COMP 3331/9331: Computer Networks and Applications

Week 3

Network Layer: Data Plane

Reading Guide: Chapter 4: Sections 4.1-4.3

Network Layer: outline

Our goals:

- understand principles behind network layer services, focusing on data plane:
  - network layer service models
  - forwarding versus routing
  - how a router works
- instantiation, implementation in the Internet

Network Layer, data plane: outline

4.1 Overview of Network layer
- data plane
- control plane

4.2 What's inside a router
-hardware design issues

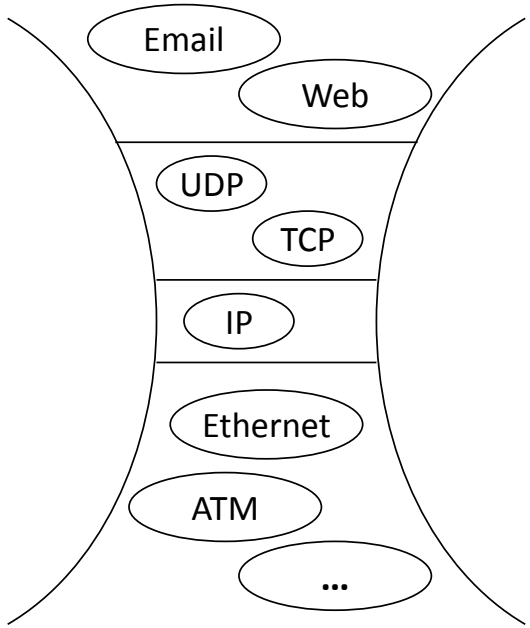
4.3 IP: Internet Protocol
- datagram format
- fragmentation
- IPv4 addressing
- network address translation

4.4 Generalized forwarding and Software Defined Networking (SDN)
- Not Covered

Some Background

- 1968: DARPAnet/ARPAnet (precursor to Internet)
  - (Defense) Advanced Research Projects Agency Network
- Mid 1970's: new networks emerge
  - SATNet, Packet Radio, Ethernet
  - All "islands" to themselves – didn't work together
- Big question: How to connect these networks?

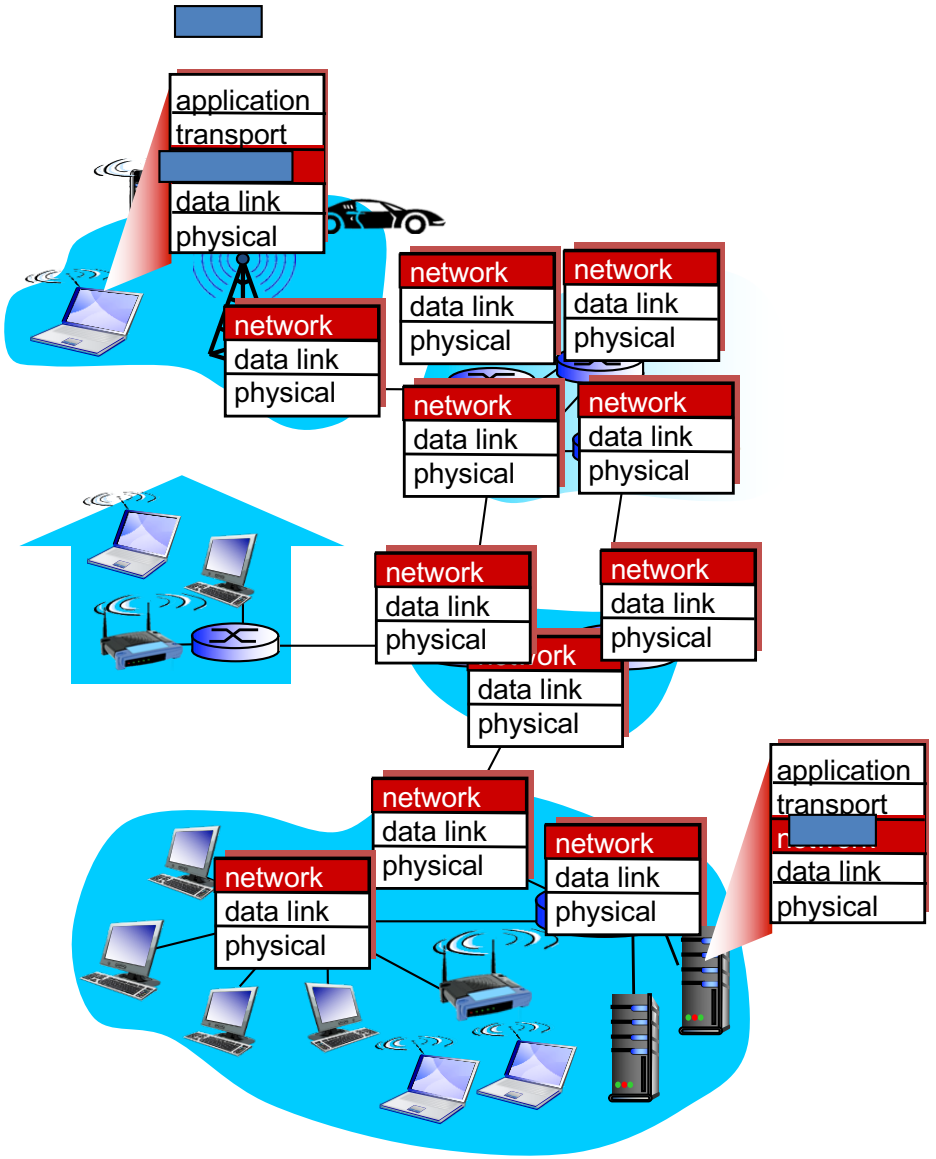
Internetworking



- Cerf & Kahn in 1974,
  - "A Protocol for Packet Network Intercommunication"
  - Foundation for the modern Internet
- **Routers** forward **packets** from source to destination
  - May cross many separate networks along the way
- All packets use a common **Internet Protocol**
  - Any underlying data link protocol
  - Any higher layer transport protocol

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



Two key nework-layer functions

- *forwarding:* move packets from router's input to appropriate router output
- *routing:* determine route taken by packets from source to dest.
  - *routing algorithms*

analogy:

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

When should a router perform routing? And forwarding ?

A: Do both when a packet arrives

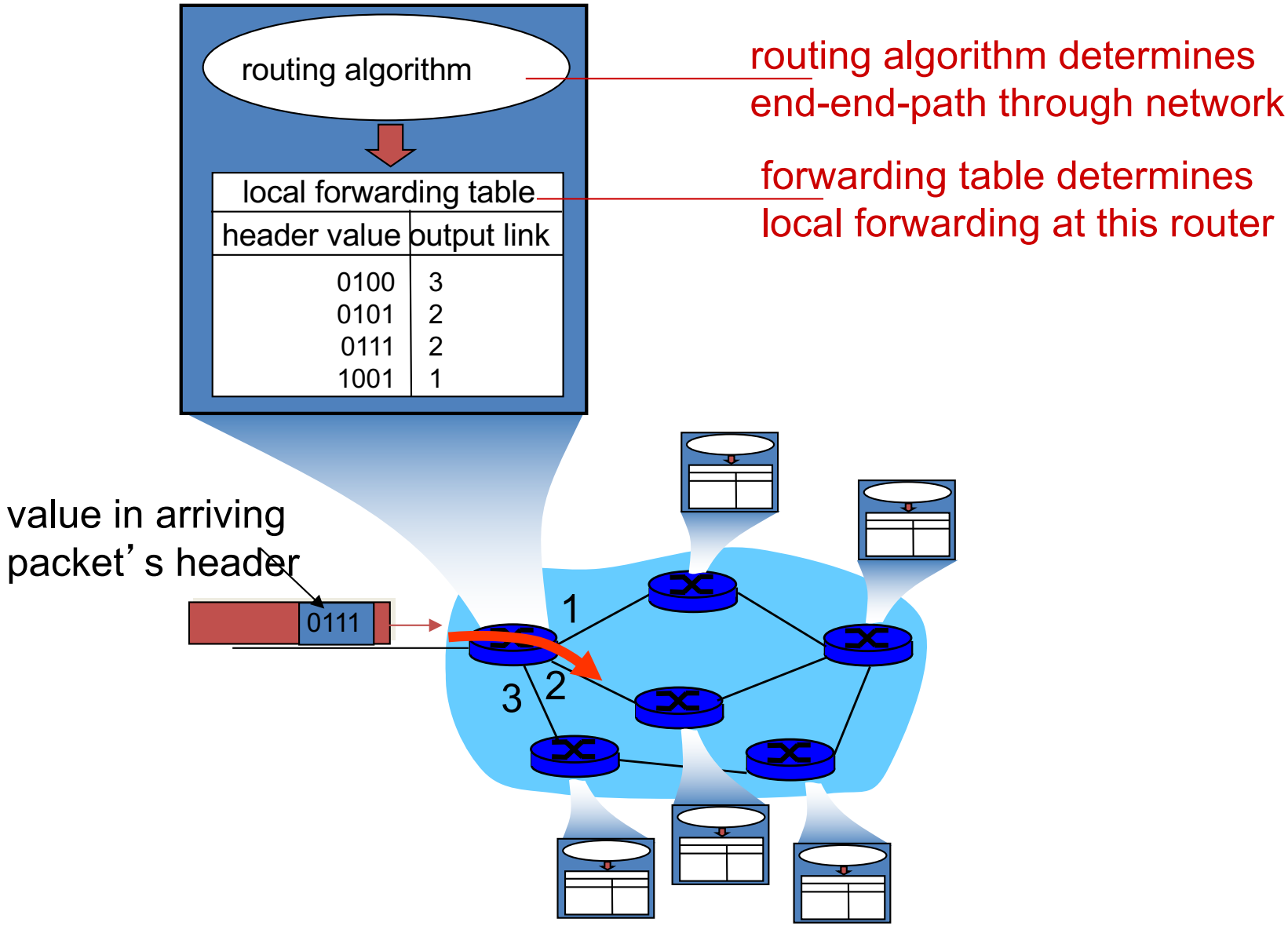
B: Route in advance, forward when a packet arrives

C: Forward in advance, route when a packet arrives

D: Do both in advance

E: Some other combination

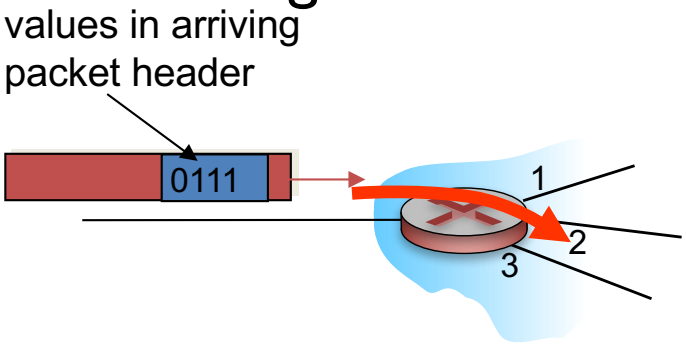
Interplay between routing and forwarding



Network Layer: data vs control plane

Data plane

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

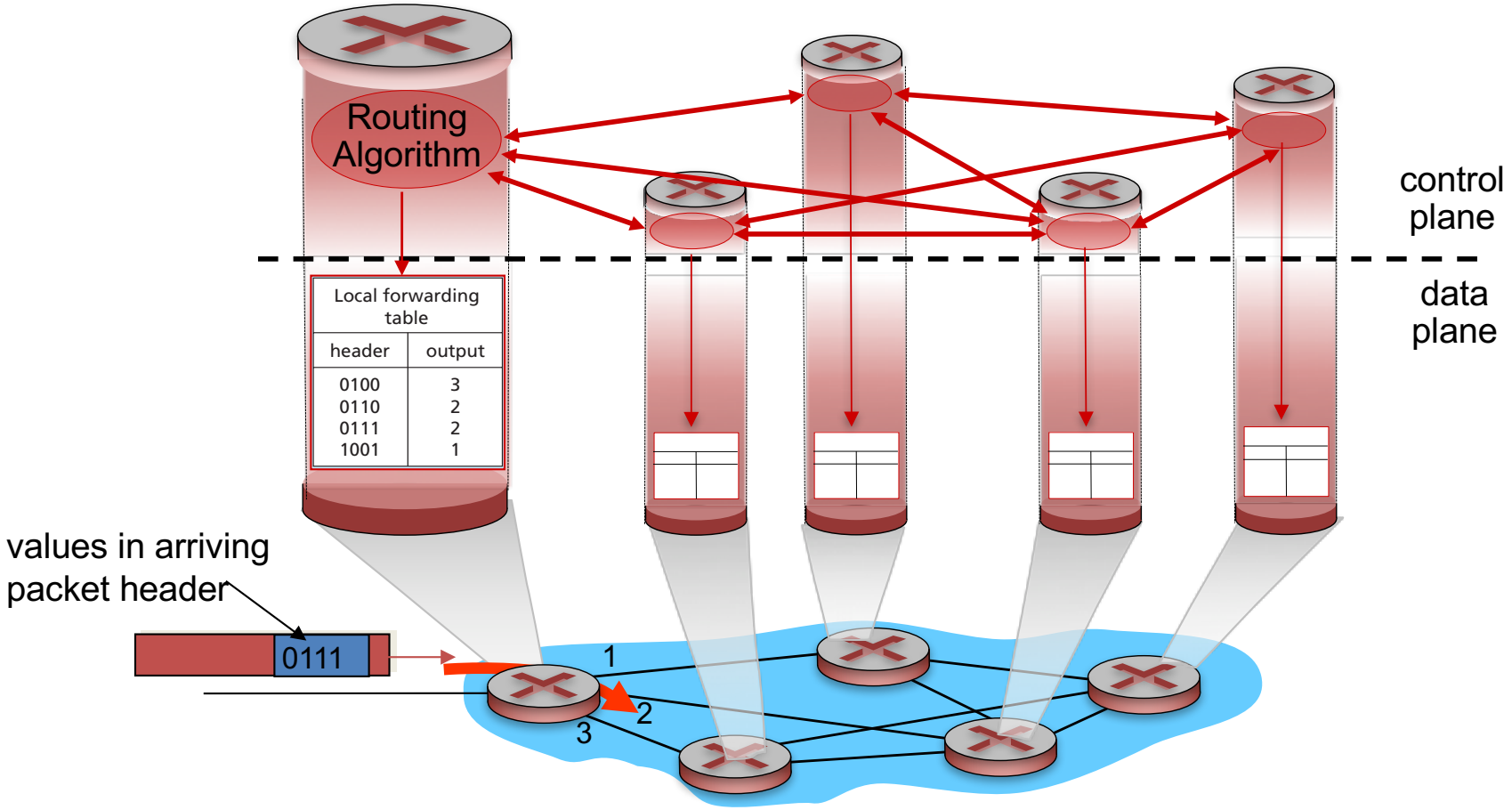


Control plane

- 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)*: centralised (remote) servers

