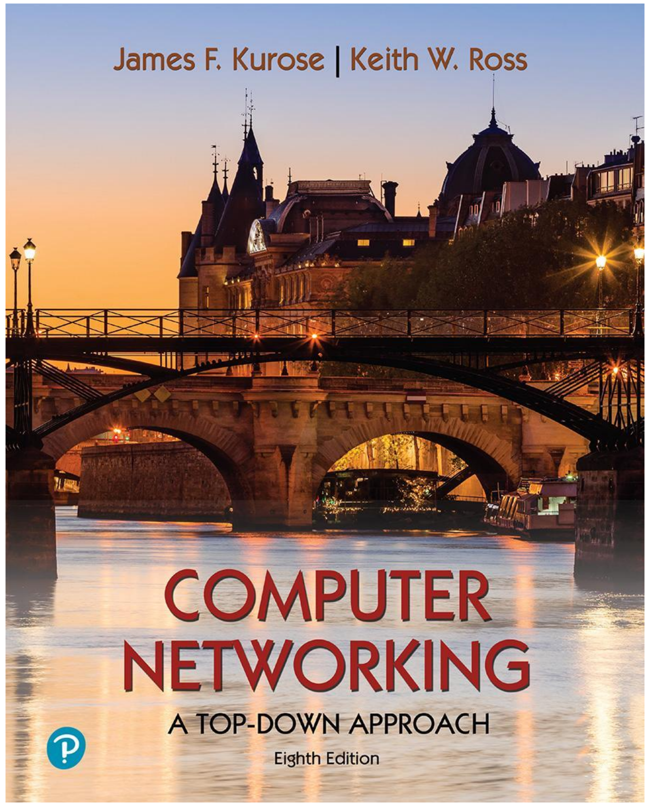
# Chapter 4
# Network Layer: Data Plane



*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

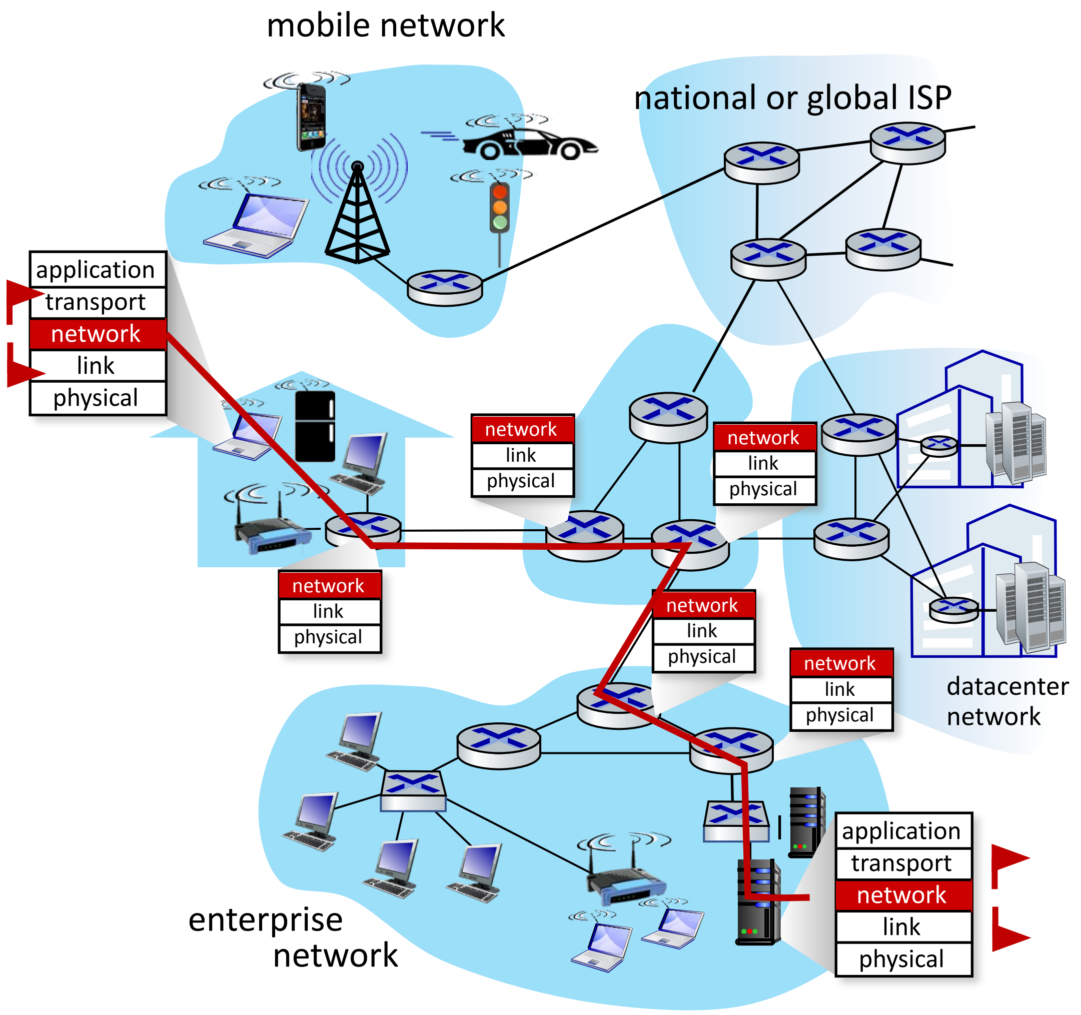
# Network layer: "data plane" roadmap

- 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

# 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



# Two key network-layer functions

network-layer functions:

- *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*

analogy: taking a trip

- *forwarding:* process of getting through single interchange
- *routing:* process of planning trip from source to destination

