CS 330: Network Applications & Protocols

Network Layer

Galin Zhelezov
Department of Physical Sciences
York College of Pennsylvania

Overview of Network Layer

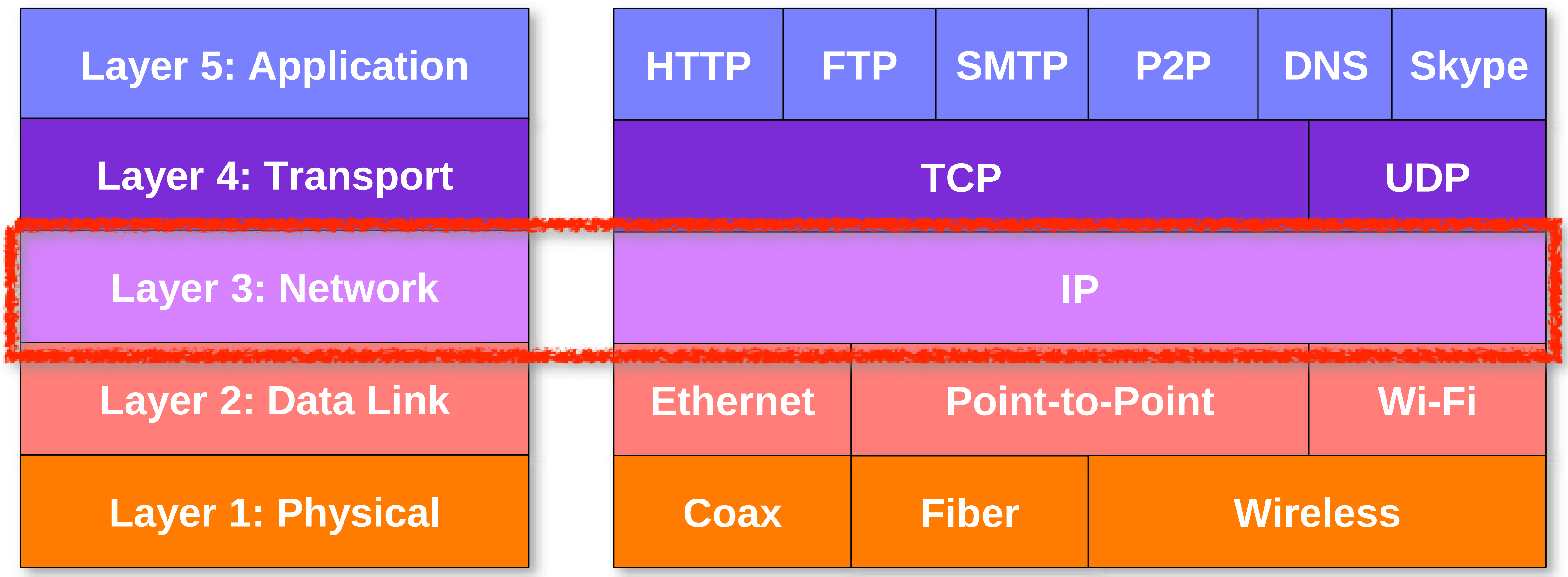
- **Introduction**
- **Virtual Circuit and Datagram Networks***
- **Router Architectures**
- **IP: Internet Protocol**
- **Routing algorithms**
- **Routing in the Internet**
- **Broadcast and multicast routing**

Overview of Network Layer

- **Introduction**
- **Virtual Circuit and Datagram Networks***
- **Router Architectures**
- **IP: Internet Protocol**
- **Routing algorithms**
- **Routing in the Internet**
- **Broadcast and multicast routing**