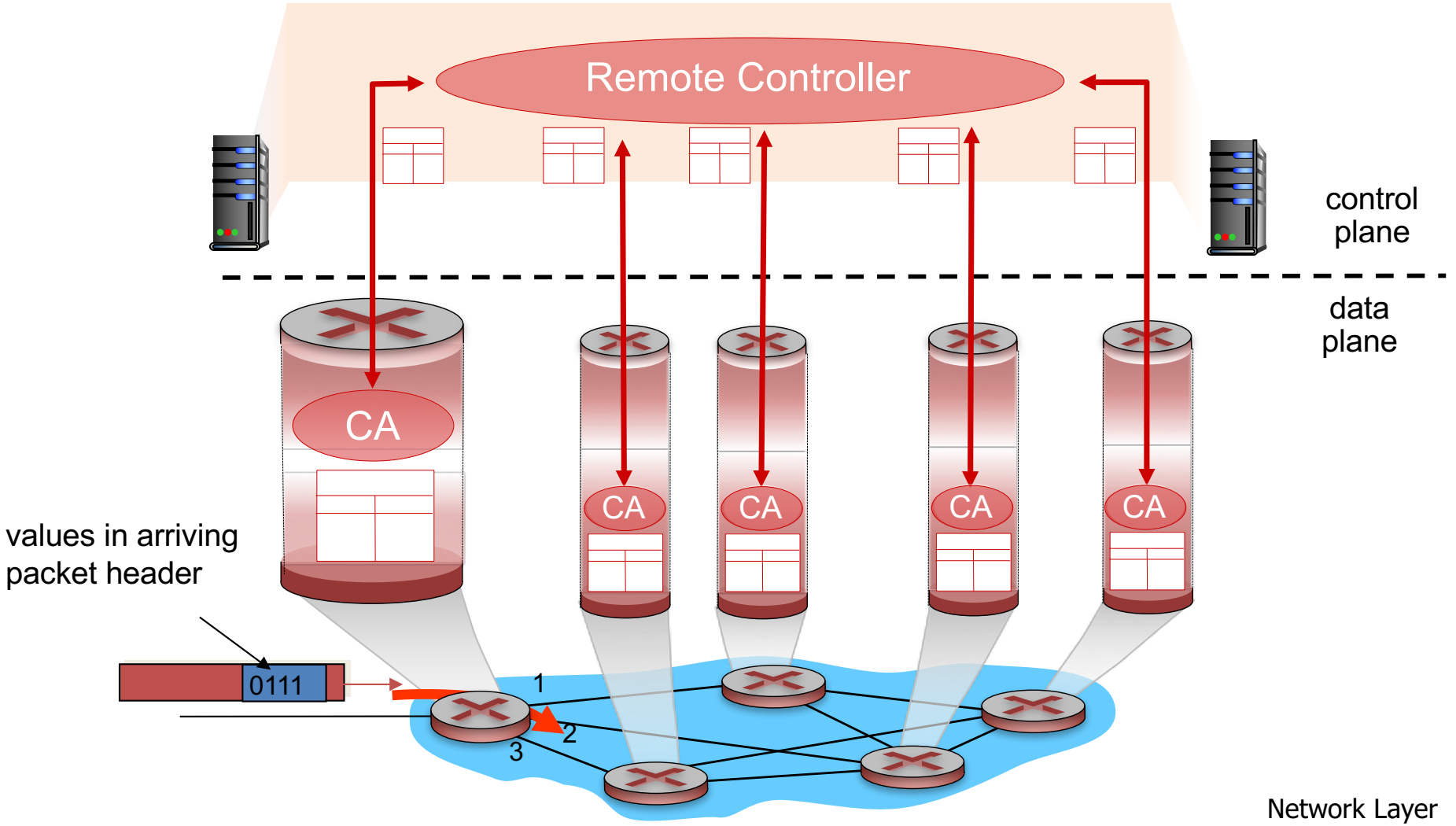
Per-router control plane

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



Logically centralized control plane (SDN)

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



Network Layer: 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

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 models

Network Architecture	Service Model	Guarantees ?				Congestion feedback
		Bandwidth	Loss	Order	Timing	
Internet	best effort	none	no	no	no	no (inferred via loss)
ATM	CBR	constant rate	yes	yes	yes	no congestion
ATM	VBR	guaranteed rate	yes	yes	yes	no congestion
ATM	ABR	guaranteed minimum	no	yes	no	yes
ATM	UBR	none	no	yes	no	no

No need to memorise this table

Network Layer, data plane: outline

4.1 Overview of Network layer
- data plane
- control plane

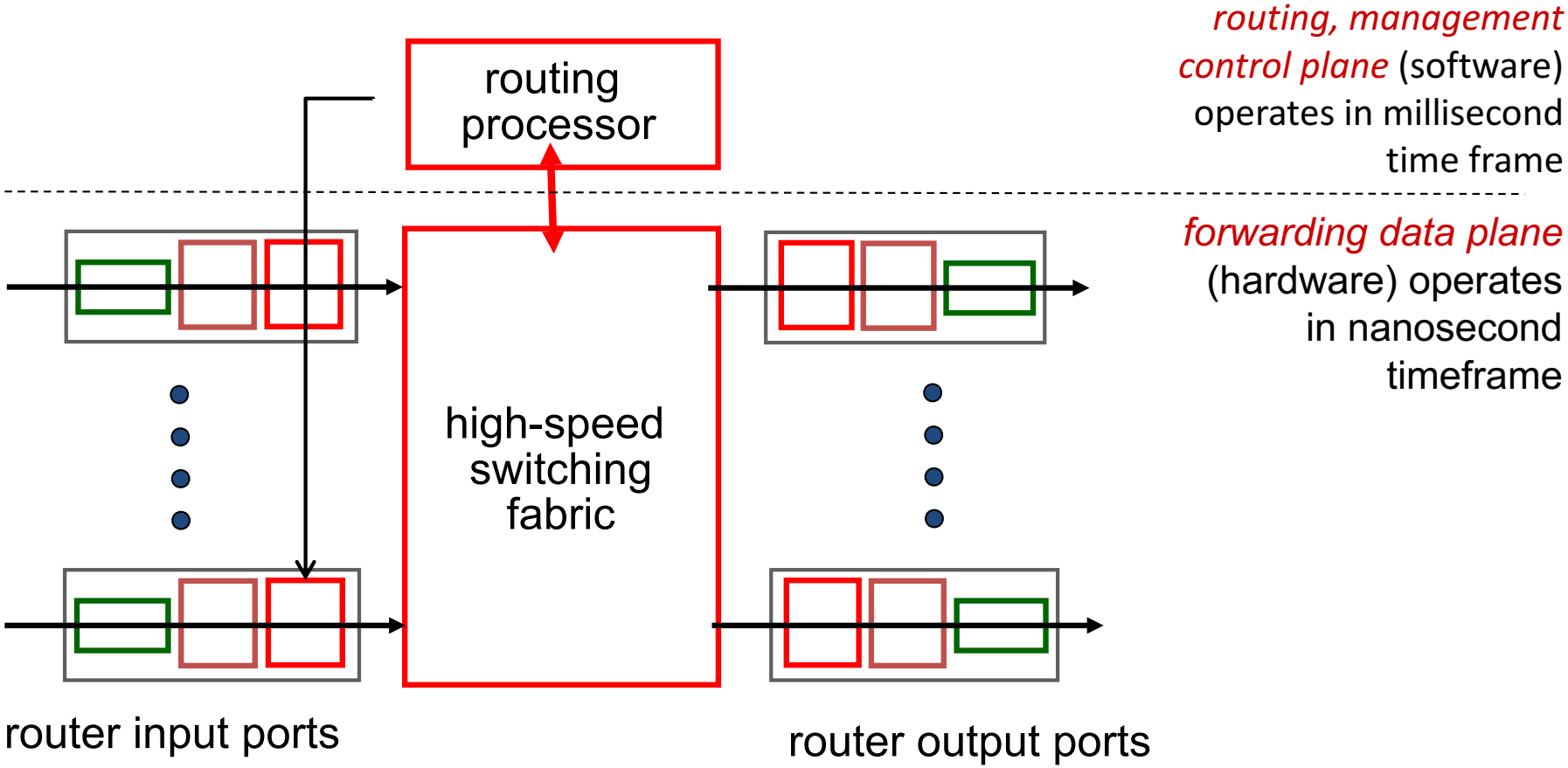
4.2 What's inside a router
-mostly self study

4.3 IP: Internet Protocol
- datagram format
- fragmentation
- IPv4 addressing
- network address translation
- IPv6

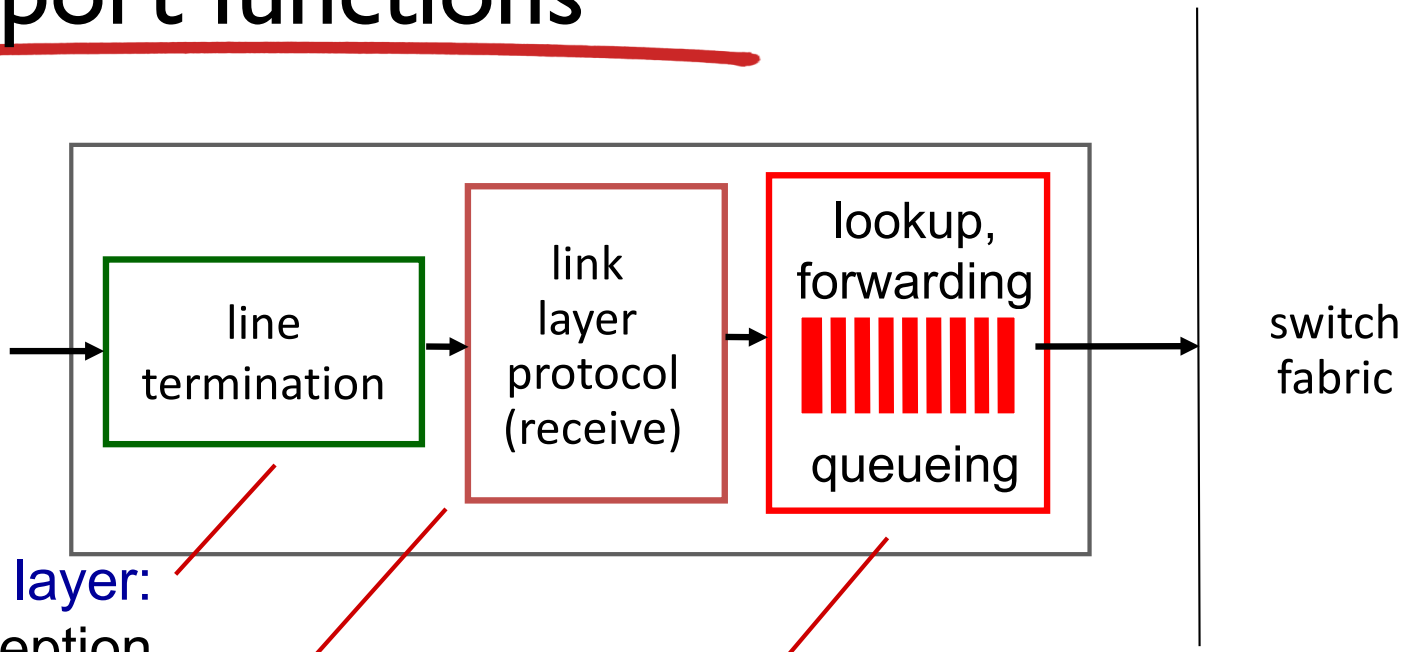
4.4 Generalized Forwarding and SDN

Router architecture overview

- high-level view of generic router architecture:



Input port functions



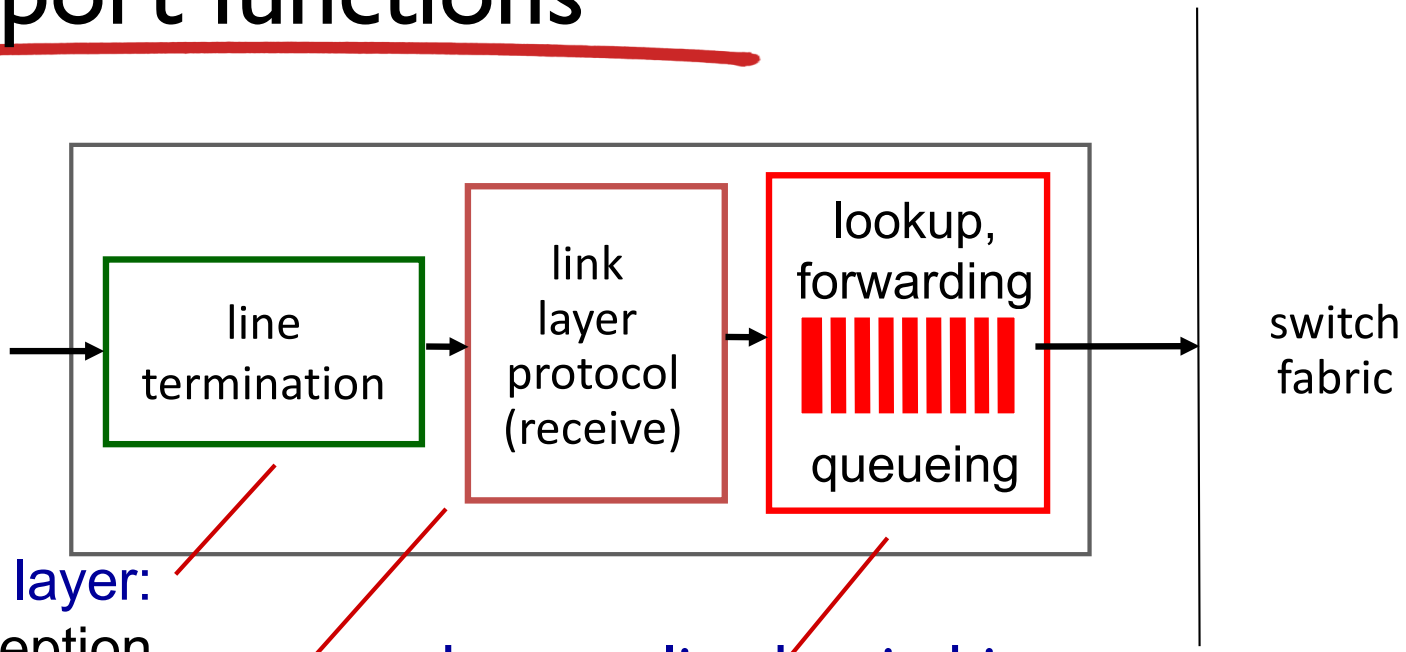
physical layer:
bit-level reception

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

Input port functions



physical layer:
bit-level reception

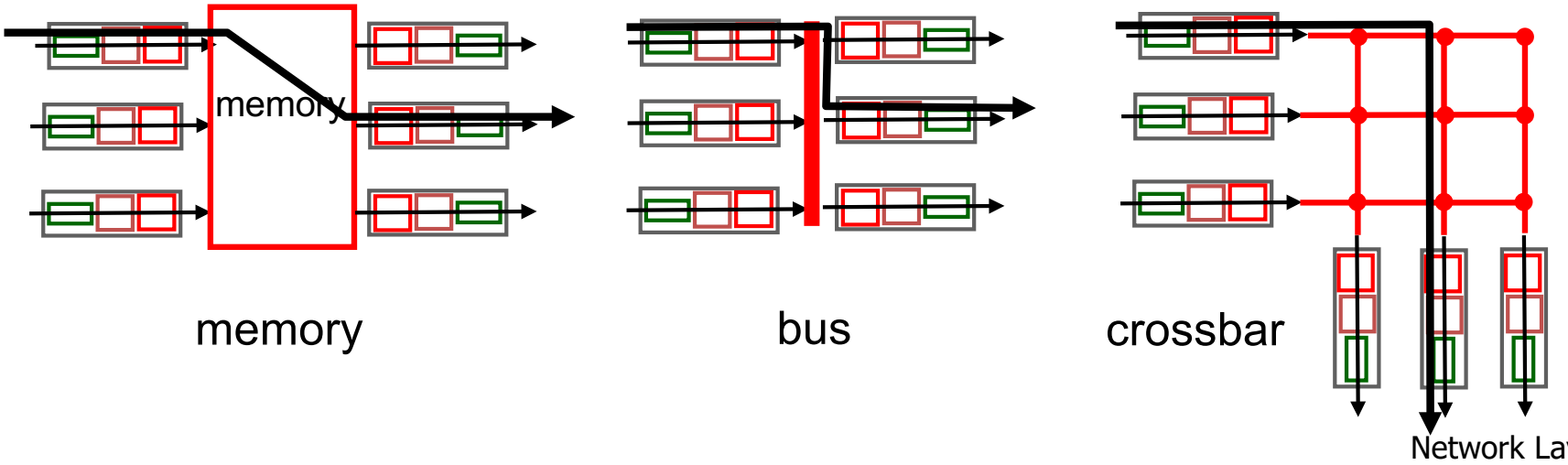
data link layer:
e.g., Ethernet

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

Switching fabrics

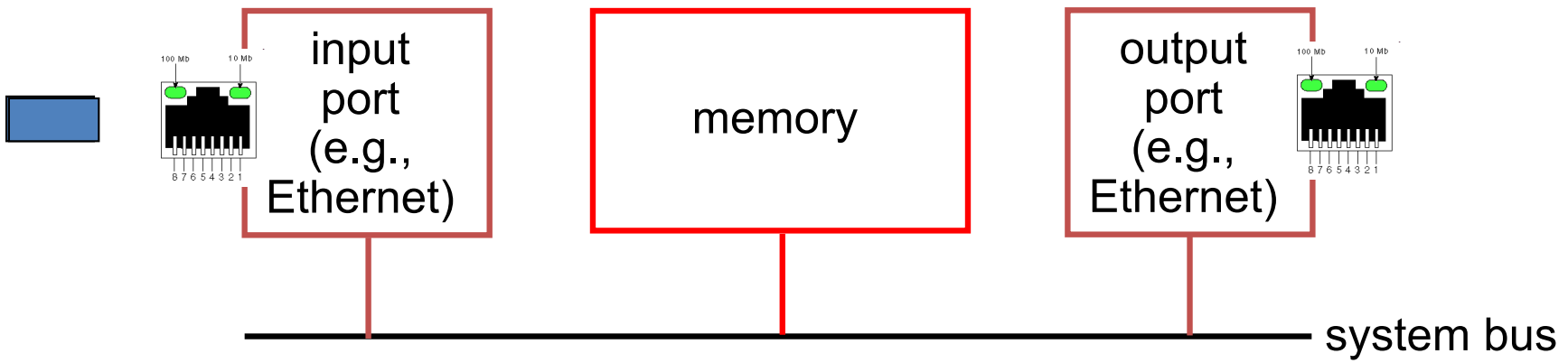
- transfer packet from input buffer to appropriate output buffer
- switching rate: rate at which packets can be transfer from inputs to outputs
  - often measured as multiple of input/output line rate
  - N inputs: switching rate N times line rate desirable
- three types of switching fabrics



Switching via memory

first generation routers:

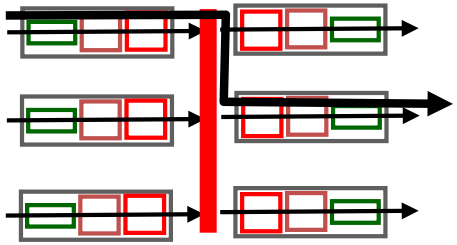
- traditional computers with switching under direct control of CPU
- packet copied to system's memory
- speed limited by memory bandwidth (2 bus crossings per datagram)



Switching via a bus

- datagram from input port memory

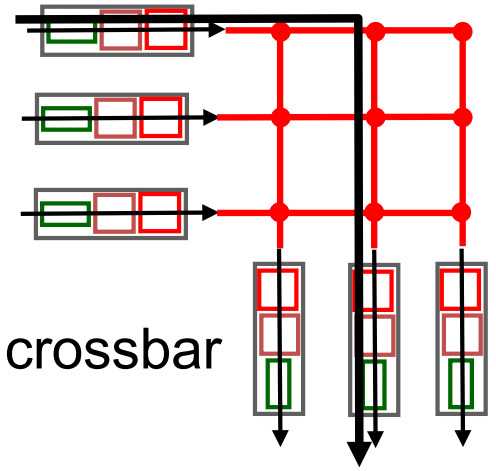
  to output port memory via a shared bus
- *bus contention:* switching speed limited by bus bandwidth
- 32 Gbps bus, Cisco 5600: sufficient speed for access and enterprise routers



bus

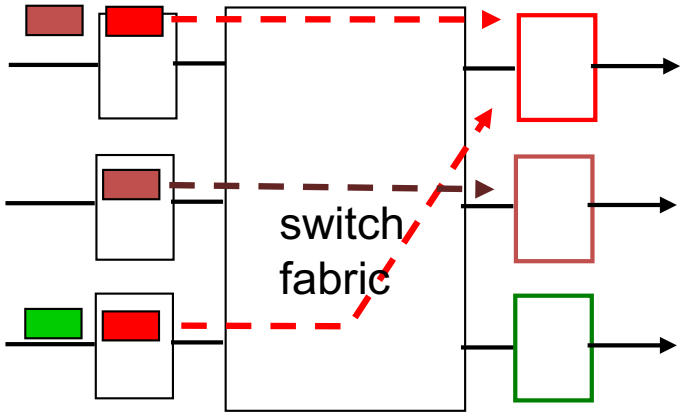
Switching via interconnection network

- 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 network

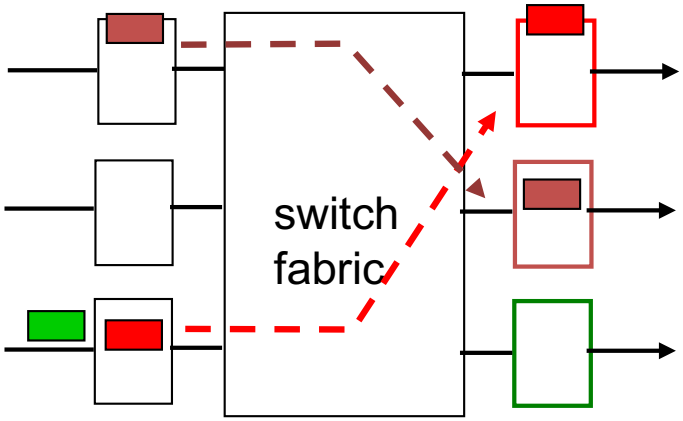


Input port queuing

- 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

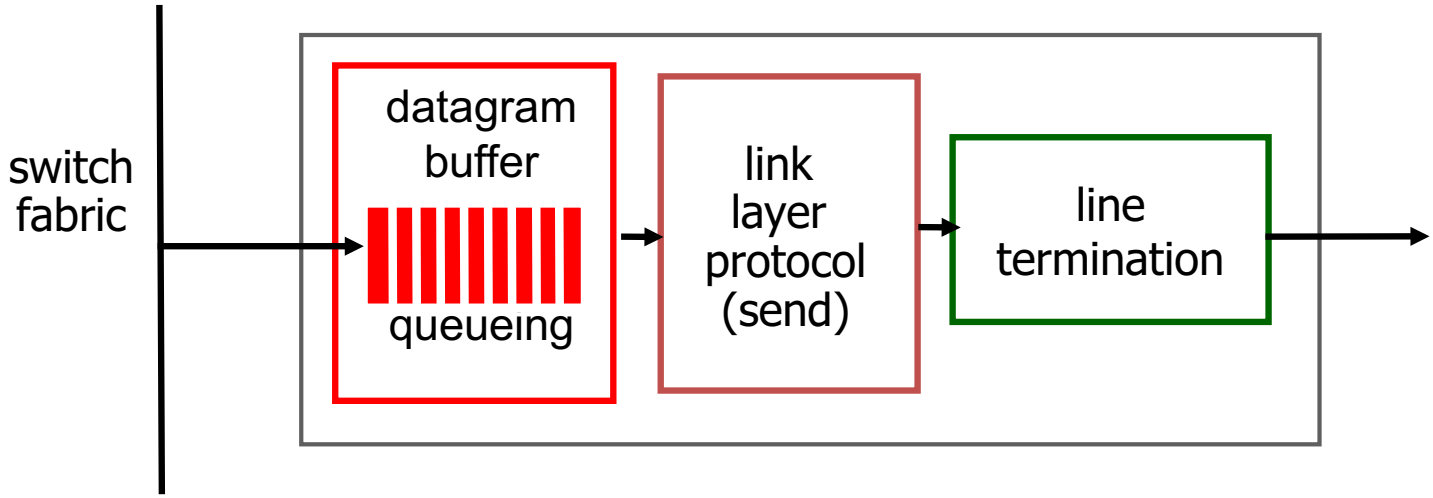


output port contention:
only one red datagram can be transferred.
lower red packet is blocked