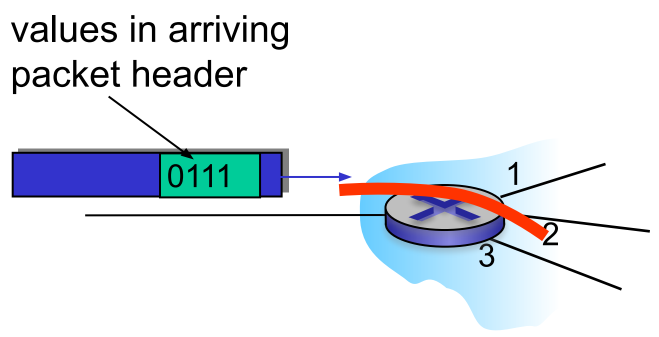
forwarding

routing

# 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

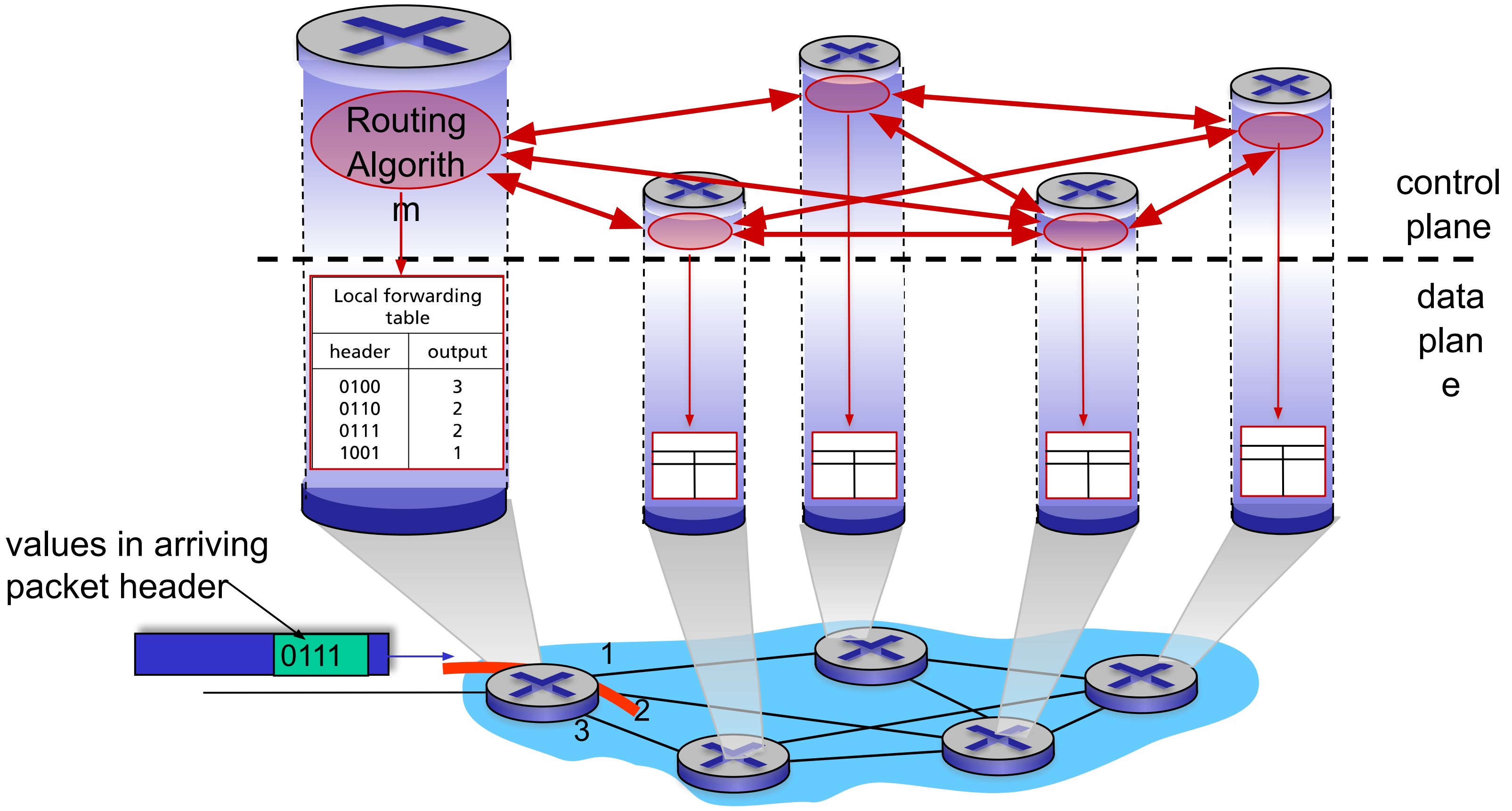


## Control plane

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

# Per-router control plane

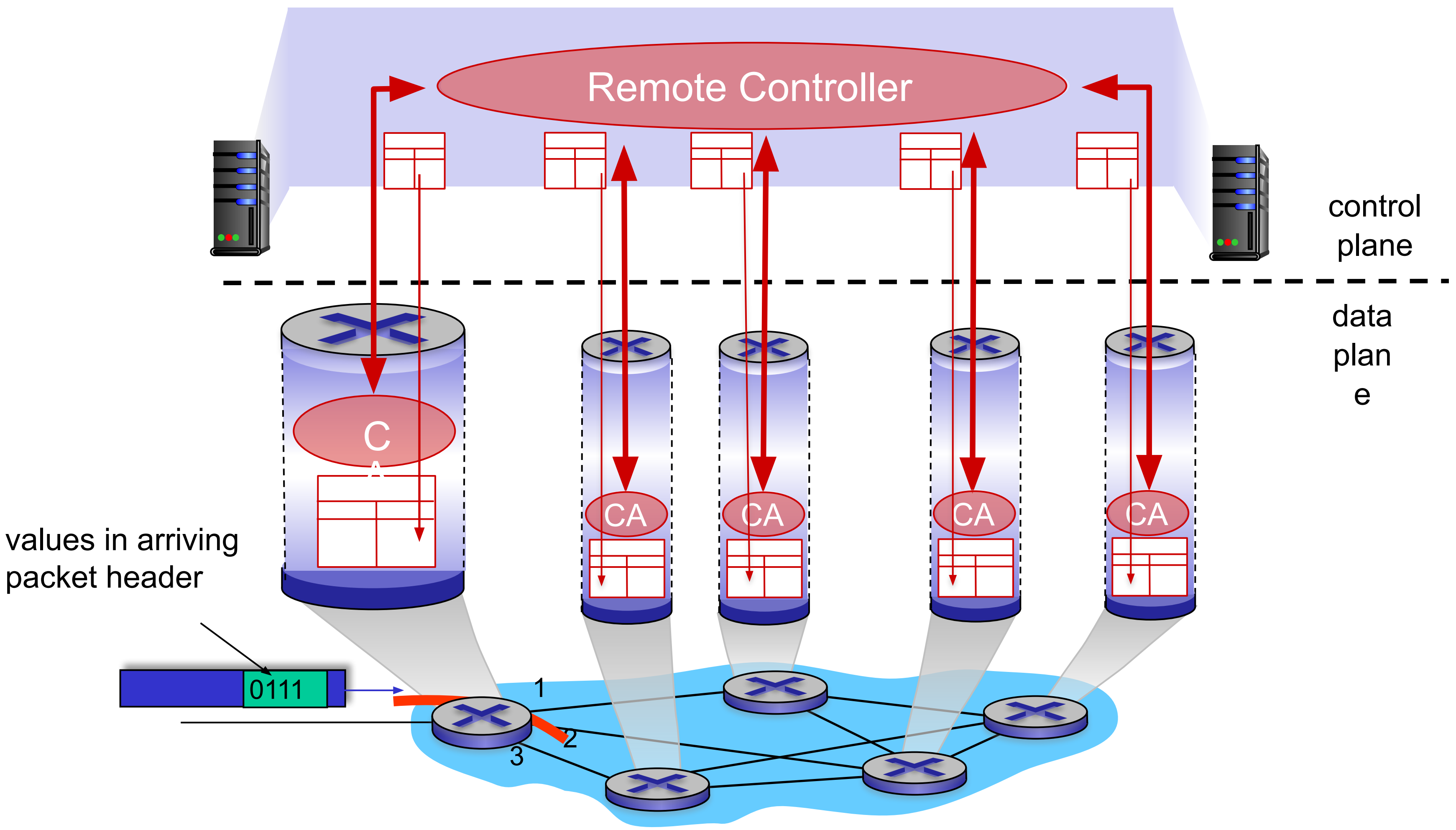
Individual routing algorithm components *in each and every router* interact in the control plane



| Local forwarding table | |
|---|---|
| header | output |
| 0100 | 3 |
| 0110 | 2 |
| 0111 | 2 |
| 1001 | 1 |

## Software-Defined Networking (SDN) control plane

Remote controller computes, installs forwarding tables in routers

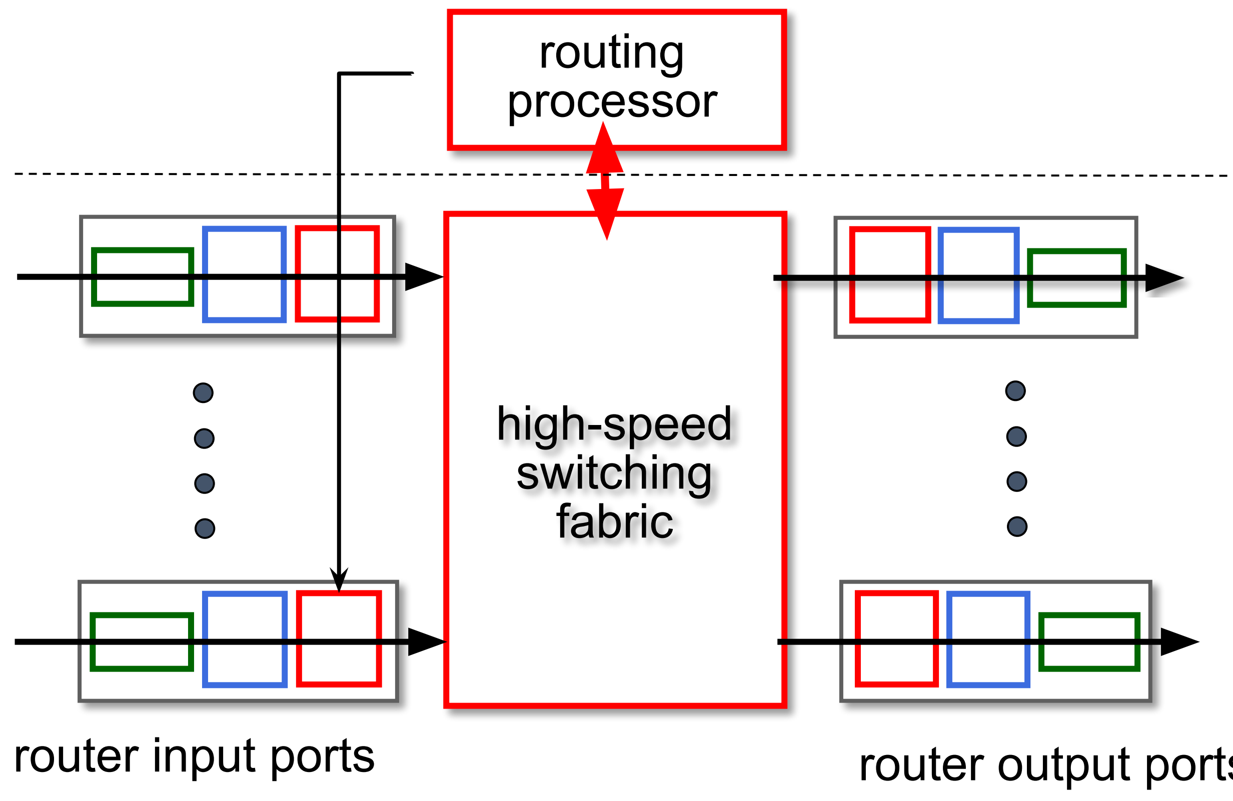


## Network layer: "data plane" roadmap

- 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:

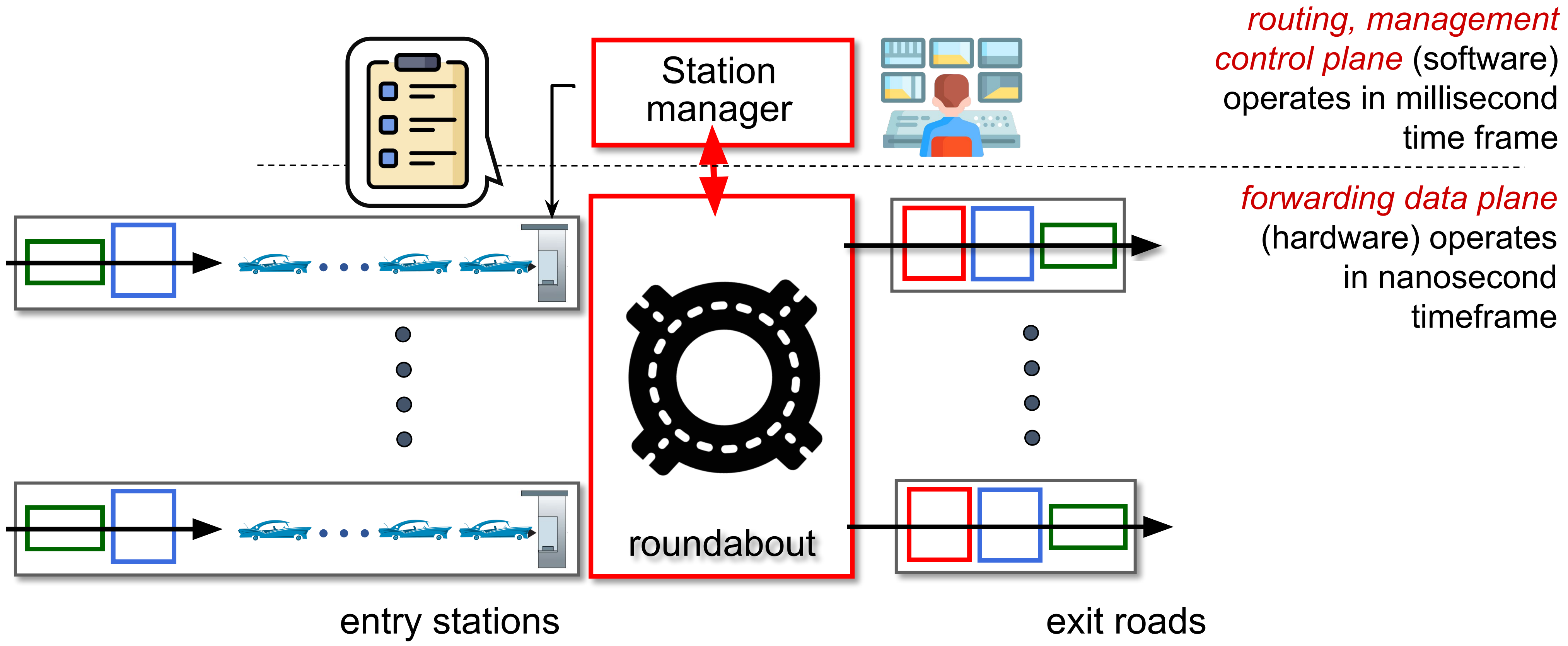


*routing, management*
*control plane* (software)
operates in millisecond
time frame

*forwarding data plane*
(hardware) operates
in nanosecond
timeframe

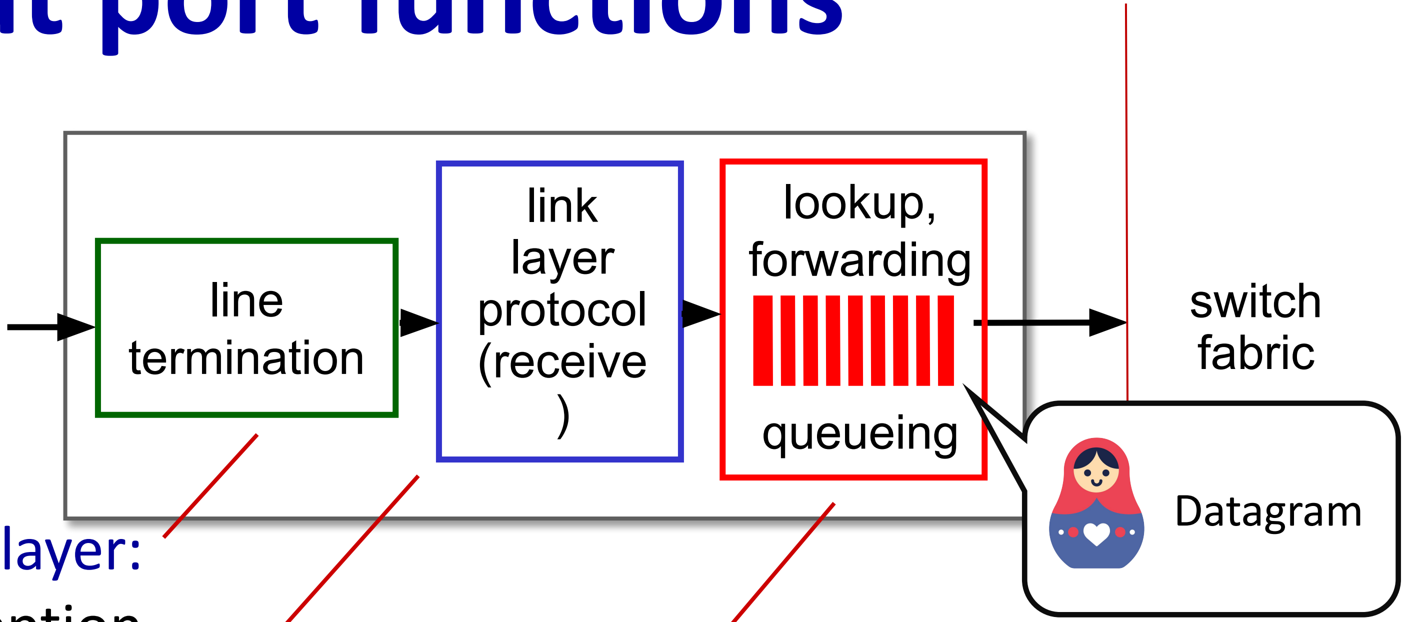
# Router architecture overview

analogy view of generic router architecture:



*routing, management control plane* (software) operates in millisecond time frame
Station manager
*forwarding data plane* (hardware) operates in nanosecond timeframe
roundabout
entry stations
exit roads

# Input port functions



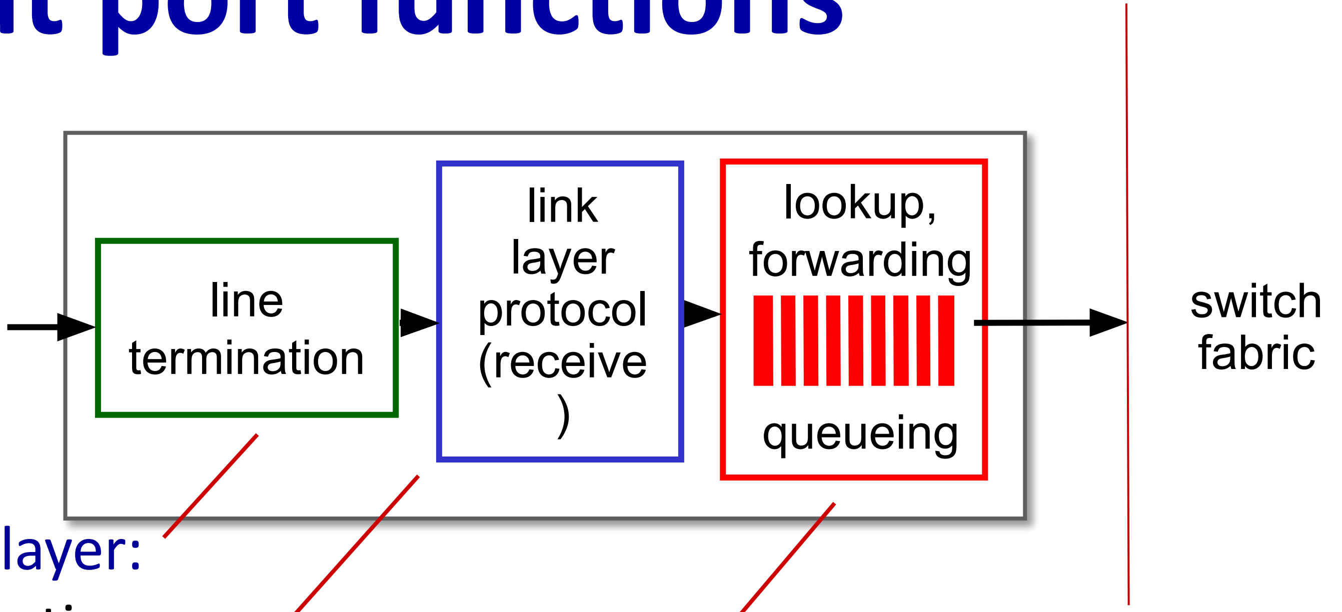
physical layer:
bit-level reception

link layer:
e.g., Ethernet

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 queuing: if datagrams arrive faster than forwarding rate into switch fabric

# Input port functions



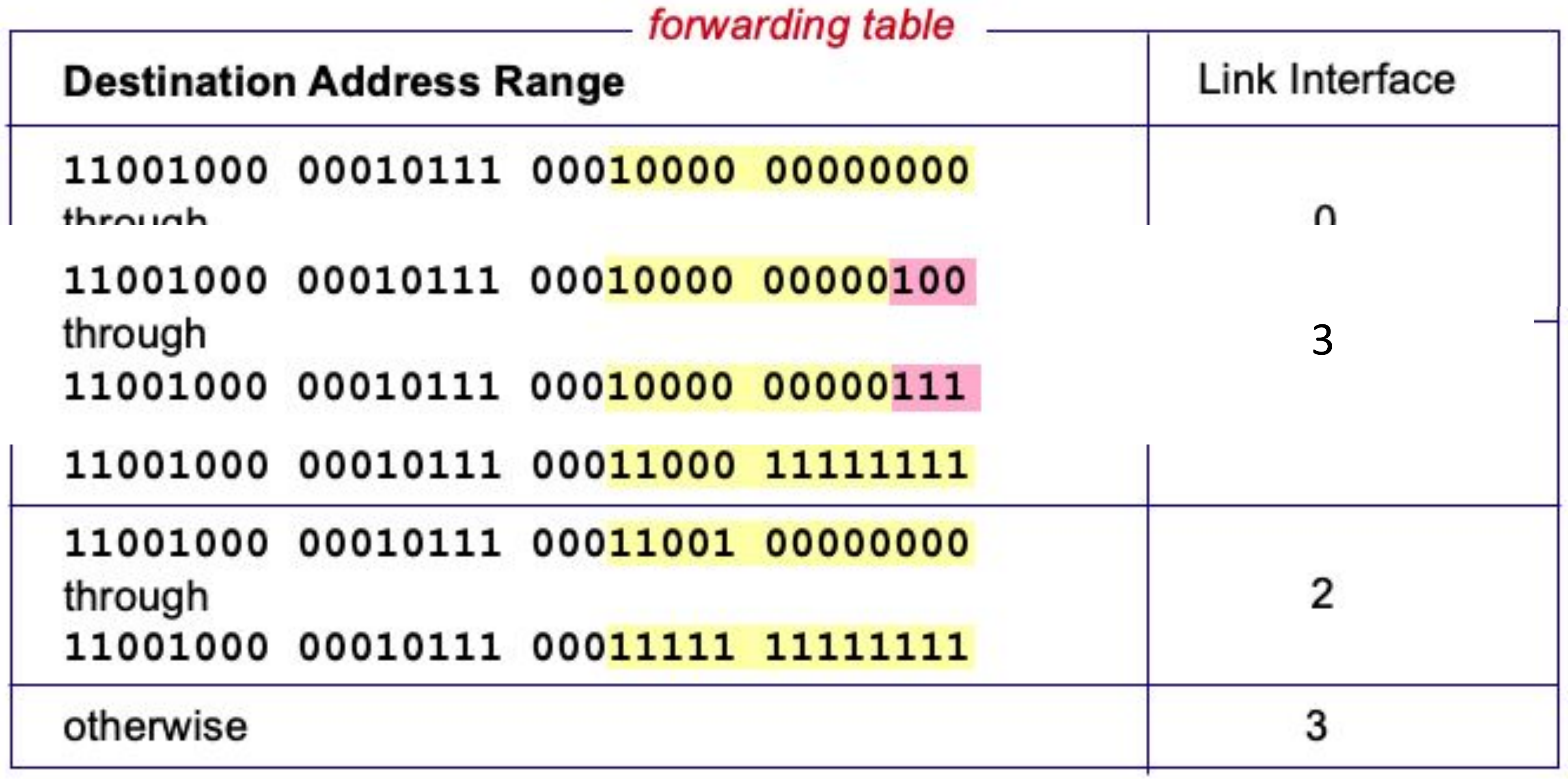
physical layer:
bit-level reception

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

decentralized switching:

- using header field values, lookup output port using forwarding table in input port memory *("match plus action")*
- destination-based forwarding: forward based only on destination IP address (traditional)
- generalized forwarding: forward based on any set of header field values

# Destination-based forwarding



forwarding table

| Destination Address Range | Link Interface |
|---|---|
| 11001000 00010111 00010000 00000000<br>through | 0 |
| 11001000 00010111 00010000 00000100<br>through<br>11001000 00010111 00010000 00000111 | 3 |
| 11001000 00010111 00011000 11111111 | |
| 11001000 00010111 00011001 00000000<br>through<br>11001000 00010111 00011111 11111111 | 2 |
| otherwise | 3 |

*Q:* but what happens if ranges don't divide up so nicely?

# 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 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

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:

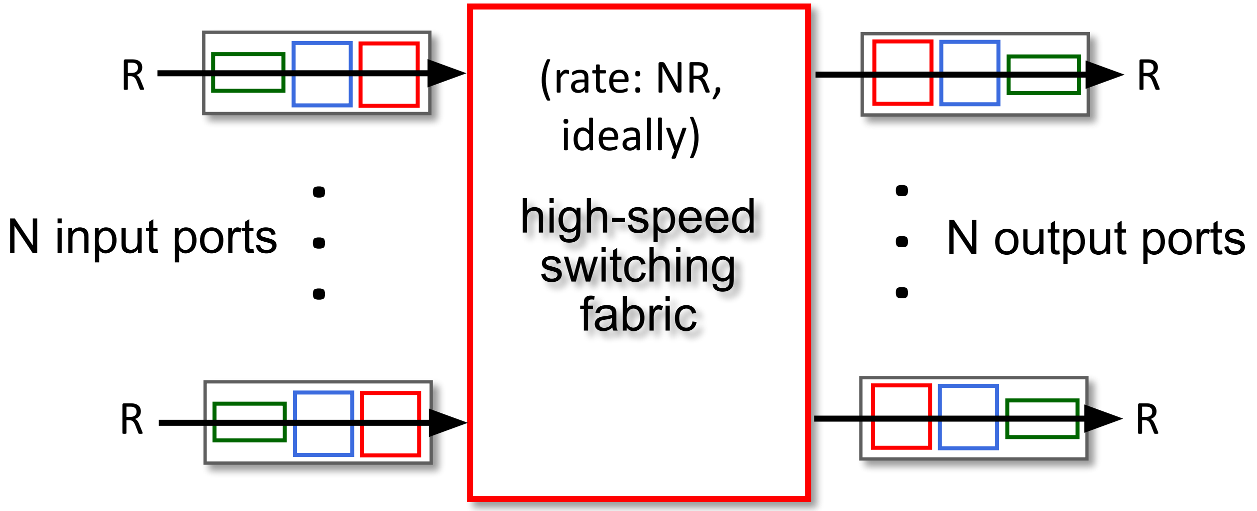
| | | | | |
|---|---|---|---|---|
| 11001000 | 00010111 | 00010110 | 10100001 | which interface? |
| 11001000 | 00010111 | 00011000 | 10101010 | which interface? |

## 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

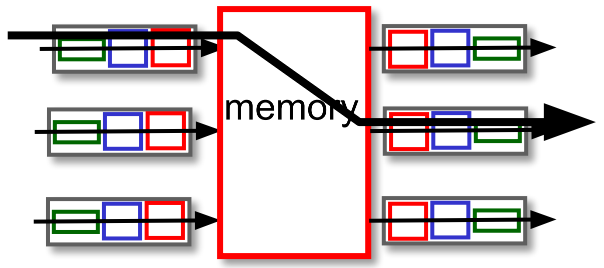
# Switching fabrics

- 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

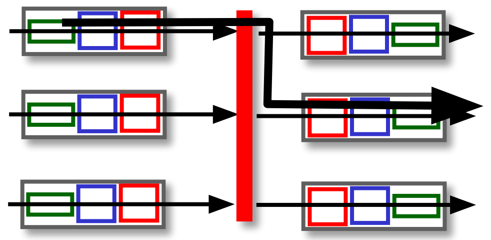


# Switching fabrics

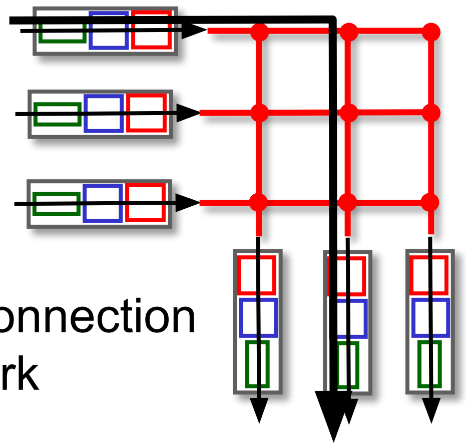
- transfer packet from input link to appropriate output link
- switching rate: rate at which packets can be transfer from inputs to outputs
  - often measured as multiple of input/output line rate
  - N inputs: switching rate N times line rate desirable
- three major types of switching fabrics:



memory



bus

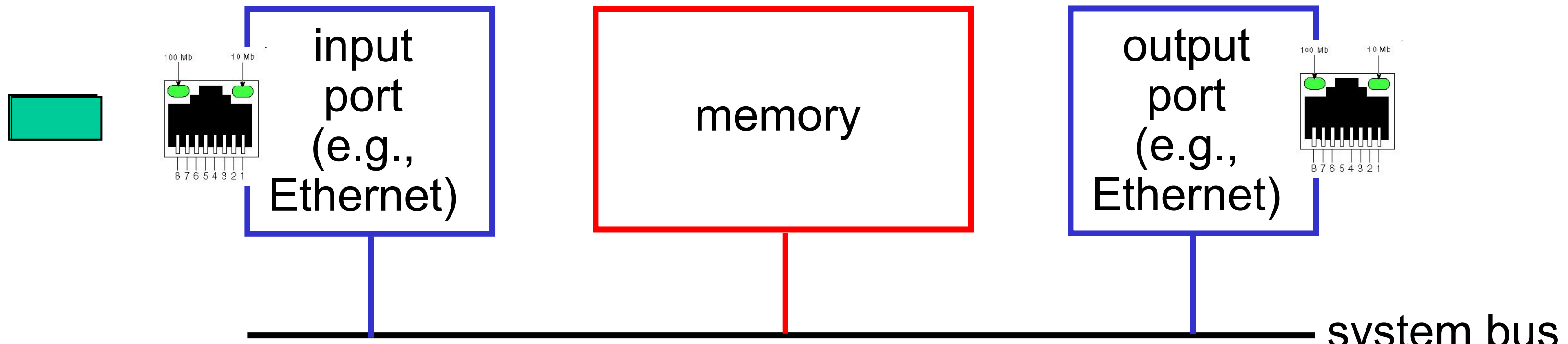


interconnection network

## Switching via memory

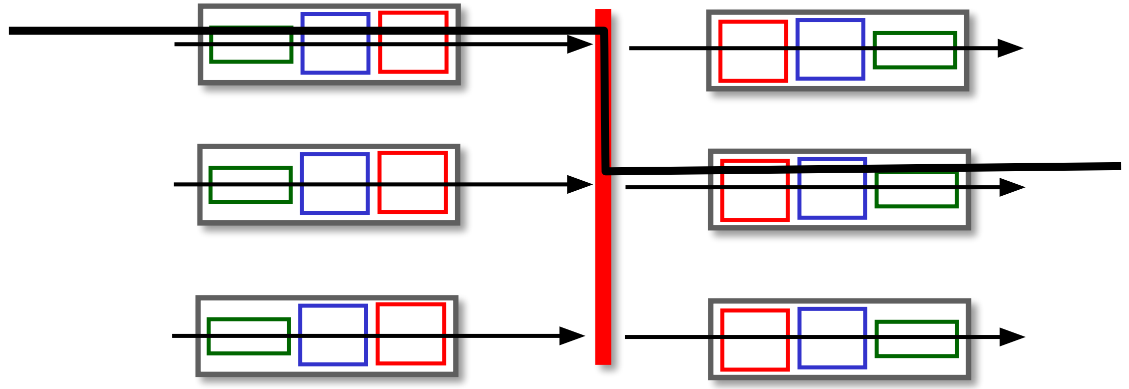
first generation routers:

- 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)



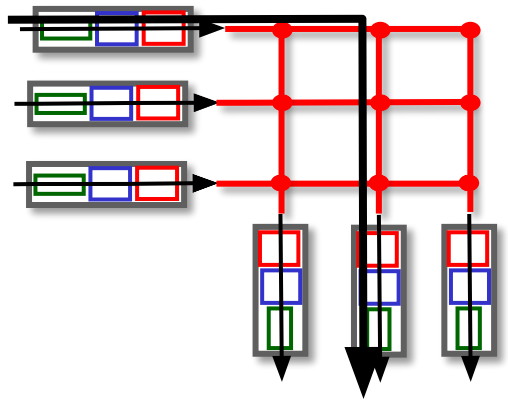
## Switching via a bus

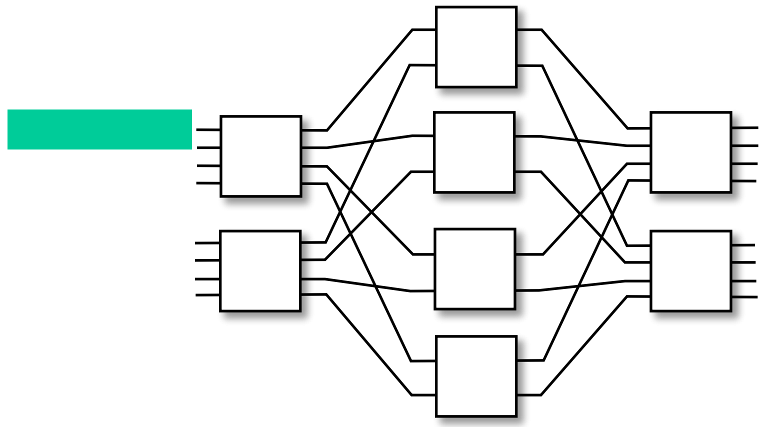
- datagram from input port memory to output port memory via a shared bus
- *bus contention:* switching speed limited by bus bandwidth
- 32 Gbps bus, Cisco 5600: sufficient speed for access routers



# Switching via interconnection network

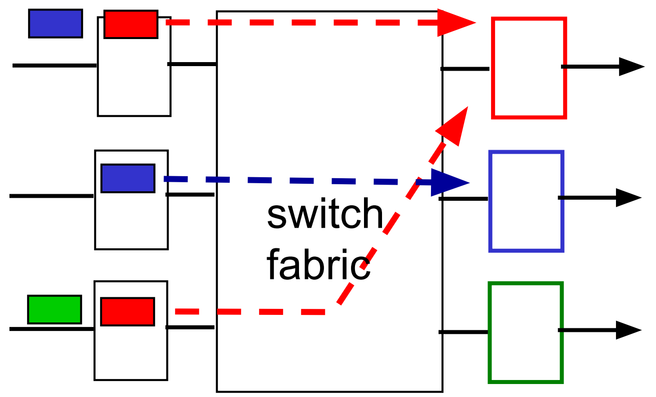
- Crossbar, Clos networks, other interconnection nets initially developed to connect processors in multiprocessor
- multistage switch: *nxn* switch from multiple stages of smaller switches
- exploiting parallelism:
  - fragment datagram into fixed length cells on entry
  - switch cells through the fabric,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


3x3 crossbar

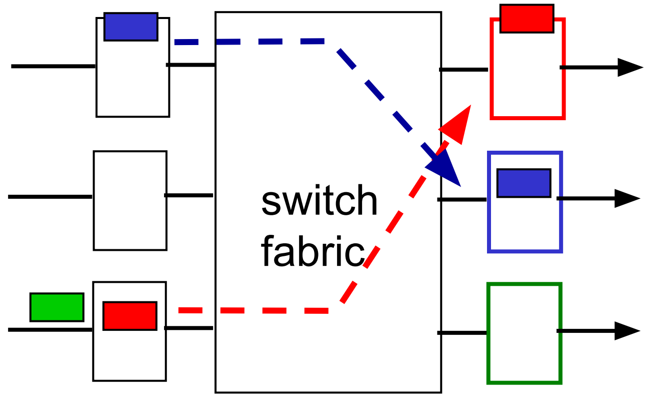

8x8 multistage switch
built from smaller-sized switches

# Input port queuing

- If switch 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

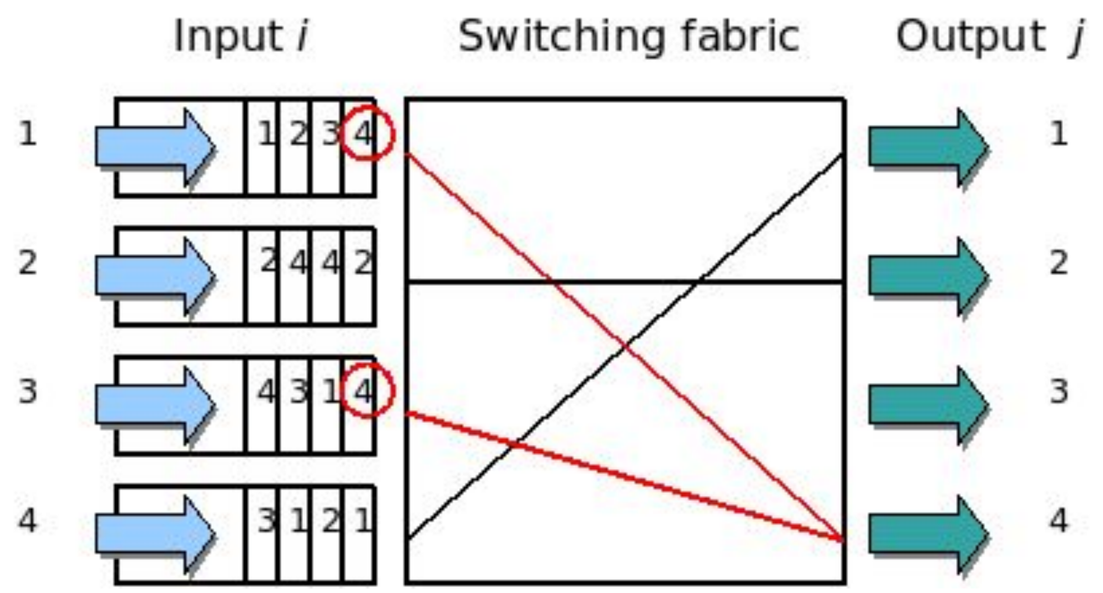


output port contention: only one red datagram can be transferred. lower red packet is *blocked*



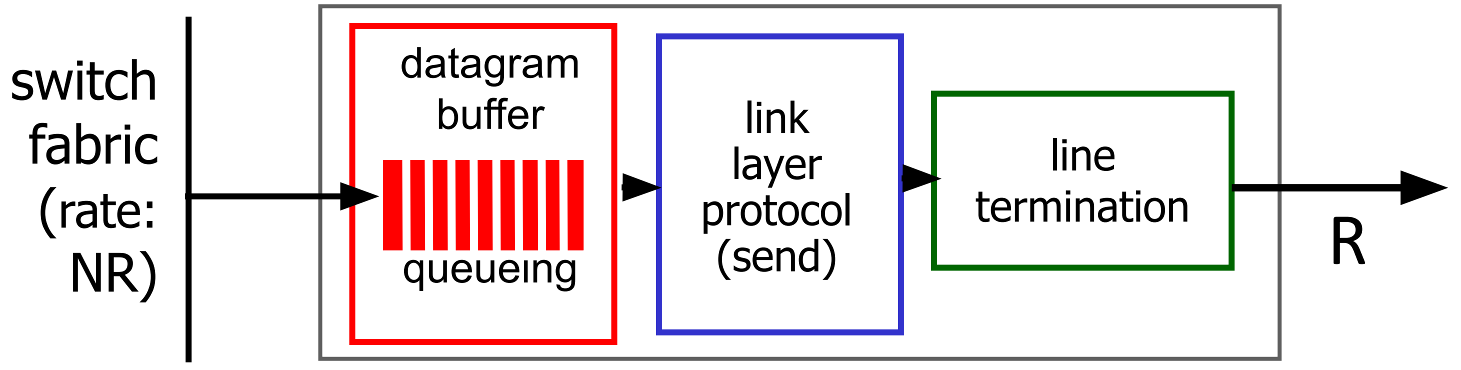
one packet time later: green packet experiences HOL blocking

# Head of Line blocking



Head-of-line blocking example: The 1st and 3rd input flows are competing to send packets to the same output interface. In this case if the switching fabric decides to transfer the packet from the 3rd input flow, the 1st input flow cannot be processed in the same time slot. Note that the 1st input flow is blocking a packet for output interface 3, which is available for processing.

