# Network Layer

Dr. Raja Vara Prasad,
IIIT Sri City, Chittoor

# Inside a Router

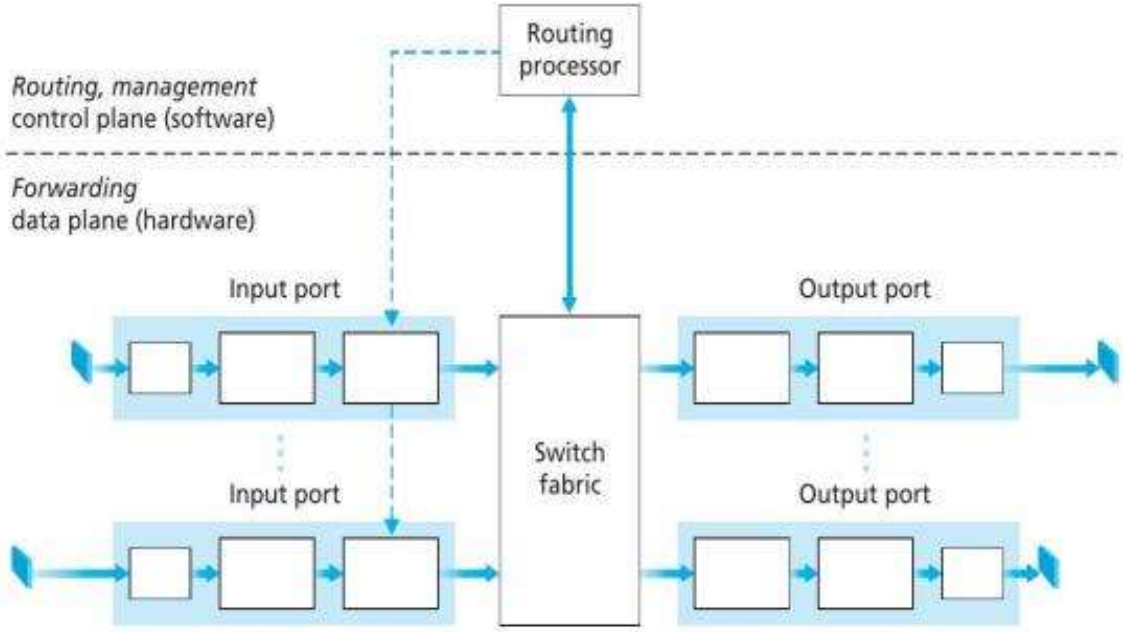



Figure 4.6 ♦ Router architecture

- Input port
- Switching fabric
- Output port
- Routing processor

# Inside a Router

***Input port:***

performs the physical layer function

performs link-layer functions

lookup function is also performed

Control packets (for example, packets carrying routing protocol information) are forwarded from an input port to the routing processor

***Switching fabric.***

connects the router's input ports to its output ports.

is completely contained within the router: a network inside of a network router!

***Output ports:***

stores packets received from the switching fabric

transmits these packets on the outgoing link by performing the necessary link-layer and physical-layer functions

***Routing processor.***

- executes the routing protocols
- maintains routing tables and attached link state information,
- computes the forwarding table for the router.
- performs the network management functions

# Inside a Router

**Router forwarding plane:**
router's input ports, output ports, and switching fabric together → forwarding function → always implemented in hardware: **router forwarding plane**

**Ex:**

- a 10 Gbps input link and a 64-byte IP datagram,
- the input port has only **51.2 ns** to process the datagram before another datagram may arrive.
- If *N* ports are combined on a line card (as is often done in practice), the datagram-processing **pipeline must operate *N* times faster**
- far too fast for software implementation

**Router's control functions**

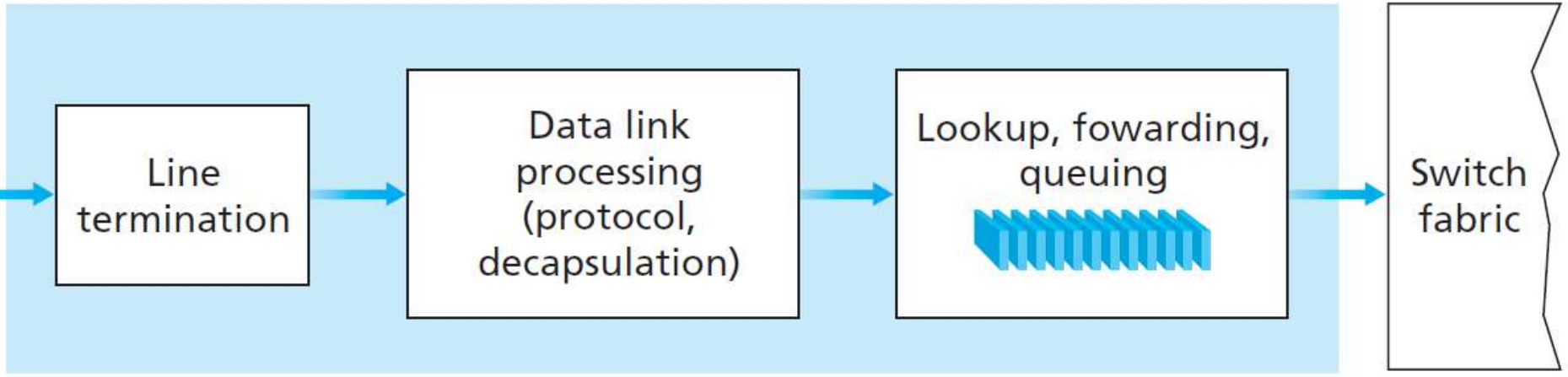
- executing the routing protocols, responding to attached links that go up or down, and performing management functions
- operate at the **millisecond** or second timescale
- **router control plane** functions are usually implemented in **software** and execute on the **routing processor**

# Inside a Router

1. Input processing
2. Switching
3. Output processing

**1. Input processing:**

- input port's line termination function and link-layer processing implement the physical and link layers for that individual input link
- lookup performed → Central to the router's operation → to look up the output port → Arriving packet will be forwarded via the switching fabric.



