Network Layer

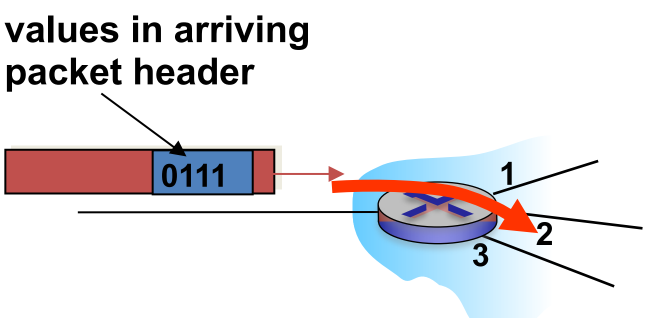
Note: The slides are adapted from the materials from Prof. Richard Han at CU Boulder and Profs. Jennifer Rexford and Mike Freedman at Princeton University, and the networking book (Computer Networking: A Top Down Approach) from Kurose and Ross.

Packet Forwarding

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



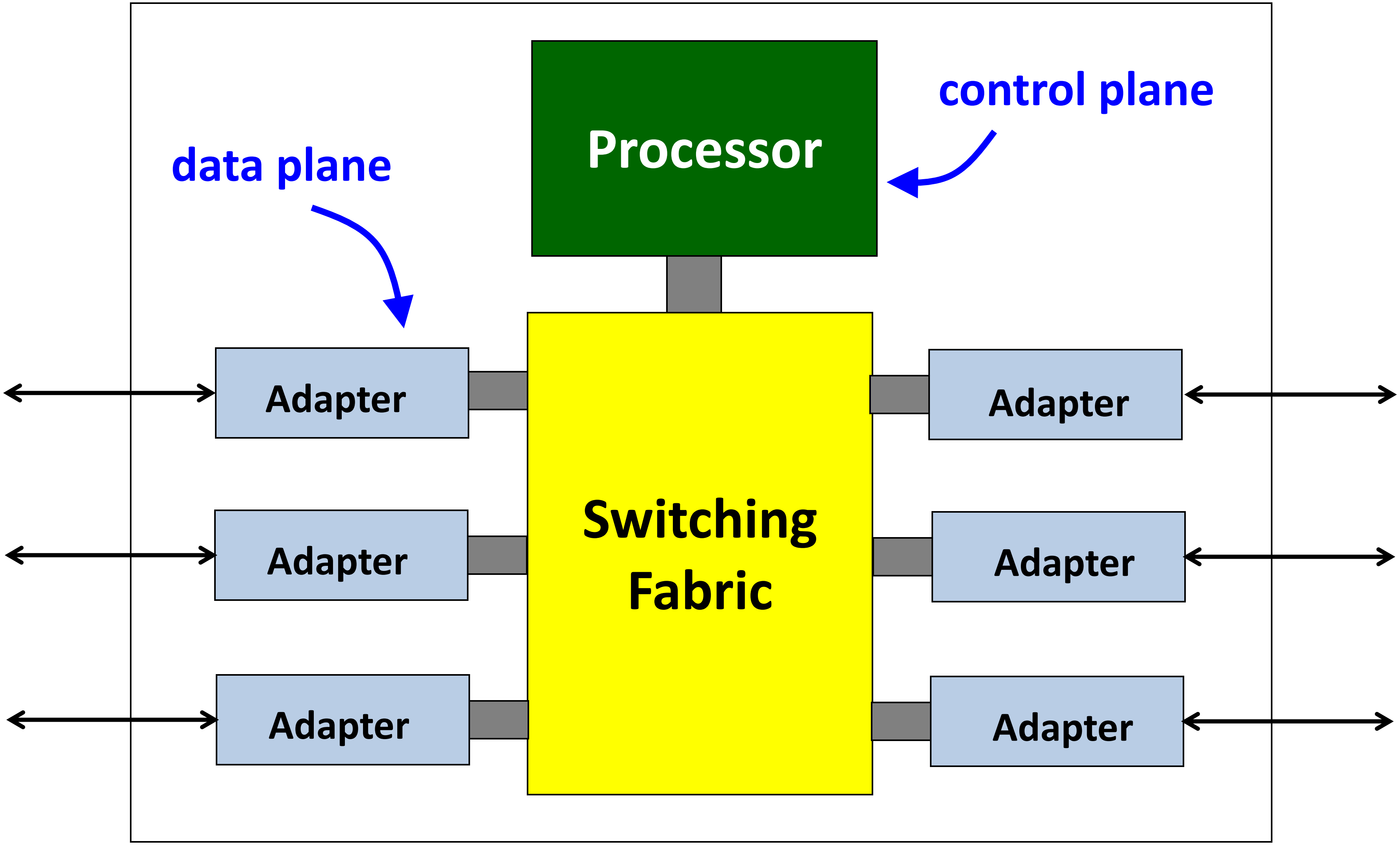
Control plane

- 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

Hop-by-Hop Packet Forwarding

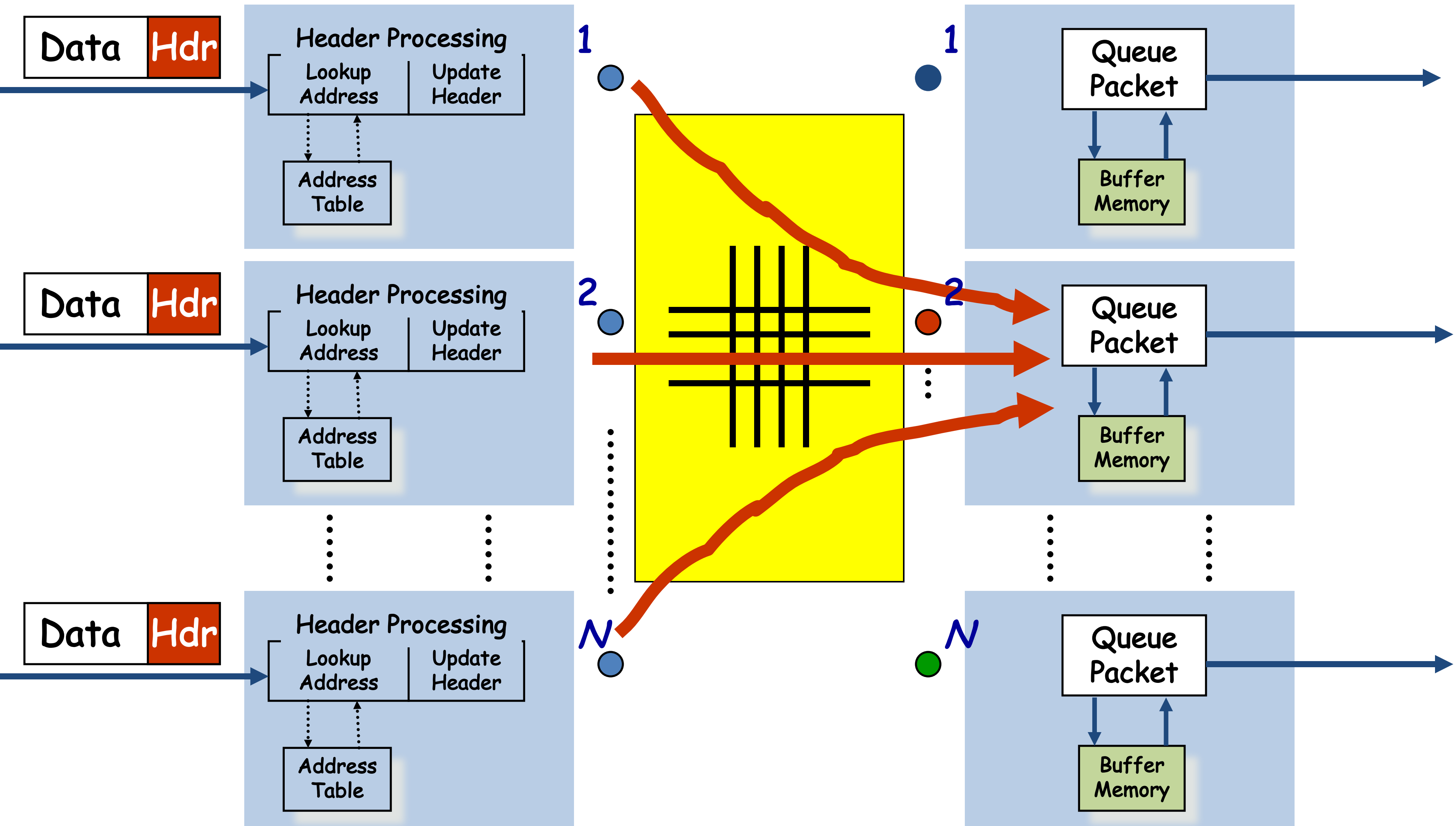
- Each router has a forwarding table
  - Maps destination address to outgoing interface
- Upon receiving a packet
  - Inspect the destination address in the header
  - Index into the table
  - Determine the outgoing interface
  - Forward the packet out that interface
- Then, the next router in the path repeats