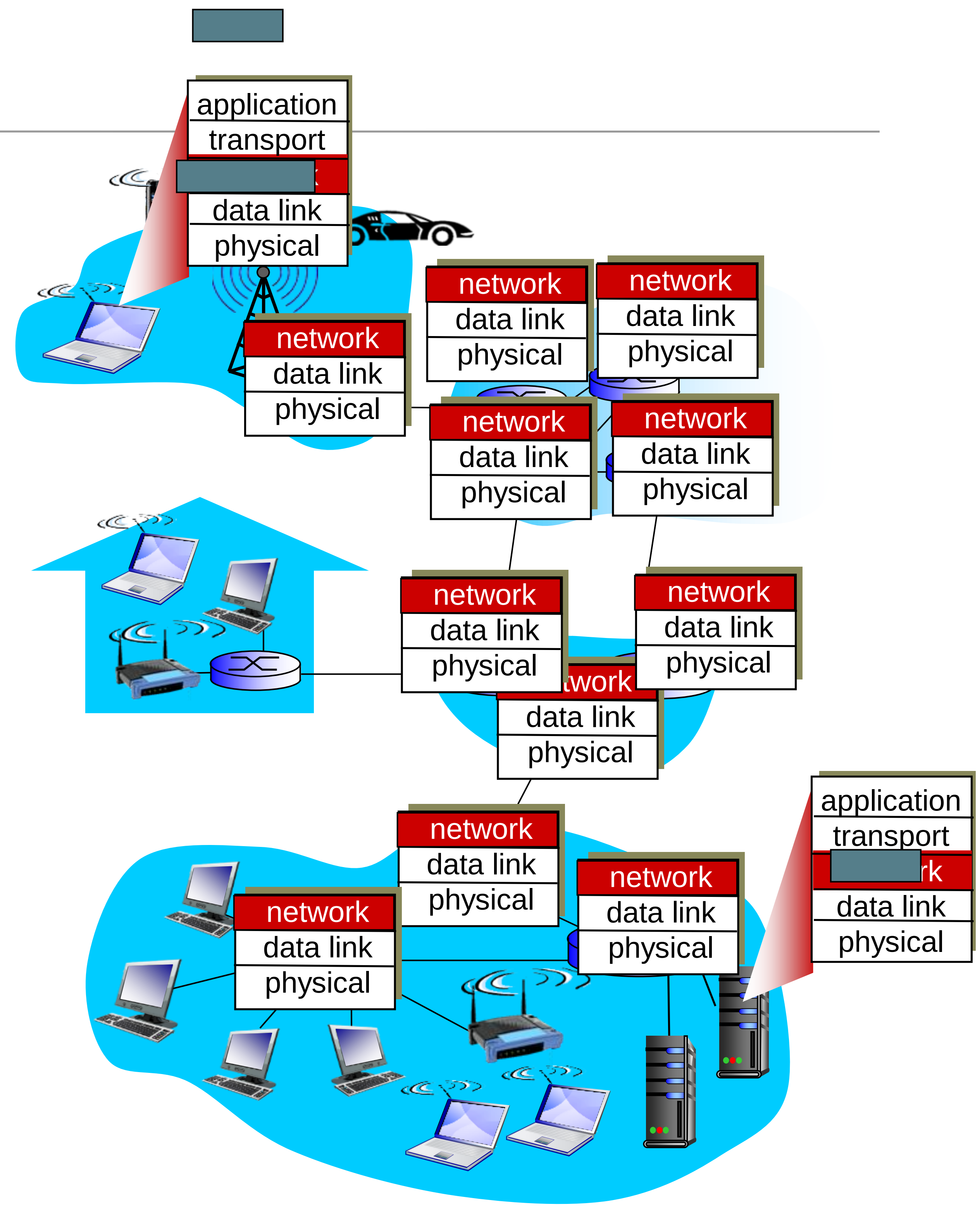
Protocol Layers

- **Top-Down Approach**

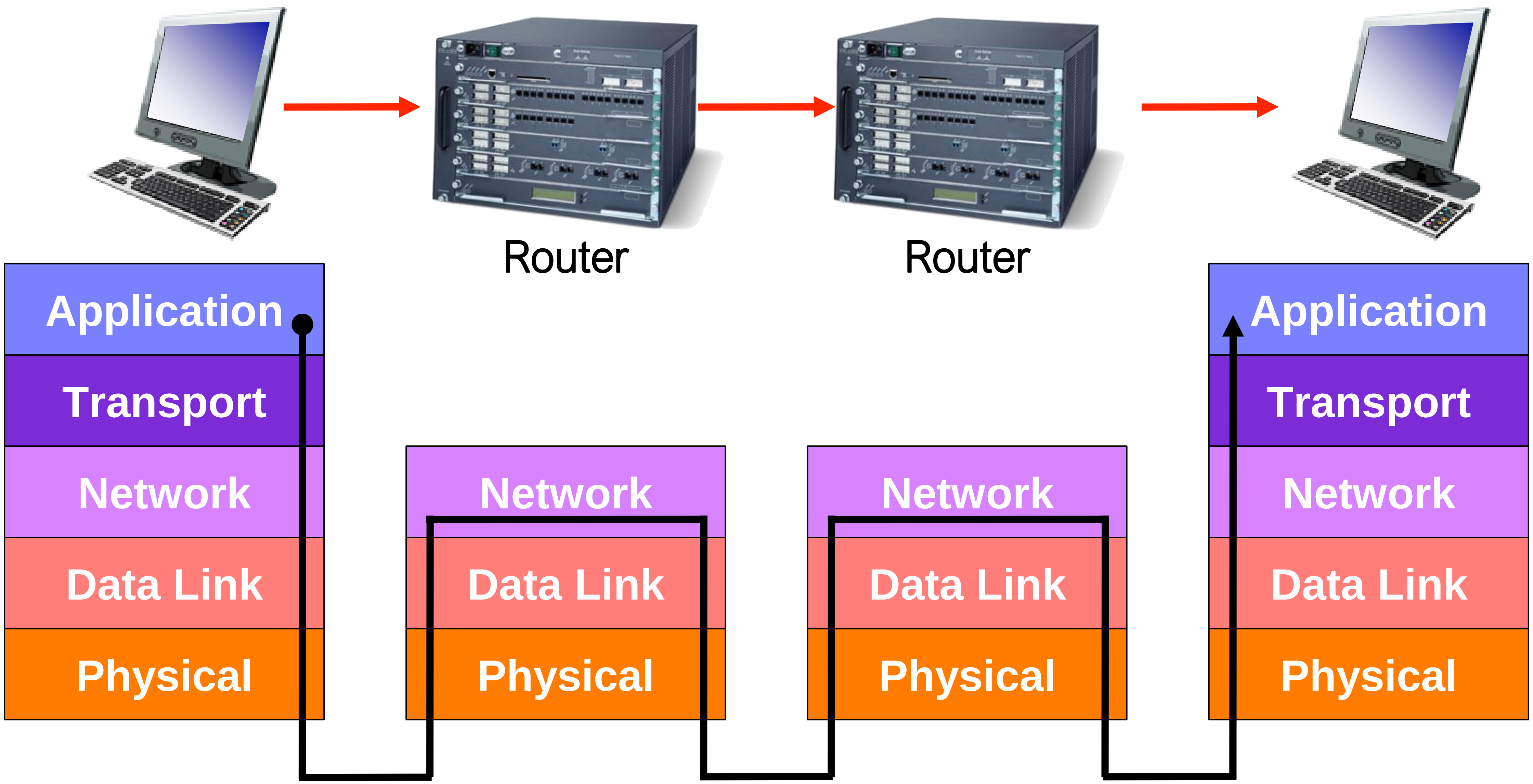


Network layer

- **transport segment from sending to receiving host**
- **on sending side encapsulates segments into datagrams**
- **on receiving side, delivers segments to transport layer**
- **network layer protocols in *every* host, router**
- **router examines header fields in all IP datagrams passing through it**



Network Layer



- **Network layer provides host to host communication service**
  - Network layer protocols in *every* host, router
  - A router is used to examine the header fields of all IP datagrams passing through it

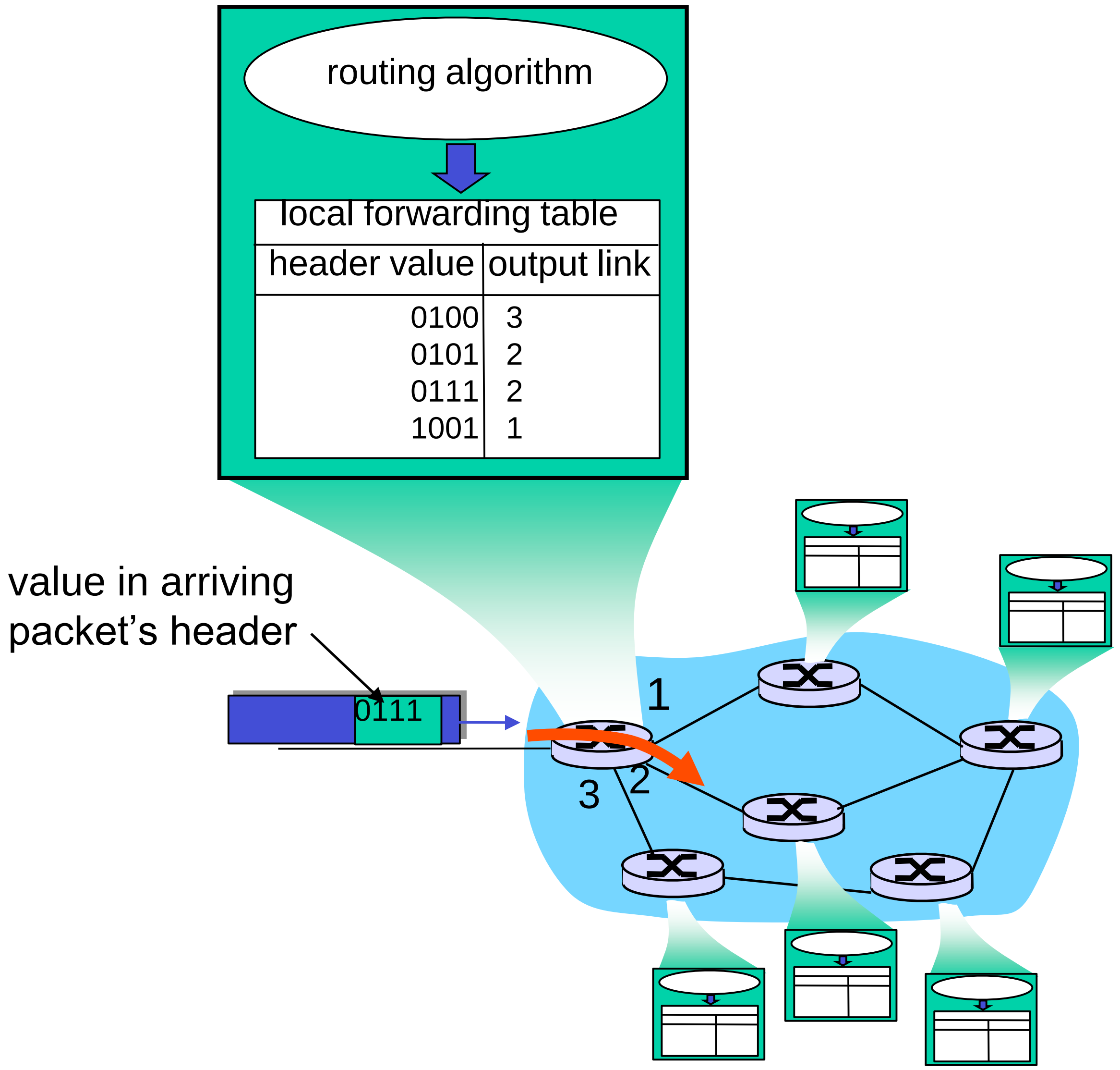
Two Key Network-Layer Functions

- **Forwarding - move packets from a single router's input to the appropriate router output**
  - Happens within a single router
  - Routers contain a forwarding table to determine output

- **Routing - determine the *complete route* taken by packets from the source host to the destination host**
  - Complete route may include multiple routers
  - Requires routing algorithms and knowledge of other routers on the network
    - Routing algorithms are used to update a router's forwarding tables

Interplay Between Routing and Forwarding

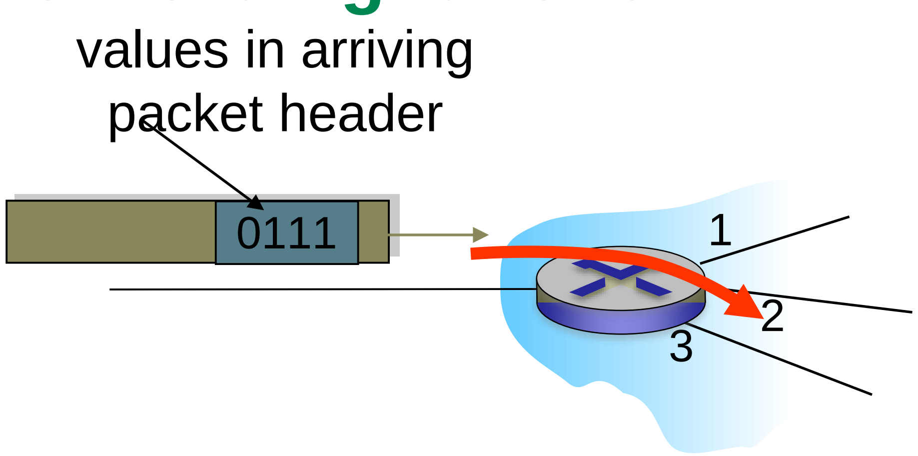
- **Routing algorithm determines end-end-path through network**
- **Forwarding table determines local forwarding at this router**
- **Example:**
  - Packet arrives with a header value of `0111`, it is output on port 2 of the router



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**
- **forwarding function**

