overview

high-level view of generic router architecture:

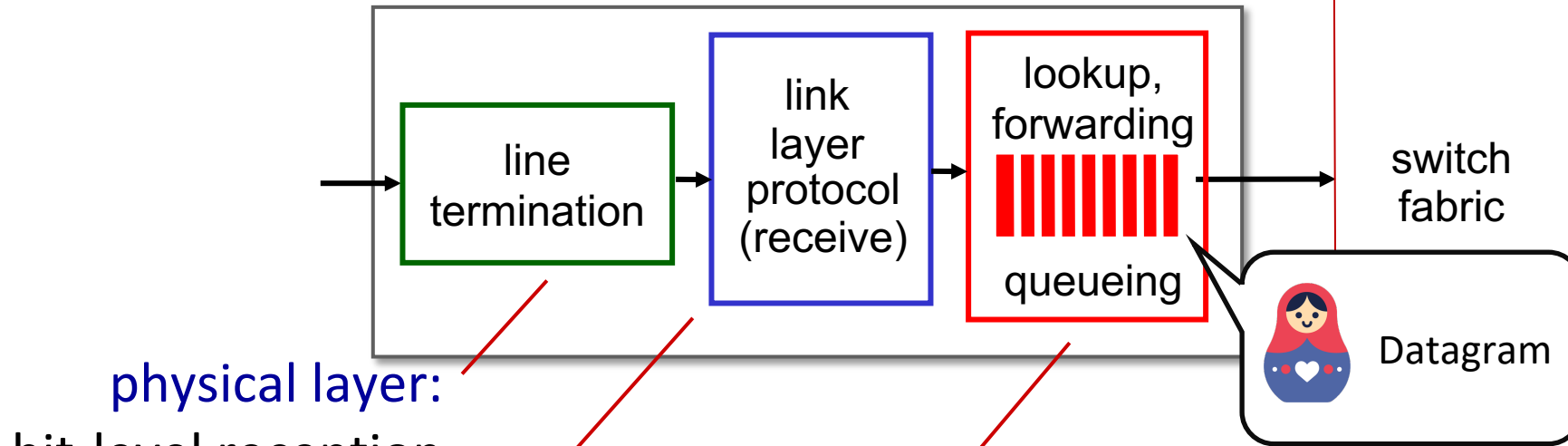


Router architecture overview

analogy view of generic router architecture:



Input port functions



physical layer:
bit-level reception

link layer:
e.g., Ethernet
(chapter 6)