# Output port queuing



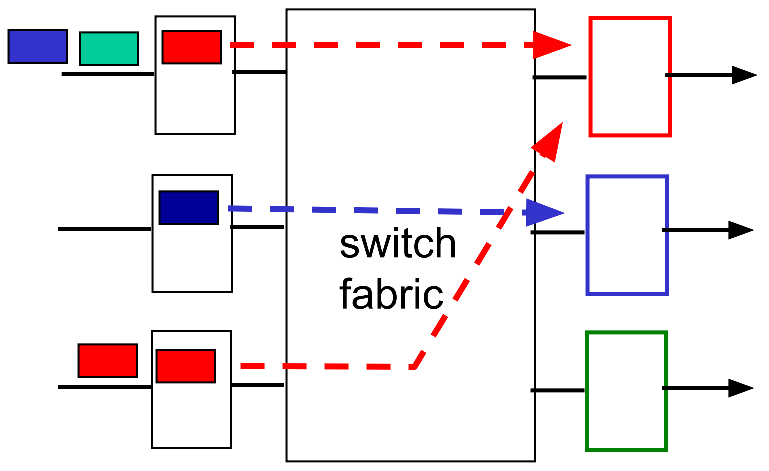
- *Buffering* required when datagrams arrive from fabric faster than link transmission rate. *Drop policy:* which datagrams to drop if no free buffers?

Datagrams can be lost due to congestion, lack of buffers

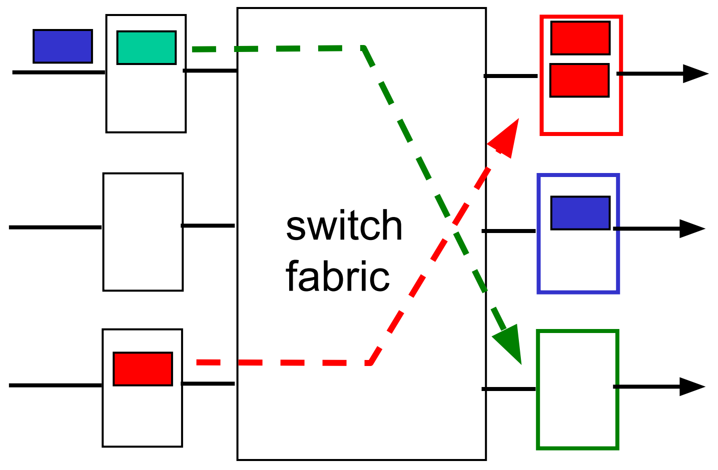
- *Scheduling discipline* chooses among queued datagrams for transmission

Priority scheduling – who gets best performance, network neutrality

## Output port queuing



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!*

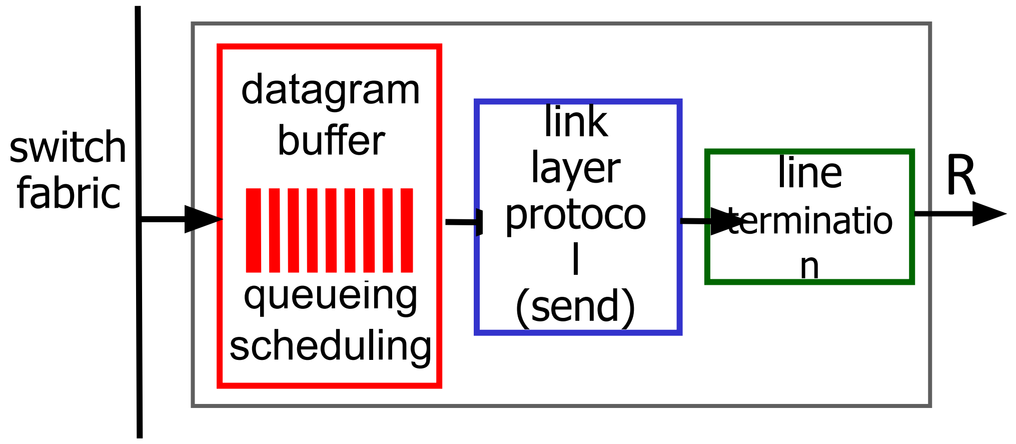
## How much buffering?

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

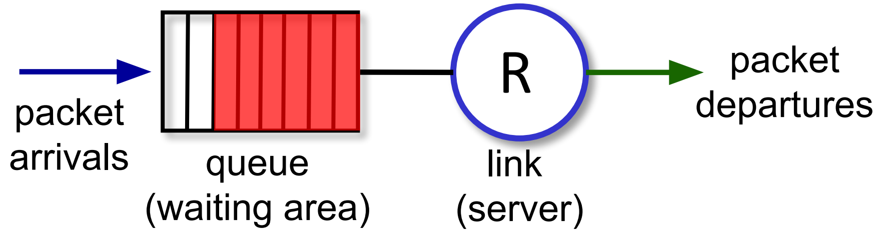
$$\frac{\mathrm{RTT} \cdot C}{\sqrt{N}}$$

- but *too* much buffering can increase delays (particularly in home routers)
  - long RTTs: poor performance for real-time apps, sluggish TCP response
  - recall delay-based congestion control: "keep bottleneck link just full enough (busy) but no fuller"

# Buffer Management



Abstraction: queue



buffer management:

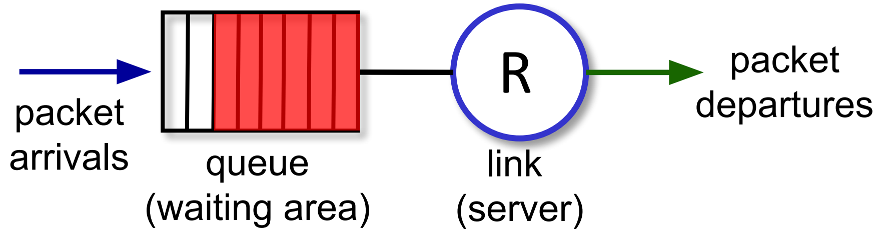
- drop: which packet to add, drop when buffers are full
  - tail drop: drop arriving packet
  - priority: drop/remove on priority basis
- marking: which packets to mark to signal congestion (ECN, RED)

# Packet Scheduling: FCFS

packet scheduling: deciding which packet to send next on link

- first come, first served
- priority
- round robin
- weighted fair queueing

Abstraction: queue



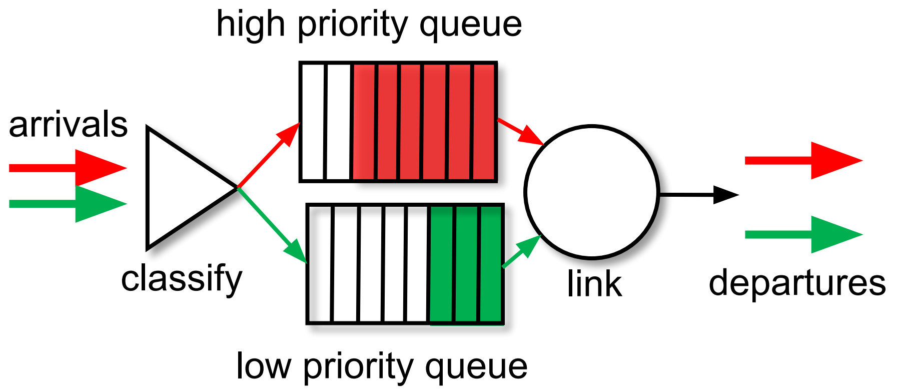
FCFS: packets transmitted in order of arrival to output port

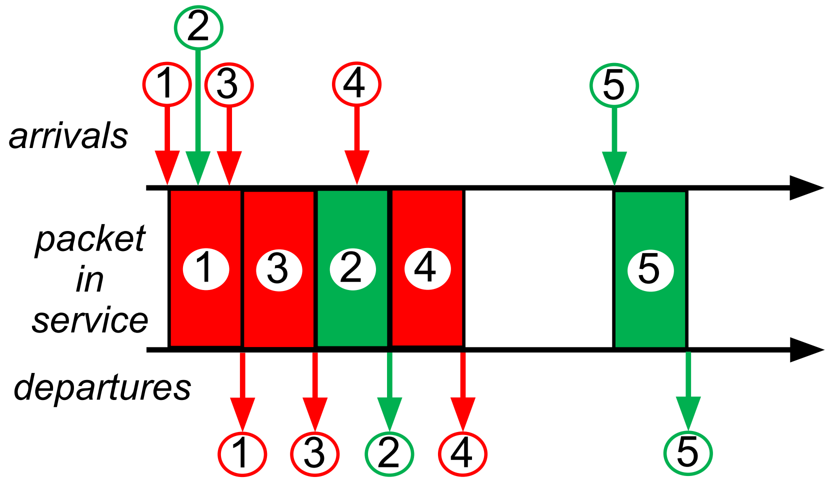
- also known as: First-in-first-out (FIFO)
- real world examples?

# Scheduling policies: priority

*Priority scheduling:*

- arriving traffic classified, queued by class
  - any header fields can be used for classification
- send packet from highest priority queue that has buffered packets
  - FCFS within priority class

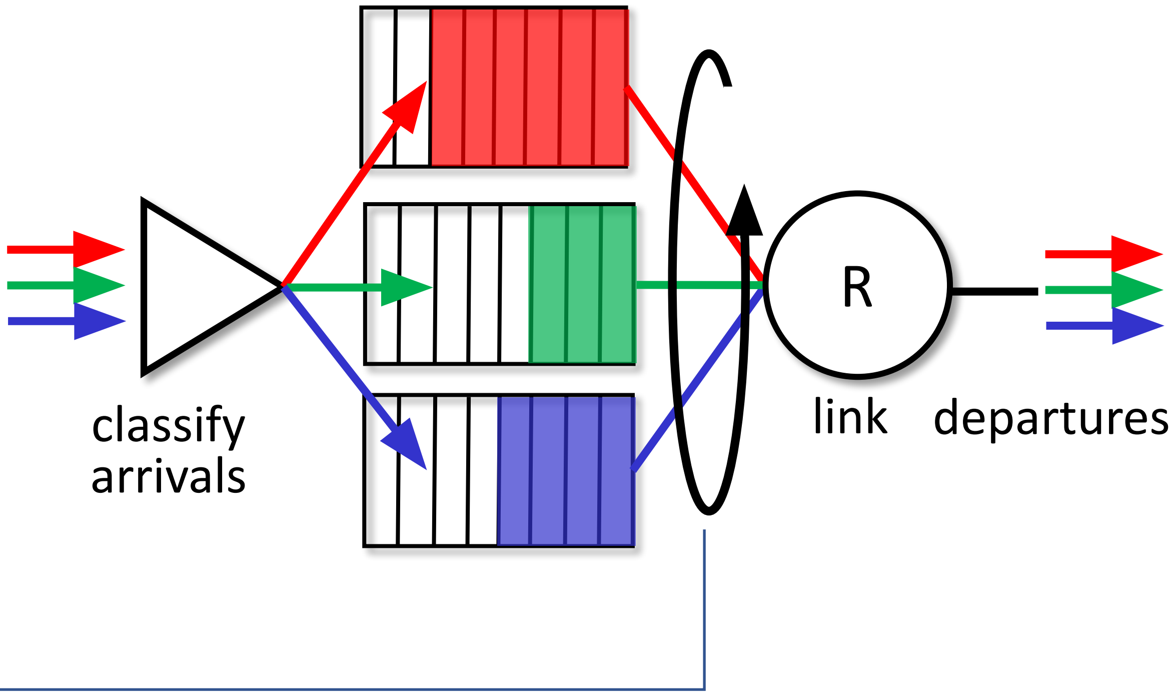




# Scheduling policies: round robin

*Round Robin (RR) scheduling:*

- arriving traffic classified, queued by class
  - any header fields can be used for classification
- server cyclically, repeatedly scans class queues, sending one complete packet from each class (if available) in turn