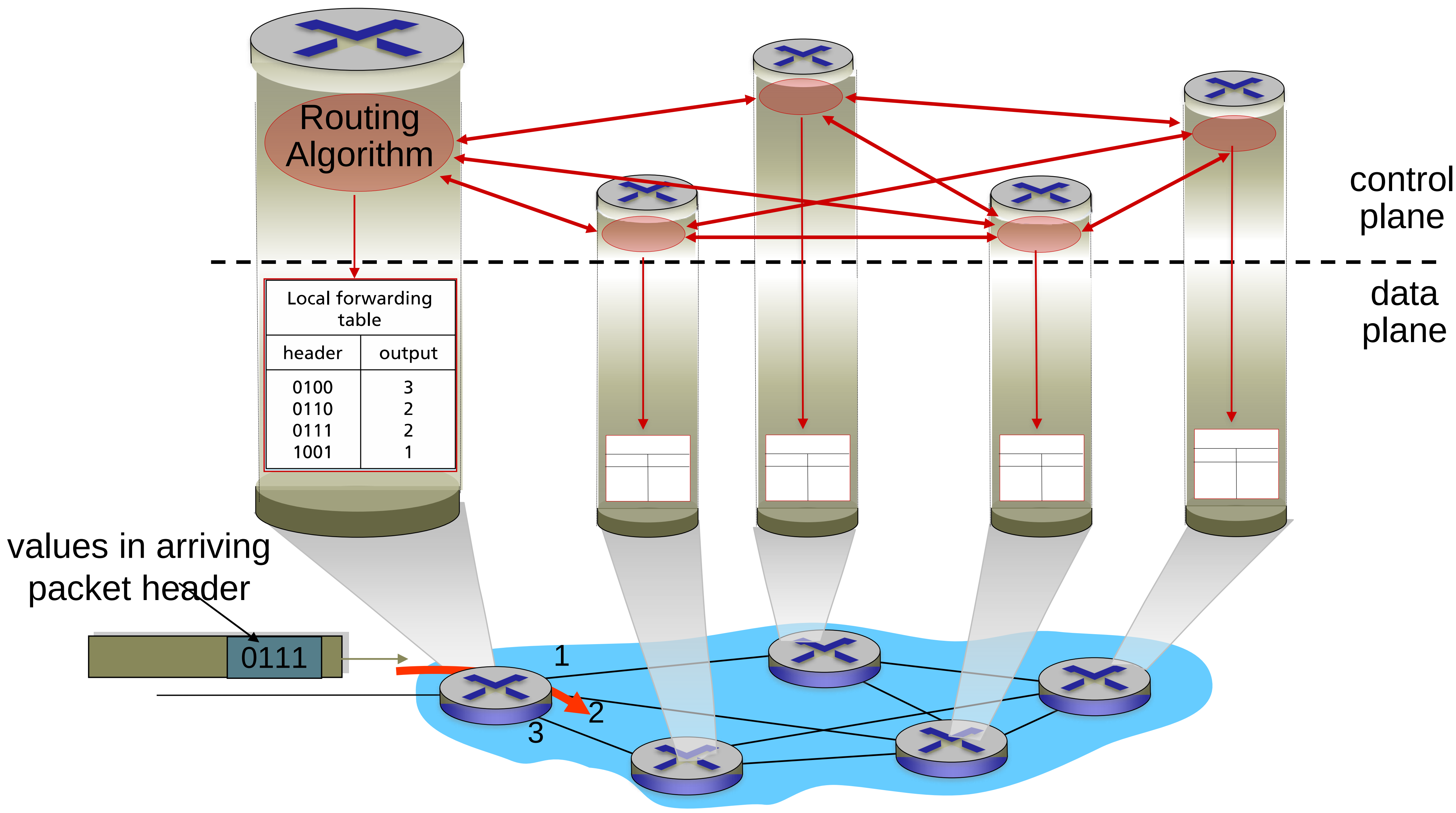


Control plane

- network-wide logic
- determines how datagram is routed among routers along end-end path from source host to destination host
- two control-plane approaches:
  - *traditional routing algorithms:* implemented in routers
  - *software-defined networking (SDN)*: implemented in (remote) servers

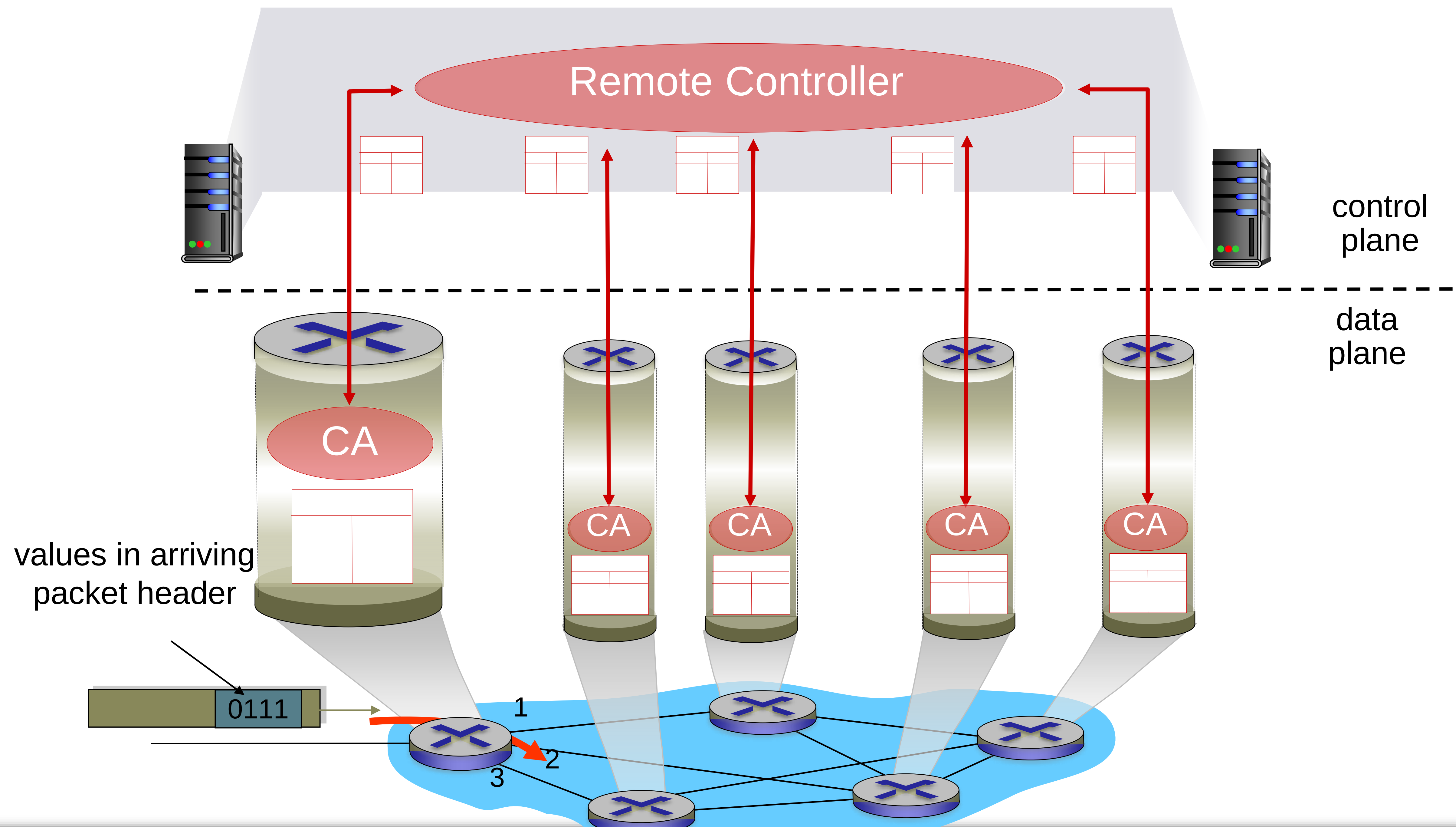
Per-router control plane

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



Logically centralized control plane

A distinct (typically remote) controller interacts with local control agents (CAs)



A Third Network-Layer Function

- **Connection Setup**
  - 3rd important function in some network architectures:
    - ATM, frame relay, X.25
  - Before datagrams flow, two end hosts and intervening routers establish virtual connection
  - Network vs. Transport layer connection service:
    - Network - connection is between two hosts (may also involve intervening routers in case of Virtual Circuits)
    - Transport - connection is between two processes

Overview of Network Layer

- **Introduction**
- **Virtual Circuit and Datagram Networks***
- **Router Architectures**
- **IP: Internet Protocol**
- **Routing algorithms**
- **Routing in the Internet**
- **Broadcast and multicast routing**

Connection-oriented or Connectionless Service

- **Network Layer can provide either connection-oriented or connectionless service (just like in the Transport Layer, TCP/UDP)**
  - Virtual Circuit Networks provide Network Layer connection-oriented service
  - Datagram Networks provide Network Layer connectionless service
- **Unlike the Transport Layer**
  - Services are offered as a host-to-host services
  - The network can provide *either* connection-oriented *or* connectionless service, but not both
  - Implemented in the core of the network (i.e. the routers)

Virtual Circuit Networks

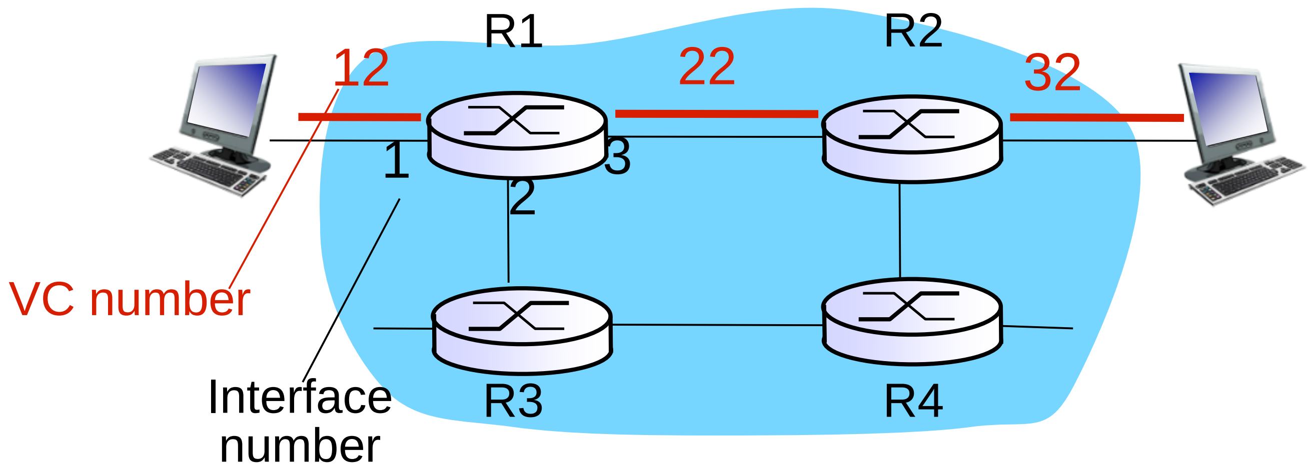
- **Provide Network Layer connection-oriented service**
- **Requires that each connection be *setup* before data can flow**
- **Each packet carries a Virtual Circuit identifier (not destination host address)**
- **Every router on source-to-destination path maintains "state" for each connection that passes through it**
- **Router resources (e.g. bandwidth, buffers) may be allocated to a Virtual Circuit during setup**
  - Dedicated resources = Predictable service