# Inside a Router

1. **Input processing:**

- a shadow copy typically stored at each input port
- forwarding table is copied from the routing processor to the line cards
- forwarding decisions → locally, at each input port --> without invoking the centralized routing processor on a per-packet basis →avoiding a centralized processing bottleneck.
- search through the forwarding table → for the longest prefix match
- at Gigabit transmission rates → lookup → nanoseconds
- techniques beyond a simple linear search through a large table
- Special attention: memory access times → embedded on-chip DRAM, faster SRAM, Ternary Content Address Memories (TCAMs) using the fabric

# Inside a Router

**1. Input processing:**

- a packet may be temporarily blocked from entering the switching fabric if packets from other input ports are currently using the fabric
- will be queued at the input port and then scheduled to cross the fabric at a later point in time

Other important aspects of Input processing:

(1) physical- and link-layer processing must occur, as discussed above;

(2) the packet's version number, checksum and time-to-live field must be checked and the latter two fields rewritten;

(3) counters used for network management (number of IP datagrams received) must be updated.

# Inside a Router

Switching:
through this fabric that the packets are actually switched from an input port to an output port.

Three types of switching:

- *Switching via memory*
- *Switching via a bus.*
- *Switching via an interconnection network*

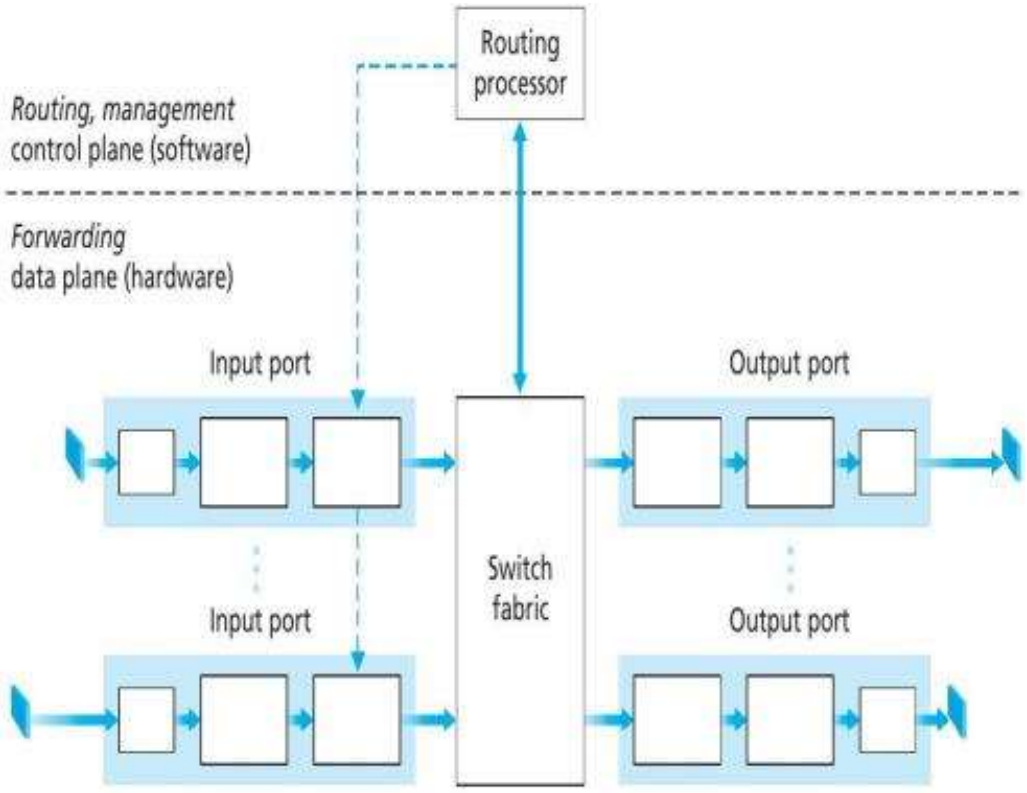


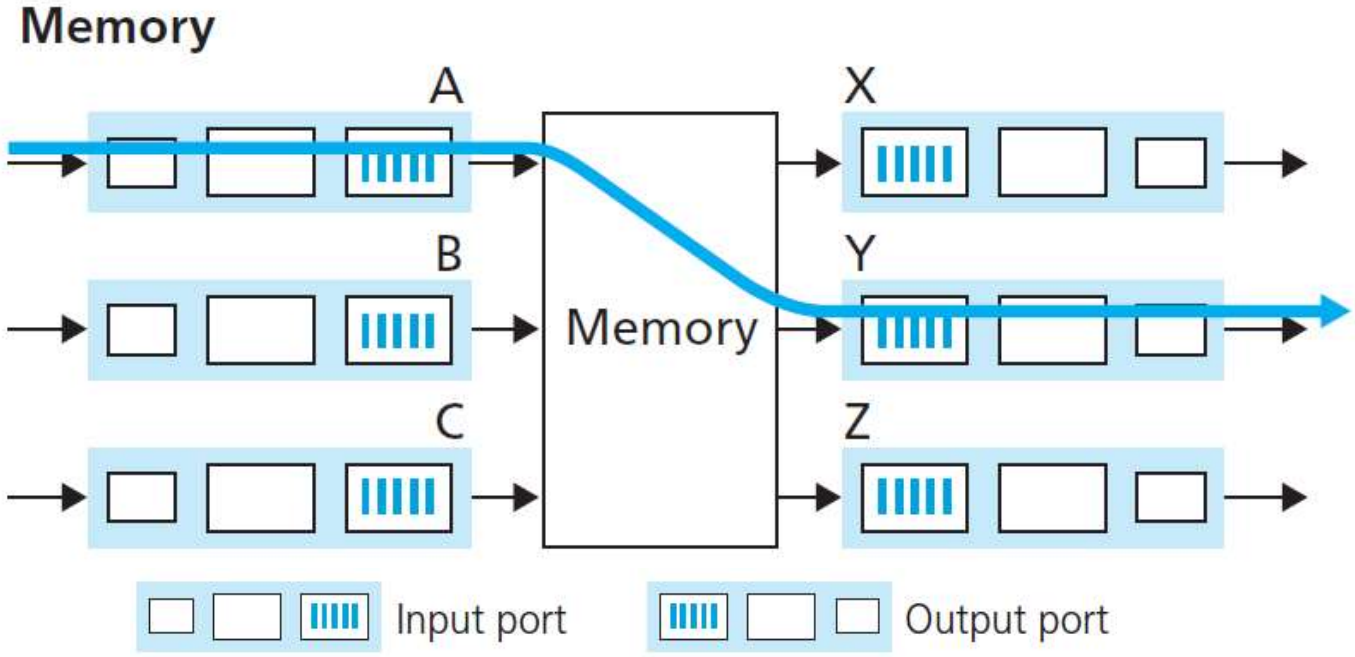

Figure 4.6 ♦ Router architecture

# Inside a Router: Switching

*Switching via memory*

- earliest routers – switching → under direct control of CPU
- *Input port signals the arrival of a packet → routing processor →Interrupt*
- *Processor completes lookup → copies packet → output buffer*
- *Present day routers* → processing on a line card
- shared-memory multiprocessors

