Introduction to Computer Networks

CS640

IP Introduction

https://pages.cs.wisc.edu/~mgliu/CS640/S26/index.html

Ming Liu
mgliu@cs.wisc.edu

Outline

- Last
  - Reliable transmission at L2
- Today
  - IP Introduction
- Announcements
  - Lab2 due on 03/03/2026

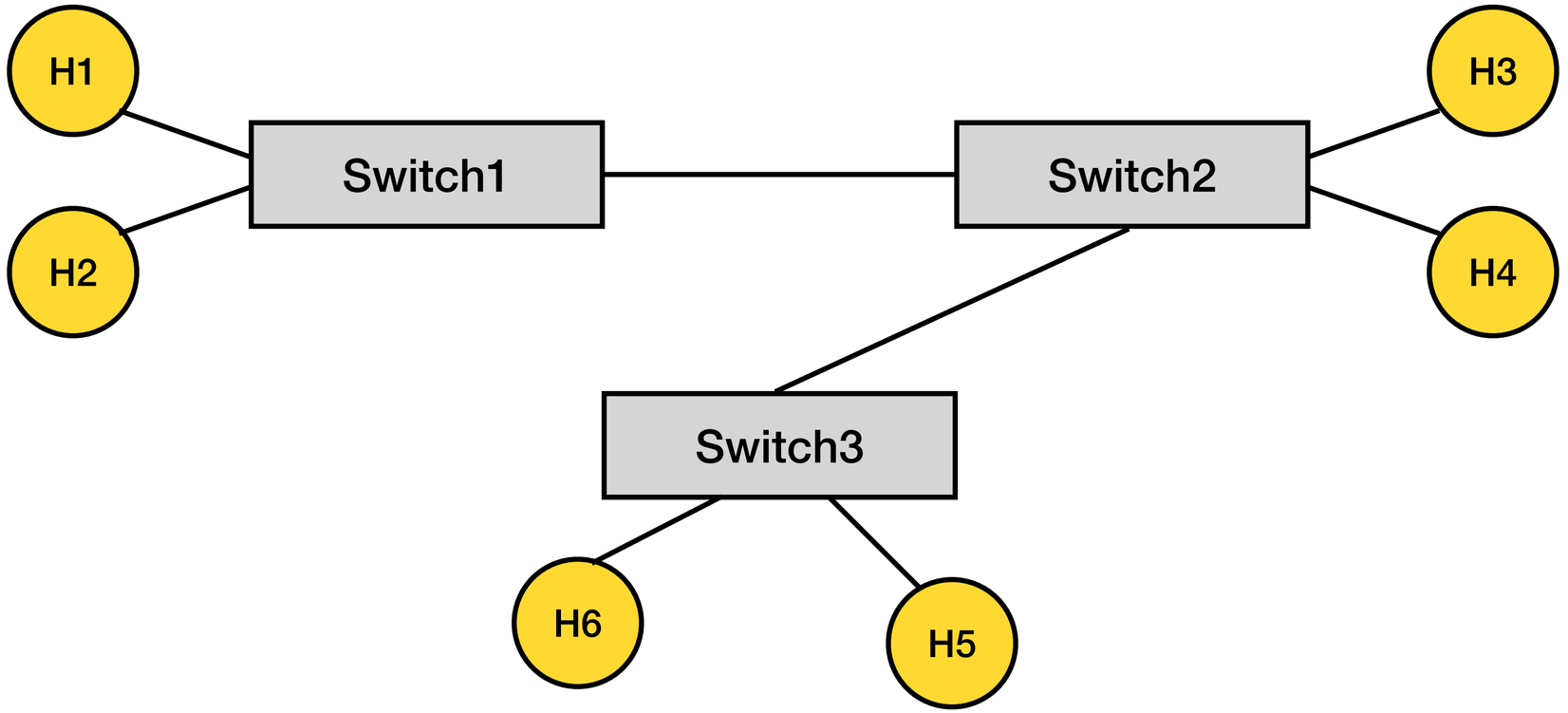
Recap

- Key Questions
  - Why is it hard to achieve reliable transmission?
  - How do we design a reliable transmission mechanism?

- Terminology
  - Acknowledgement and timeout
  - Stop-and-Wait
  - Logical channels
  - Sliding window