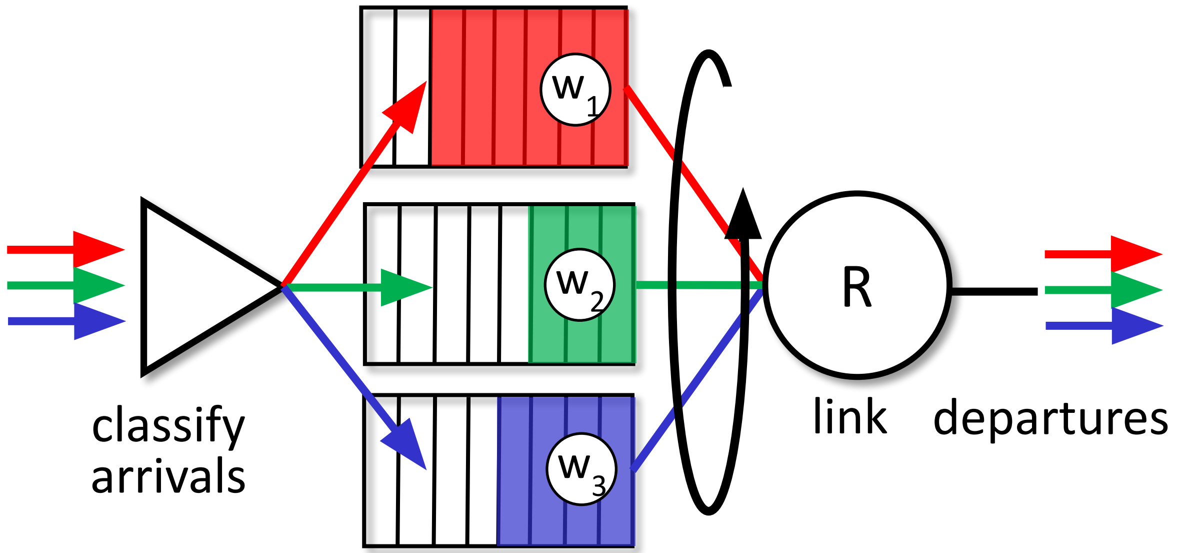
# Scheduling policies: weighted fair queueing

*Weighted Fair Queuing (WFQ):*

- generalized Round Robin
- each class, *i*, has weight, $w_i$, and gets weighted amount of service in each cycle:

$$\frac{w_i}{\Sigma_j w_j}$$

- minimum bandwidth guarantee (per-traffic-class)



## Sidebar: Network Neutrality

What is network neutrality?

- *technical:* how an ISP should share/allocation its resources
  - packet scheduling, buffer management are the *mechanisms*
- *social, economic* principles
  - protecting free speech
  - encouraging innovation, competition
- enforced *legal* rules and policies

*Different countries have different "takes" on network neutrality*

## Sidebar: Network Neutrality

2015 US FCC *Order on Protecting and Promoting an Open Internet:* three “clear, bright line” rules:

- no blocking … “shall not block lawful content, applications, services, or non-harmful devices, subject to reasonable network management.”
- no throttling … “shall not impair or degrade lawful Internet traffic on the basis of Internet content, application, or service, or use of a non-harmful device, subject to reasonable network management.”
- no paid prioritization. … “shall not engage in paid prioritization”

## ISP: telecommunications or information service?

Is an ISP a "telecommunications service" or an "information service" provider?

- the answer *really* matters from a regulatory standpoint!

US Telecommunication Act of 1934 and 1996:

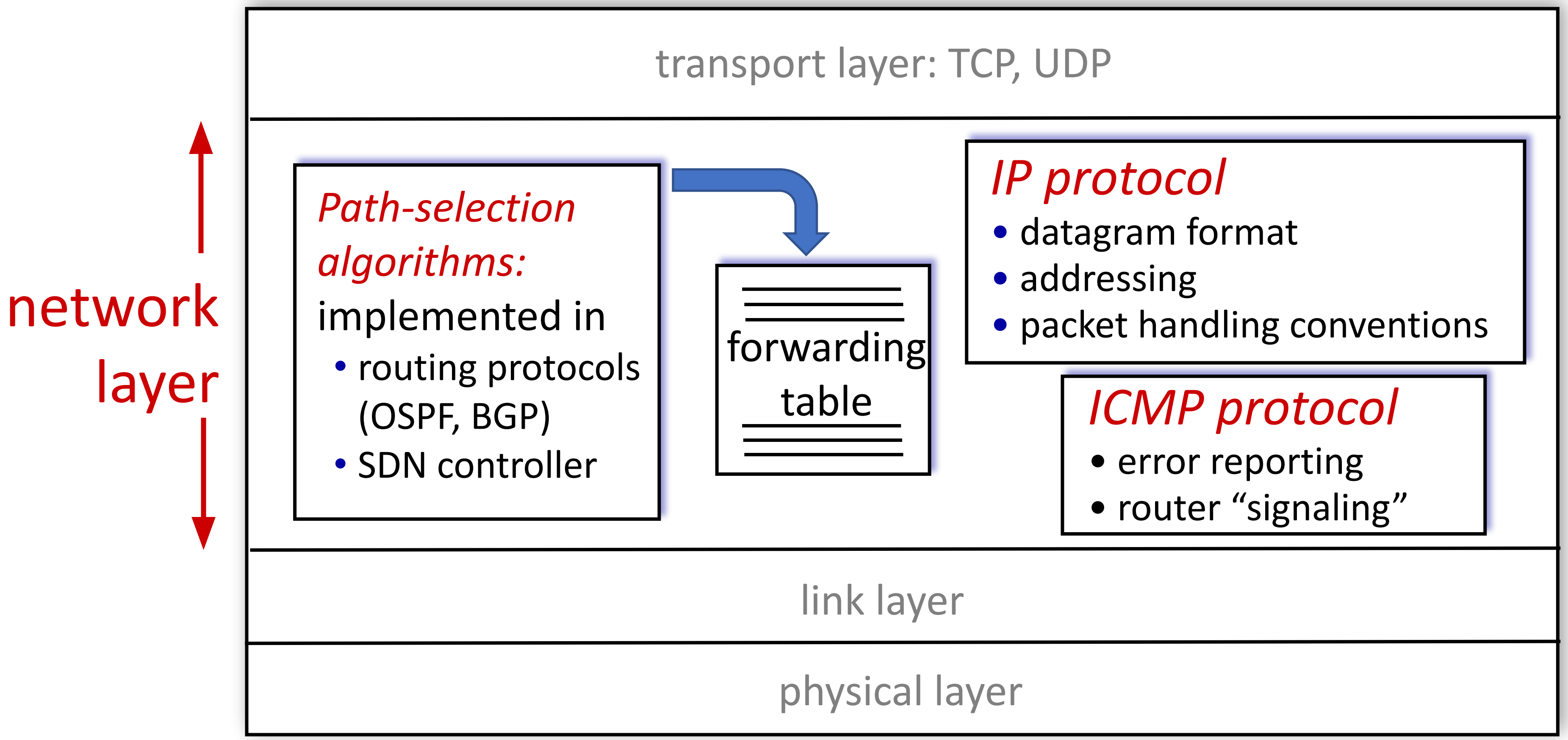
- *Title II:* imposes "common carrier duties" on *telecommunications services*: reasonable rates, non-discrimination and *requires regulation*
- *Title I:* applies to *information services:*
  - no common carrier duties (*not regulated*)
  - but grants FCC authority "… as may be necessary in the execution of its functions"[4]

# Network layer: "data plane" roadmap

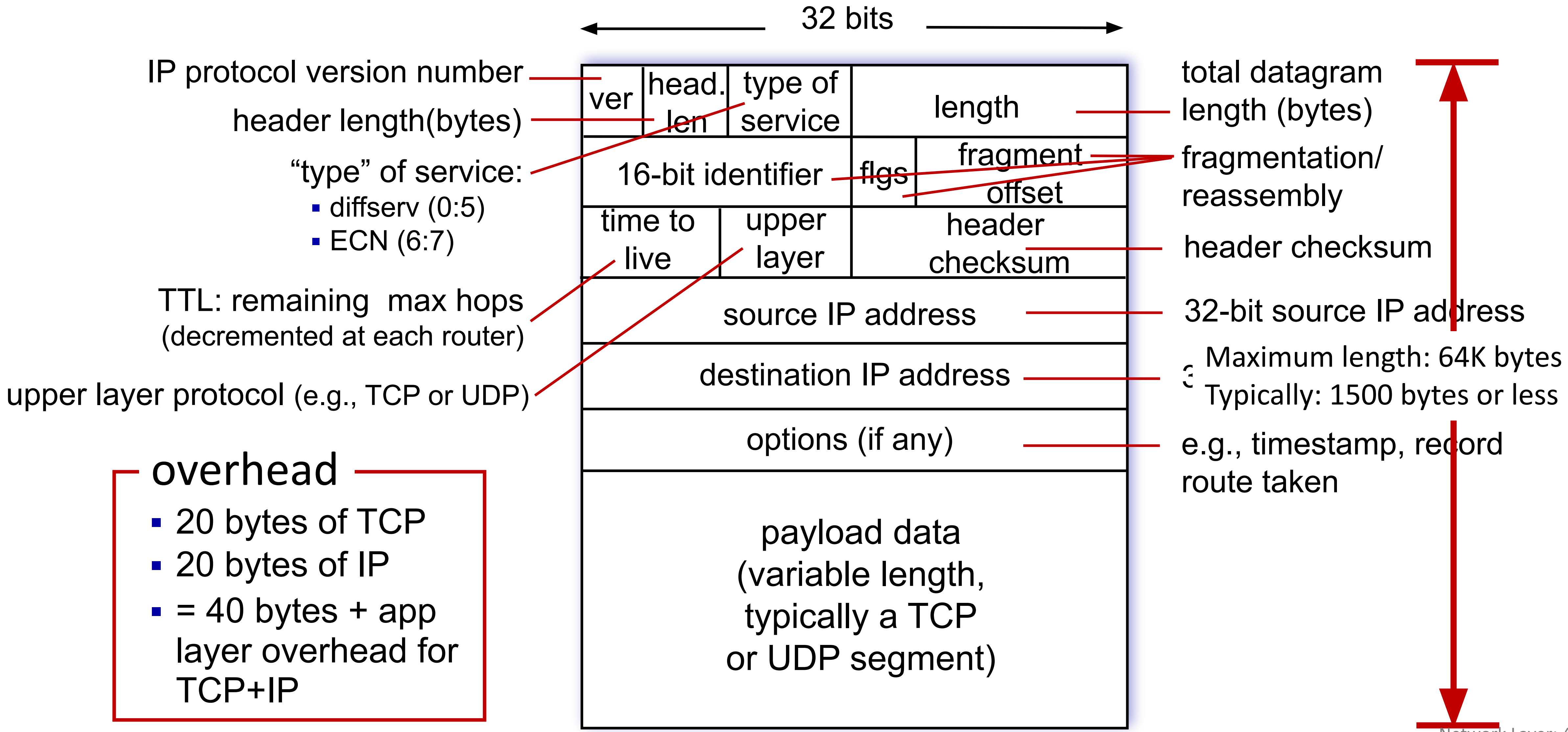
- Network layer: overview
  - data plane
  - control plane
- What's inside a router
  - input ports, switching, output ports
  - buffer management, scheduling
- IP: the Internet Protocol
  - datagram format
  - addressing
  - network address translation
  - IPv6
- Generalized Forwarding, SDN
  - match+action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

# Network Layer: Internet

host, router network layer functions:



# IP Datagram format


32 bits
ver
head. len
type of service
length
16-bit identifier
flgs
fragment offset
time to live
upper layer
header checksum
source IP address
destination IP address
options (if any)
payload data (variable length, typically a TCP or UDP segment)
IP protocol version number
header length(bytes)
“type” of service:
diffserv (0:5)
ECN (6:7)
TTL: remaining max hops (decremented at each router)
upper layer protocol (e.g., TCP or UDP)
total datagram length (bytes)
fragmentation/ reassembly
header checksum
32-bit source IP address
Maximum length: 64K bytes
Typically: 1500 bytes or less
e.g., timestamp, record route taken