*Ex:* Cisco's Catalyst 8500 series switches

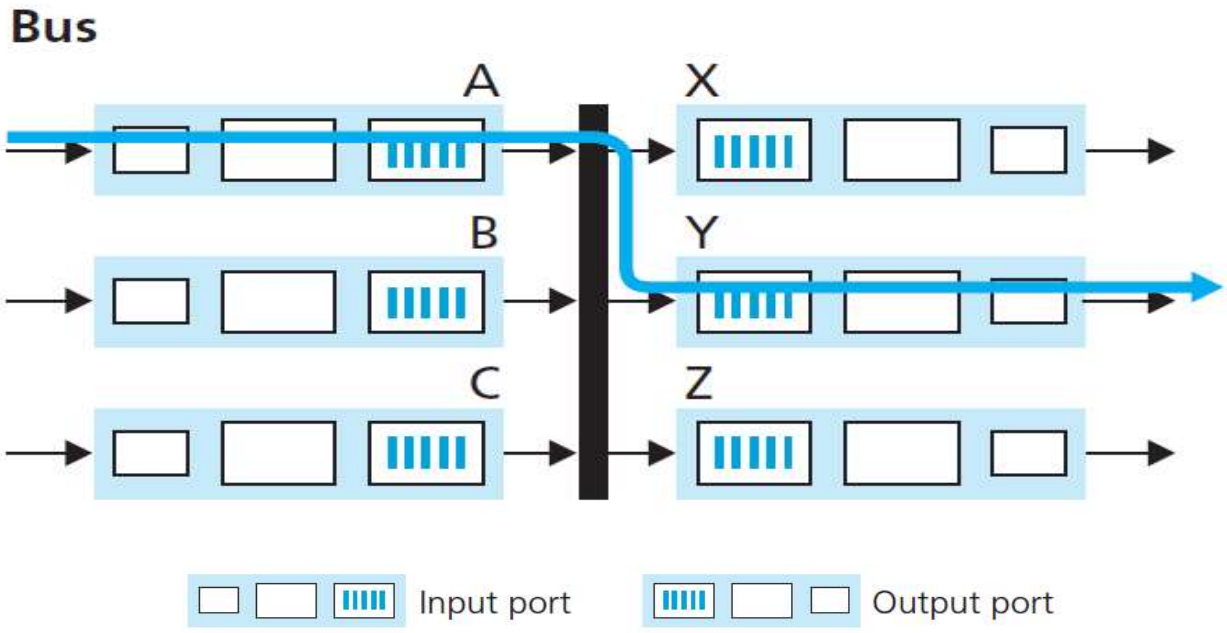


# Inside a Router: Switching:

***Switching via a bus:***

- an input port transfers a packet directly to the output port over a shared bus, without intervention by the routing processor
- input port pre-pend a switch-internal label → indicating the local output port → transmitting the packet onto the bus
- received by all output ports, but only the port that matches the label

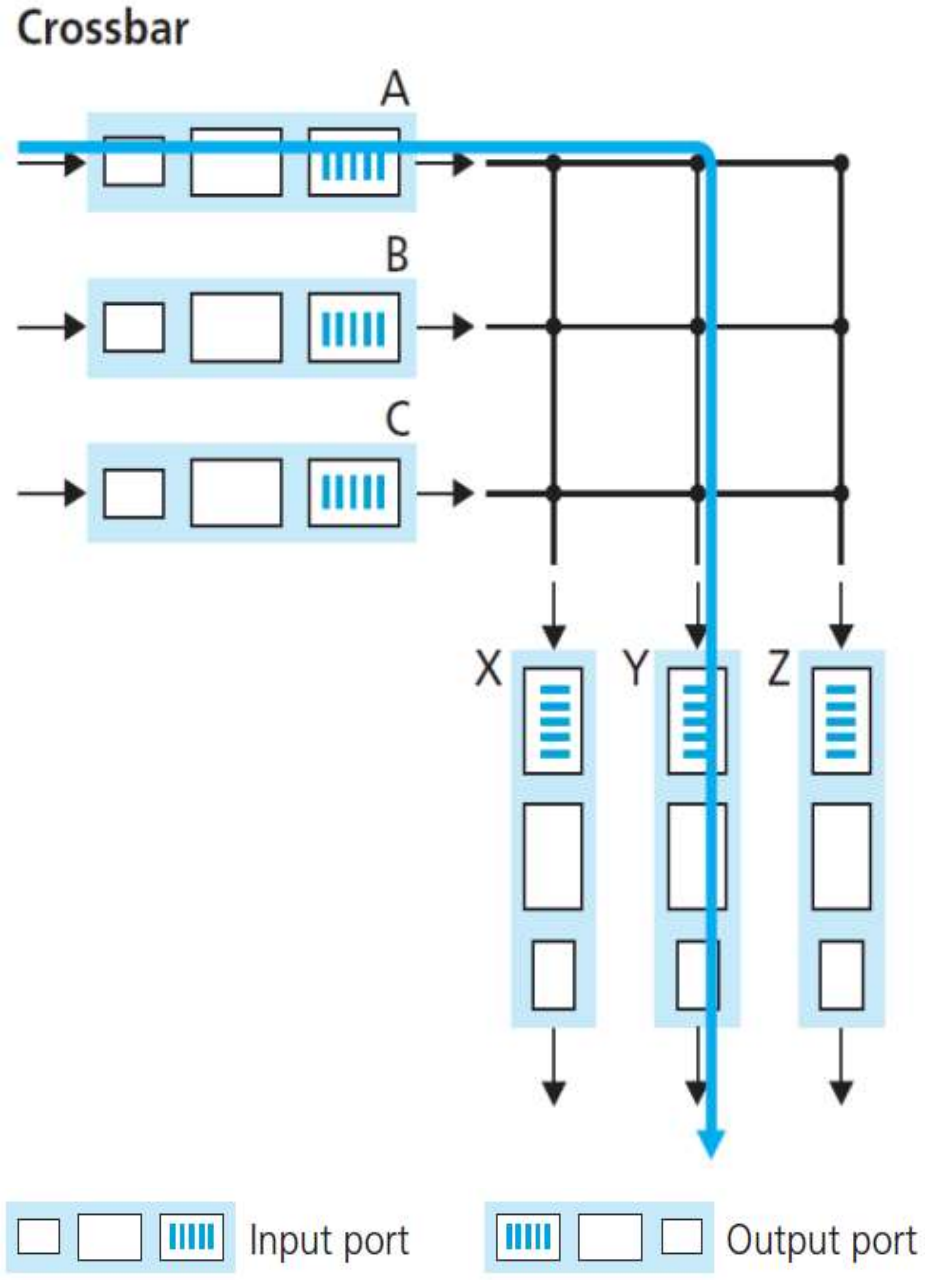
- Even multiple packets at input ports → one packet on bus
- switching speed of the router is limited to the bus speed