Virtual Circuit Implementation

- **A Virtual Circuit consists of:**
  - (1) A path between the source and destination hosts
    - This path is made of a series of links and routers
  - (2) Virtual Circuit numbers, one number for each link along the path
  - (3) Entries in the forwarding tables of the routers along the path
- **A packet belonging to virtual circuit carries a virtual circuit number (rather than dest address)**
  - VC number can be changed on each link
  - At each router along the path, a new VC number comes from the forwarding table

Virtual Circuit Forwarding Table

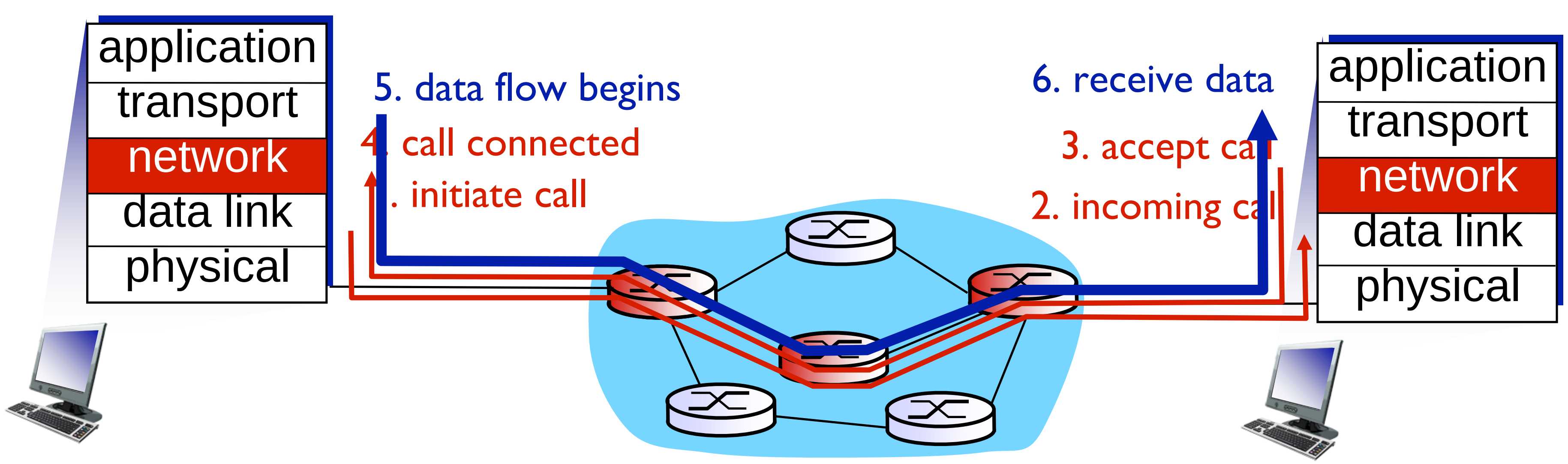
- **Each VC router maintains connection state information**
  - When a connection is setup, entries are added to the VC forwarding table
  - When a connection is torn down, entries are removed from the table



Incoming Interface	Incoming VC #	Outgoing Interface	Outgoing VC #
1	12	3	22
2	63	1	18
3	7	2	17
1	97	3	87
...	...	...	...

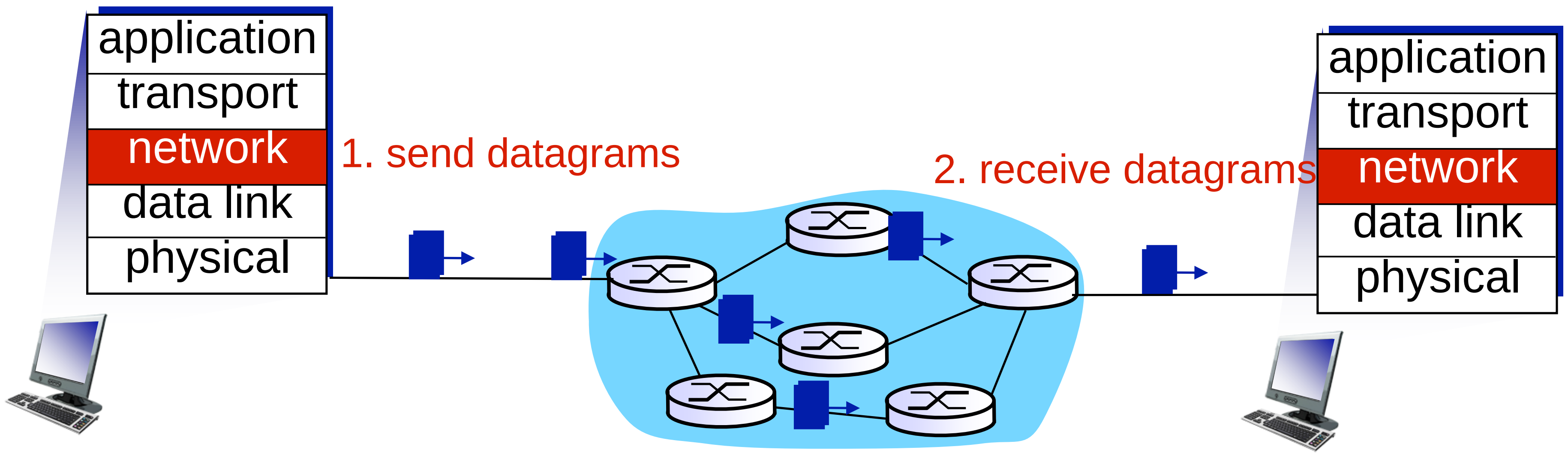
Virtual Circuits Signaling Protocols

- **Protocols used to setup, maintain, and teardown VCs**
  - All protocol message pass through all routers on the way to the destination
  - As messages pass through routers, routers setup the VC



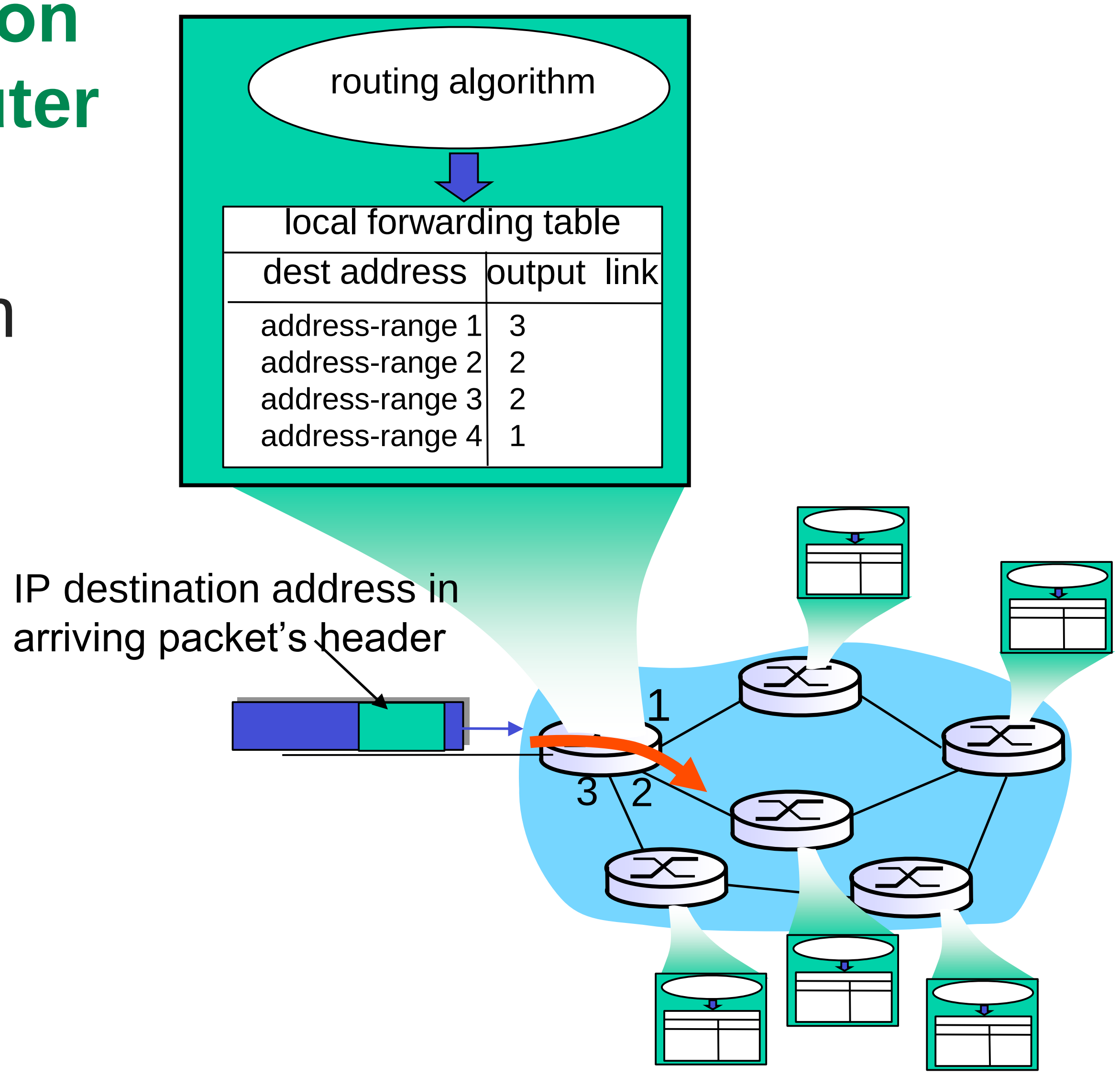
Datagram Networks

- **Requires no call setup at network layer**
- **Routers contain no state about end-to-end connections**
- **No network-level concept of a "connection"**
- **Packets are forwarded using destination host address**
  - To send a packet, a host puts a destination address on the packet and simply inserts it into the network