IP Router



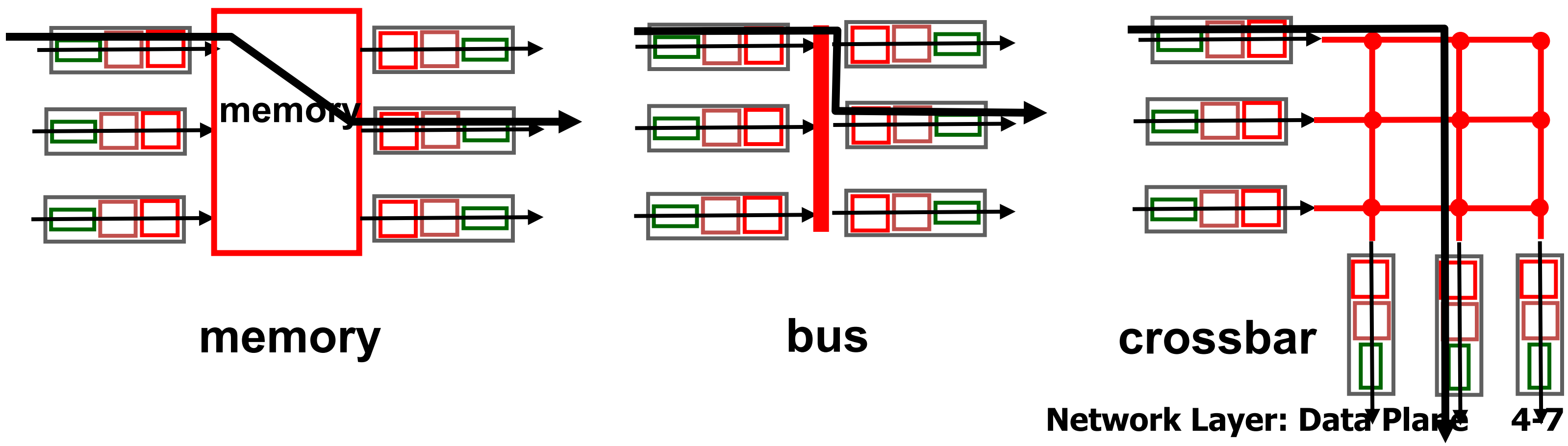
Processor
control plane
data plane
Adapter
Adapter
Adapter
Switching Fabric
Adapter
Adapter
Adapter

Switch Fabric: From Input to Output



Switching fabrics

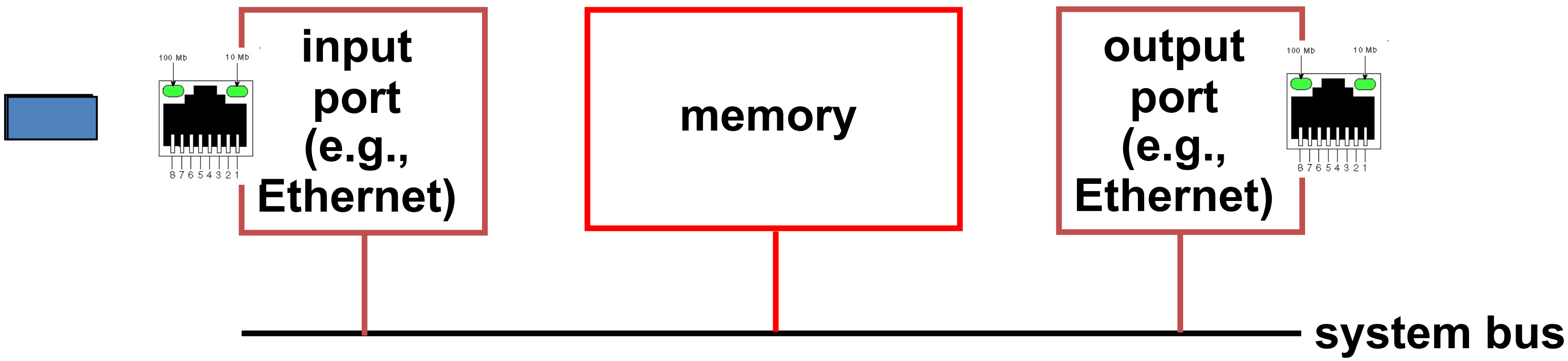
- transfer packet from input buffer to appropriate output buffer
- switching rate: rate at which packets can be transfer from inputs to outputs
  - often measured as multiple of input/output line rate
  - N inputs: switching rate N times line rate desirable
- three types of switching fabrics



Switching via memory

first generation routers:

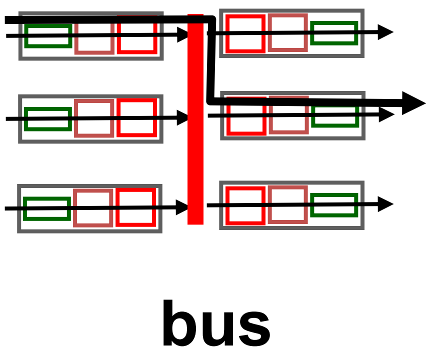
- traditional computers with switching under direct control of CPU
- packet copied to system's memory
- speed limited by memory bandwidth (2 bus crossings per datagram)



Switching via a bus

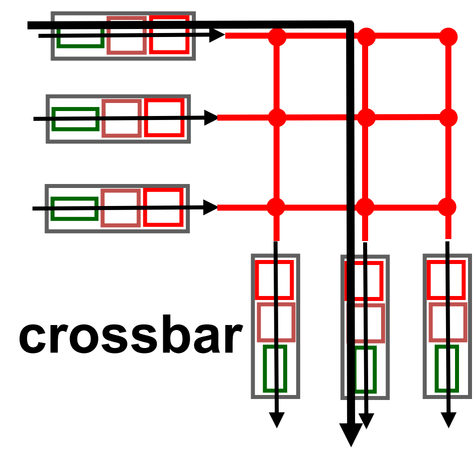
- datagram from input port memory

  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 and enterprise routers



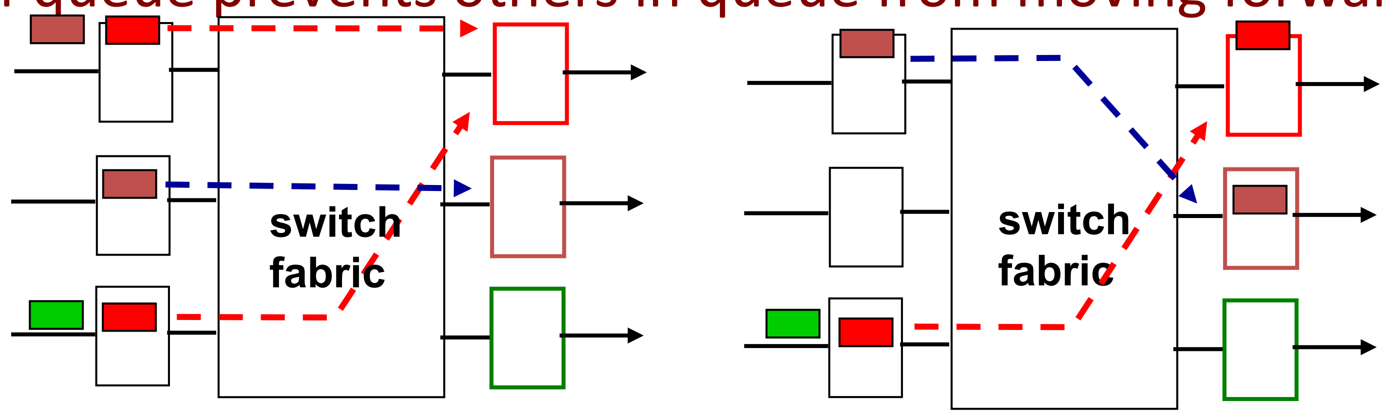
Switching via interconnection network

- overcome bus bandwidth limitations
- banyan networks, crossbar, other interconnection nets initially developed to connect processors in multiprocessor
- advanced design: fragmenting datagram into fixed length cells, switch cells through the fabric.
- Cisco 12000: switches 60 Gbps through the interconnection