# Inside a Router: Switching

***Switching via an interconnection network:***

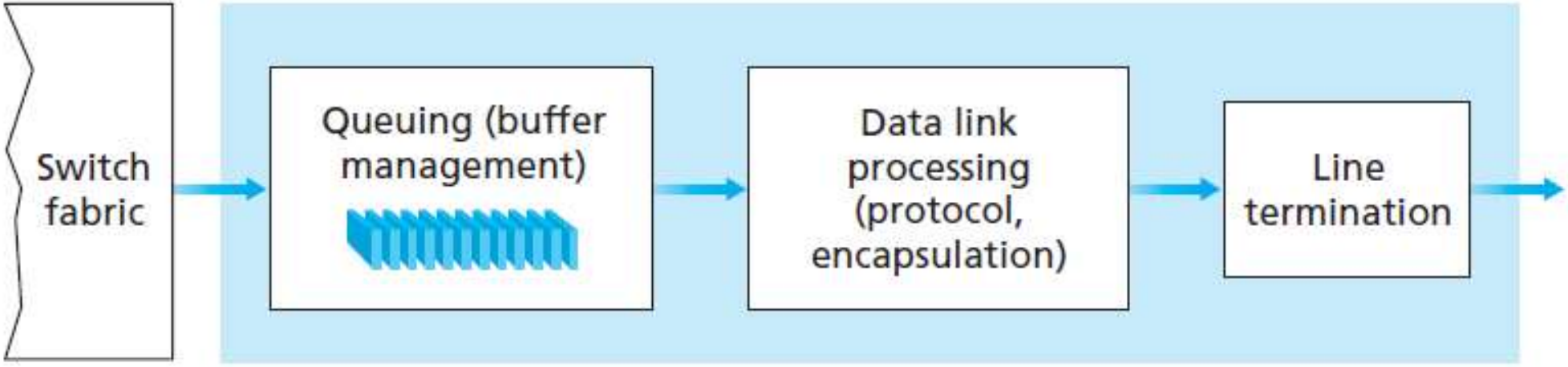
- more sophisticated interconnection network
- crossbar switch is an interconnection network consisting of 2*N* buses that connect *N* input ports to *N* output ports
- Each vertical bus intersects each horizontal bus at a crosspoint → can be opened or closed at any time by the switch fabric controller
- crossbar networks are capable of forwarding multiple packets in parallel
- if two packets from two different input ports → to the same output port → one will have to wait at the input



# Inside a Router

Output processing:

- takes stored packets in the output port's memory → transmits them over the output link
- selecting and de-queueing packets for transmission
- performing the needed link layer and physical-layer transmission functions



# Inside a Router: Output processing

**Queueing at input and output ports:**

- packet queues may form at both the input ports *and* the output ports
- extent of queueing depend on
  - →the traffic load → the relative speed of the switching fabric,
  - →the line speed
- queues grow large → router's memory exhaust → **packet loss** will occur when no memory is available to store arriving packets
- an identical input and output transmission rate of $R_{line}$ packets/sec
- $R_{switch}$ rate at which packets can be moved from input to output port
- if $R_{switch} = N^* R_{line}$ negligible queuing will occur at input ports
- If all packets at N input ports are destined to same output port ?
- output port can transmit only a single packet in a unit of time
- *N* arriving packets will have to queue for transmission over the outgoing link.
- number of queued packets can grow large enough → exhaust available memory at the output port → packets are dropped