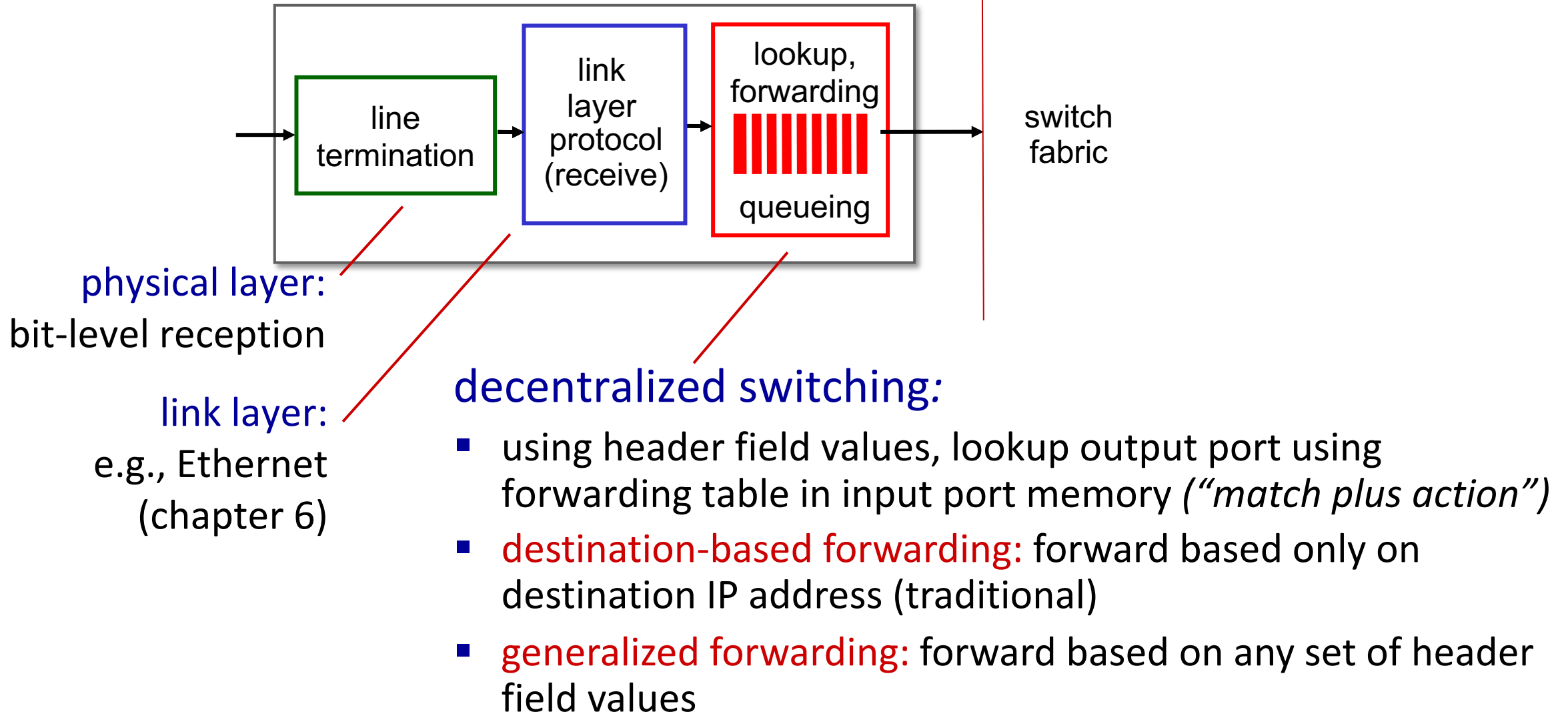


Frame

decentralized switching:

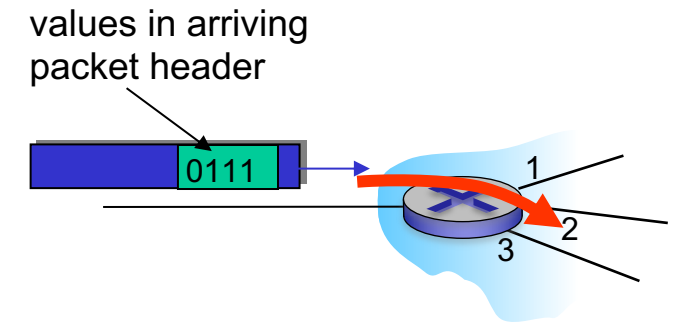
- using header field values, lookup output port using forwarding table in input port memory (*“match plus action”*)
- goal: complete input port processing at ‘line speed’
- **input port queueing**: if datagrams arrive faster than forwarding rate into switch fabric

Input port functions



Forwarding Table

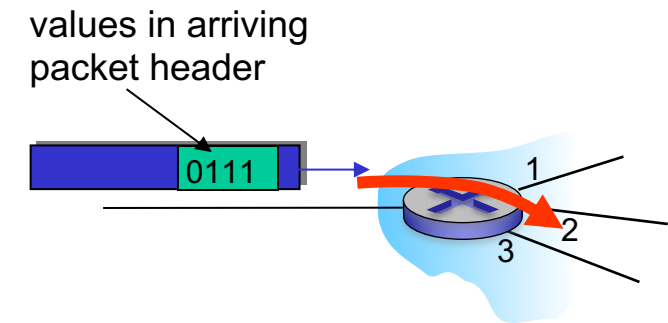
IP Address Range		Forwarding Interface
192.168.0.1	192.168.0.20	1
192.168.0.40	192.168.0.60	2
192.168.0.80	192.168.0.100	3



To speed up this process, Internet uses
longest pre-fix matching algorithm

Forwarding Table

IP Address Range		Forwarding Interface
192.168.0.1	192.168.0.20	1
192.168.0.10	192.168.0.15	3
192.168.0.40	192.168.0.60	2
192.168.0.80	192.168.0.100	3



Longest prefix matching

IP Address Range	Forwarding Interface
192.168.0.1 192.168.0.20	1

IP Address Range	Forwarding Interface
11000000.10101000.00000000.00000001 11000000.10101000.00000000.00010100	1

Longest prefix matching

11000000.10101000.00000000.000*****

11000000.10101000.00000000.00000000

192.168.0.1

11000000.10101000.00000000.00011111

192.168.0.31

11000000.10101000.00000000.0000****

11000000.10101000.00000000.00000000

192.168.0.1

11000000.10101000.00000000.00001111

192.168.0.15

Longest prefix matching

longest prefix match

when looking for forwarding table entry for given destination address, use *longest* address prefix that matches destination address.