Input port queuing

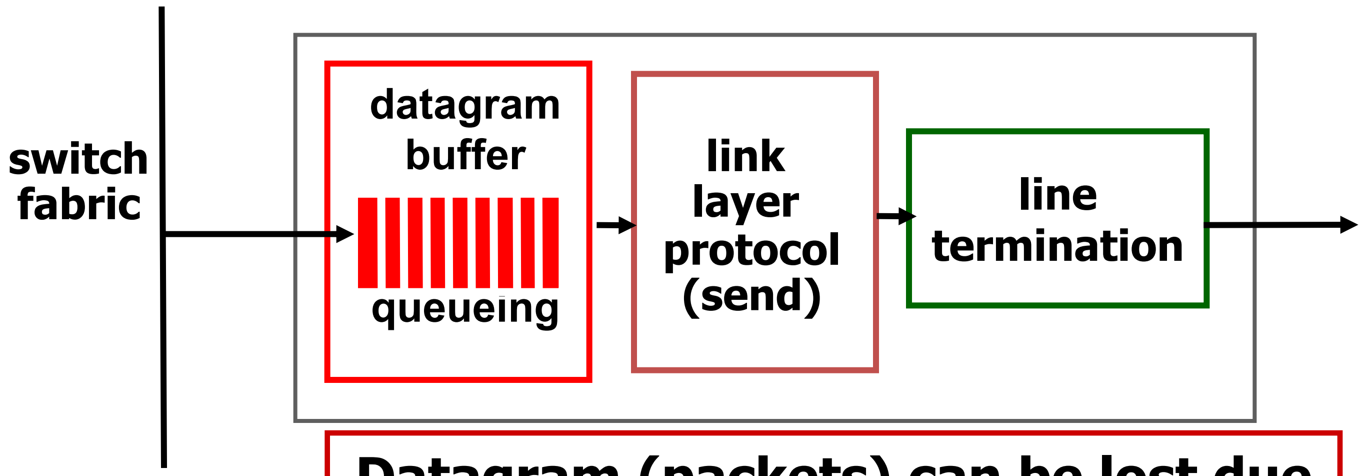
- fabric slower than input ports combined -> queueing may occur at input queues
  - *queueing delay and loss due to input buffer overflow!*
- Head-of-the-Line (HOL) blocking: queued datagram at front of queue prevents others in queue from moving forward



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

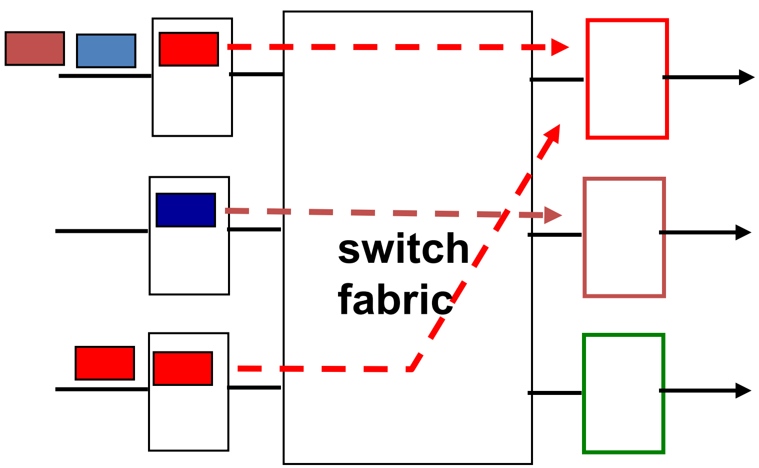


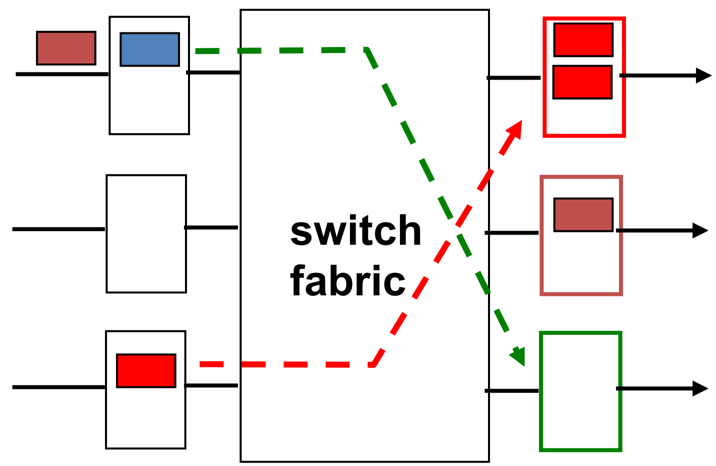
Datagram (packets) can be lost due to congestion, lack of buffers

- *buffering* required when datagrams arrive from fabric faster than the transmission rate
- *scheduling discipline* chooses among queued datagrams for transmission

Priority scheduling – who gets best performance, network neutrality

Output port queueing





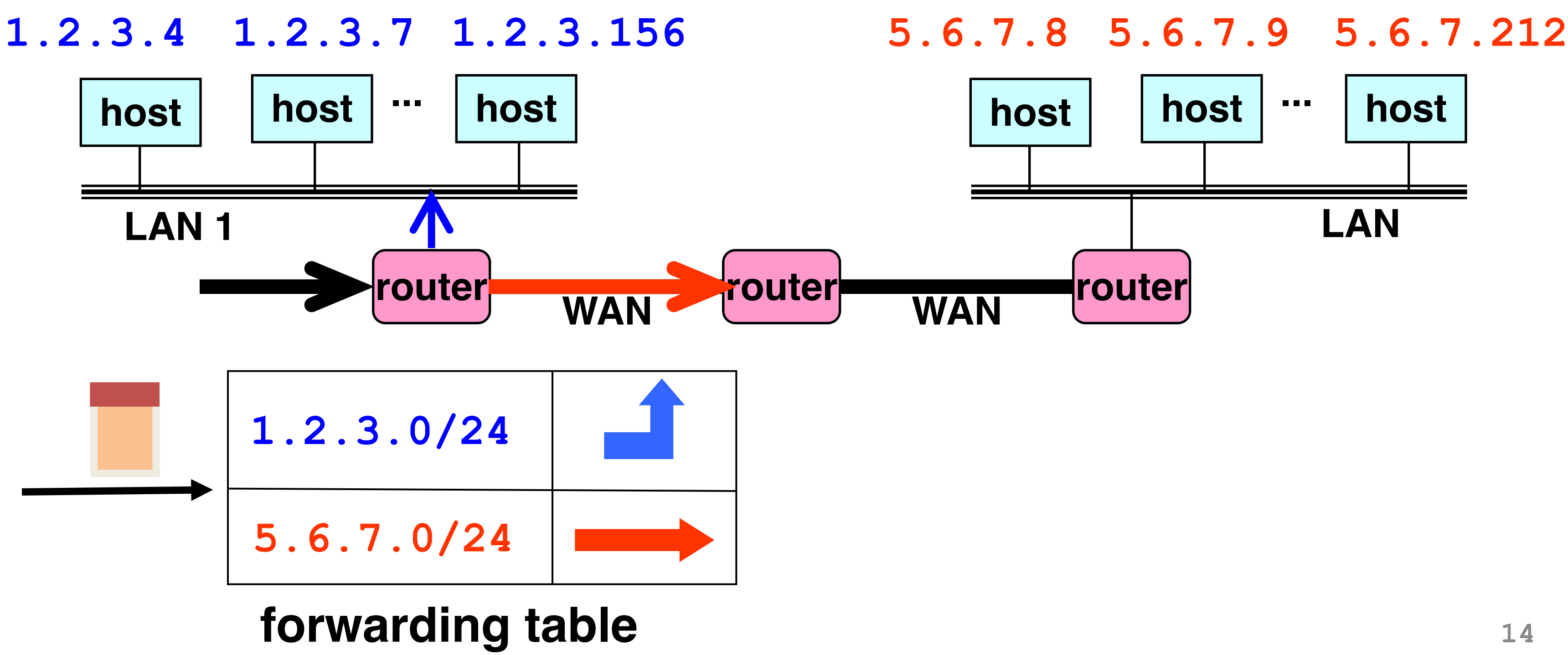
at *t,* packets more from input to output