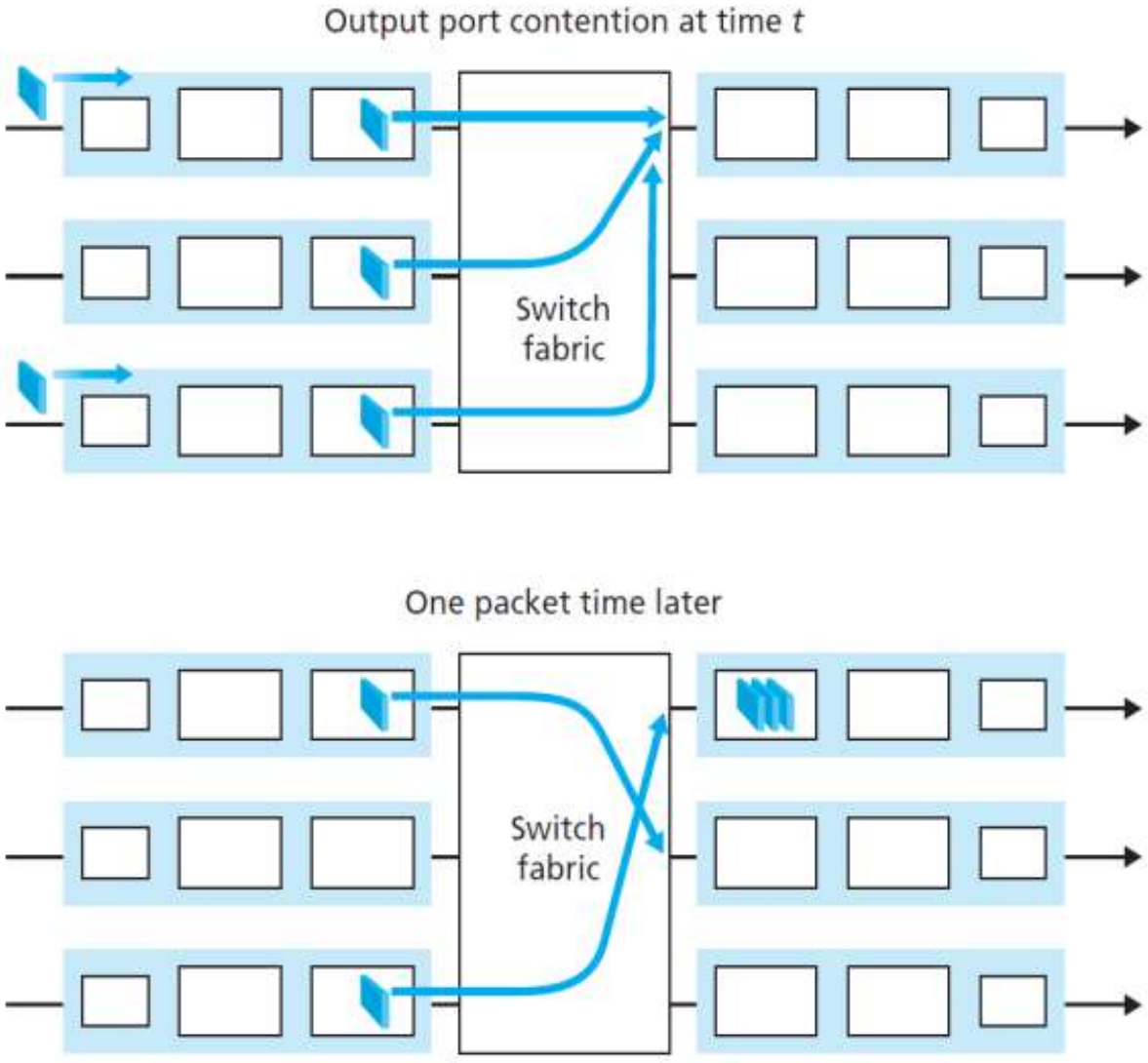
# Inside a Router: Output processing

**Queueing at input and output ports:**

- packet queues may form at both the input ports *and* the output ports
- extent of queueing depend on
  - →the traffic load → the relative speed of the switching fabric,
  - →the line speed
- queues grow large → router's memory exhaust → **packet loss** will occur when no memory is available to store arriving packets
- an identical input and output transmission rate of $R_{line}$ packets/sec
- $R_{switch}$ rate at which packets can be moved from input to output port
- if $R_{switch} = N^* R_{line}$ negligible queuing will occur at input ports
- If all packets at N input ports are destined to same output port ?
- output port can transmit only a single packet in a unit of time
- *N* arriving packets will have to queue for transmission over the outgoing link.
- number of queued packets can grow large enough → exhaust available memory at the output port → packets are dropped

# Inside a Router: Output processing:

- **packet scheduler** at the output port must choose one packet among those queued for transmission
- first-come-first-served (FCFS)
- weighted fair queuing (WFQ) → shares the outgoing link fairly among the different end-to-end connections that have packets queued for transmission
- no enough memory to buffer an incoming packet either drop the arriving packet or remove one or more already-queued packets
- **Random Early Detection -** probabilistic marking/dropping functions

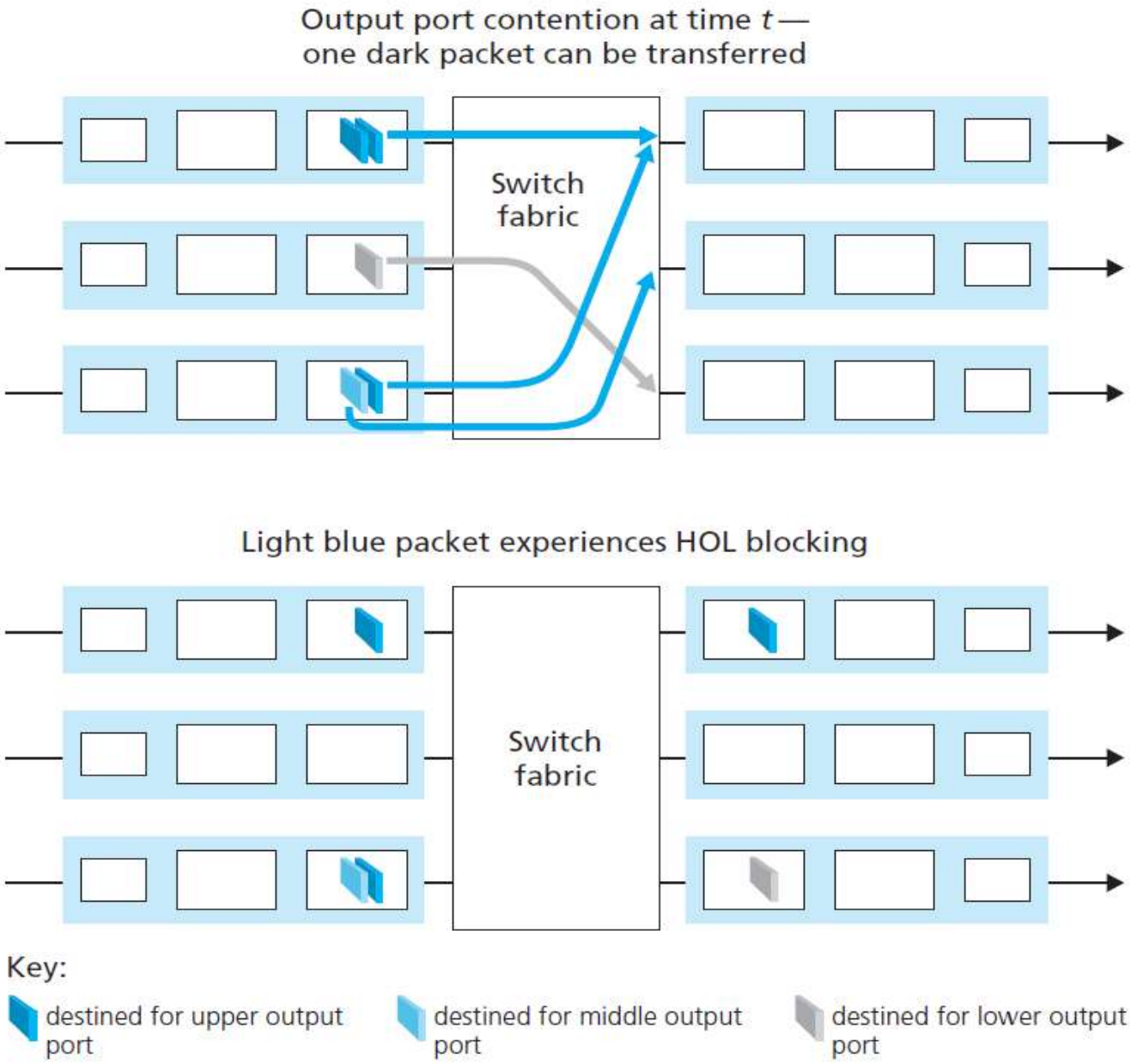


# Inside a Router: Output processing:

what if the Switch fabric is not fast enough: → packet queuing at the input ports