Destination Address Range	Link interface
11001000 00010111 00010*** *****	0
11001000 00010111 00011000 *****	1
11001000 00010111 00011*** *****	2
otherwise	3

examples:

11001000 00010111 00010110 10100001 which interface?

11001000 00010111 00011000 10101010 which interface?

Longest prefix matching

longest prefix match

when looking for forwarding table entry for given destination address, use *longest* address prefix that matches destination address.

Destination Address Range	Link interface
11001000 00010111 00010*** *****	0
11001000 00010111 00011000 *****	1
11001000 match! 1 00011*** *****	2
otherwise	3

examples:

11001000 00010111 00010110 10100001 which interface?
11001000 00010111 00011000 10101010 which interface?

Longest prefix matching

longest prefix match

when looking for forwarding table entry for given destination address, use *longest* address prefix that matches destination address.

Destination Address Range	Link interface
11001000 00010111 00010*** *****	0
11001000 00010111 00011000 *****	1
11001000 00010111 00011*** *****	2
otherwise	3

match!

examples:

11001000 00010111 00010110 10100001	which interface?
11001000 00010111 00011000 10101010	which interface?

Longest prefix matching

longest prefix match

when looking for forwarding table entry for given destination address, use *longest* address prefix that matches destination address.

Destination Address Range	Link interface
11001000 00010111 00010*** *****	0
11001000 00010111 00011000 *****	1
11001000 00010111 00011*** *****	2
otherwise	3

match!

examples:

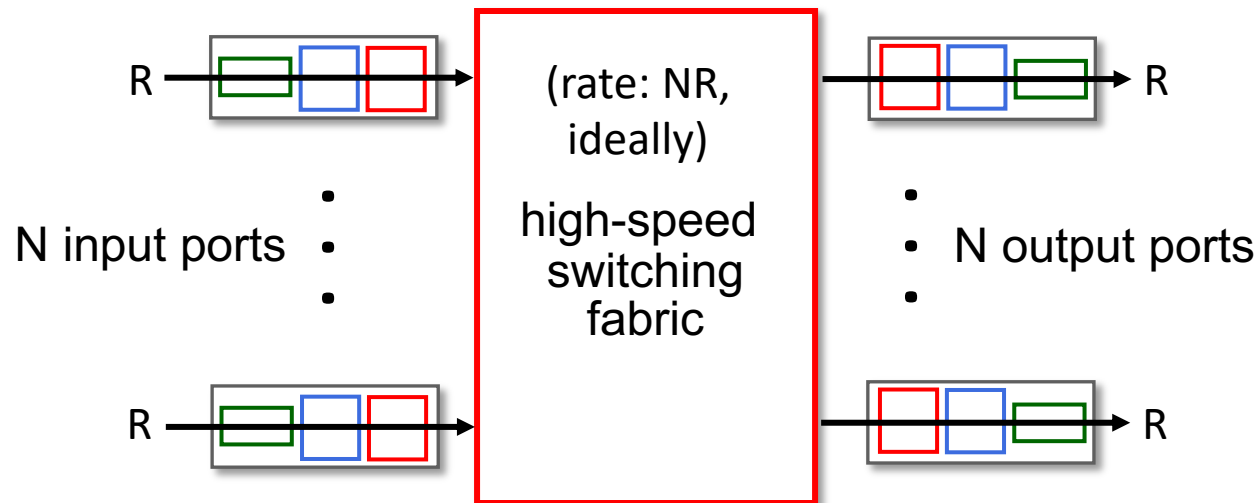
11001000 00010111 00010110 10100001	which interface?
11001000 00010111 00011000 10101010	which interface?

Longest prefix matching

- we'll see *why* longest prefix matching is used shortly, when we study addressing
- longest prefix matching: often performed using ternary content addressable memories (TCAMs)
 - *content addressable*: present address to TCAM: retrieve address in one clock cycle, regardless of table size
 - Cisco Catalyst: ~1M routing table entries in TCAM