Datagram Forwarding Table

- **Each router contains a forwarding table that maps a packet's destination address to an output link on the router**
  - 4 billion possible IP addresses so rather than list individual destination addresses, list range of addresses (aggregate table entries)
  - Use Longest-Prefix Matching to determine which address-range a packet falls into



Building a Router Forwarding Table

- **Given a router with multiple output links, specify ranges of addresses to determine which addresses are output on which output link**
  - Example below shows a router with four output links
    - Packet with destination `200.23.24.17` would go out on interface 1
    - Packet with destination `122.17.10.1` would go out on interface 3

Destination Address Ranges	Destination Address Ranges (again)	Outgoing Link Interface
11001000 00010111 00010000 00000000 – 11001000 00010111 00010111 11111111	200.23.16.0 – 200.23.23.255	0
11001000 00010111 00011000 00000000 – 11001000 00010111 00011000 11111111	200.23.24.0 – 200.23.24.255	1
11001000 00010111 00011001 00000000 – 11001000 00010111 00011111 11111111	200.23.25.0 – 200.23.31.255	2
otherwise		3

Longest-Prefix Matching

- **An implementation of ranged matching**
- **When searching the forwarding table for given destination address, use the longest address prefix that matches the destination address**
  - Example 1: `11001000 00010111 00010110 10100001` → `0`
  - Example 2: `11001000 00010111 00011000 10101010` → `1`
    - Matches multiple table entries, send out on interface that has the *longest* match

Destination Address Ranges	Destination Address Ranges (again)	Outgoing Link Interface
`11001000 00010111 00010*** *********`	200.23.16.0 – 200.23.23.255	0
`11001000 00010111 00011000 *********`	200.23.24.0 – 200.23.24.255	1
`11001000 00010111 00011*** *********`	200.23.24.0 – 200.23.31.255	2
otherwise		3

Datagram Network or Virtual Circuit Network

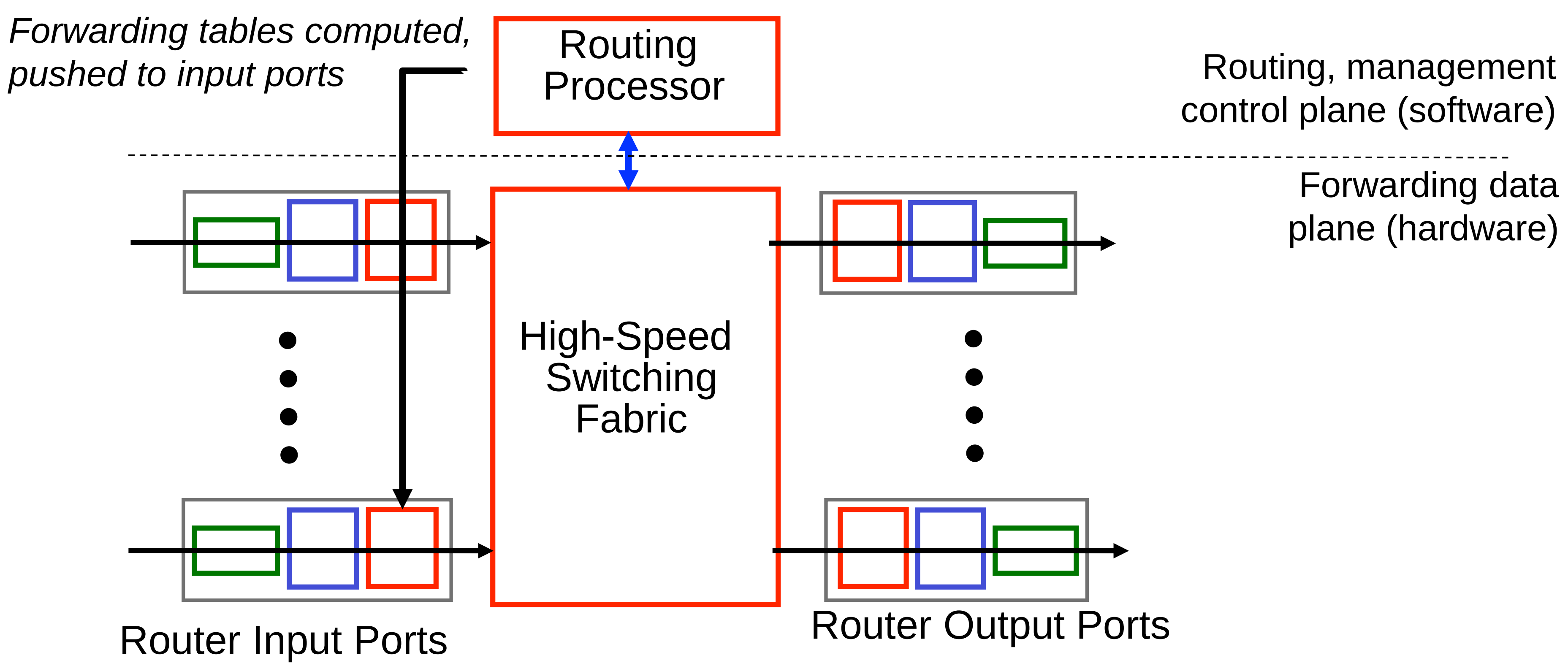
- **Internet (Datagram Network)**
  - Data exchange among computers
    - "Elastic" service, no strict timing requirements
  - Many link types
    - Different characteristics
    - Uniform service difficult
  - "Smart" end systems (computers)
    - Can adapt, perform control, error recovery
    - Simple inside network, complexity at "edge"
- **ATM (Virtual Circuit Network)**
  - Evolved from telephony
  - Like human conversation:
    - Strict timing, reliability requirements
    - Need for guaranteed service
  - "Dumb" end systems
    - Telephones
    - Complexity inside network

Overview of Network Layer

- **Introduction**
- **Virtual Circuit and Datagram Networks***
- **Router Architectures**
- **IP: Internet Protocol**
- **Routing algorithms**
- **Routing in the Internet**
- **Broadcast and multicast routing**

Router Architecture Overview

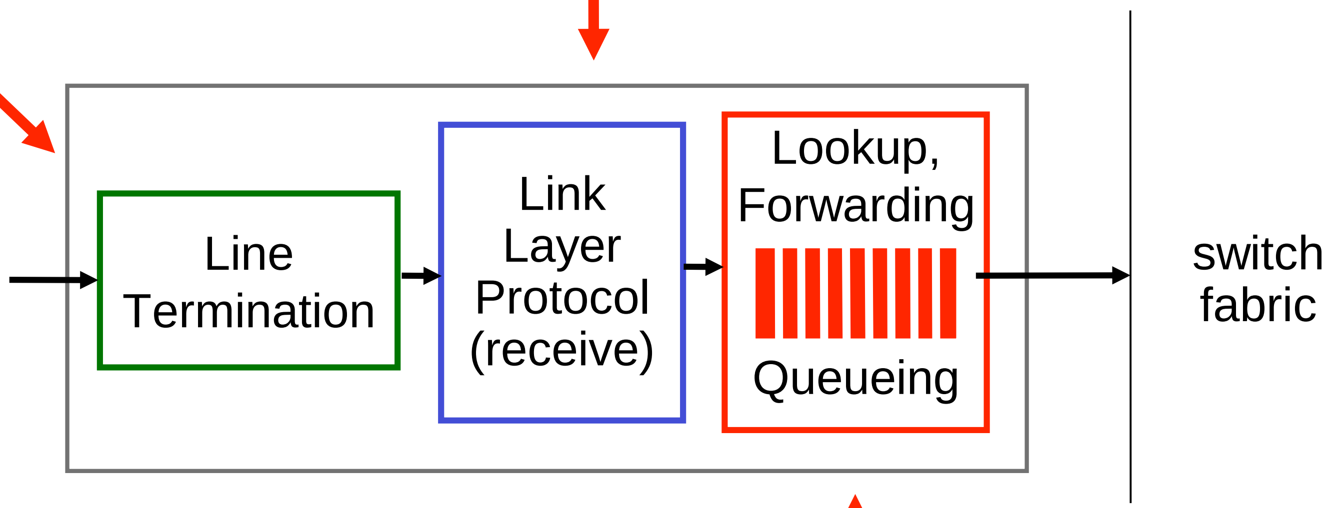
- **Routers have two key functions**
  - Run routing algorithms
  - Forward datagrams from incoming to outgoing link