Assume:
(1) all link speeds are identical
(2) one packet can be transferred from any one input port to a given output port in the same amount of time it takes for a packet to be received on an input link
(3) packets are moved from a given input queue to their desired output queue in an FCFS manner
→Multiple packets can be transferred in parallel, as long as their output ports are different
→ if two packets of two input queues are destined for the same output queue one of the packets will be blocked and must wait at the input queue.
→ **head-of-the-line (HOL) blocking** → queue will grow to unbounded length

# Inside a Router: Output processing:



# Inside a Router: Routing plane

- fully resides and executes in a routing processor within the router
- network-wide routing control plane → decentralized

→ with different pieces executing at different routers and interacting by sending control messages to each other

New router control plane architectures

→ part of the control plane is implemented in the routers along with the data plane
→ part of the control plane can be implemented externally to the router
→ A well-defined API dictates how these two parts interact and communicate with each other

**Software Defined Networking (SDN)**

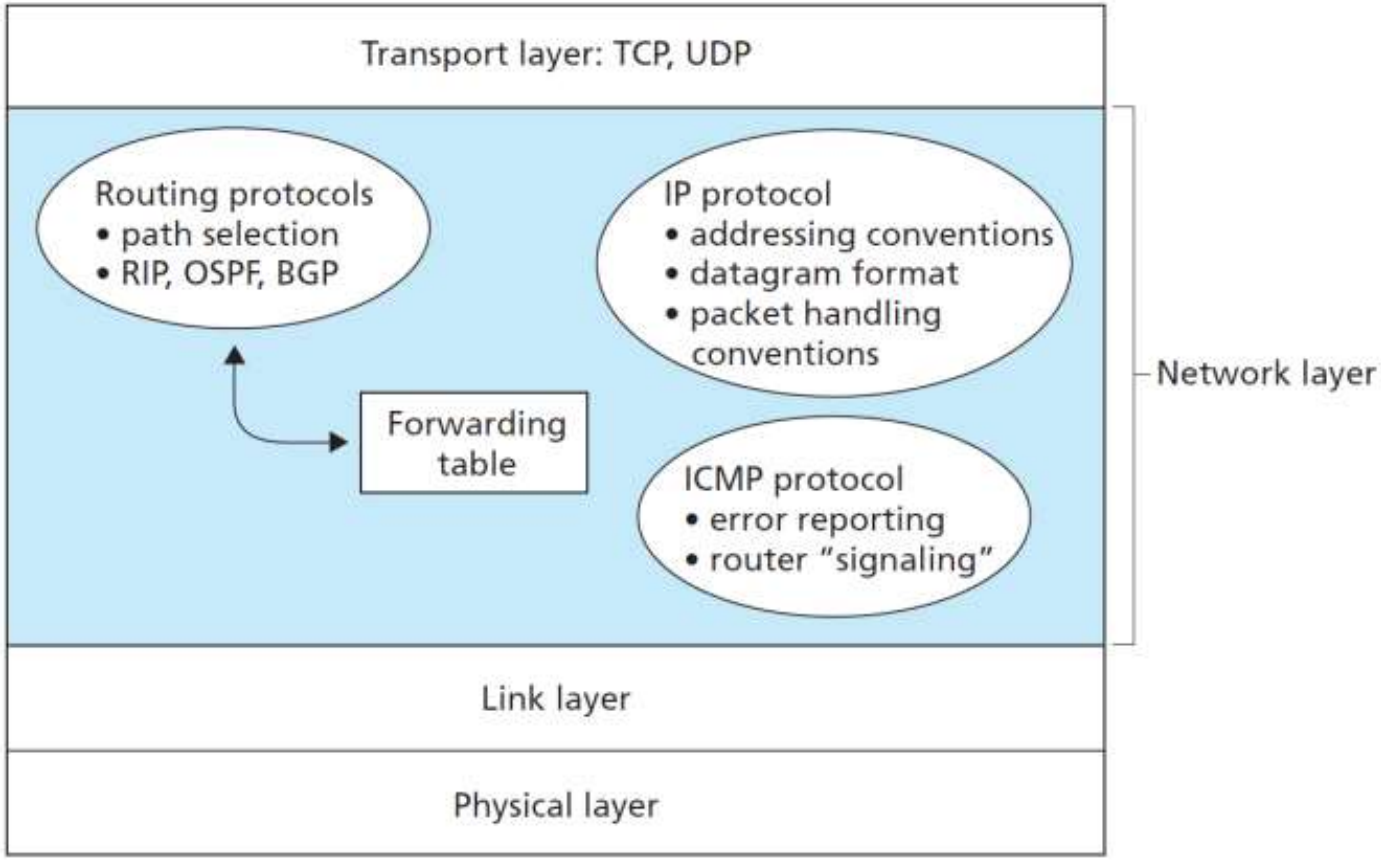
- separating the software control plane from the hardware data plane
- allowing different customized control planes to operate over fast hardware data planes