one packet time later

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

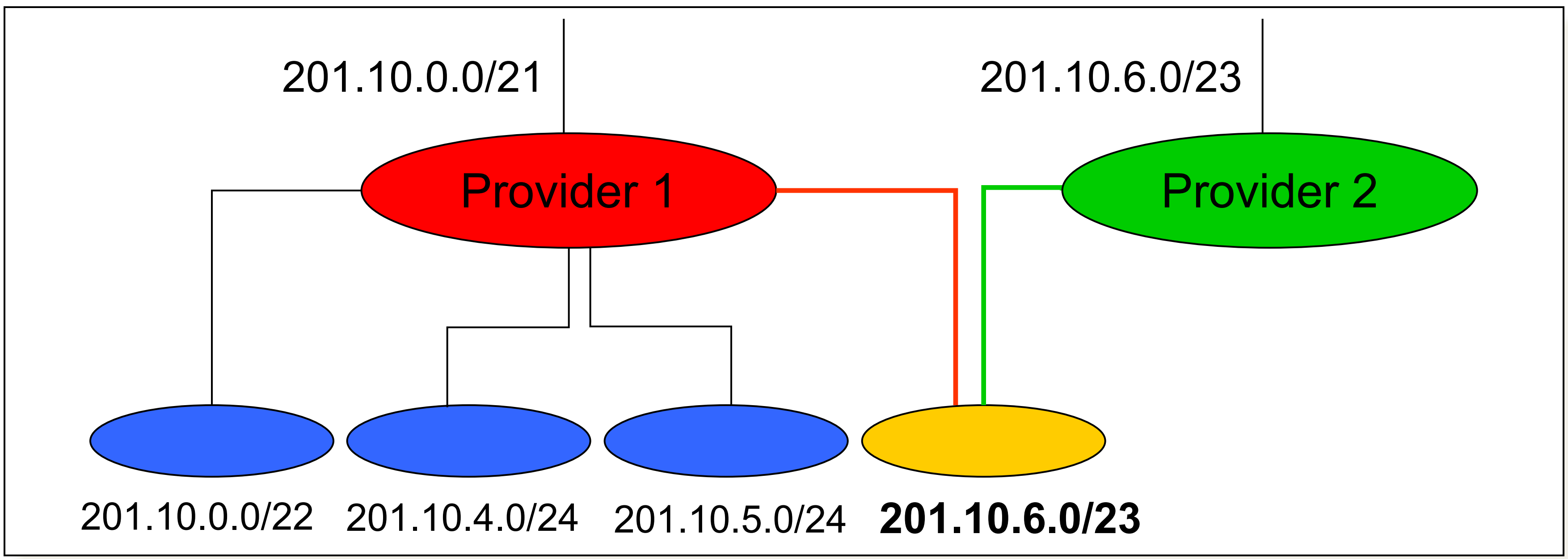
Separate Forwarding Entry Per Prefix

- Prefix-based forwarding
  - Map the destination address to matching prefix
  - Forward to the outgoing interface



CIDR Makes Packet Forwarding Harder

- Forwarding table may have many matches
  - E.g., entries for 201.10.0.0/21 and 201.10.6.0/23
  - The IP address 201.10.6.17 would match both!



Longest Prefix Match Forwarding

- Destination-based forwarding
  - Packet has a destination address
  - Router identifies longest-matching prefix
  - Cute algorithmic problem: very fast lookups

forwarding table

destination

201.10.6.17

4.0.0.0/8
4.83.128.0/17
201.10.0.0/21
201.10.6.0/23
126.255.103.0/24

outgoing link

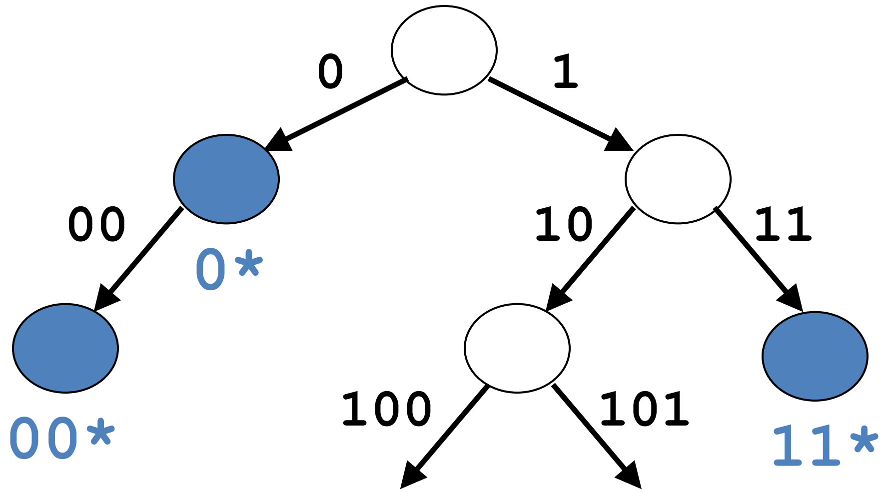
Serial0/0.1

Simplest Algorithm is Too Slow

- Scan the forwarding table one entry at a time
  - Keep track of entry with longest-prefix (by netmask)
- Overhead is linear in size of forwarding table
  - Today, that means 350,000 entries!
  - How much time do you have to process?
    - Consider 10Gbps routers and 64B packets
    - 10^{10} / 8 / 64: 19,531,250 packets per second
    - 51 nanoseconds per packet
- Need greater efficiency to keep up with *line rate*
  - Better algorithms
  - Hardware implementations

Patricia Tree (1968)

- Store prefixes as a tree
  - One bit for each level of tree
  - Some nodes correspond to valid prefixes
  - ... which have next-hop interfaces in a table
- When a packet arrives
  - Traverse tree based on destination address
  - Stop upon reaching longest matching prefix



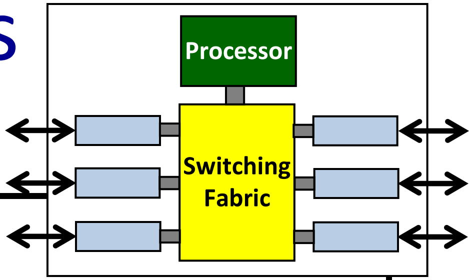
Even Faster Lookups

- Patricia tree is faster than linear scan
  - Proportional to number of bits in address
  - Speed-up further by time vs. space tradeoff
    - Each node in 4-ary tree has 4 children, cuts depth by half
- Still somewhat slow, major concern in mid-to-late 1990s
  - … after CIDR was introduced and LPM major bottleneck
  - Reintroduction of circuit switching via pre-established paths: individual paths named by labels added to packets (MPLS)
- Innovation of special hardware
  - Content Addressable Memories (CAMs): assoc. array in h/w
    - Compares key in parallel to each entry
  - Ternary CAMs (TCAMS): Stored data is 0, 1, <don't care>
    - Least sig. bits represented by <don't care> (netmask=0)

Creating a Forwarding Table

- Entries can be statically configured
  - E.g., "map 12.34.158.0/24 to Serial0/0.1"
- But, this doesn't adapt
  - To failures
  - To new equipment
  - To the need to balance load
- That is where the *control plane* comes in
  - Routing protocols

Data, Control, & Management Planes



	Data	Control	Management
Time-scale	Packet (ns)	Event (10 ms to sec)	Human (min to hours)
Tasks	Forwarding, buffering, filtering, scheduling	Routing, signaling	Analysis, configuration
Location	Line-card hardware	Router software	Humans or scripts