Input Port Functions

Physical layer, bit-level receiving

Data Link Layer communication with router/host at other end of incoming link

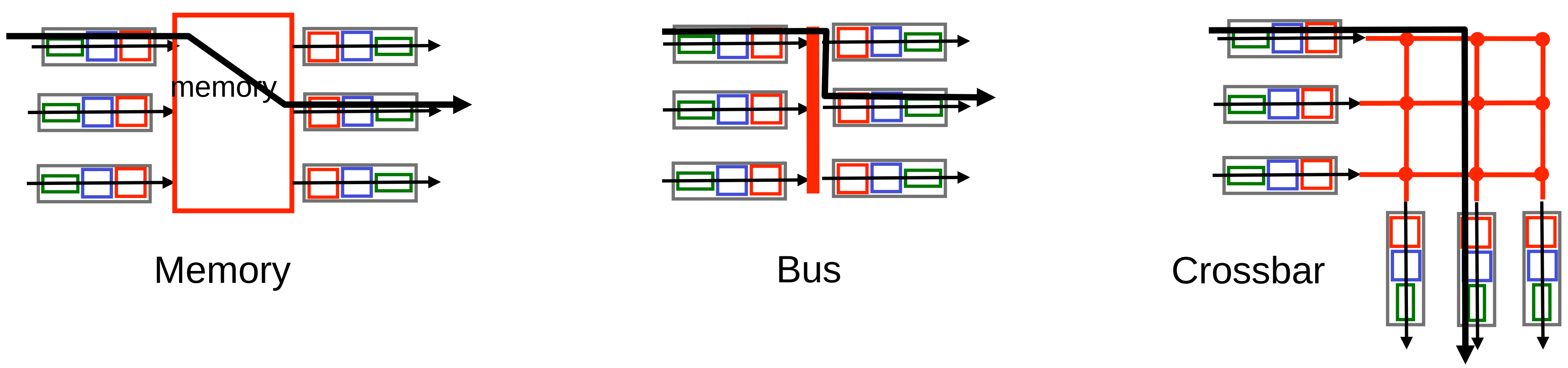


Decentralized Switching

- Given datagram destination, lookup output port using the forwarding table in input port memory
- Complete input port processing at ‘line speed’, or close to it
- If datagrams arrive faster than they can be processed, queue them up

Switching Fabrics

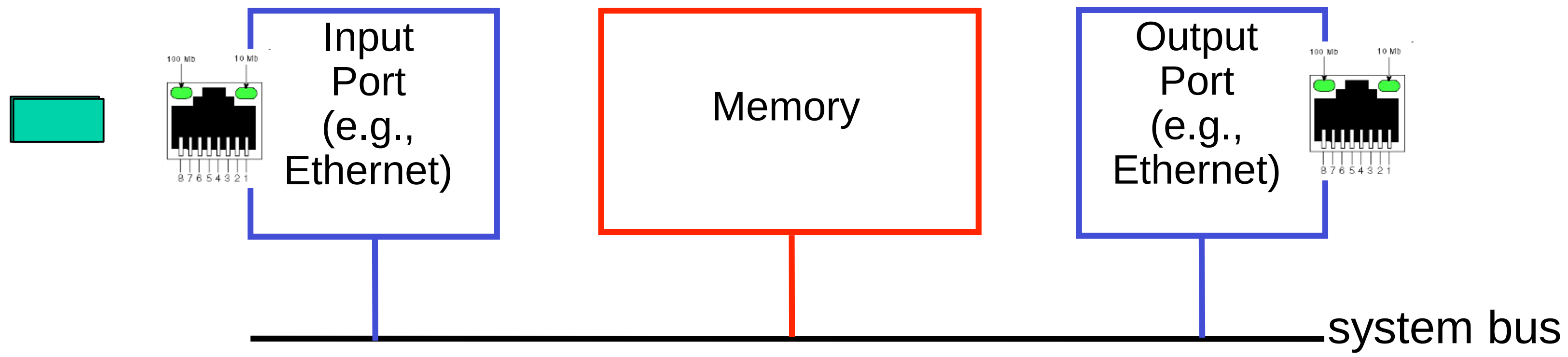
- **Used to transfer packets from input buffers to appropriate output buffers**
- **Switching rate is the rate at which packets can be transfer from inputs to outputs**
  - Often measured as multiple of input/output line rate
  - If N inputs, switching rate N times line rate is desirable
- **Three types of switching fabrics**



Memory Bus Crossbar

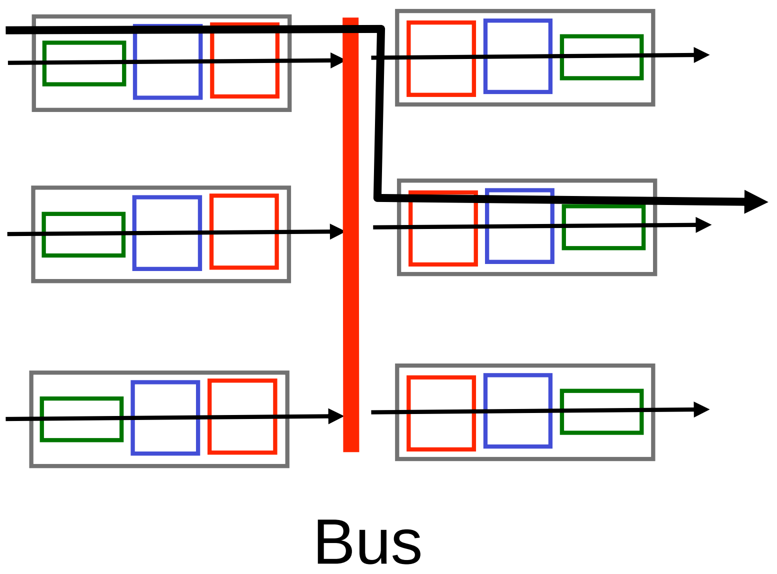
Switching Via Memory

- **Used in 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)
- **Still used in some systems today**



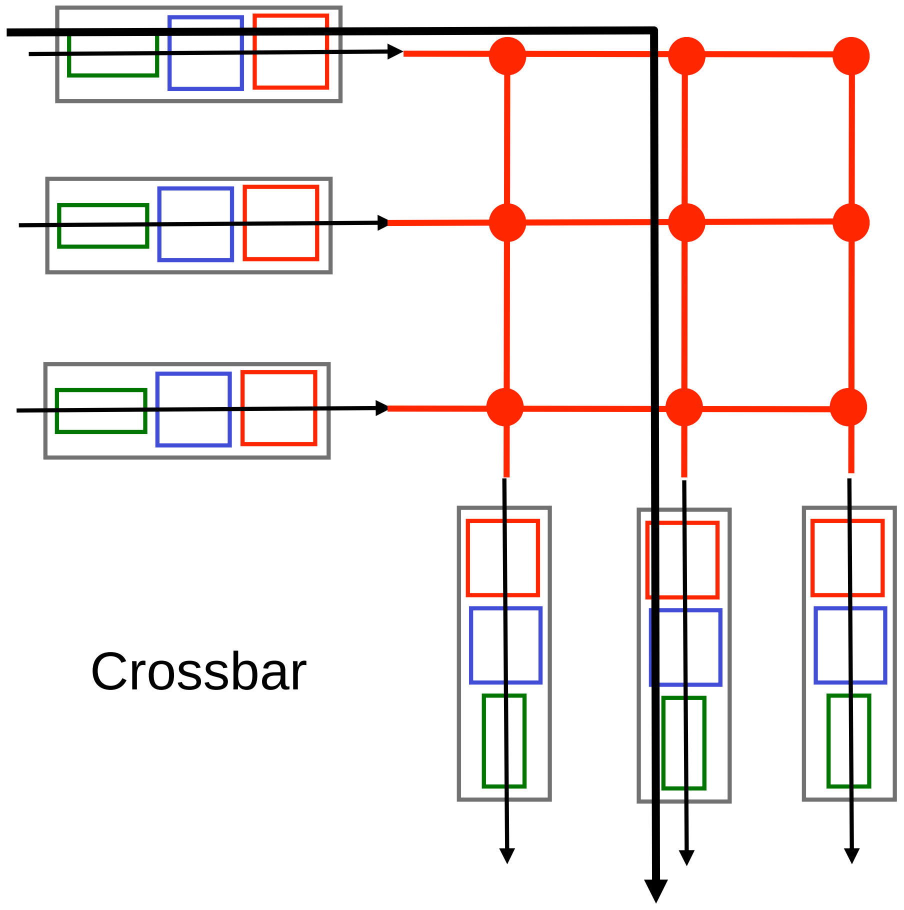
Switching Via a Bus

- **Datagram is transferred from input port memory to output port memory via a shared bus**
  - Prepend header internal to router to get packet to appropriate output port
- **Must deal with bus contention**
  - Switching speed limited by bus bandwidth
- **Example: 32 Gbps bus, Cisco 5600: sufficient speed for access and enterprise routers**