one packet time later:
green packet
experiences HOL
blocking

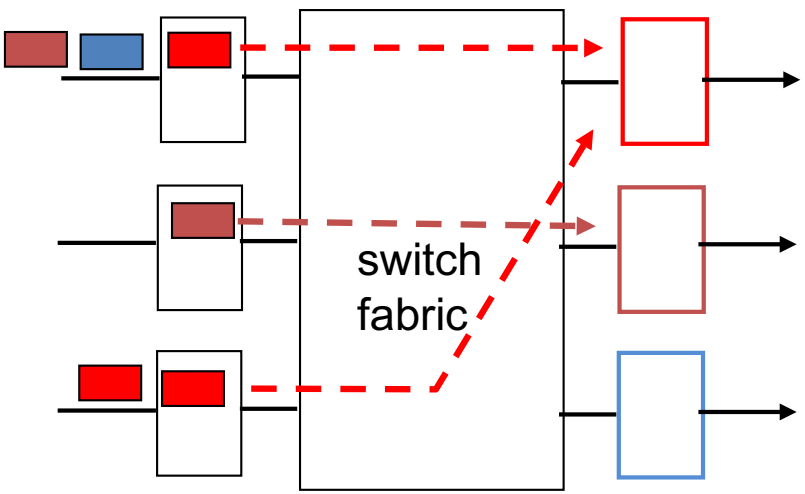
Output ports



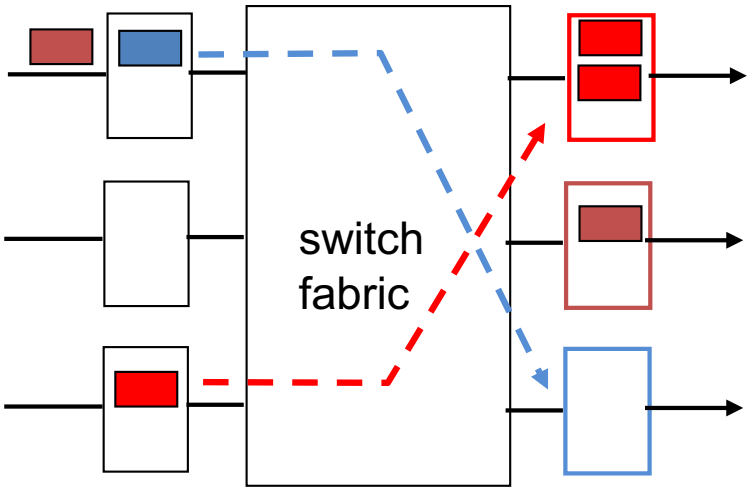
- *buffering* required when datagrams arrive from fabric faster than the transmission rate
- *scheduling discipline* chooses among queued datagrams for transmission

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

Output port queueing



at *t*, packets move
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!*

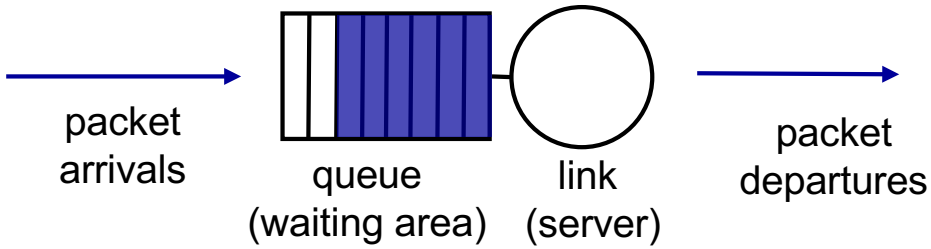
How much buffering?

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

G. Appenzeller, I. Keslassy, and N. McKeown. Sizing Router Buffers. In ACM SIGCOMM, USA, 2004

Scheduling mechanisms

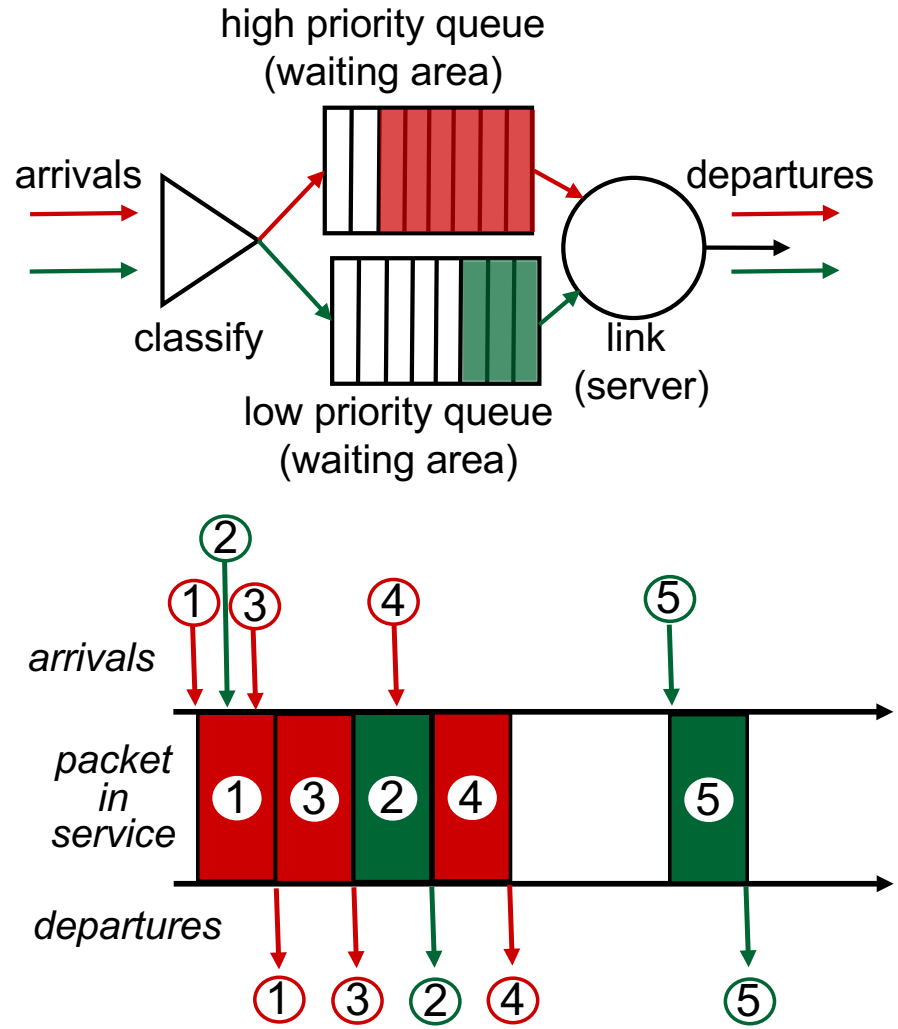
- *scheduling:* choose next packet to send on link
- *FIFO (first in first out) scheduling:* send in order of arrival to queue
  - *discard policy:* if packet arrives to full queue: who to discard?
    - *tail drop:* drop arriving packet
    - *priority:* drop/remove on priority basis
    - *random:*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



Scheduling policies: priority

priority scheduling: send highest priority queued packet

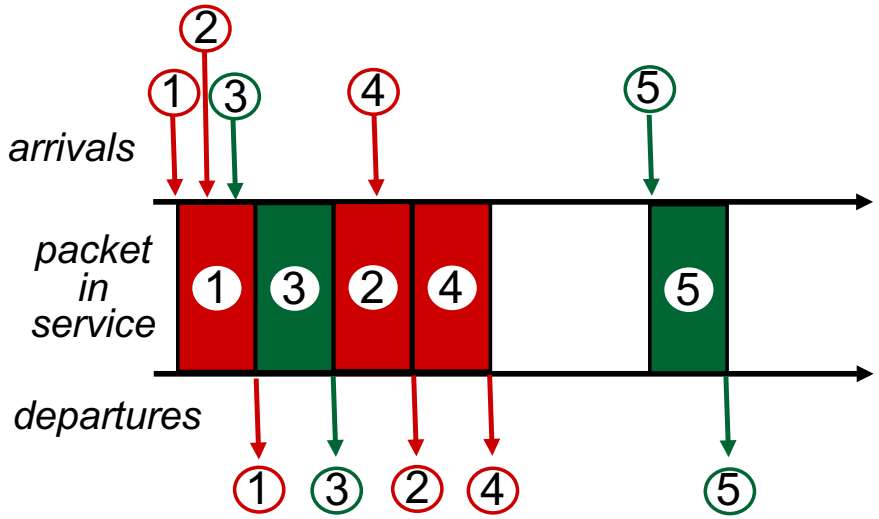
- multiple *classes*, with different priorities
  - class may depend on marking or other header info, e.g. IP source/dest, port numbers, etc.



Scheduling policies: still more

Round Robin (RR) scheduling:

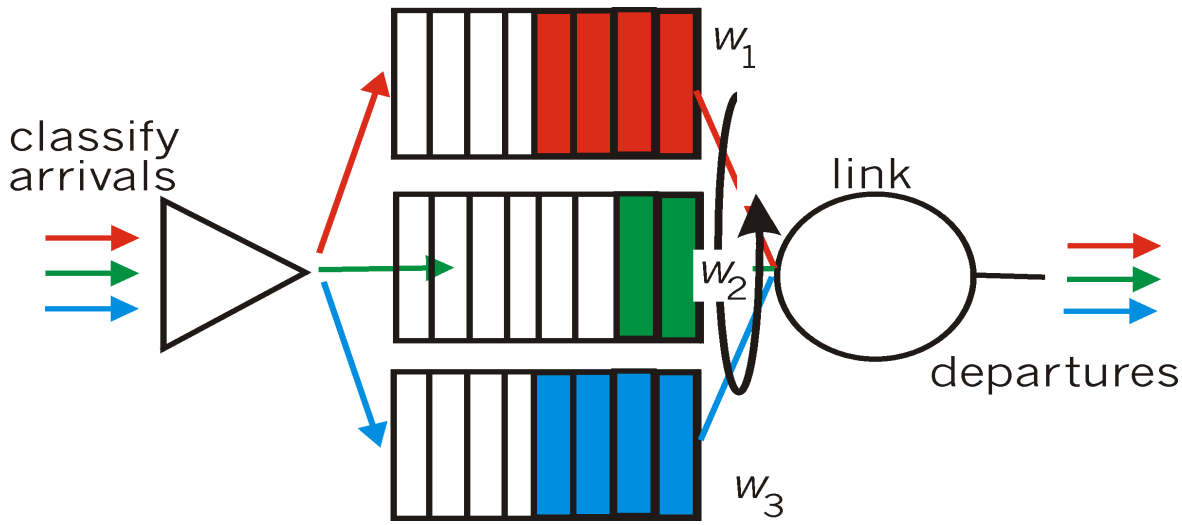
- multiple classes
- cyclically scan class queues, sending one complete packet from each class (if available)



Scheduling policies: still more

Weighted Fair Queuing (WFQ):

- generalized Round Robin
- each class gets weighted amount of service in each cycle