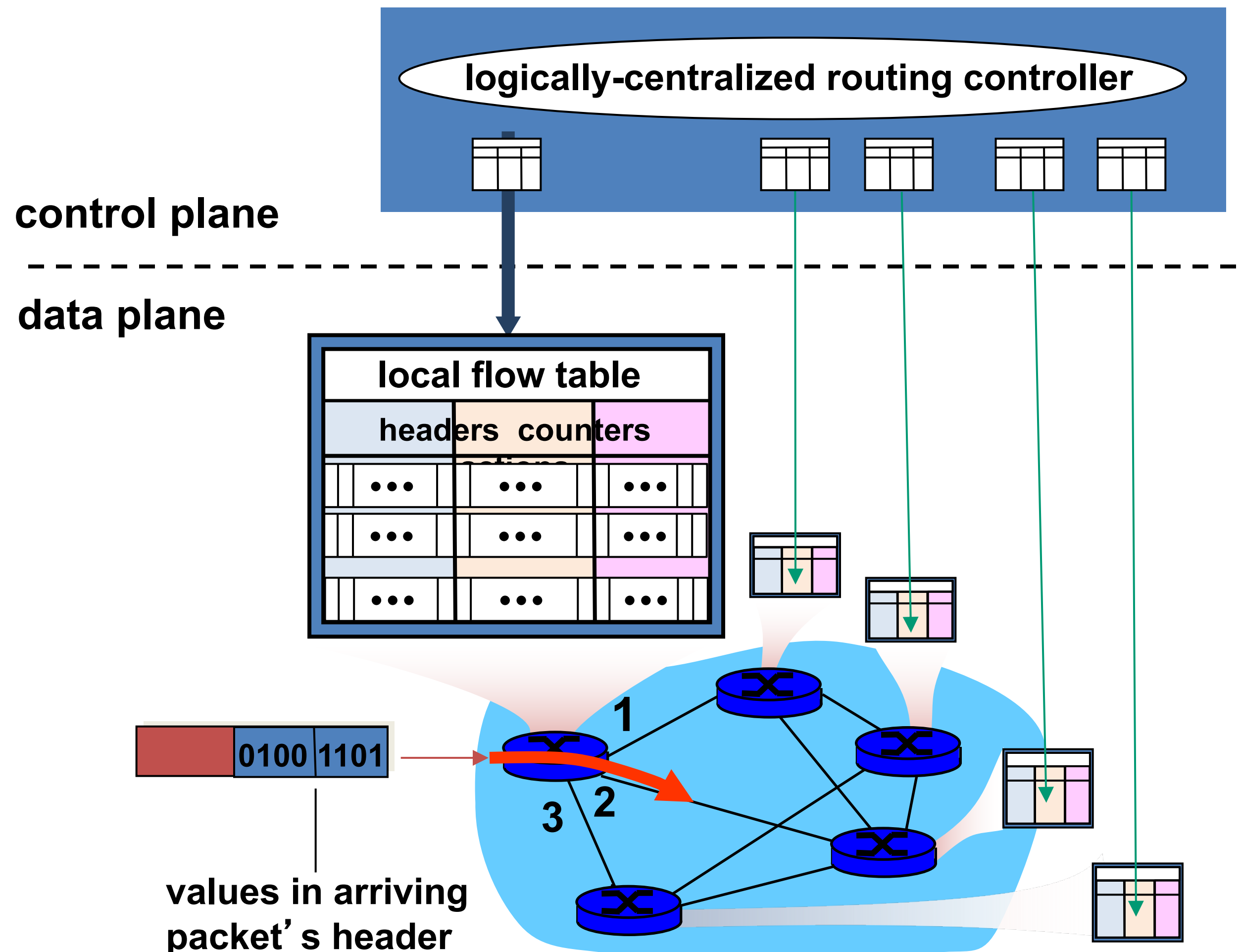
Q's: MAC vs. IP Addressing

- Hierarchically allocated
  - A) MAC B) IP C) Both D) Neither
- Organized topologically
  - A) MAC B) IP C) Both D) Neither
- Forwarding via exact match on address
  - A) MAC B) IP C) Both D) Neither
- Automatically calculate forwarding by observing data
  - A) Ethernet switches B) IP routers C) Both D) Neither
- Per connection state in the network
  - A) MAC B) IP C) Both D) Neither
- Per host state in the network
  - A) MAC B) IP C) Both D) Neither

Software Defined Networking (SDN)

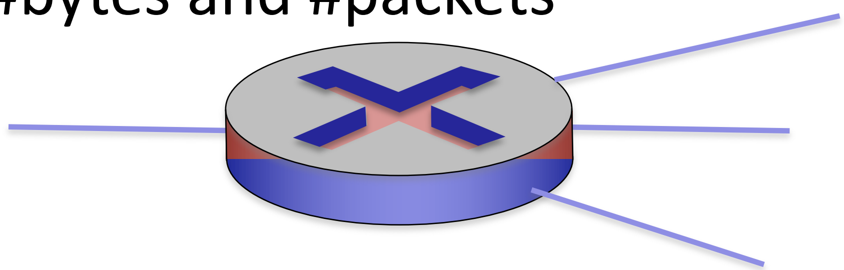
Generalized Forwarding and SDN

Each router contains a *flow table* that is computed and distributed by a *logically centralized* routing controller



OpenFlow data plane abstraction

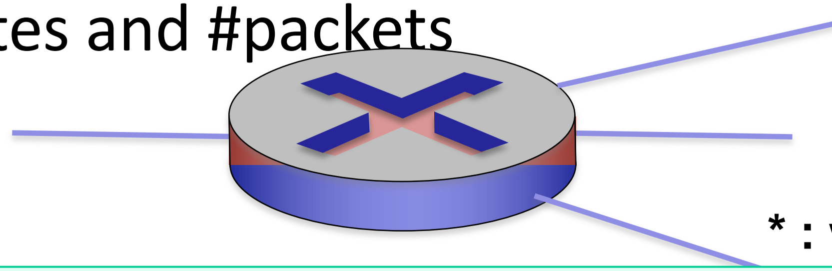
- *flow*: defined by header fields
- generalized forwarding: simple packet-handling rules
  - *Pattern:* match values in packet header fields
  - *Actions: for matched packet:* drop, forward, modify, matched packet or send matched packet to controller
  - *Priority*: disambiguate overlapping patterns
  - *Counters:* #bytes and #packets



Flow table in a router (computed and distributed by controller) define router's match+action rules

OpenFlow data plane abstraction

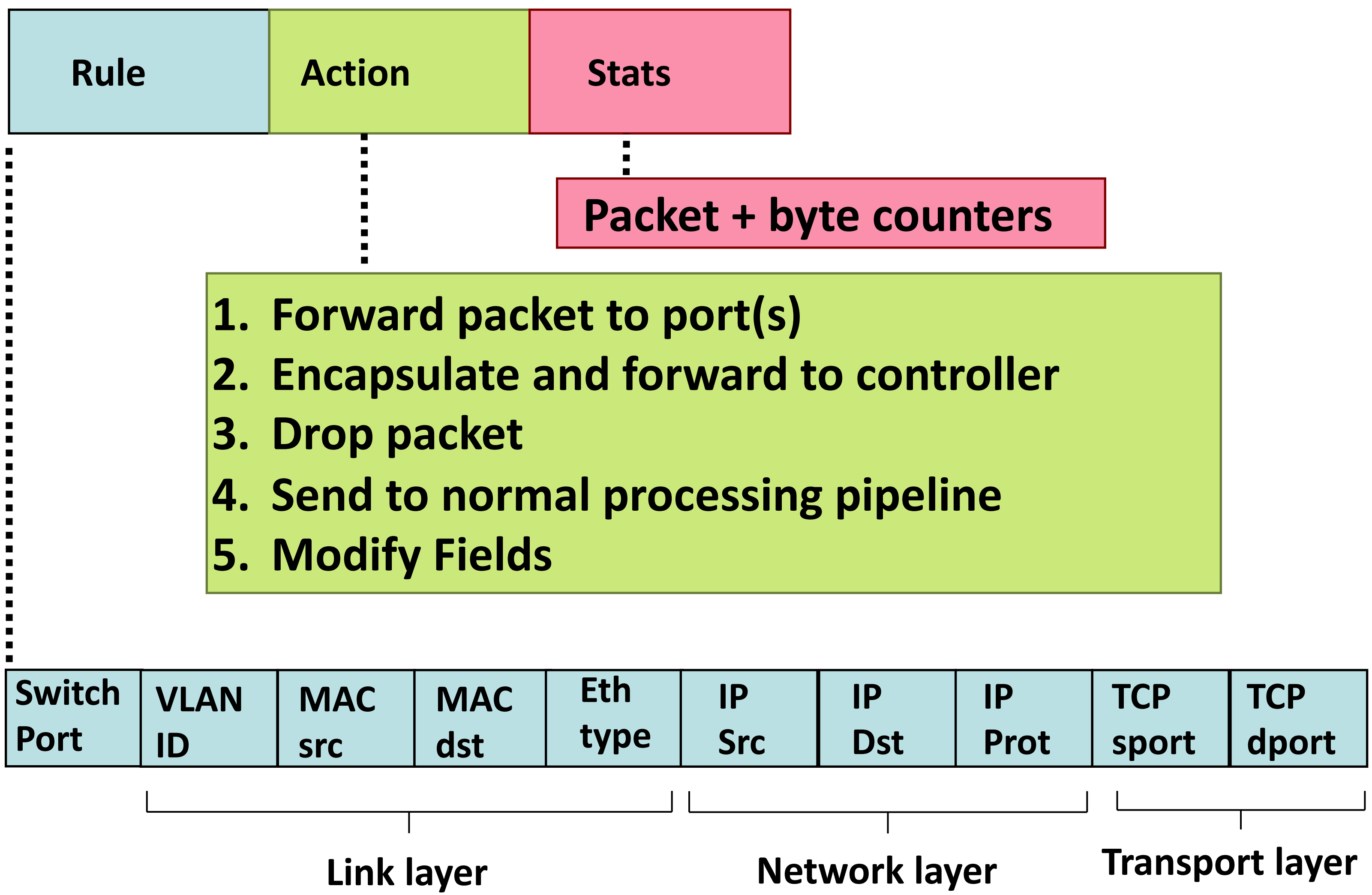
- *flow*: defined by header fields
- generalized forwarding: simple packet-handling rules
  - *Pattern:* match values in packet header fields
  - *Actions: for matched packet:* drop, forward, modify, matched packet or send matched packet to controller
  - *Priority*: disambiguate overlapping patterns
  - *Counters:* #bytes and #packets