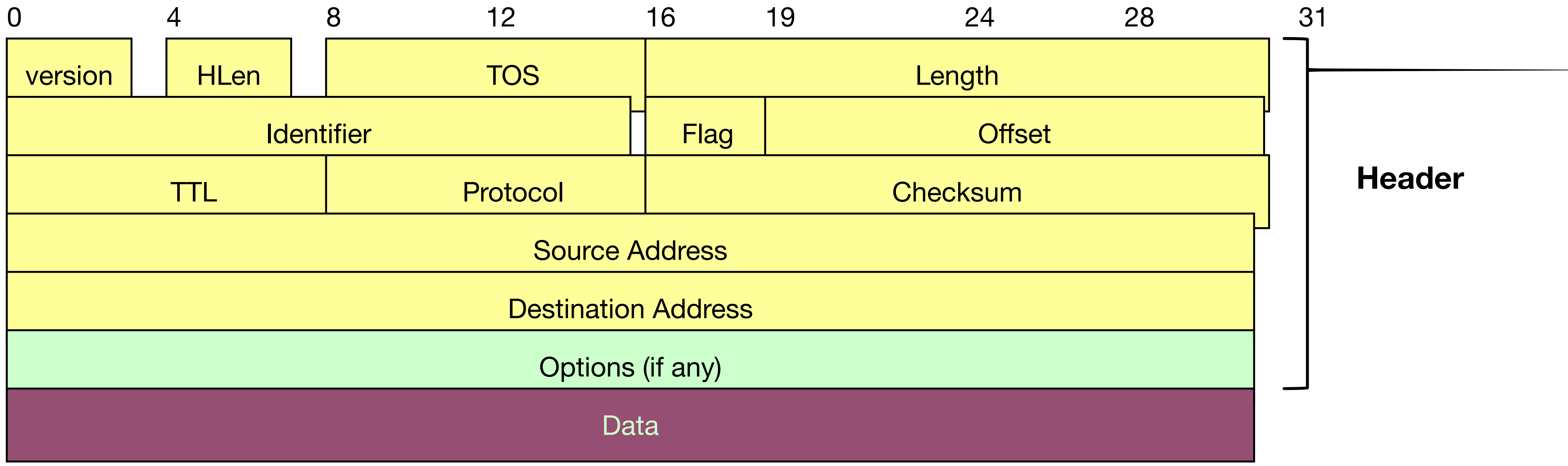
overhead

- 20 bytes of TCP
- 20 bytes of IP
- = 40 bytes + app layer overhead for TCP+IP

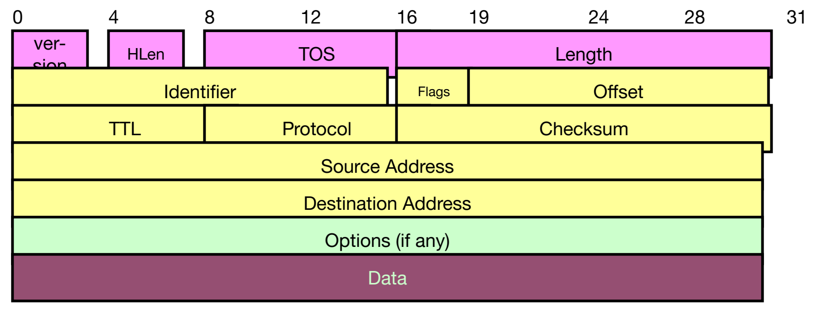
## IP Service Model

- Low-level communication model provided by Internet
- Datagram
  - Each packet self-contained
    - All information needed to get to destination
    - No advance setup or connection maintenance
  - Analogous to letter or telegram

**IPv4 Packet Format**



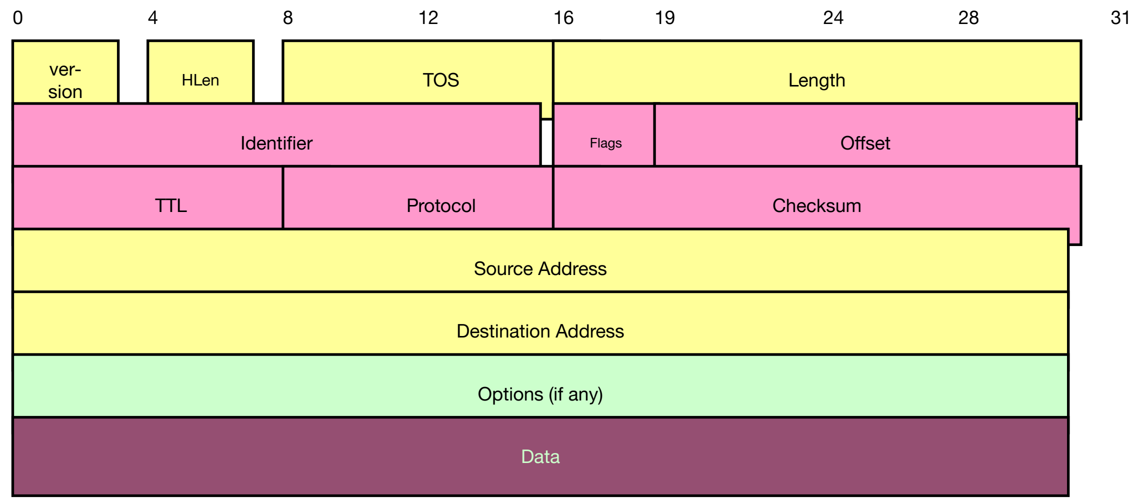
## IPv4 Header Fields



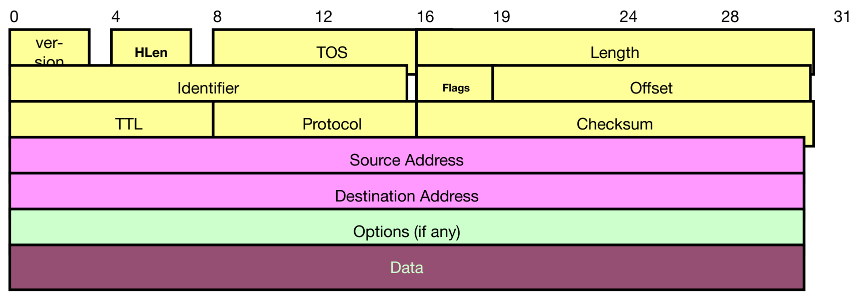
- Version: IP Version
  - 4 for IPv4
- HLen: Header Length
  - 32-bit words (typically 5)
- TOS: Type of Service
  - Priority information
- Length: Packet Length
  - Bytes (including header)
- Header format can change with versions
  - First byte identifies version
- Length field limits packets to 65,535 bytes
  - In practice, break into much smaller packets for network performance considerations

# IPv4 Header Fields

- Identifier, flags, fragment offset are used primarily for fragmentation
- Time to live
  - Must be decremented at each router
  - Packets with TTL=0 are thrown away
  - Ensure packets exit the network
- Protocol
  - Demultiplexing to higher layer protocols
  - TCP = 6, ICMP = 1, UDP = 17…
- Header checksum
  - Ensures some degree of header integrity
  - Relatively weak – 16 bit
- Options
  - E.g. Source routing, record route, etc.
  - Performance issues
    - Poorly supported



# IPv4 Header Fields



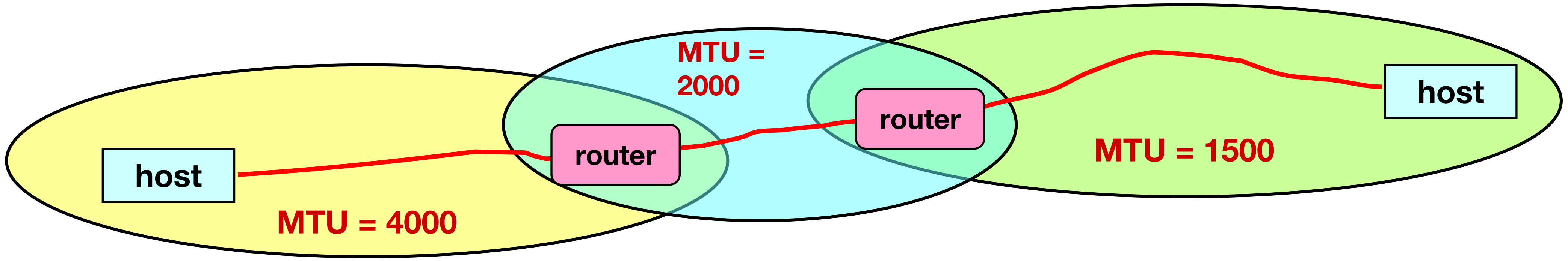
- Source Address
  - 32-bit IP address of sender
- Destination Address
  - 32-bit IP address of destination

- Like the addresses on an envelope
- Globally unique identification of sender & receiver

# IP Delivery Model

- *Best effort service*
  - Network will do its best to get packet to destination
- Does NOT guarantee:
  - Any maximum latency or even ultimate success
  - Sender will be informed if packet doesn't make it
  - Packets will arrive in same order sent
  - Just one copy of packet will arrive
- Implications
  - Scales very well
  - Higher level protocols must make up for shortcomings
    - Reliably delivering ordered sequence of bytes □ TCP
  - Some services not feasible
    - Latency or bandwidth guarantees

# IP Fragmentation



- Every Network has Own Maximum Transmission Unit (MTU)
  - Largest IP datagram it can carry within its own packet frame
    - E.g., Ethernet is 1500 bytes
  - Don’t know MTUs of all intermediate networks in advance
- IP Solution
  - When hit network with small MTU, fragment packets
    - Might get further fragmentation as proceed farther

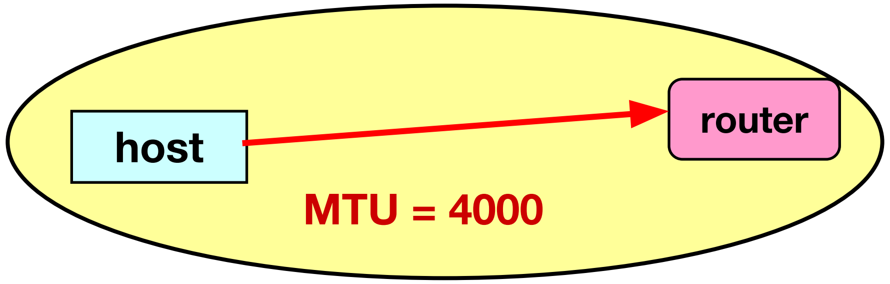
## Reassembly

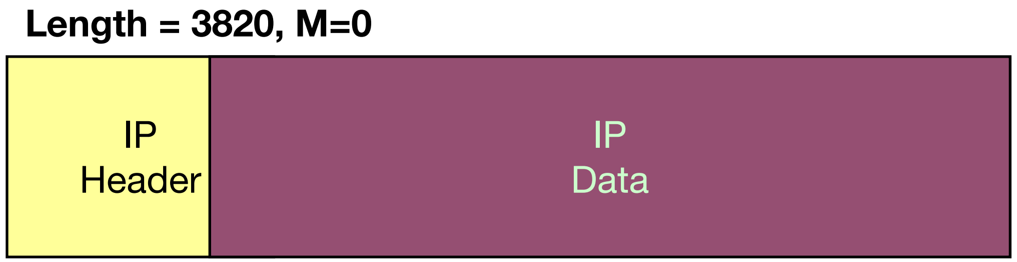
- Where to do reassembly?
  - End nodes or at routers?
- End nodes
  - If any fragment missing, delete entire packet
- Dangerous to do at intermediate nodes
  - How much buffer space required at routers?
  - What if routes in network change?
    - Multiple paths through network
    - All fragments only required to go through destination