# Internet Protocol

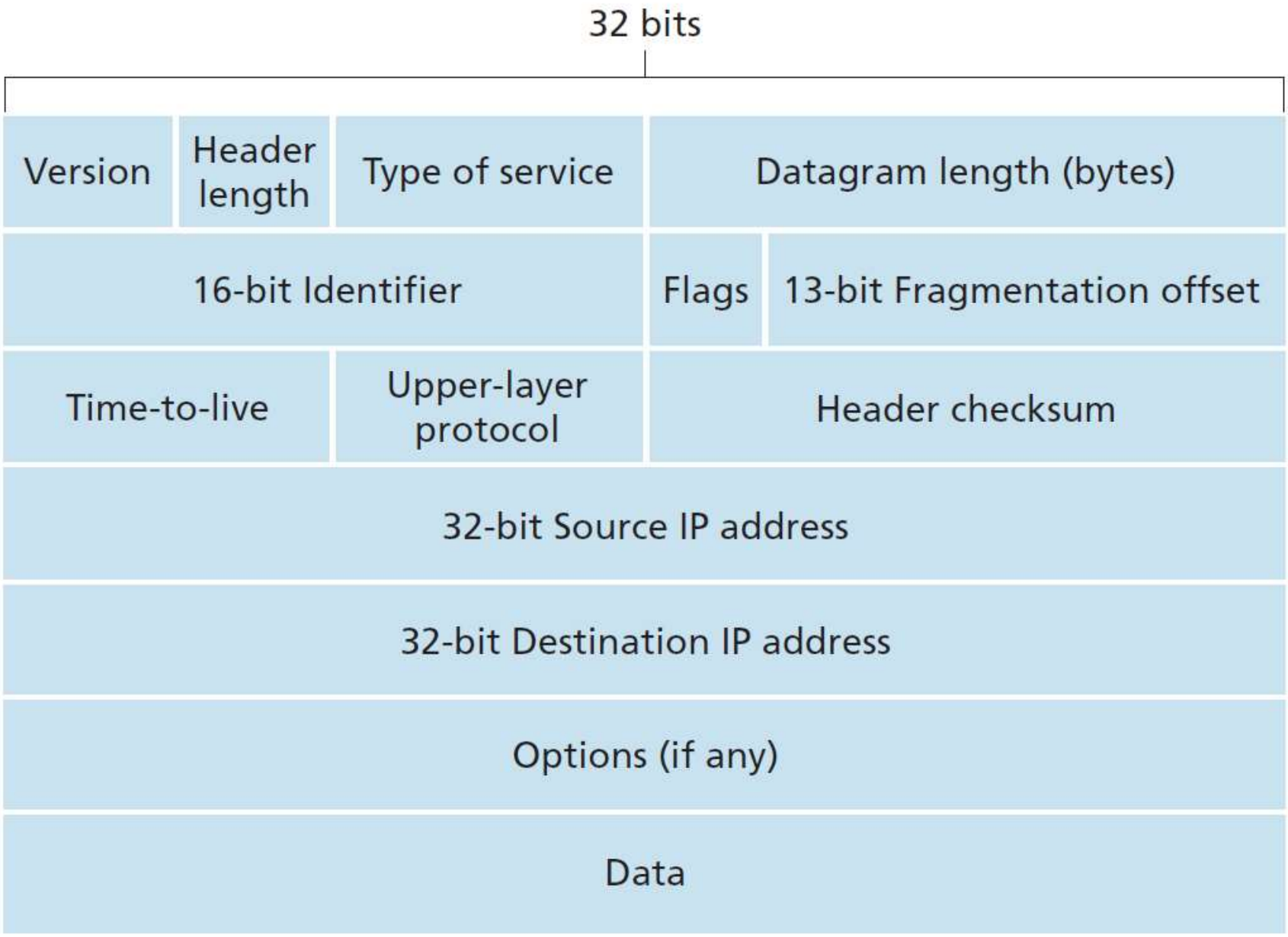
Important modules:

1. IP Protocol
2. Routing component
3. Internet Control Message Protocol

IPV4 and IPV6



# Internet Protocol



# Internet Protocol

***Version number:*** 4 bits specify the IP protocol version
router can determine how to interpret the remainder of the IP datagram

***Header length:*** IPv4 datagram can contain a variable number of options
→ 4 bits are needed → where in the IP datagram the data actually begins
→Most of IP datagrams do not contain options → 20-byte header

***Type of service:*** to allow different types of IP datagrams
→differentiating datagrams requiring low delay, high throughput, or reliability
→ distinguish real-time and non-real time datagrams

***Datagram length:*** total length of the IP datagram → header + data
→16 bits long → theoretical maximum size: 65,535 bytes → rarely > than 1500 bytes

***Identifier, flags, fragmentation offset:*** *fields required for* IP fragmentation

**Time to Live:**
→ to ensure that datagrams do not circulate forever: long-lived routing loop
→ field is decremented by one each time

# Internet Protocol

***Protocol:***

field is used  when an IP datagram reaches its final destination.

value indicates → specific transport-layer protocol to which the data portion of this IP datagram should be passed.

Ex:value-6 indicates → data portion is passed to TCP,  17 indicates to UDP.

***Header checksum:***

→ aids a router in detecting bit errors in a received IP datagram.

→ computed by treating each 2 bytes in the header as a number and summing these numbers using 1s complement arithmetic.

→ stored in the checksum field and compares with router computed checksum

→ detects an error condition → Routers typically discard datagrams

→ checksum must be recomputed and stored again at each router → TTL field, and possibly the options field as well, may change.

TCP already has checksum, why do datagrams need checksum?

***Source and destination IP addresses:***

it inserts  its IP address into the source IP address field and inserts the address of the ultimate destination into the destination IP address field

# Internet Protocol

***Options:***

→ allow an IP header to be extended.
→ existence of options does complicate
→ datagram headers can be of variable length,
→ cannot determine a priori where the data field will start
→ amount of time needed to process an IP datagram at a router can vary greatly
→ IP options were dropped in the IPv6 header

***Data (payload):***

→ IP datagram contains the transport-layer segment to be delivered to the destination.
→ can carry other types of data → ICMP messages

***datagram carrying a TCP segment → each nonfragmented datagram carries a total of 40 bytes of header → 20 bytes of IP header plus 20 bytes of TCP header along with the application-layer message.

# Internet Protocol

**IP Datagram Fragmentation:**

→ Not all link-layer protocols can carry network-layer packets of the same size.

→ Some protocols can carry big datagrams→ other protocols carry only little packets.
Ex: Ethernet frames up to 1,500 bytes
wide-area links no more than 576bytes.

**maximum amount of data that a link-layer frame can carry
→ maximum transmission unit (MTU).
→ IP datagram is encapsulated within the link-layer frame for transport from one router to the next router
→ MTU link-layer protocol places a hard limit on the length of an IP datagram.
→ problem: each of the links along the route between sender and destination can use different link-layer protocols
→ each of these protocols can have different MTUs.

→ **Router with interconnects several links with different link layer MTU's may receive a datagram and outgoing link MTU my be smaller**

***squeeze this oversized IP datagram into the payload field of the link-layer frame**

Solution: Fragment the IP datagram into two or more smaller datagrams

# Internet Protocol - Fragmentation

**Fragment:**

→ IP datagram into two or more smaller IP datagrams
→ encapsulate each smaller IP datagrams in a separate link-layerframe;
→ send these frames over the outgoing link. Each smaller datagrams is a **fragment**.

- Fragments need to be reassembled before they reach destination.
- TCP and UDP: expecting to receive complete, unfragmented segments from nwk layer
- reassembling datagrams in the routers → significant complication damp router performance.
- datagram reassembly in the end systems rather than in network routers.