Network Layer, data plane: outline

4.1 Overview of Network layer
- data plane
- control plane

4.2 What's inside a router

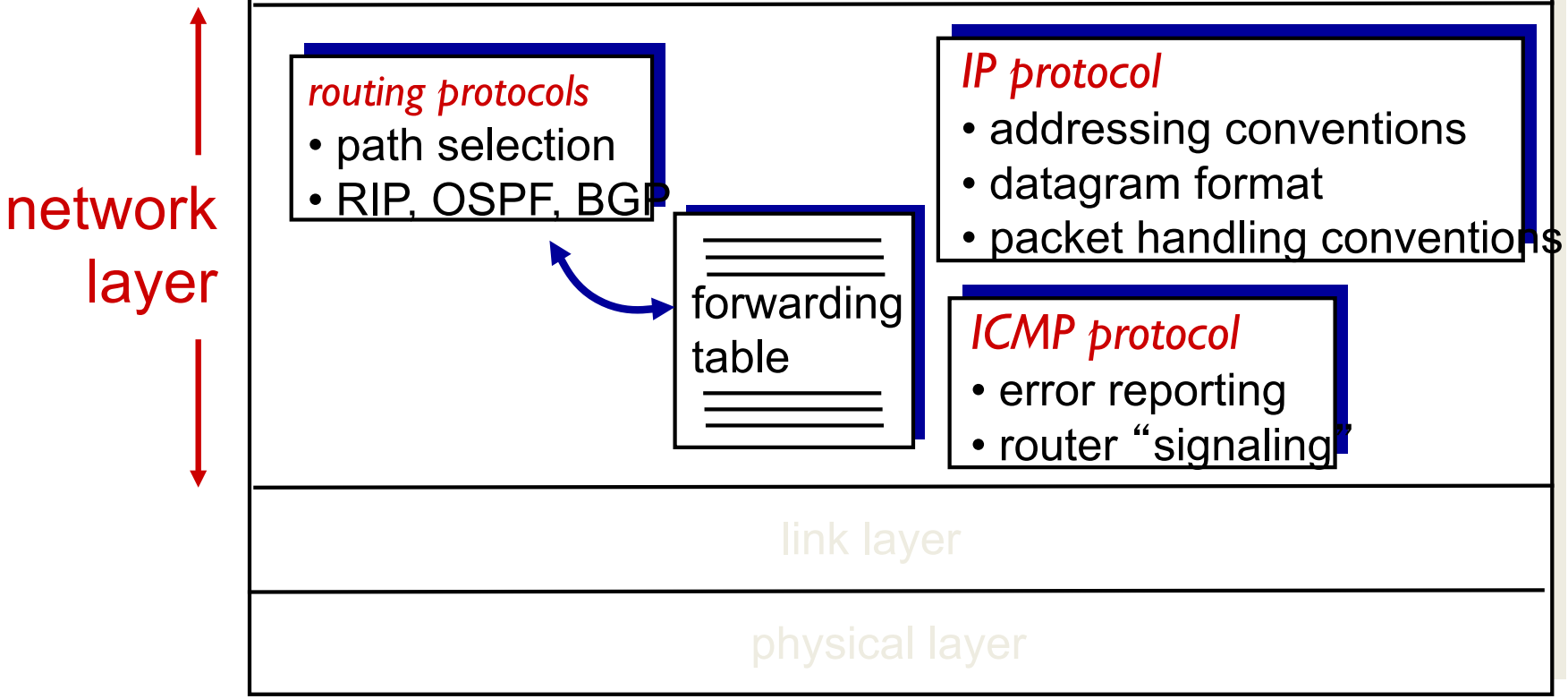
4.3 IP: Internet Protocol
- datagram format
- fragmentation
- IPv4 addressing
- network address translation
- IPv6

4.4 Generalized forwarding

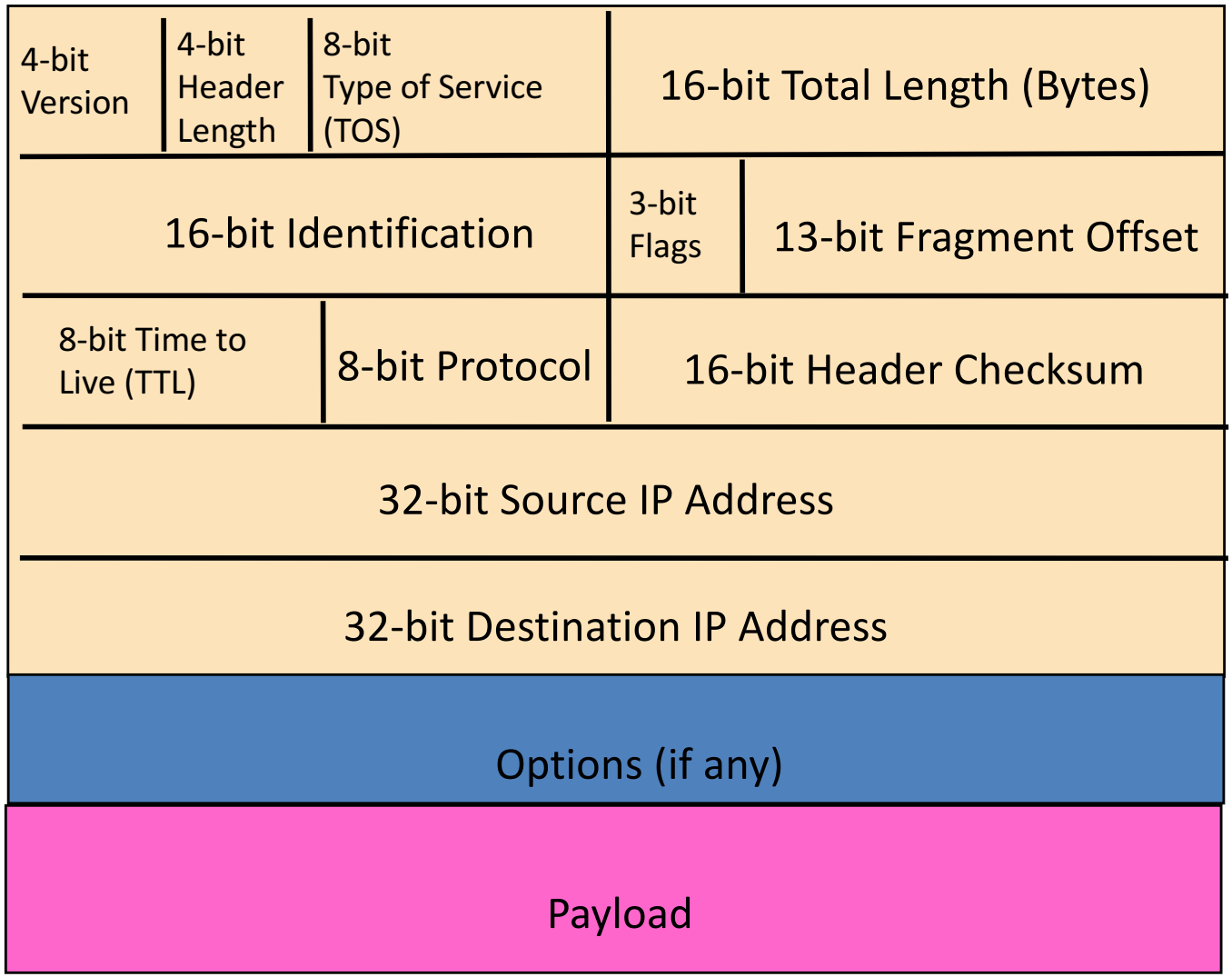
(Not Covered)

The Internet network layer

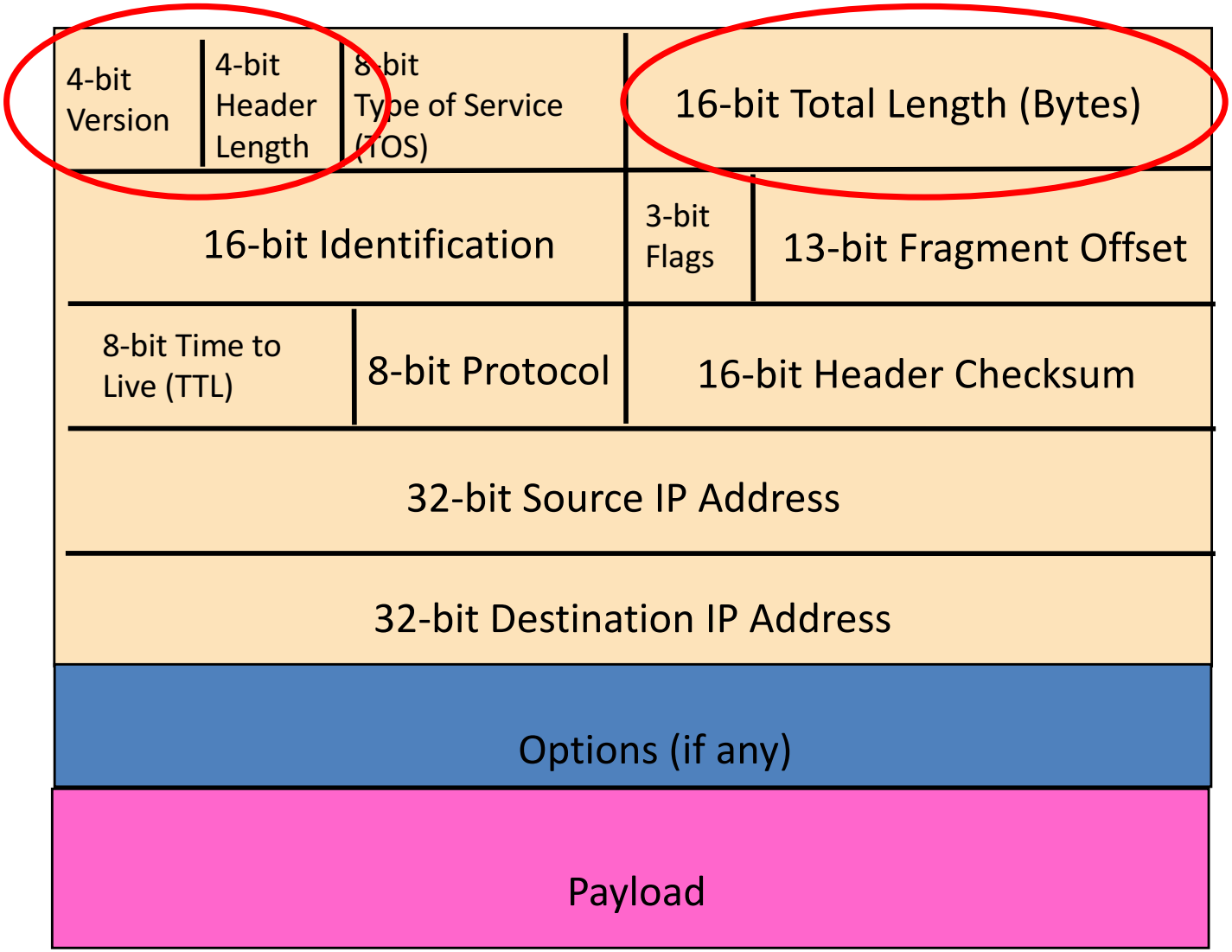
host, router network layer functions:



IP Packet Structure



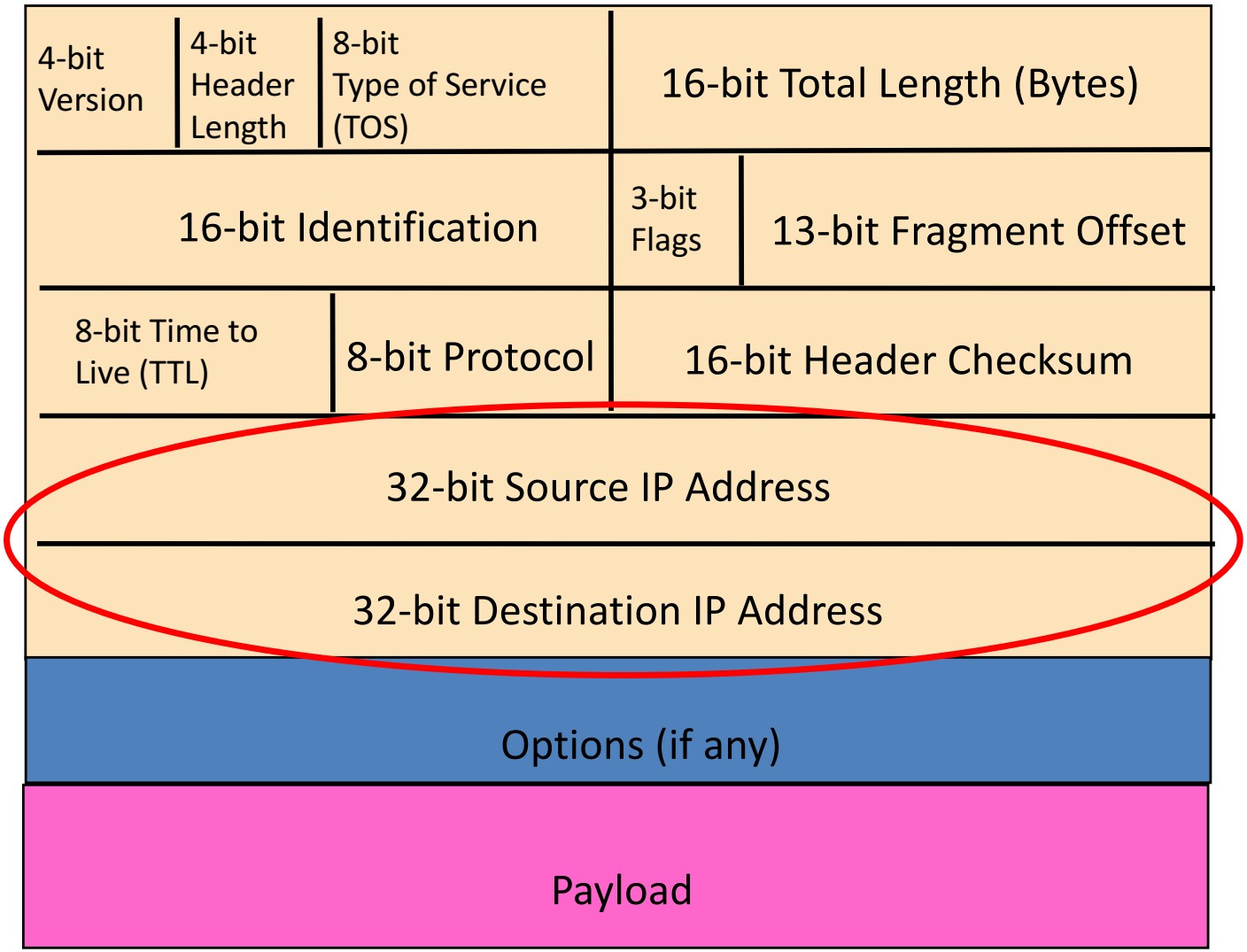
Fields for Reading Packet Correctly



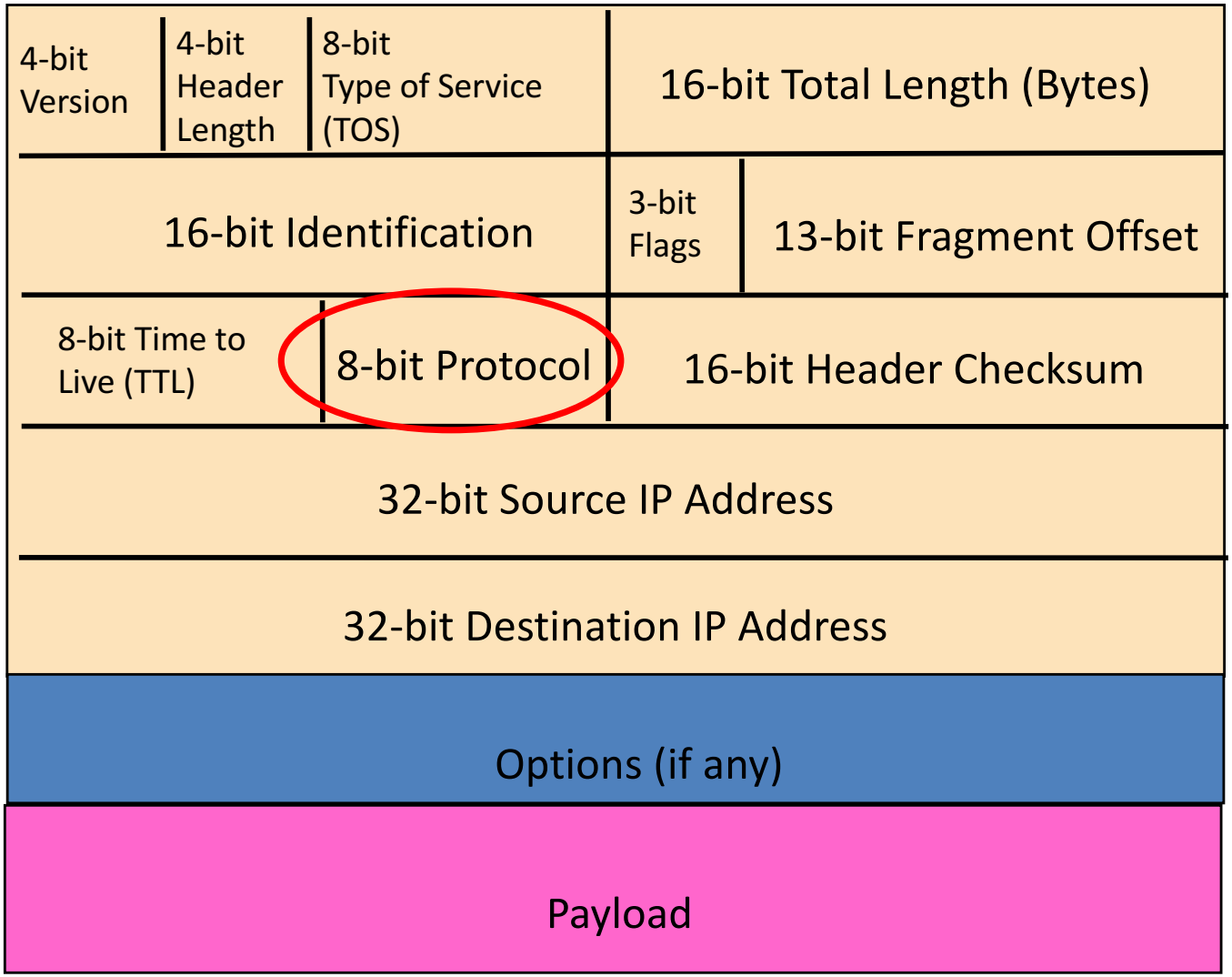
Reading Packet Correctly

- Version number (4 bits)
  - Indicates the version of the IP protocol
  - Necessary to know what other fields to expect
  - Typically “4” (for IPv4)
- 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
- Total length (16 bits)
  - Number of bytes in the packet
  - Maximum size is 65,535 bytes (2^{16} -1)
  - … though underlying links may impose smaller limits

Fields for Reaching Destination and Back

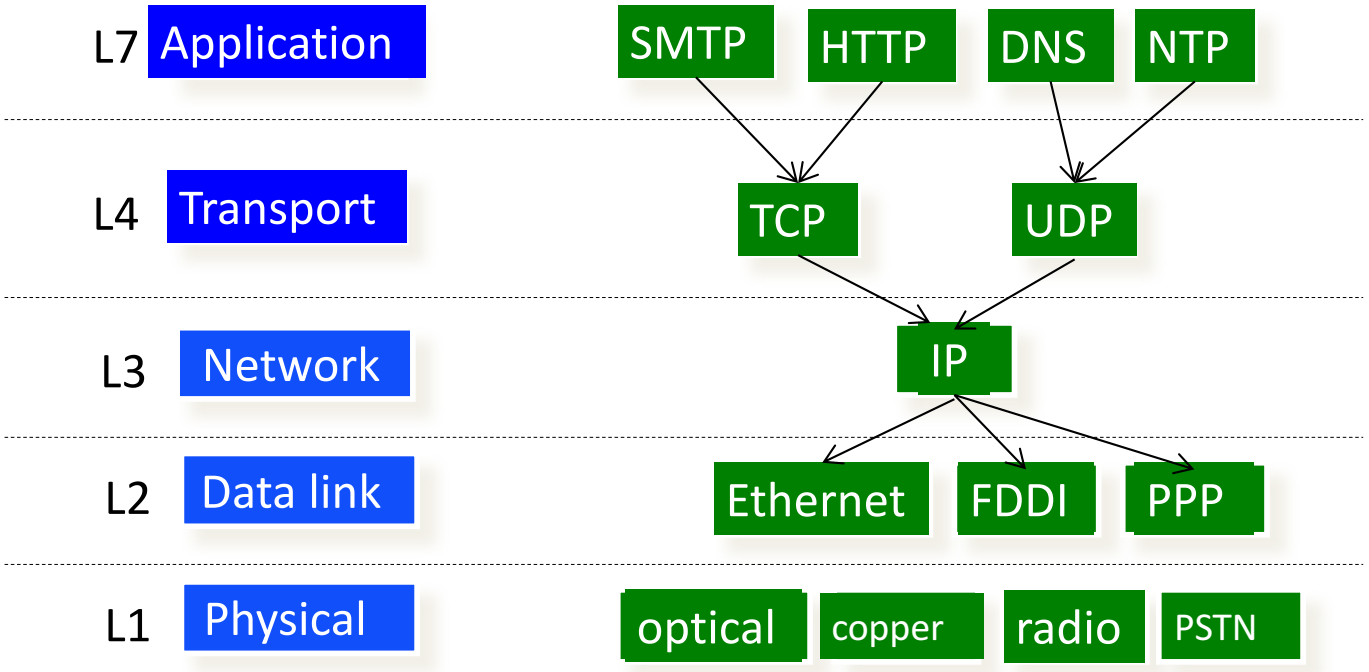


Telling End-Host How to Handle Packet



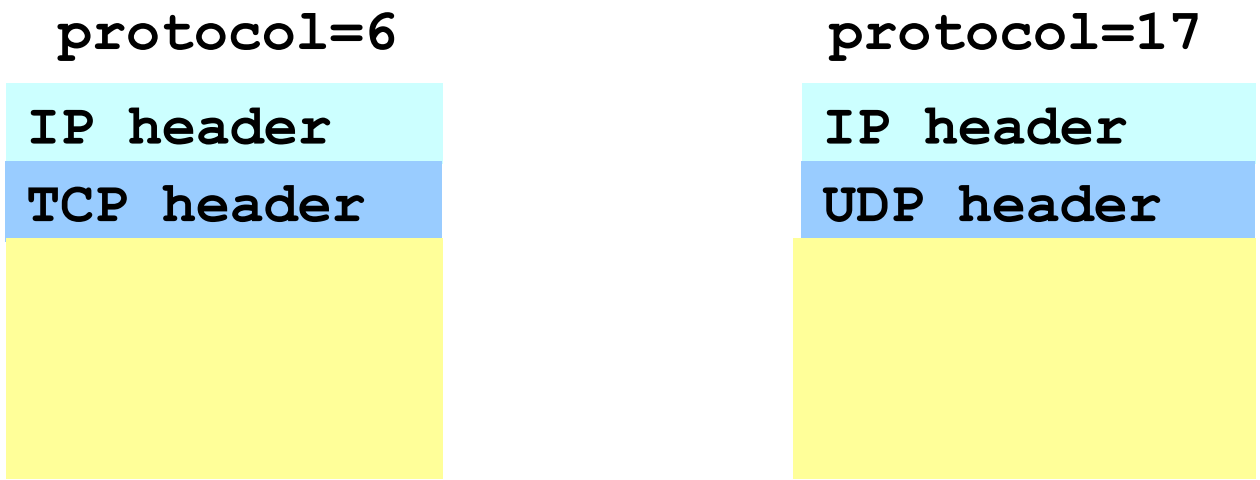
Telling End-Host How to Handle Packet

- Protocol (8 bits)
  - Identifies the higher-level Transport protocol
  - Important for demultiplexing at receiving host

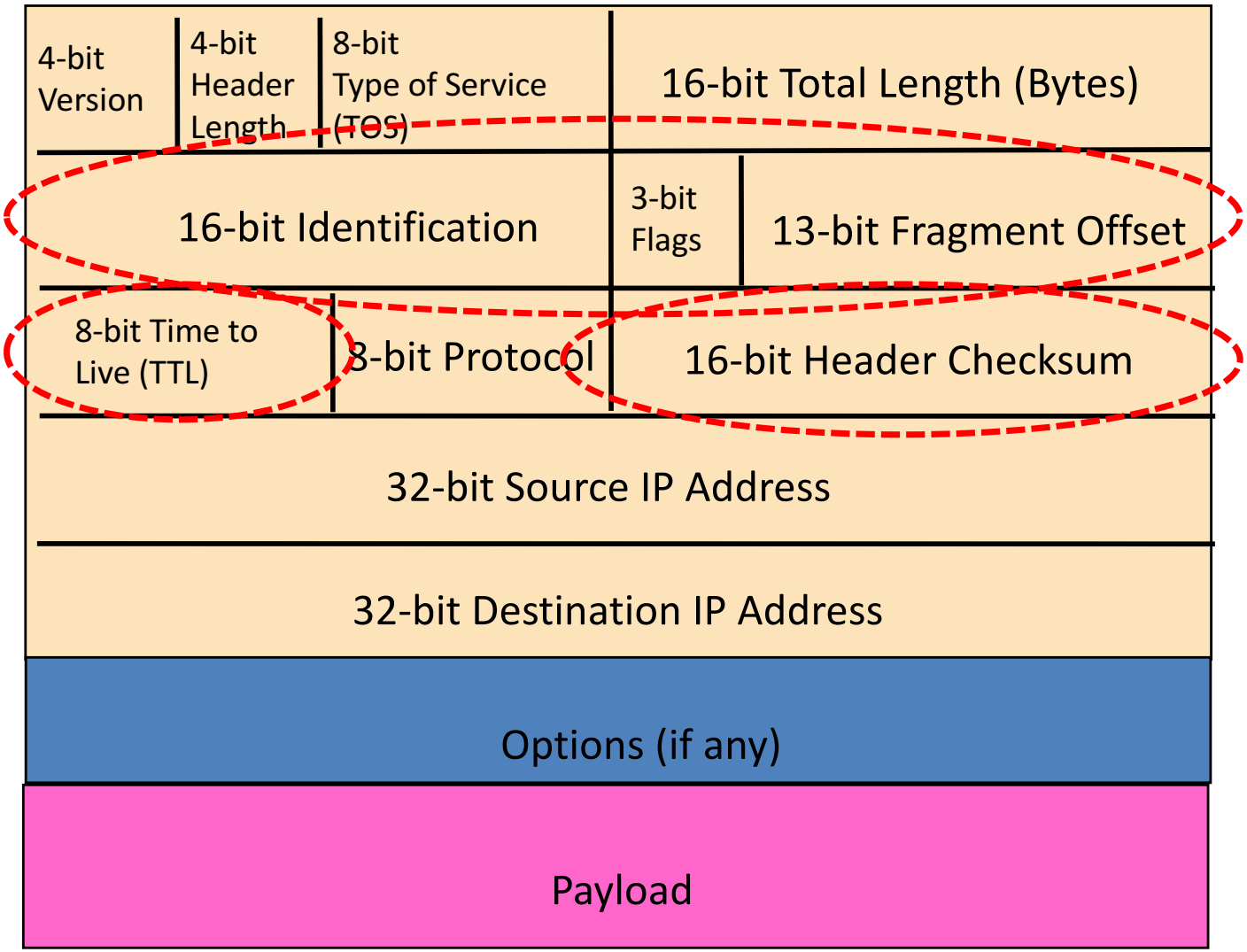


Telling End-Host How to Handle Packet

- Protocol (8 bits)
  - Identifies the higher-level protocol
  - Important for de-multiplexing at receiving host
- Most common examples
  - E.g., “6” for the Transmission Control Protocol (TCP)
  - E.g., “17” for the User Datagram Protocol (UDP)