Switching Via Interconnection Network

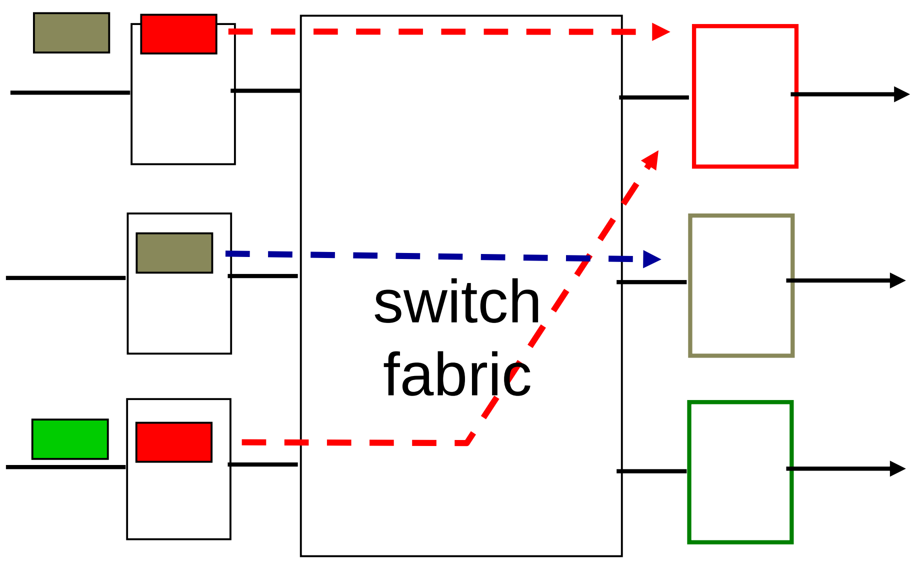
- **Overcomes bus bandwidth limitations**
- **Banyan networks, crossbar, other interconnection nets**
  - Interconnect contains 2*N buses that connects N input ports to N output ports
- **Can forward multiple packets at the same time (unless they are destined for same output)**
- **Example: The Cisco 12000 switches 60 Gbps through the interconnection network**



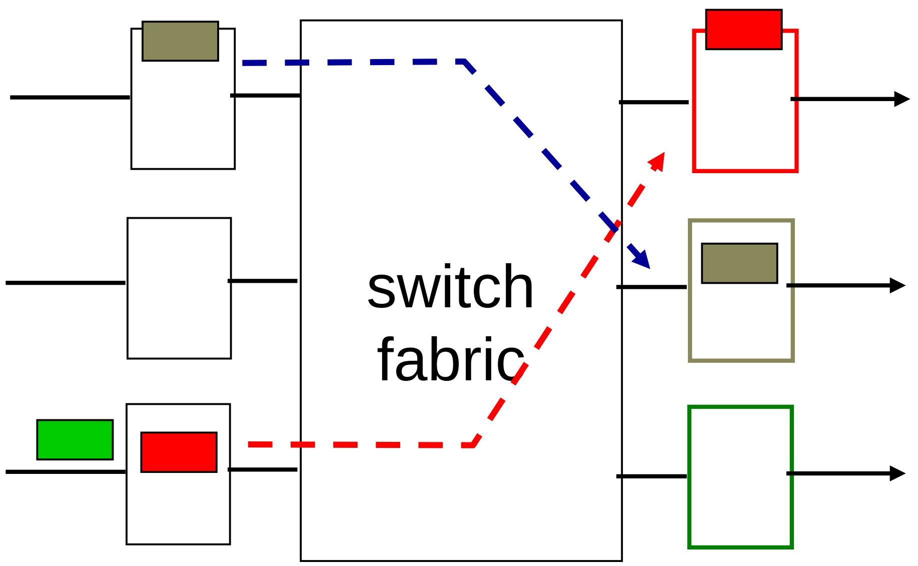
Input port queuing

- **fabric slower than input ports combined -> queueing may occur at input queues**
    - *queueing delay and loss due to input buffer overflow!*

- **Head-of-the-Line (HOL) blocking: queued datagram at front of queue prevents others in queue from moving forward**

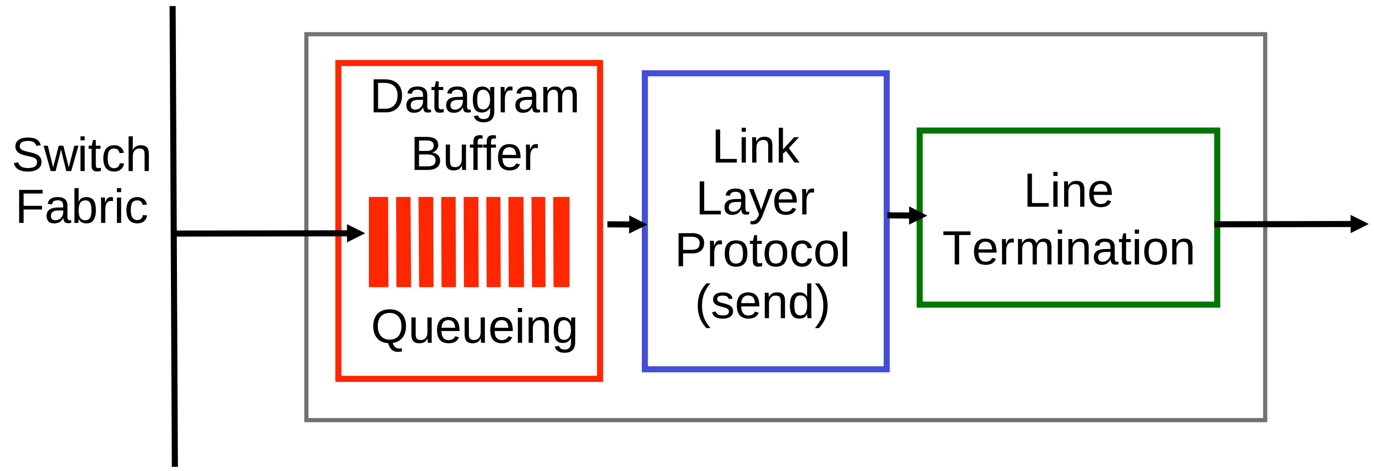


output port contention:
only one red datagram can be transferred.
lower red packet is blocked



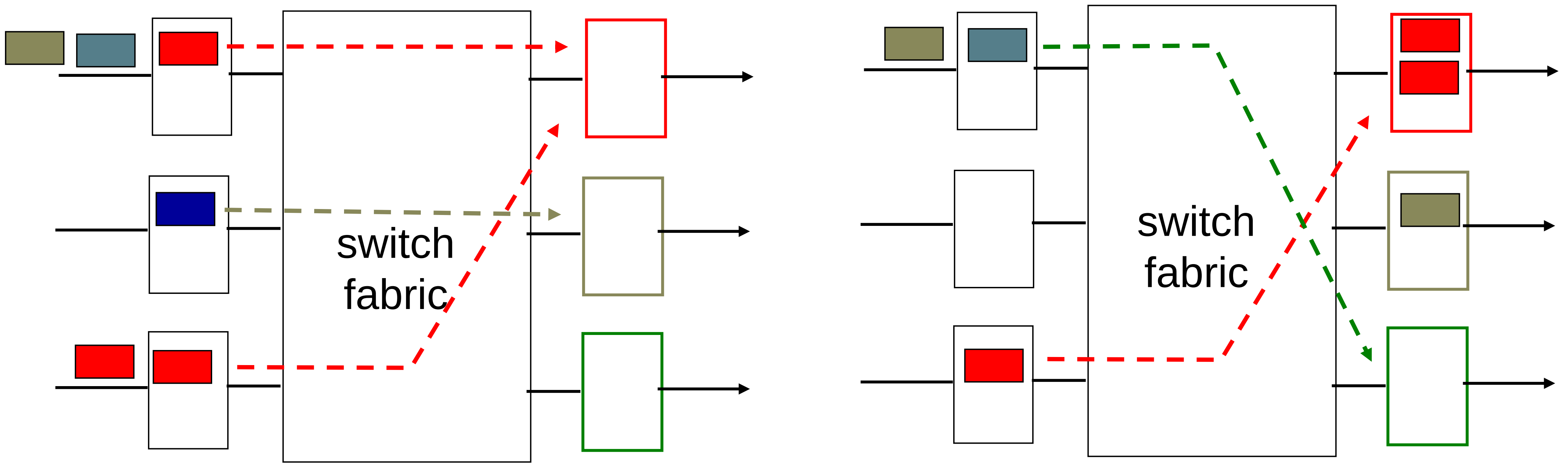
one packet time later:
green packet experiences HOL blocking

Output Ports



- **Buffers outgoing datagrams as required**
  - When datagrams arrive from fabric faster than they can be transmitted
  - Datagram (packets) can be lost due to congestion, lack of buffers
- **Scheduling to choose among queued datagrams for transmission**
  - Priority scheduling – who gets best performance, network neutrality
- **Data Link Layer communication with router/host at other end of outgoing link**
- **Physical Layer bit transmission**

Output port queueing



at *t,* packets more from input to output

one packet time later

- **buffering when arrival rate via switch exceeds output line speed**
- ***queueing (delay) and loss due to output port buffer overflow!***

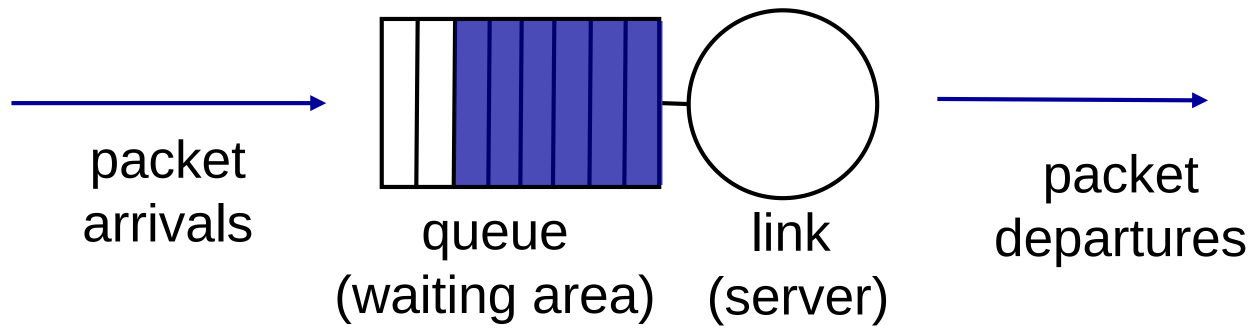
How much buffering?