*** : wildcard**

1. src=1.2.*.*, dest=3.4.5.* → drop
2. src = *.*.*.*, dest=3.4.*.* → forward(2)
3. src=10.1.2.3, dest=*.*.*.* → send to controller

OpenFlow: Flow Table Entries



Examples

Destination-based forwarding:

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	IP Src	IP Dst	IP Prot	TCP sport	TCP dport	Action
*	*	*	*	*	*	51.6.0.8	*	*	*	port6

IP datagrams destined to IP address 51.6.0.8 should be forwarded to router output port 6

Firewall:

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	IP Src	IP Dst	IP Prot	TCP sport	TCP dport	Forward
*	*	*	*	*	*	*	*	*	22	drop

do not forward (block) all datagrams destined to TCP port 22

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	IP Src	IP Dst	IP Prot	TCP sport	TCP dport	Forward
*	*	*	*	*	128.119.1.1	*	*	*	*	drop

do not forward (block) all datagrams sent by host 128.119.1.1

Examples

Destination-based layer 2 (switch) forwarding:

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	IP Src	IP Dst	IP Prot	TCP sport	TCP dport	Action
*	22:A7:23: 11:E1:02	*	*	*	*	*	*	*	*	port6

layer 2 frames from MAC address 22:A7:23:11:E1:02 should be forwarded to output port 6

OpenFlow abstraction

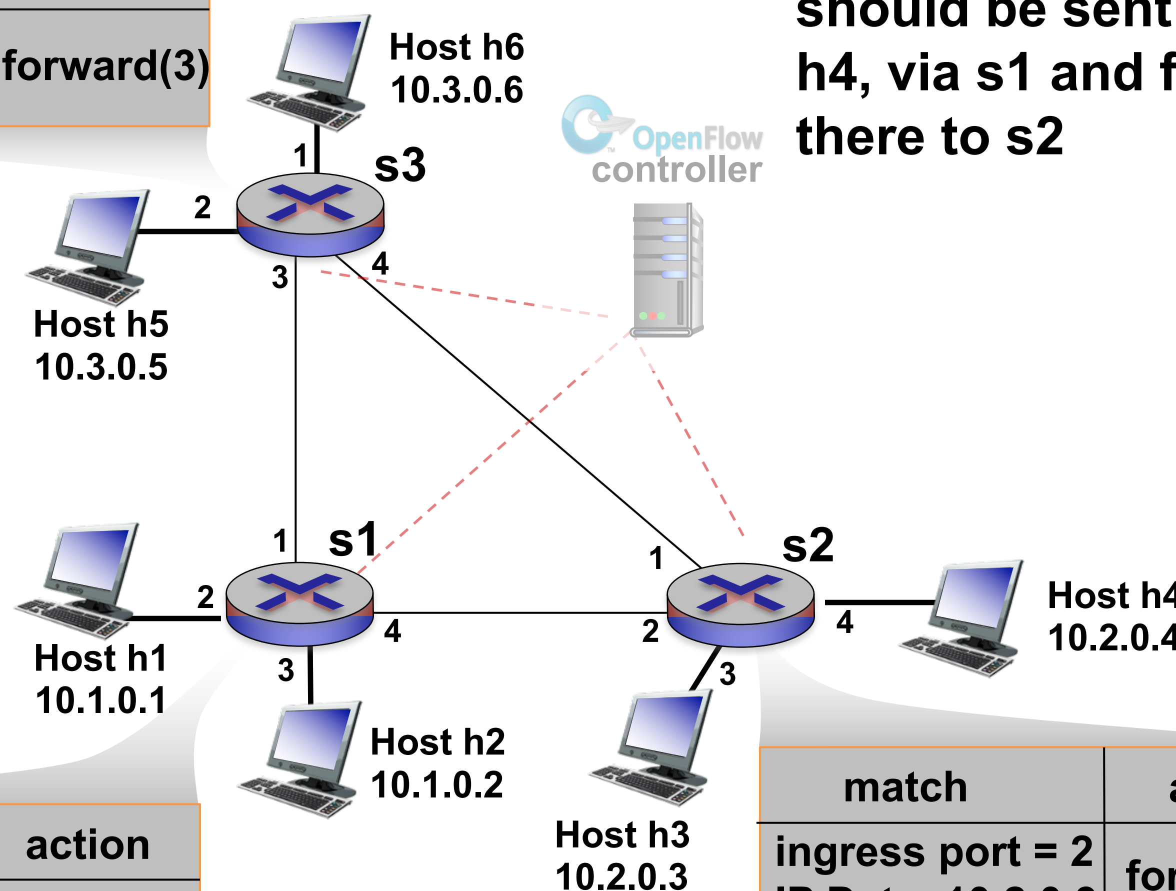
- ***match+action:*** **unifies different kinds of devices**

- Router
  - *match:* longest destination IP prefix
  - *action:* forward out a link
- Switch
  - *match:* destination MAC address
  - *action:* forward or flood
- Firewall
  - *match*: IP addresses and TCP/UDP port numbers
  - *action:* permit or deny
- NAT
  - *match:* IP address and port
  - *action:* rewrite address and port

OpenFlow example

Example: **datagrams from hosts h5 and h6 should be sent to h3 or h4, via s1 and from there to s2**

match	action
IP Src = 10.3.*.* IP Dst = 10.2.*.*	forward(3)

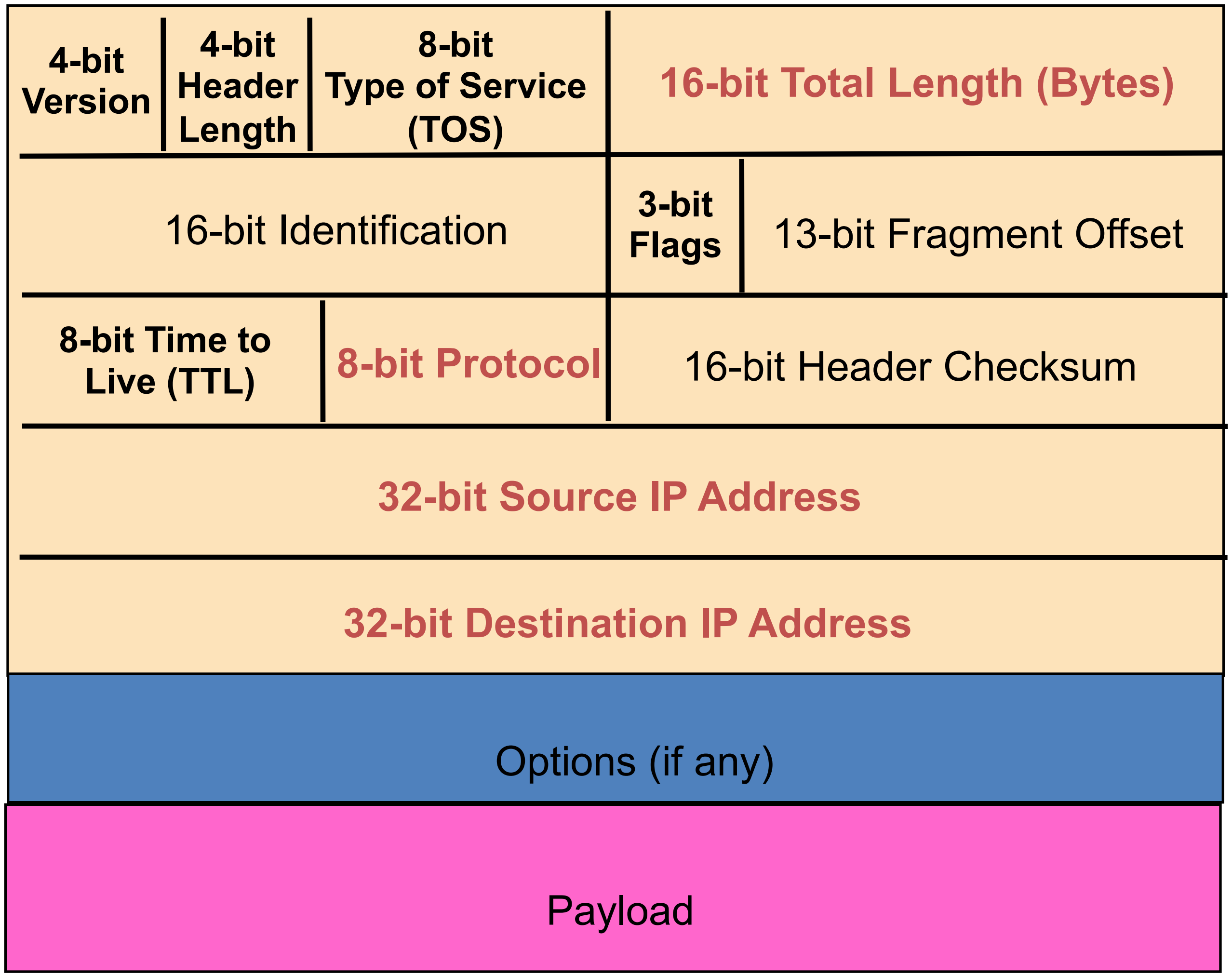


match	action
ingress port = 1 IP Src = 10.3.*.* IP Dst = 10.2.*.*	forward(4)

match	action
ingress port = 2 IP Dst = 10.2.0.3	forward(3)
ingress port = 2 IP Dst = 10.2.0.4	forward(4)