Checksum, TTL and Fragmentation Fields



Potential Problems

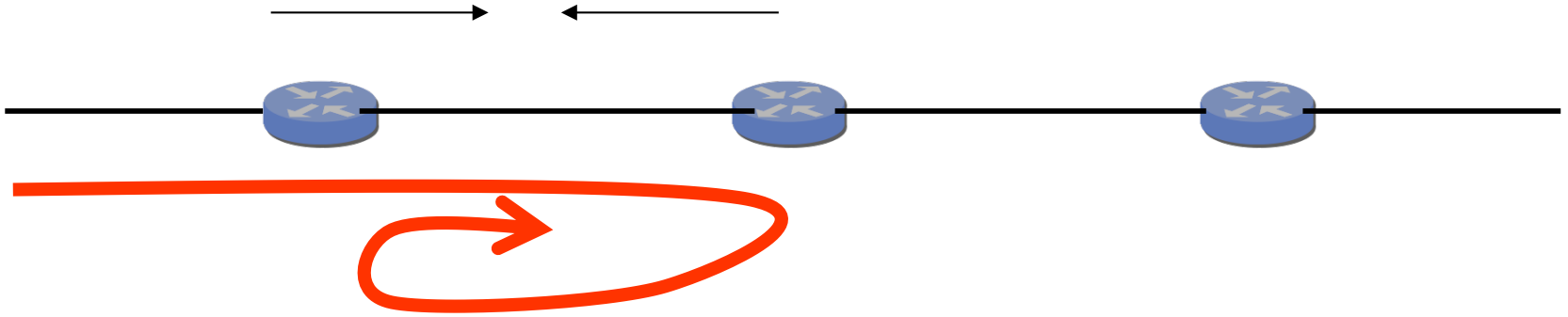
- Header Corrupted: **Checksum**
- Loop: **TTL**
- Packet too large: **Fragmentation**

Header Corruption (Checksum)

- Checksum (16 bits)
  - Particular form of checksum over packet header
- If not correct, router discards packets
  - So it doesn't act on bogus information
- Checksum recalculated at every router
  - **Why?**
  - **Why include TTL?**
  - **Why only header?**

Preventing Loops (TTL)

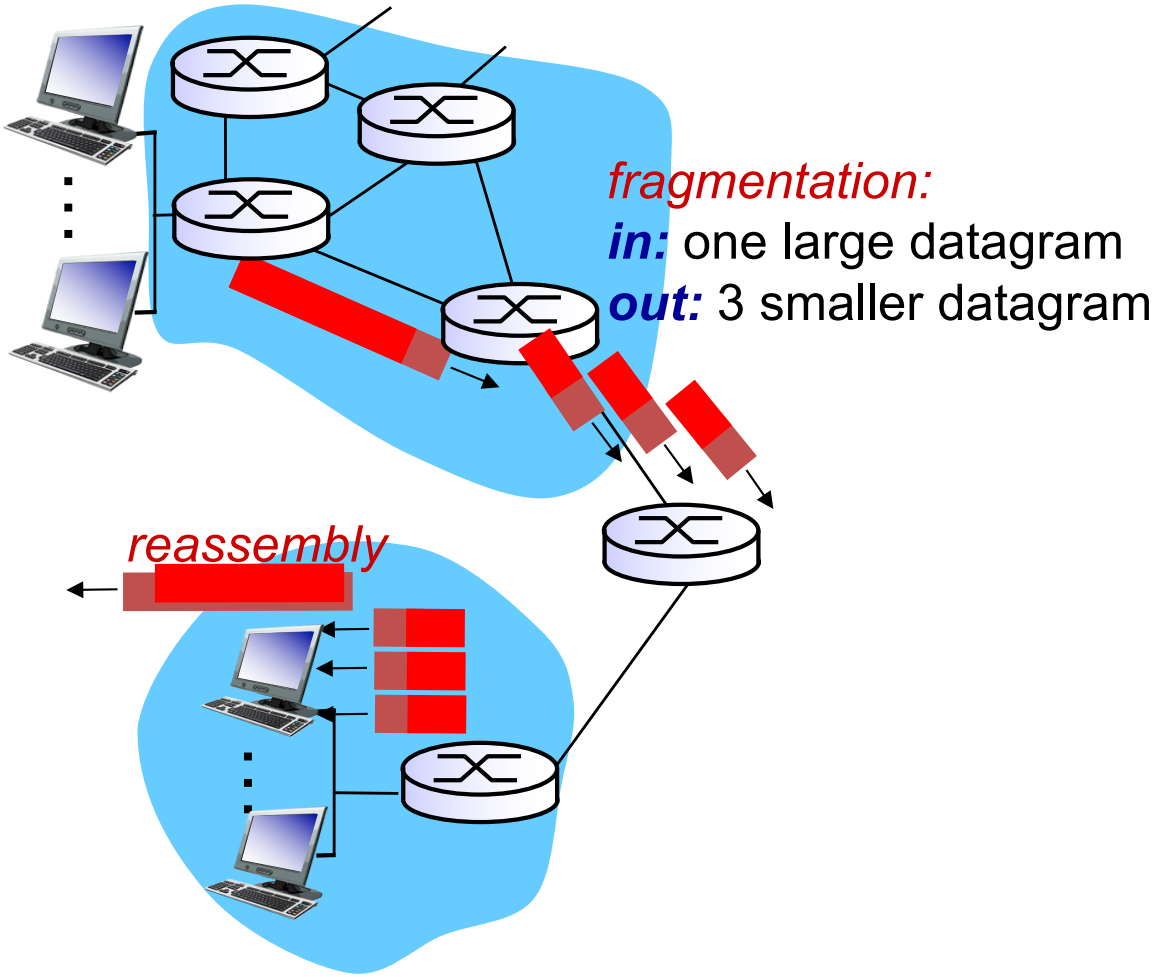
- Forwarding loops cause packets to cycle for a looong time
  - As these accumulate, eventually consume **all** capacity



- Time-to-Live (TTL) Field (8 bits)
  - Decremented at each hop, packet discarded if reaches 0
  - …and "time exceeded" message is sent to the source

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 fragments



IP fragmentation, reassembly

example:

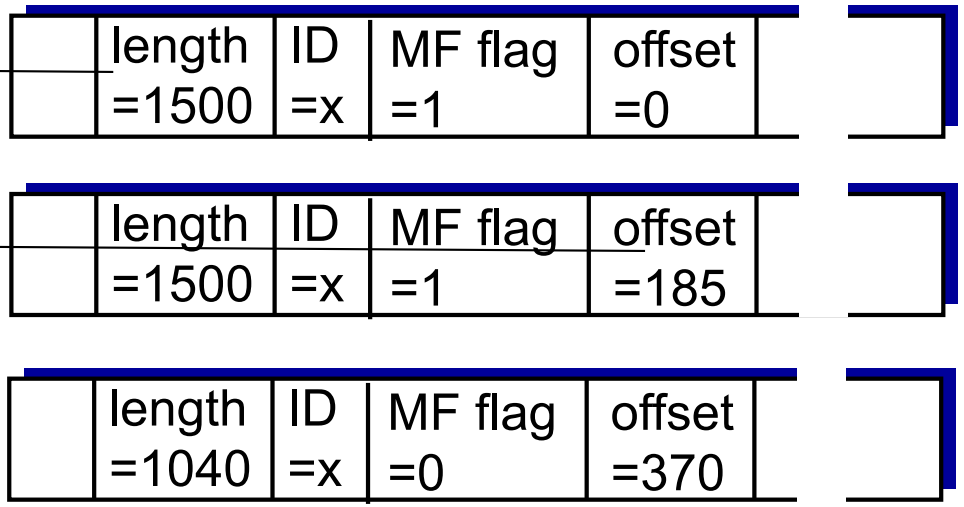
- 4000 byte datagram
- MTU = 1500 bytes

one large datagram becomes several smaller datagrams

Data = 4000 – 20 (IP header) =3980
F1=1480
F2=1480
F3=1020

1480 bytes in data field

offset = 1480/8

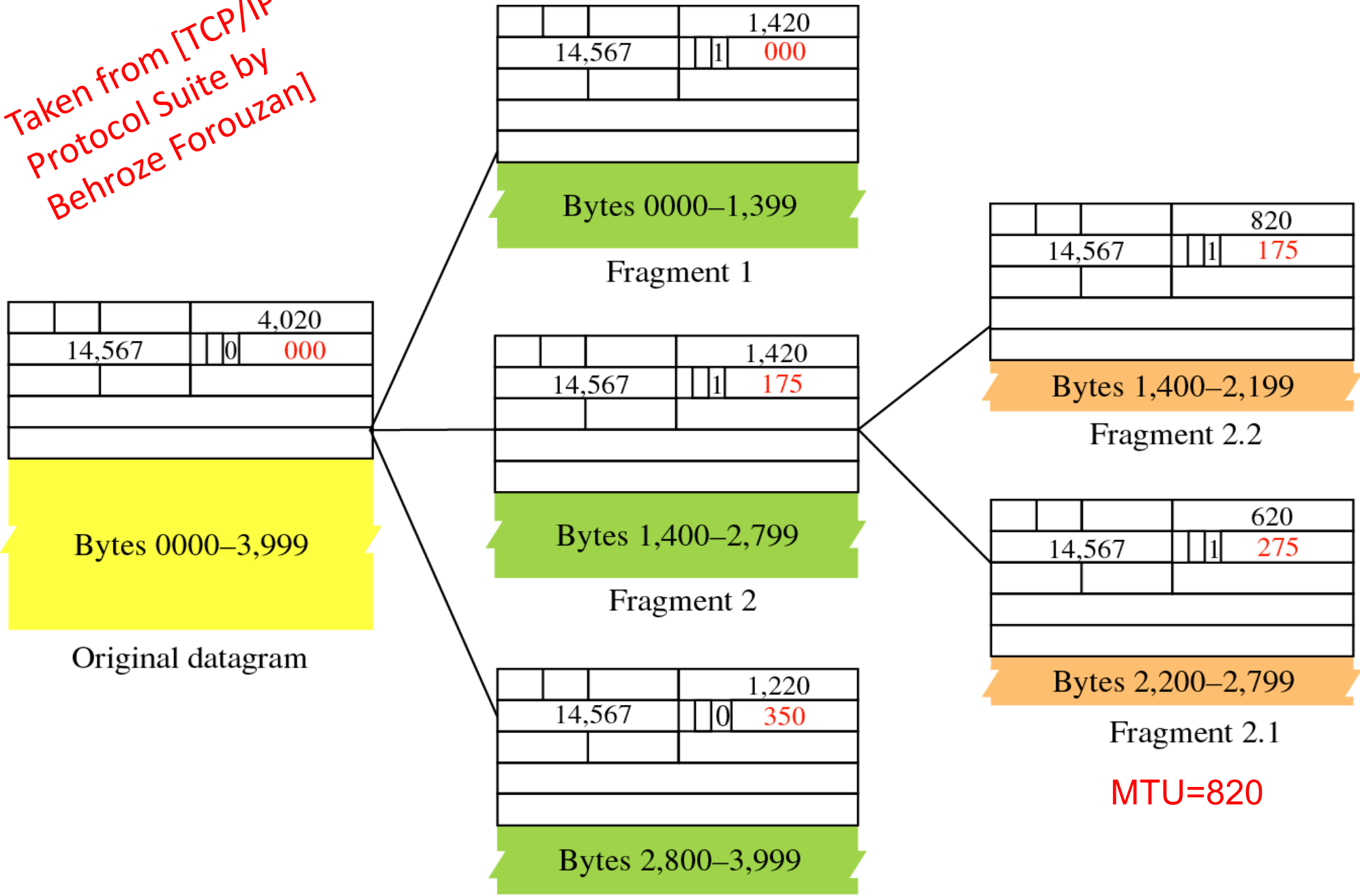


Applet:
http://media.pearsoncmg.com/aw/aw_kurose_network_2/applets/ip/ipfragmentation.html