## Fragmentation Related Fields

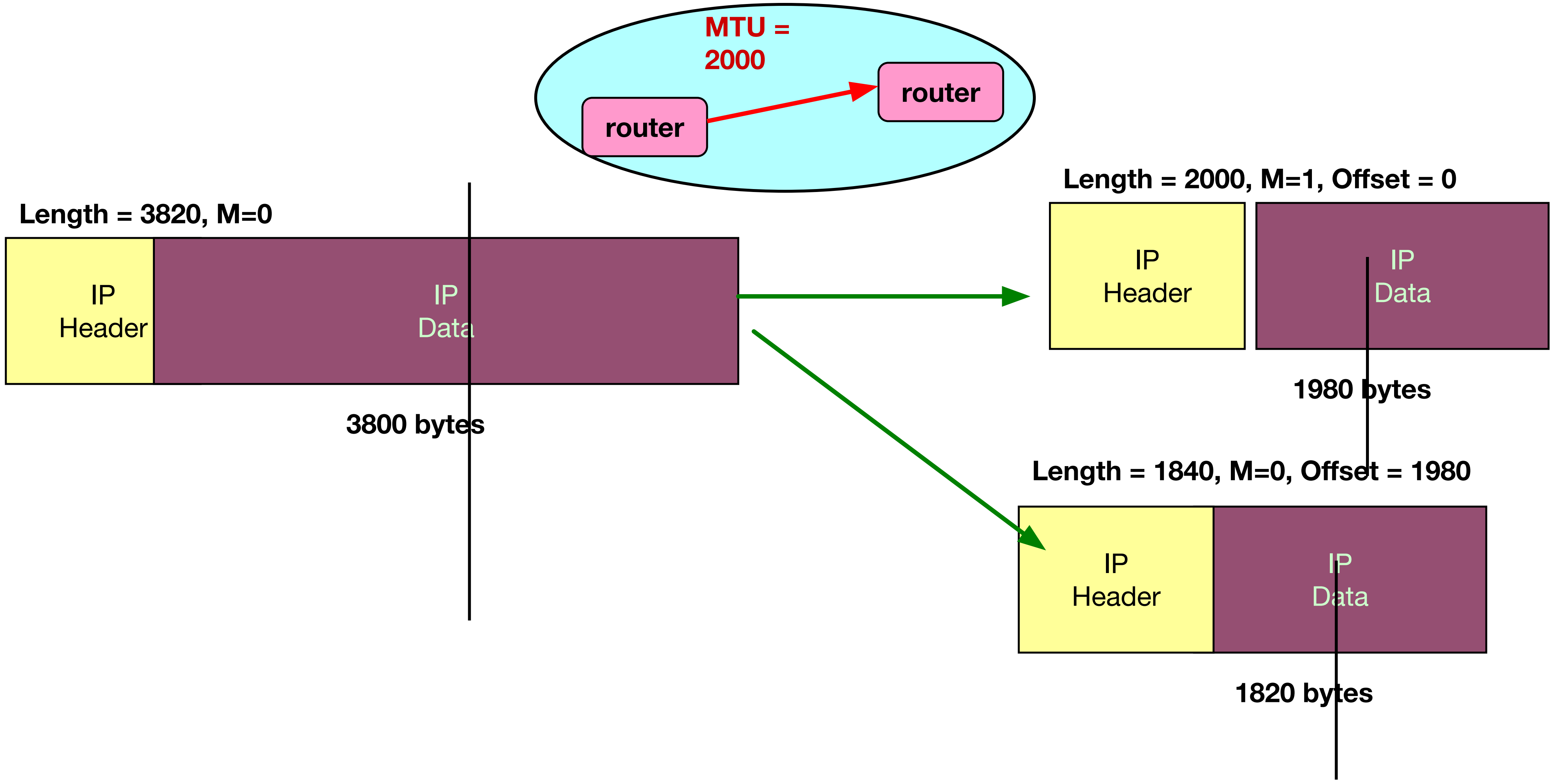
- Length
  - Length of IP fragment
- Identification
  - To match up with other fragments
- Flags
  - Don’t fragment flag
  - More fragments flag
- Fragment offset
  - Where this fragment lies in entire IP datagram
  - Measured in 8 octet units (13 bit field)

# IP Fragmentation Example #1

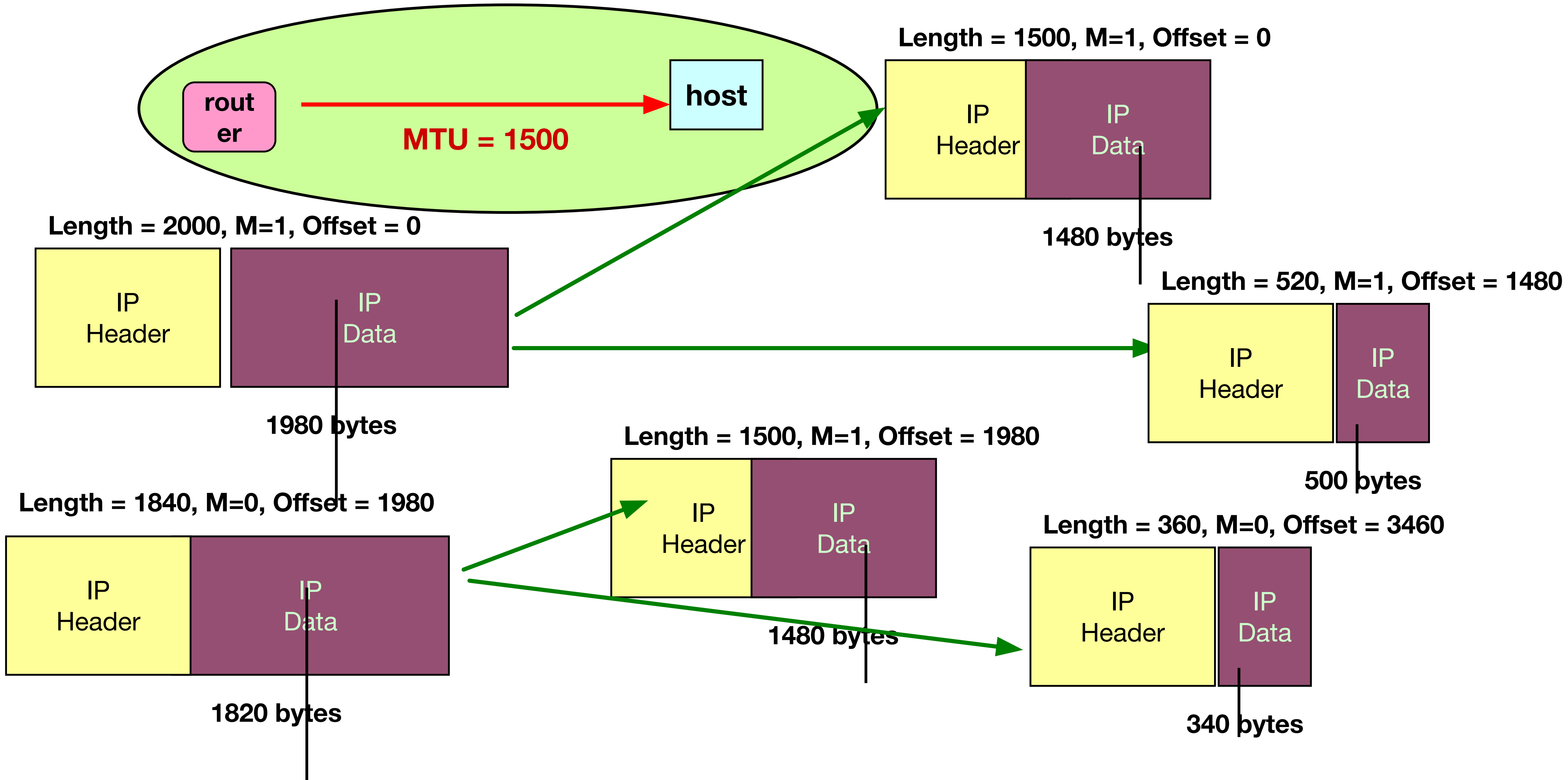




# IP Fragmentation Example #2

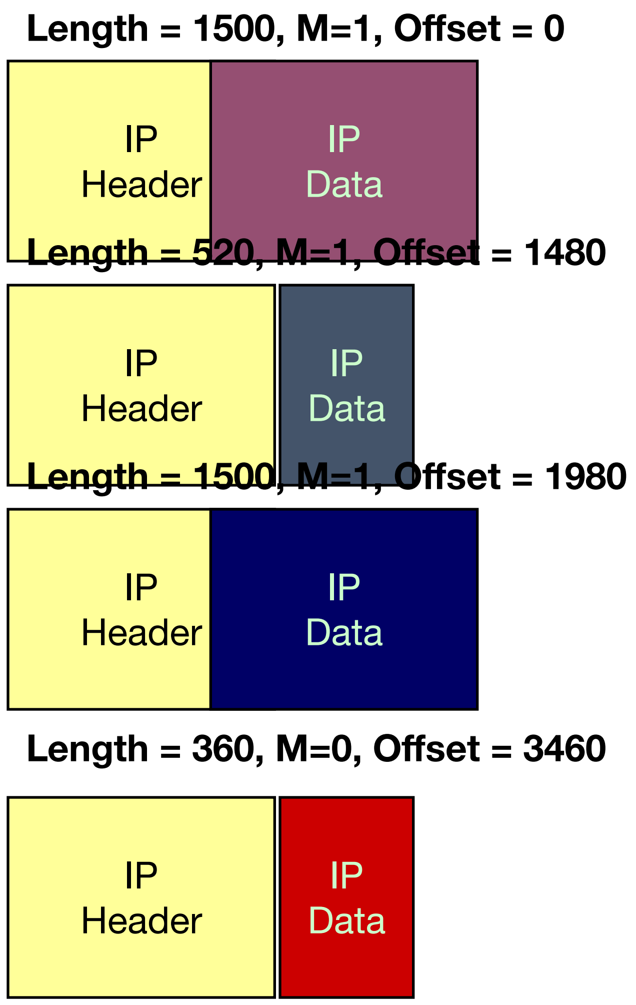


# IP Fragmentation Example #3

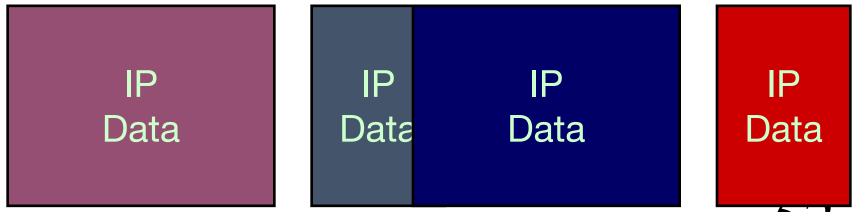


rout er
host
MTU = 1500
Length = 2000, M=1, Offset = 0
IP Header
IP Data
1980 bytes
Length = 1500, M=1, Offset = 0
IP Header
IP Data
1480 bytes
Length = 520, M=1, Offset = 1480
IP Header
IP Data
500 bytes
Length = 1840, M=0, Offset = 1980
IP Header
IP Data
1820 bytes
Length = 1500, M=1, Offset = 1980
IP Header
IP Data
1480 bytes
Length = 360, M=0, Offset = 3460
IP Header
IP Data
340 bytes

# IP Reassembly


Length = 1500, M=1, Offset = 0
IP
Header
IP
Data
Length = 520, M=1, Offset = 1480
IP
Header
IP
Data
Length = 1500, M=1, Offset = 1980
IP
Header
IP
Data
Length = 360, M=0, Offset = 3460
IP
Header
IP
Data

- Fragments might arrive out-of-order
  - Don't know how much memory required until receive final fragment
- Some fragments may be duplicated
  - Keep only one copy
- Some fragments may never arrive
  - After a while, give up entire process



## Fragmentation and Reassembly Concepts

- Demonstrates many Internet concepts
- Decentralized
  - Every network can choose MTU
- Connectionless
  - Each (fragment of) packet contains full routing information
  - Fragments can proceed independently and along different routes
- Best effort
  - Fail by dropping packet
  - Destination can give up on reassembly
  - No need to signal sender that failure occurred
- Complex endpoints and simple routers
  - Reassembly at endpoints

## Fragmentation is Harmful

- Uses resources poorly
  - Forwarding costs per packet
  - Best if we can send large chunks of data
  - Worst case: packet just bigger than MTU
- Poor end-to-end performance
  - Loss of a fragment

- Path MTU discovery protocol  determines minimum MTU along route
  - Uses ICMP error messages
- Common theme in system design
  - Assure correctness by implementing complete protocol
  - Optimize common cases to avoid full complexity