IP Packet Format

IP Packet Structure



IP Header: Version, Length, ToS

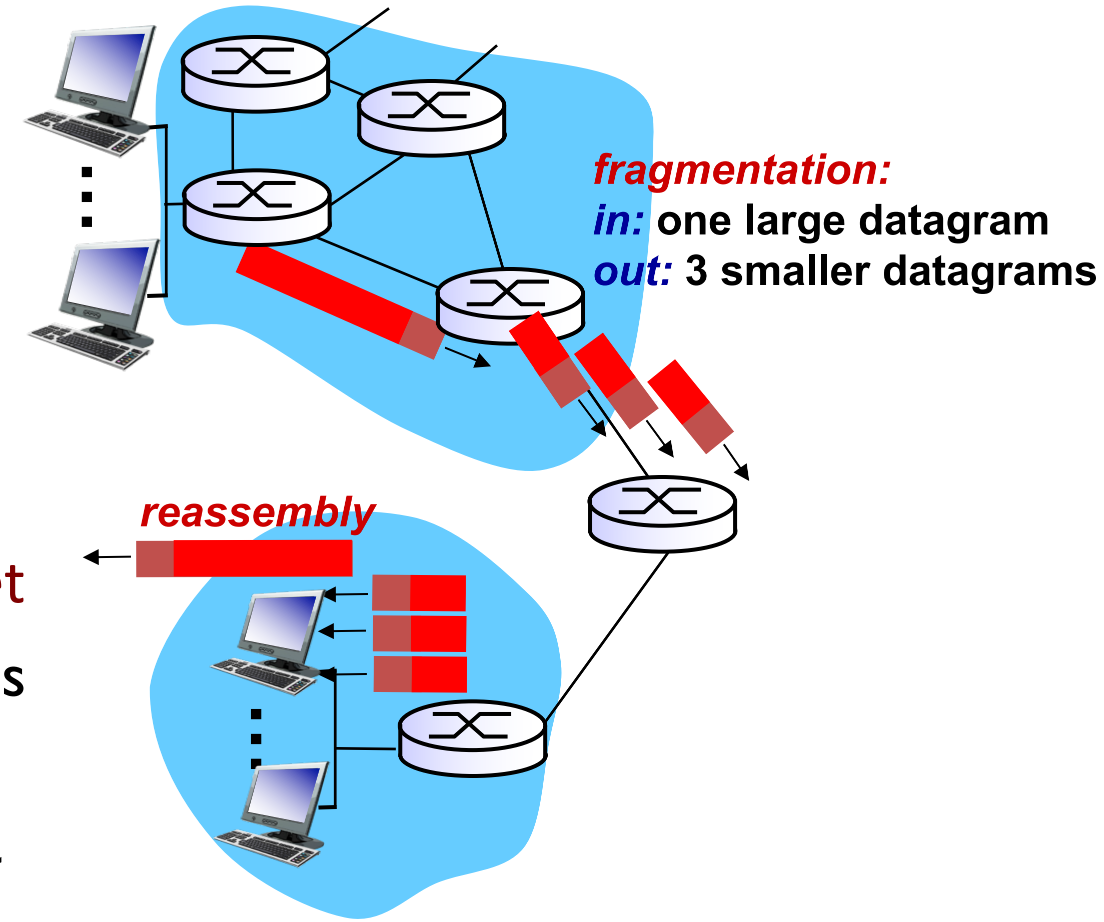
- Version number (4 bits)
  - Necessary to know what other fields to expect
  - Typically "4" (for IPv4), and sometimes "6" (for IPv6)
- Header length (4 bits)
  - Number of 32-bit words in the header
  - Typically "5" (for a 20-byte IPv4 header)
  - Can be more when "IP options" are used
- Type-of-Service (8 bits)
  - Allow different packets to be treated differently
  - Low delay for audio, high bandwidth for bulk transfer

IP Header: Length, Fragments, TTL

- Total length (16 bits)
  - Number of bytes in the packet
  - Max size is 63,535 bytes (2^{16} -1)
  - … though most links impose smaller limits
- Fragmentation information (32 bits)
  - Supports dividing a large IP packet into fragments
  - … in case a link cannot handle a large IP packet
- Time-To-Live (8 bits)
  - Used to identify packets stuck in forwarding loops
  - … and eventually discard them from the network

IP fragmentation, reassembly

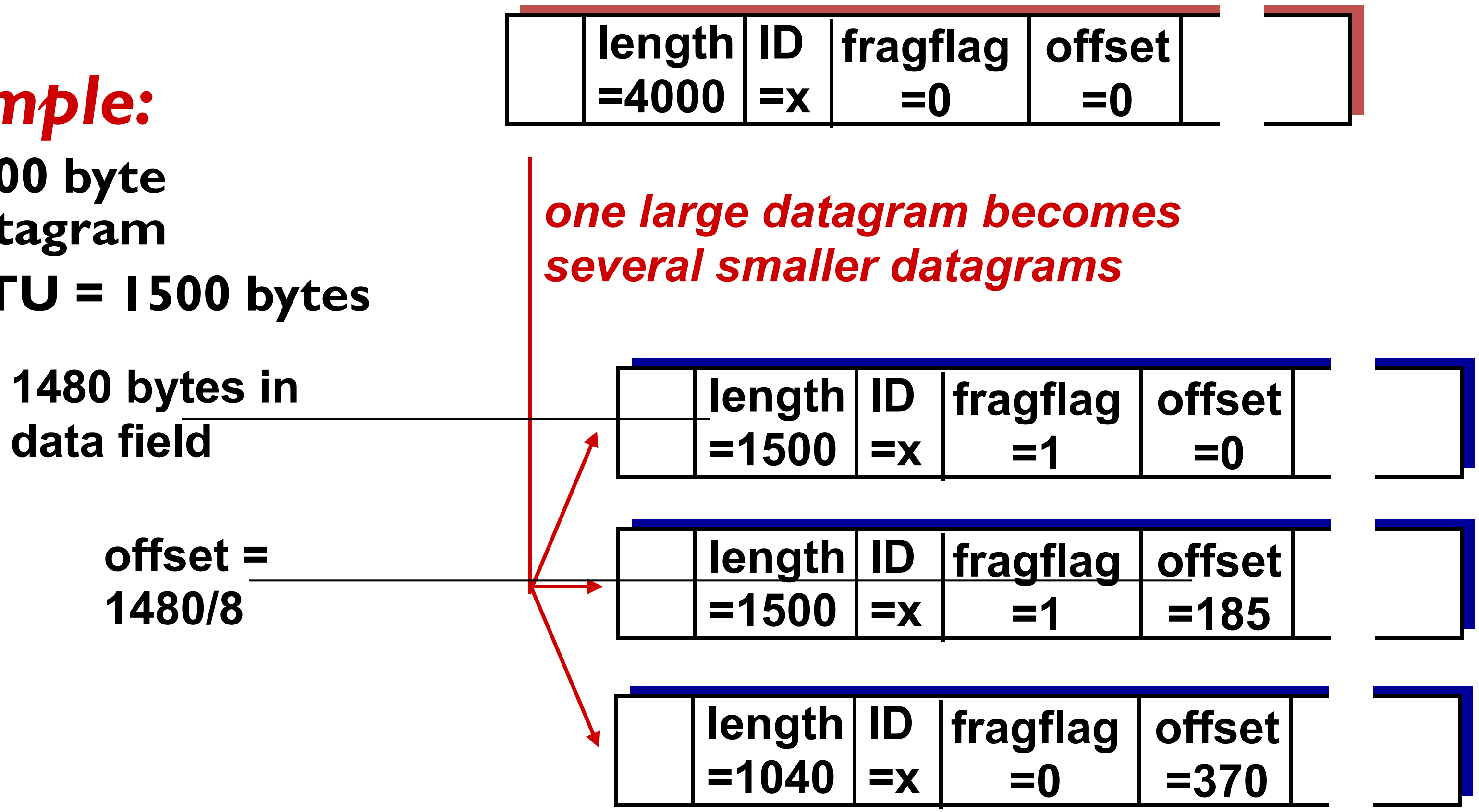
- network links have MTU (max.transfer size) - largest possible link-level frame
  - different link types, different MTUs
- large IP datagram divided ("fragmented") within net
  - one datagram becomes several datagrams
  - "reassembled" only at final destination
  - IP header bits used to identify, order related