- **RFC 3439 rule of thumb: average buffering equal to "typical" RTT (say 250 msec) times link capacity C**
  - e.g., C = 10 Gpbs link: 2.5 Gbit buffer
- **recent recommendation: with *N* flows, buffering equal to**

$$\frac{RTT \cdot C}{\sqrt{N}}$$

Scheduling mechanisms

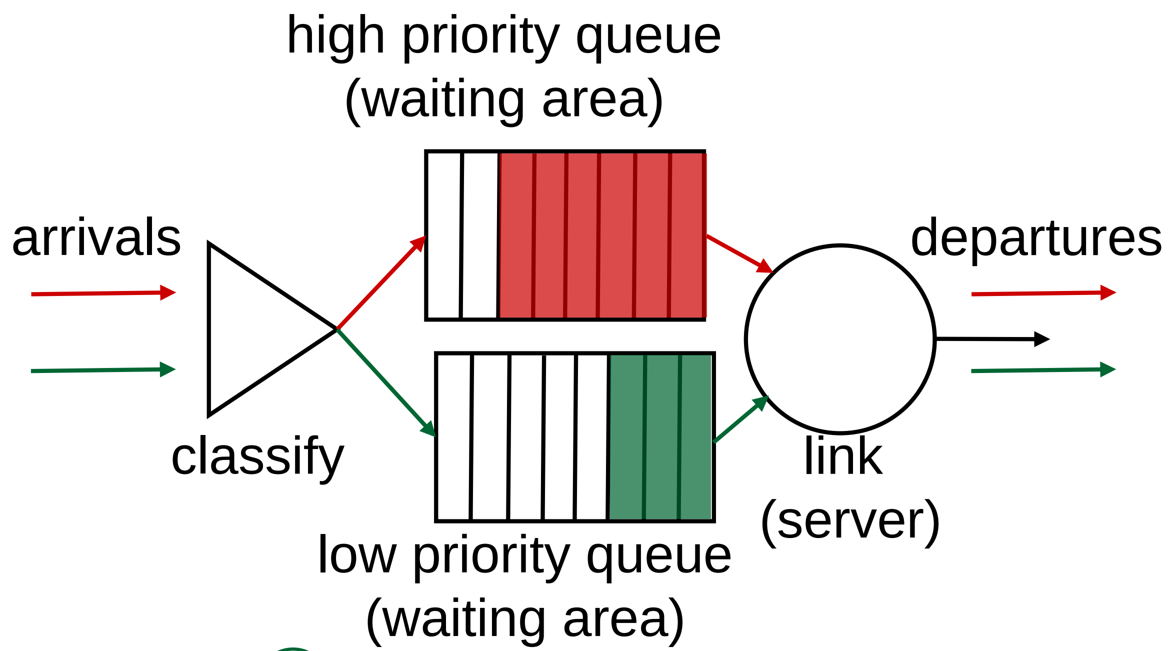
- ***scheduling:*** **choose next packet to send on link**
- ***FIFO (first in first out) scheduling:*** **send in order of arrival to queue**
  - real-world example?
  - *discard policy:* if packet arrives to full queue: who to discard?
    - *tail drop:* drop arriving packet
    - *priority:* drop/remove on priority basis
    - *random:* drop/remove randomly

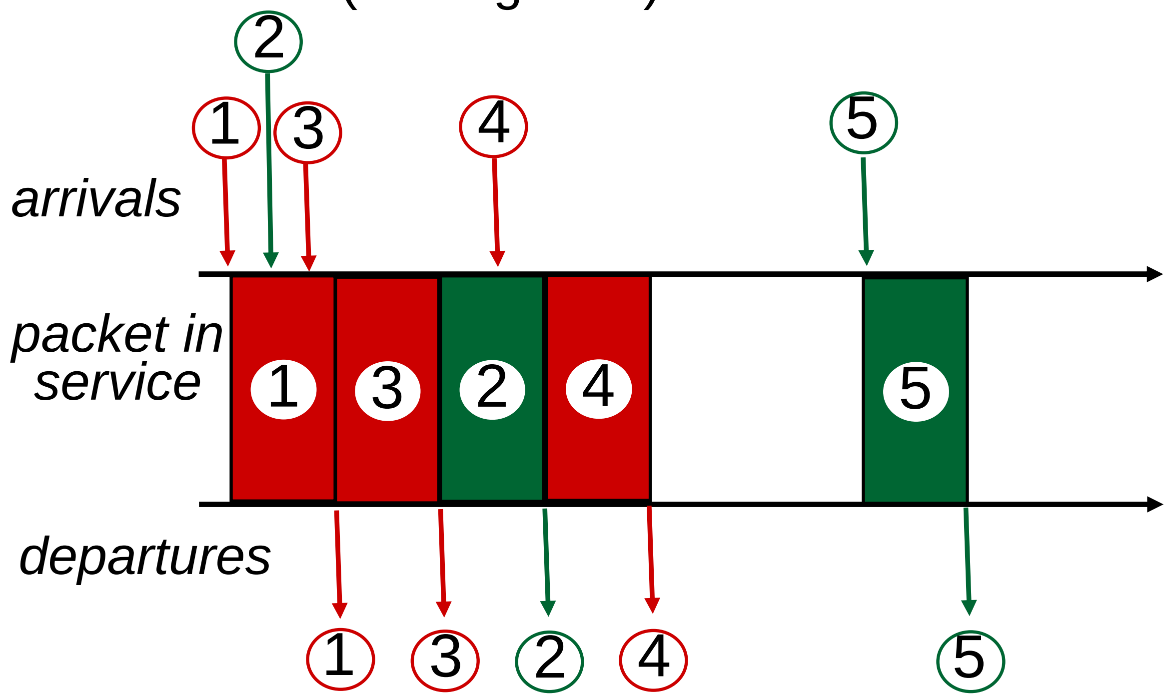


Scheduling policies: priority

priority scheduling: **send highest priority queued packet**

- **multiple *classes*, with different priorities**
  - class may depend on marking or other header info, e.g. IP source/dest, port numbers, etc.
  - real world example?

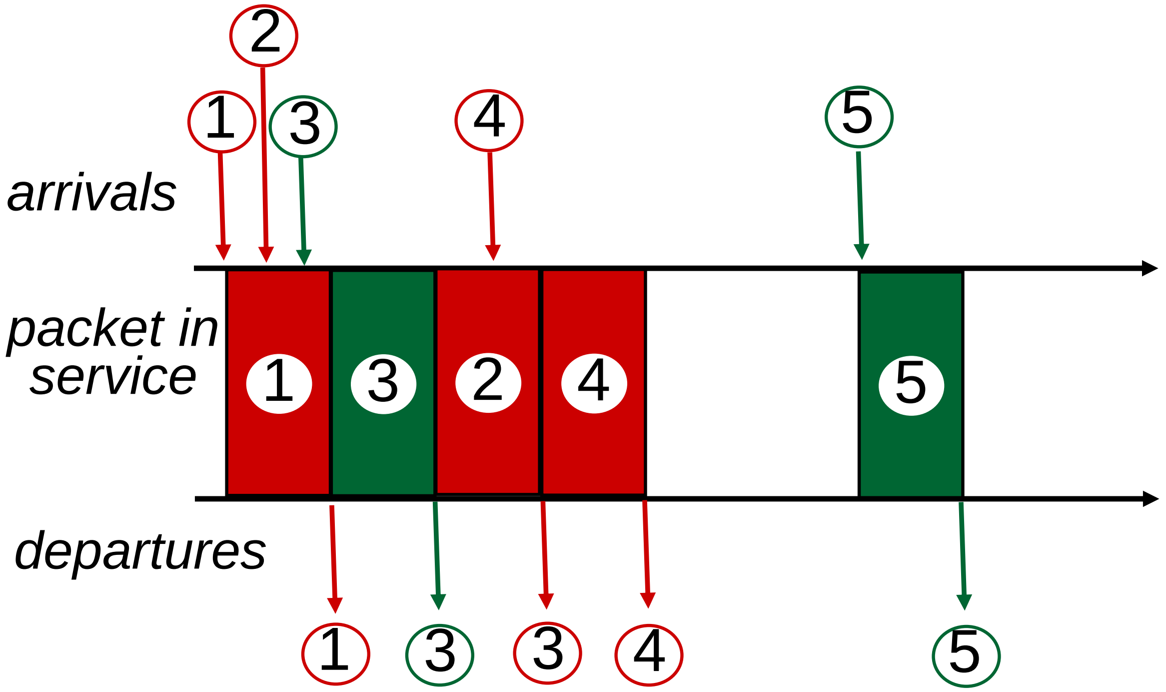




Scheduling policies: still more

Round Robin (RR) scheduling:

- **multiple classes**
- **cyclically scan class queues, sending one complete packet from each class (if available)**
- **real world example?**



Scheduling policies: still more

Weighted Fair Queuing (WFQ):

- **generalized Round Robin**
- **each class gets weighted amount of service in each cycle**
- **real-world example?**