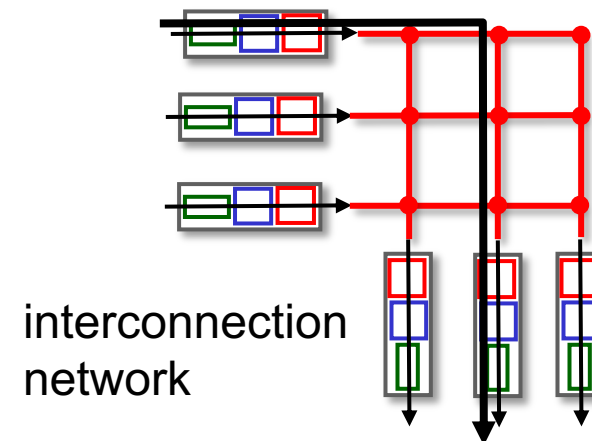
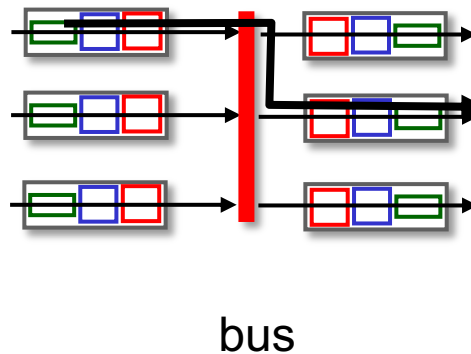
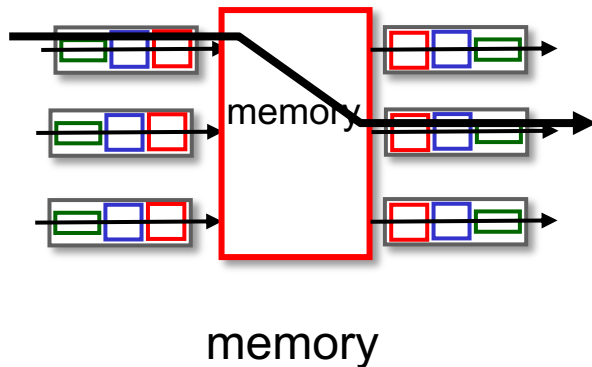
Switching fabrics

- transfer packet from input link to appropriate output link
- **switching rate**: rate at which packets can be transfer from inputs to outputs
 - often measured as multiple of input/output line rate
 - N inputs: switching rate N times line rate desirable



Switching fabrics

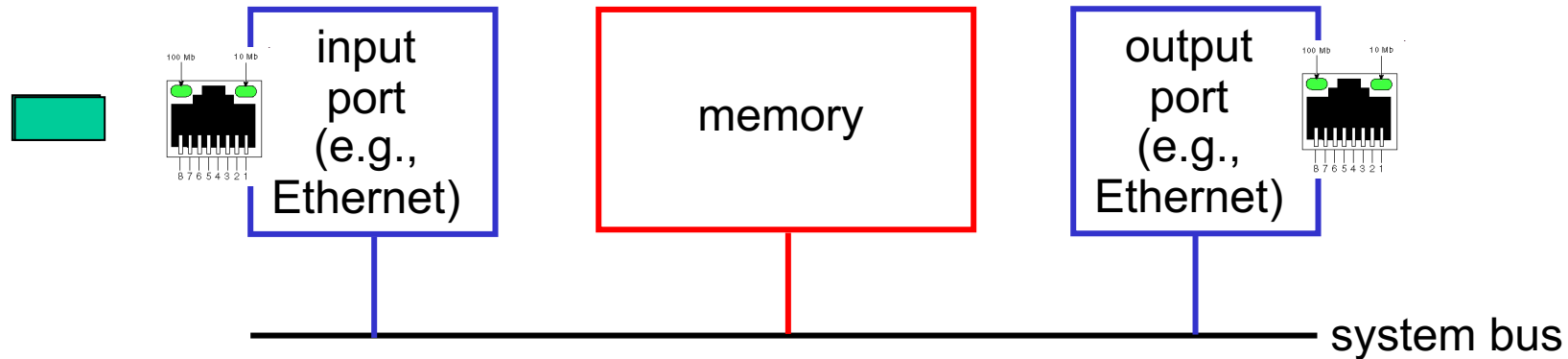
- transfer packet from input link to appropriate output link
- **switching rate**: rate at which packets can be transfer from inputs to outputs
 - often measured as multiple of input/output line rate
 - N inputs: switching rate N times line rate desirable
- three major types of switching fabrics:



Switching via memory

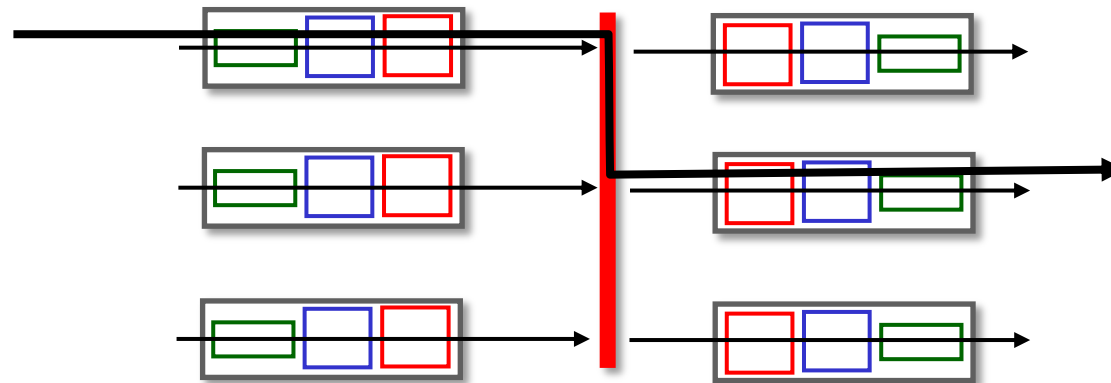
first generation routers:

- traditional computers with switching under direct control of CPU
- packet copied to system's memory
- speed limited by memory bandwidth (2 bus crossings per datagram)



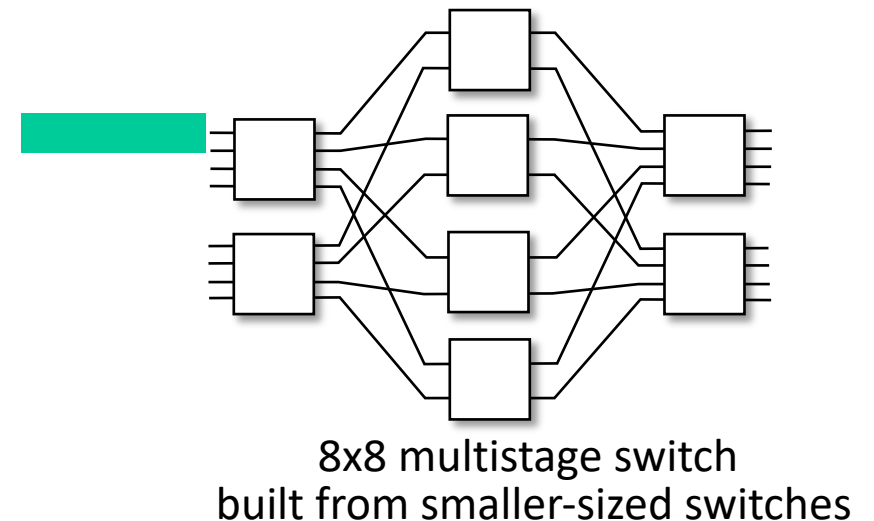
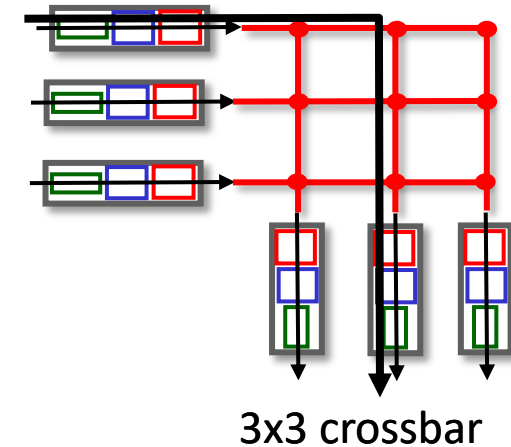
Switching via a bus

- datagram from input port memory to output port memory via a shared bus
- *bus contention*: switching speed limited by bus bandwidth
- 32 Gbps bus, Cisco 5600: sufficient speed for access routers



Switching via interconnection network

- Crossbar, Clos networks, other interconnection nets initially developed to connect processors in multiprocessor
- **multistage switch**: $n \times n$ switch from multiple stages of smaller switches
- **exploiting parallelism**:
 - fragment datagram into fixed length cells on entry
 - switch cells through the fabric, reassemble datagram at exit



Switching via interconnection network

- scaling, using multiple switching “planes” in parallel:
 - speedup, scaleup via parallelism
- Cisco CRS router:
 - basic unit: 8 switching planes
 - each plane: 3-stage interconnection network
 - up to 100's Tbps switching capacity