IP fragmentation, reassembly

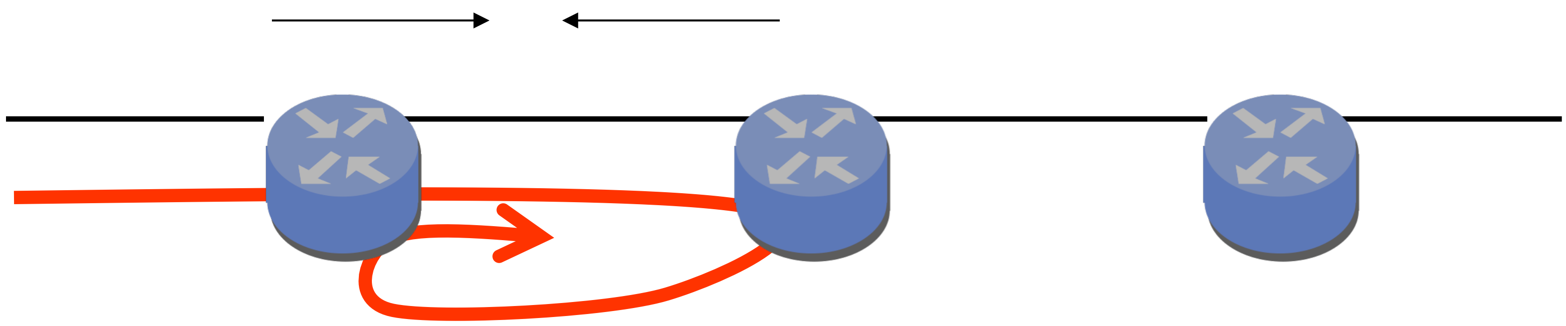
example:

- **4000 byte datagram**
- **MTU = 1500 bytes**



IP Header: More on Time-to-Live (TTL)

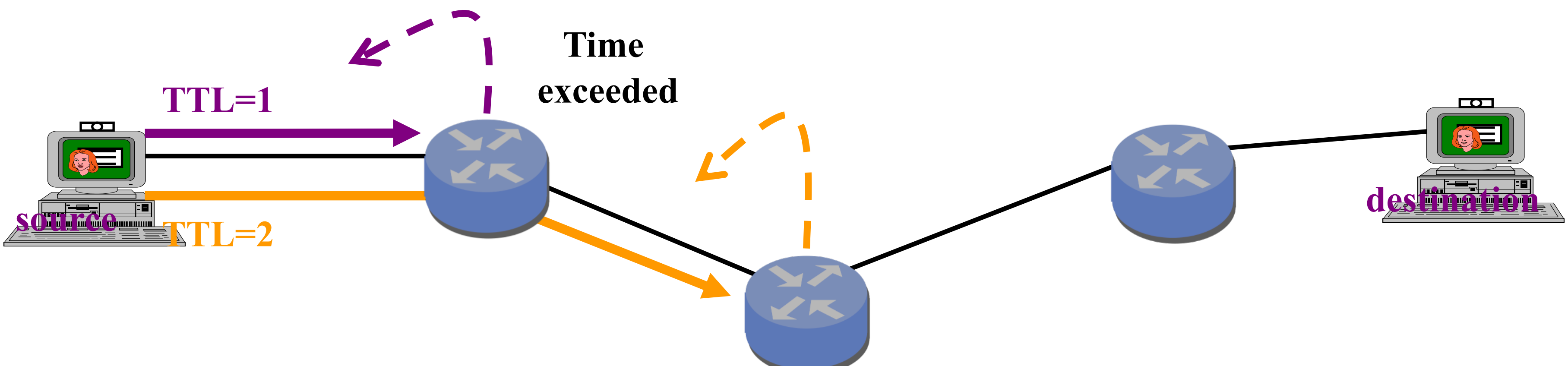
- Potential robustness problem
  - Forwarding loops can cause packets to cycle forever
  - Confusing if the packet arrives much later



- Time-to-live field in packet header
  - TTL field decremented by each router on path
  - Packet is discarded when TTL field reaches 0…
  - …and "time exceeded" message (ICMP) sent to source

IP Header: Use of TTL in Traceroute

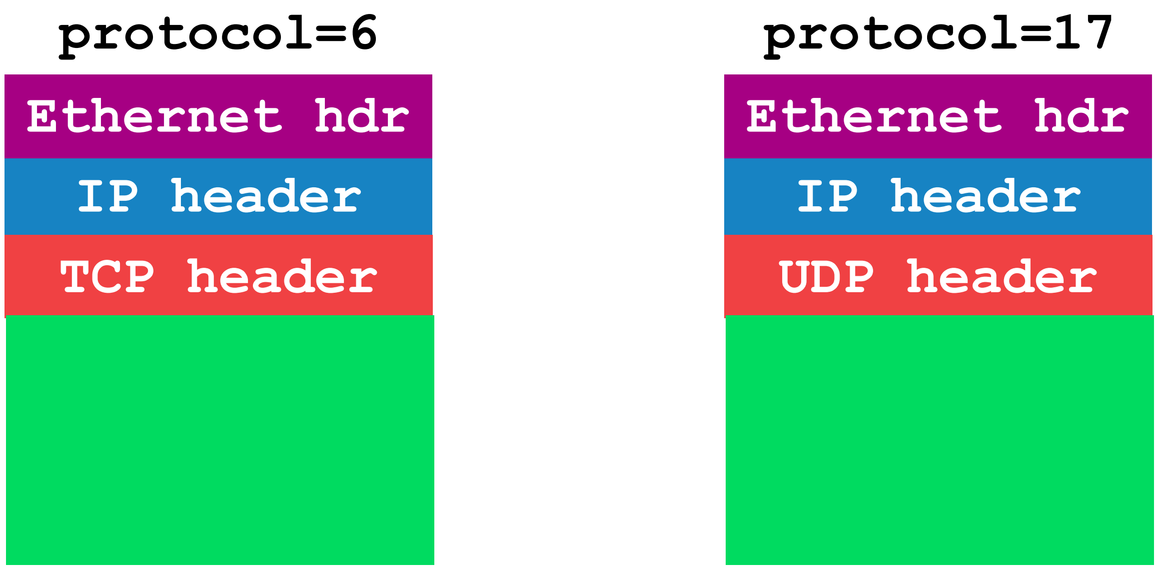
- Time-To-Live field in IP packet header
  - Source sends a packet with a TTL of *n*
  - Each router along the path decrements the TTL
  - "TTL exceeded" sent when TTL reaches *0*
- Traceroute tool exploits this TTL behavior



Send packets with TTL=1, 2, …
and record source of "time exceeded" message

IP Header: Transport Protocol

- Protocol (8 bits)
  - Identifies the higher-level protocol
    - E.g., "6" for TCP, "17" for UDP
  - Important for demultiplexing at receiving host
    - Indicates what kind of header to expect next



IP Header: Header Checksum

- Checksum (16 bits)
  - Sum of all 16-bit words in the header
  - If header bits are corrupted, checksum won't match
  - Receiving discards corrupted packets

```
  134
+ 212
-----
= 346
```

Mismatch!

```
  134
+ 216
-----
= 350
```

IP Header: To and From Addresses

- Destination IP address (32 bits)
  - Unique identifier for the receiving host
  - Allows each node to make forwarding decisions
- Source IP address (32 bits)
  - Unique identifier for the sending host
  - Recipient can decide whether to accept packet
  - Enables recipient to send a reply back to source

Conclusion

- Best-effort global packet delivery
  - Simple end-to-end abstraction
  - Enables higher-level abstractions on top
  - Doesn't rely on much from the links below
- IP addressing and forwarding
  - Hierarchy for scalability and decentralized control
  - Allocation of IP prefixes
  - Longest prefix match forwarding
- Next time: naming