# Internet Protocol - Fragmentation

*Identification flag,* and *fragmentation offset* fields in the IP datagram header:

→destination host to determine whether it has received the last fragment

→how the fragments it has received should be pieced back together to form the original datagram

→ sending host stamps the datagram with an identification number When a router needs to fragment a datagram

→ each resulting datagram is stamped with the source address, destination address, and identification number of the original datagram.

→ destination receives a series of datagrams

→ can examine the identification numbers of the datagrams

→ determine which of the datagrams are actually fragments of the same larger datagram

→ to make sure, if Destination host has received last fragment of the original datagram,

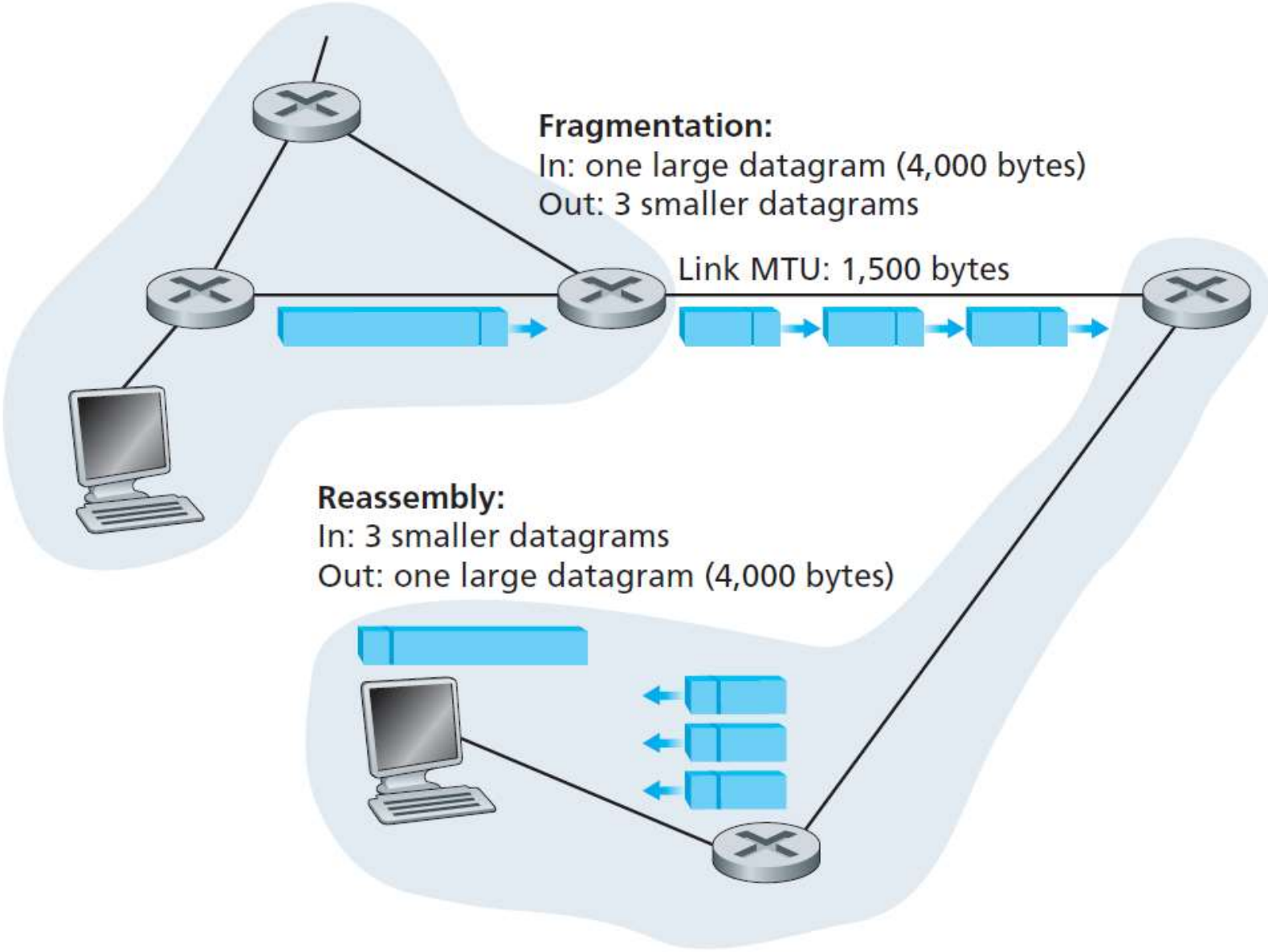
**the last fragment has a flag bit set to 0

**other fragments have this flag bit set to 1

→ a fragment is missing or reassemble the fragments in their proper order

*** the offset field is used to specify where the fragment fits within the original IP datagram.

# Internet Protocol - Fragmentation



# Internet Protocol - Fragmentation

| Fragment | Bytes | ID | Offset | Flag |
|---|---|---|---|---|
| 1st fragment | 1,480 bytes in the data field of the IP datagram | identification = 777 | offset = 0 (meaning the data should be inserted beginning at byte 0) | flag = 1 (meaning there is more) |
| 2nd fragment | 1,480 bytes of data | identification = 777 | offset = 185 (meaning the data should be inserted beginning at byte 1,480. Note that 185 · 8 = 1,480) | flag = 1 (meaning there is more) |
| 3rd fragment | 1,020 bytes (= 3,980–1,480–1,480) of data | identification = 777 | offset = 370 (meaning the data should be inserted beginning at byte 2,960. Note that 370 · 8 = 2,960) | flag = 0 (meaning this is the last fragment) |

** payload of the datagram is passed to the transport layer only after the IP layer has fully reconstructed the original IP datagram.

** If one or more of the fragments does not arrive at the destination, the incomplete datagram is discarded and not passed to the transport layer

**TCP will recover from this loss by having the source retransmit the data in the original datagram

# Internet Protocol - Fragmentation

Challenges:
complicates routers and end systems → to accommodate datagram fragmentation and reassembly

→ fragmentation can be used to create lethal DoS attacks, attacker sends a series of bizarre and unexpected fragments.

Ex: Jolt2 attack → attacker sends a stream of small fragments to the target host, none of which has an offset of zero. The target can collapse as it attempts to rebuild datagrams out of the degenerate packets.

→ Another class of exploits:
sends overlapping IP fragments, fragments whose offset values are set so that the fragments do not align properly.

IPv6, does away with fragmentation altogether, thereby streamlining IP packet processing and making IP less vulnerable to attack.