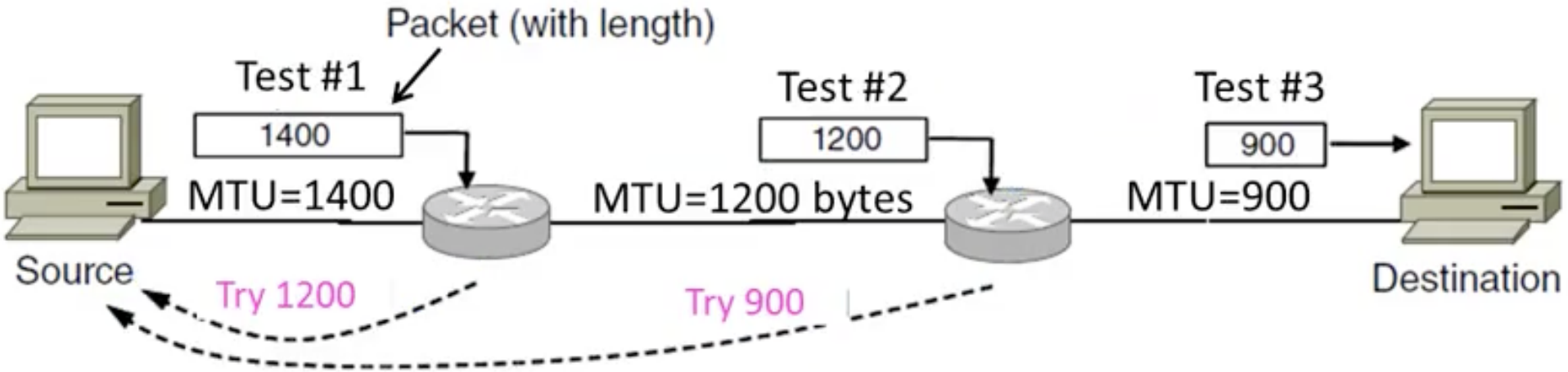
IPv4 fragmentation procedure

- Fragmentation
  - Router breaks up datagram in size that output link can support
  - Copies IP header to pieces
  - Adjust length on pieces
  - Set offset to indicate position
  - Set MF (More fragments) flag on pieces except the last
  - Re-compute checksum
- Re-assembly
  - Receiving host uses identification field with MF and offsets to complete the datagram.
- Fragmentation of fragments also supported

Taken from [TCP/IP Protocol Suite by Behroze Forouzan]



Path MTU Discovery procedure



- Host
  - Sends a big packet to test whether all routers in path to the destination can support or not
  - Set DF (Do not fragment) flag
- Routers
  - Drops the packet if it is too large (as DF is set)
  - Provides feedback to Host with ICMP message telling the maximum supported size

How can we use this for evil?

A: Send fragments that overlap.

B: Send many tiny fragments, none of which have offset 0.

C: Send segments that when assembled, are bigger than the maximum IP datagram.

D: More than one of the above.

E: Nah, networks (and operating systems) are too robust for this to cause problems.

IP Fragmentation Attacks …

https://en.wikipedia.org/wiki/IP_fragmentation_attack

IP fragmentation exploits [edit]

IP fragment overlapped [edit]

The IP fragment overlapped exploit occurs when two fragments contained within the same IP datagram have offsets that indicate that they overlap each other in positioning within the datagram. This could mean that either fragment A is being completely overwritten by fragment B, or that fragment A is partially being overwritten by fragment B. Some operating systems do not properly handle fragments that overlap in this manner and may throw exceptions or behave in other undesirable ways upon receipt of overlapping fragments. This is the basis for the teardrop Denial of service attacks.

IP fragmentation buffer full [edit]

The IP fragmentation buffer full exploit occurs when there is an excessive amount of incomplete fragmented traffic detected on the protected network. This could be due to an excessive number of incomplete fragmented datagrams, a large number of fragments for individual datagrams or a combination of quantity of incomplete datagrams and size/number of fragments in each datagram. This type of traffic is most likely an attempt to bypass security measures or Intrusion Detection Systems by intentional fragmentation of attack activity.

IP fragment overrun [edit]

The IP Fragment Overrun exploit is when a reassembled fragmented datagram exceeds the declared IP data length or the maximum datagram length. By definition, no IP datagram should be larger than 65,535 bytes. Systems that try to process these large datagrams can crash, and can be indicative of a denial of service attempt.

IP fragment overwrite [edit]

Overlapping fragments may be used in an attempt to bypass Intrusion Detection Systems. In this exploit, part of an attack is sent in fragments along with additional random data; future fragments may overwrite the random data with the remainder of the attack. If the completed datagram is not properly reassembled at the IDS, the attack will go undetected.

IP fragment too many datagrams [edit]

The Too Many Datagrams exploit is identified by an excessive number of incomplete fragmented datagrams detected on the network. This is usually either a denial of service attack or an attempt to bypass security measures. An example of "Too Many Datagrams", "Incomplete Datagram" and "Fragment Too Small" is the Rose Attack.[1]

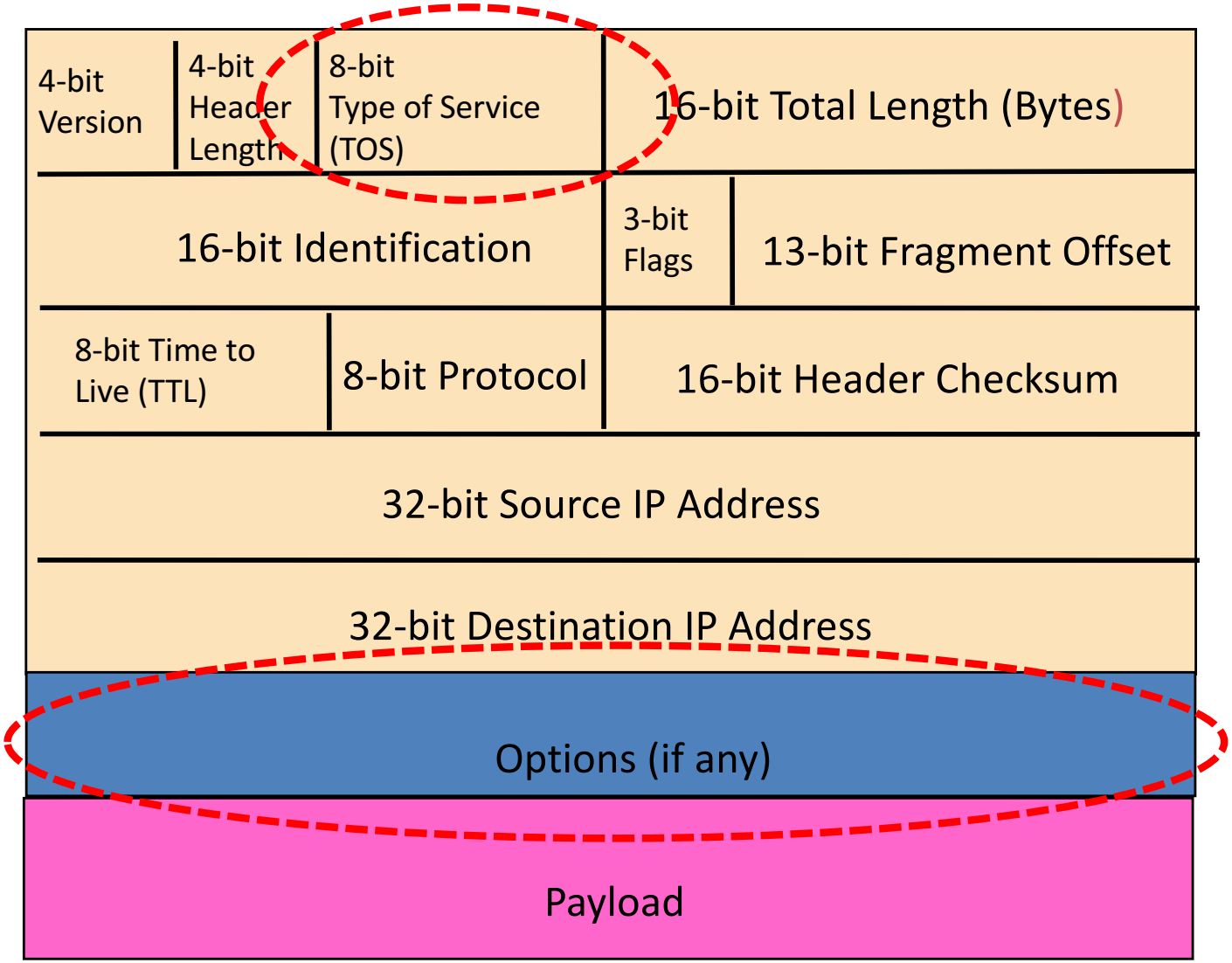
IP fragment incomplete datagram [edit]

This exploit occurs when a datagram can not be fully reassembled due to missing data. This can indicate a denial of service attack or an attempt to defeat packet filter security policies.

IP fragment too small [edit]

An IP Fragment Too Small exploit is when any fragment other than the final fragment is less than 400 bytes, indicating that the fragment is likely intentionally crafted. Small fragments may be used in denial of service attacks or in an attempt to bypass security measures or detection.

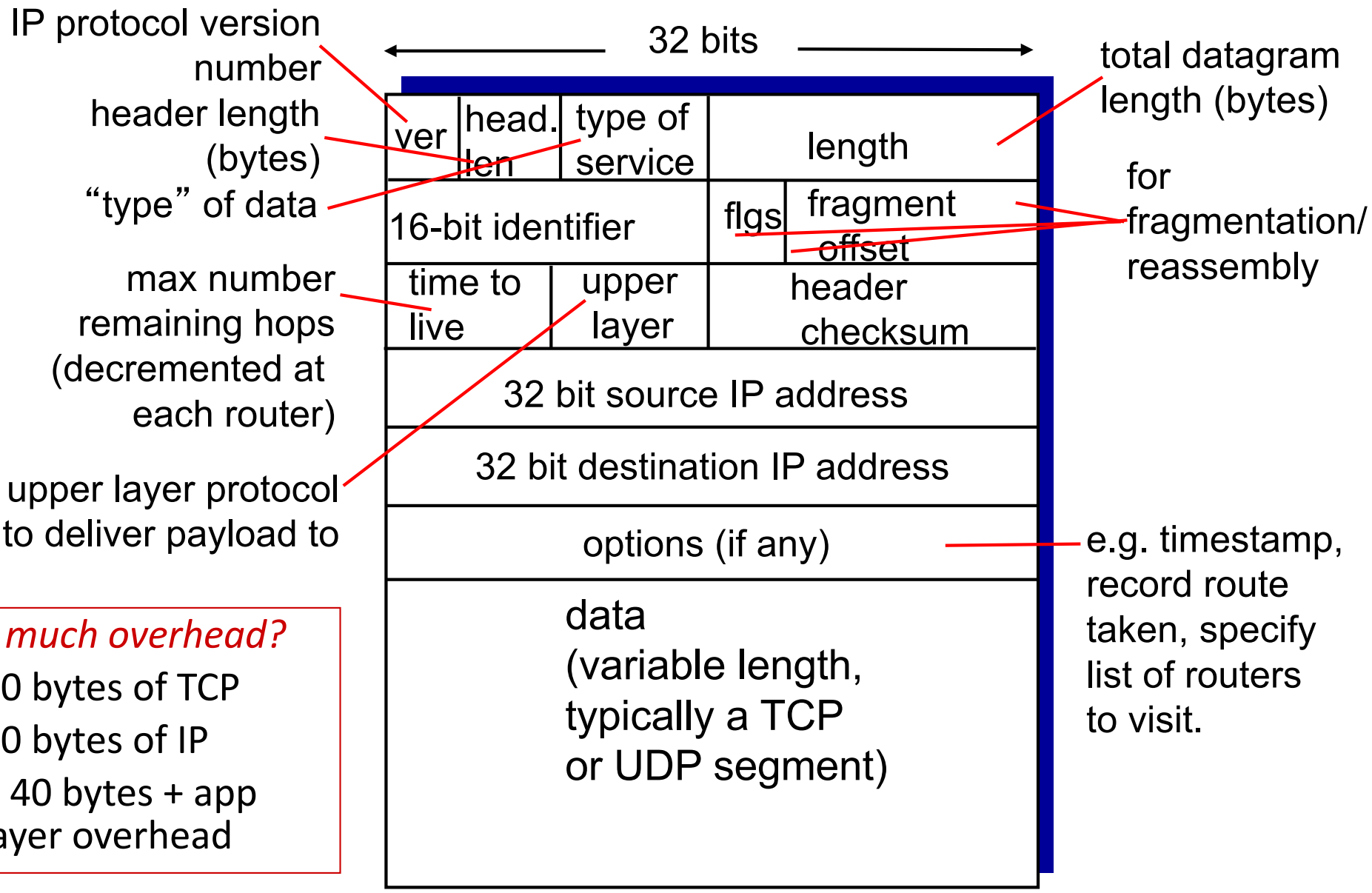
Fields for Special Handling



Special Handling

- "Type of Service", or "Differentiated Services Code Point (DSCP)" (8 bits)
  - Allow packets to be treated differently based on needs
  - E.g., low delay for audio, high bandwidth for bulk transfer
  - Has been redefined several times
- Options (not often used)

RECAP: IP datagram format



how much overhead?

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