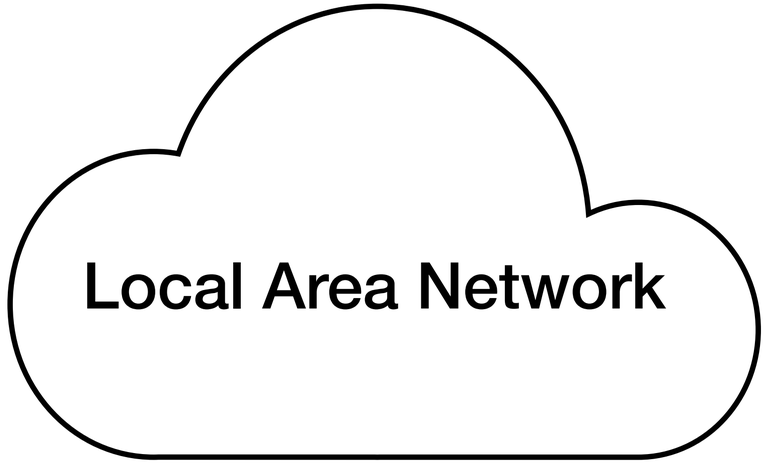
Switched Local Area Network

- So far, we are able to build a small-scaled switched network



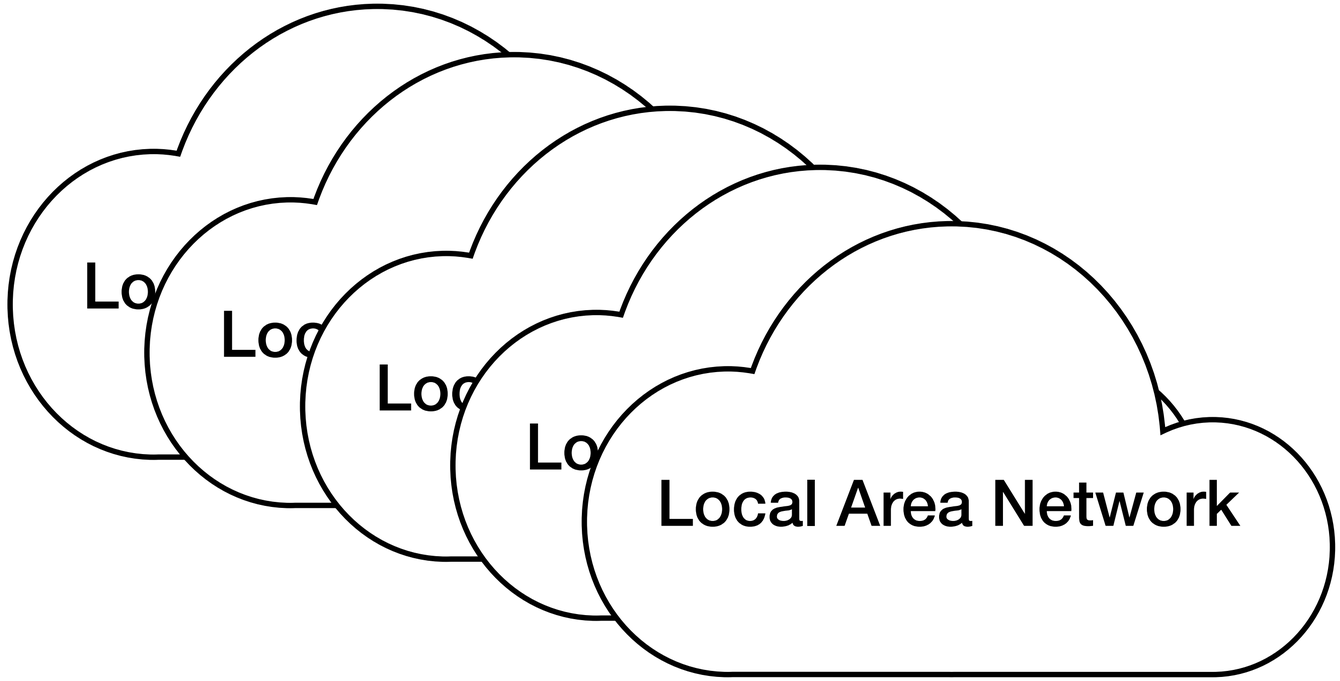
Switched Local Area Network

- So far, we are able to build a small-scaled switched network



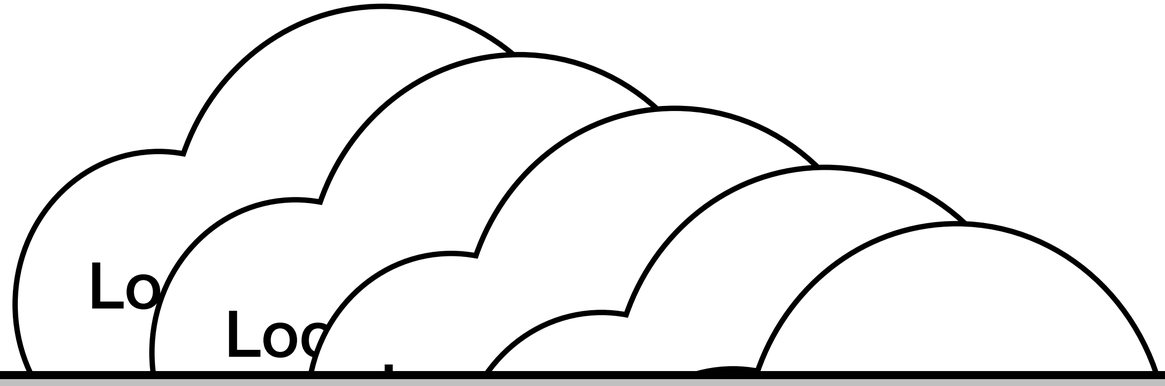
Switched Local Area Network

- So far, we are able to build a small-scaled switched network



Switched Local Area Network

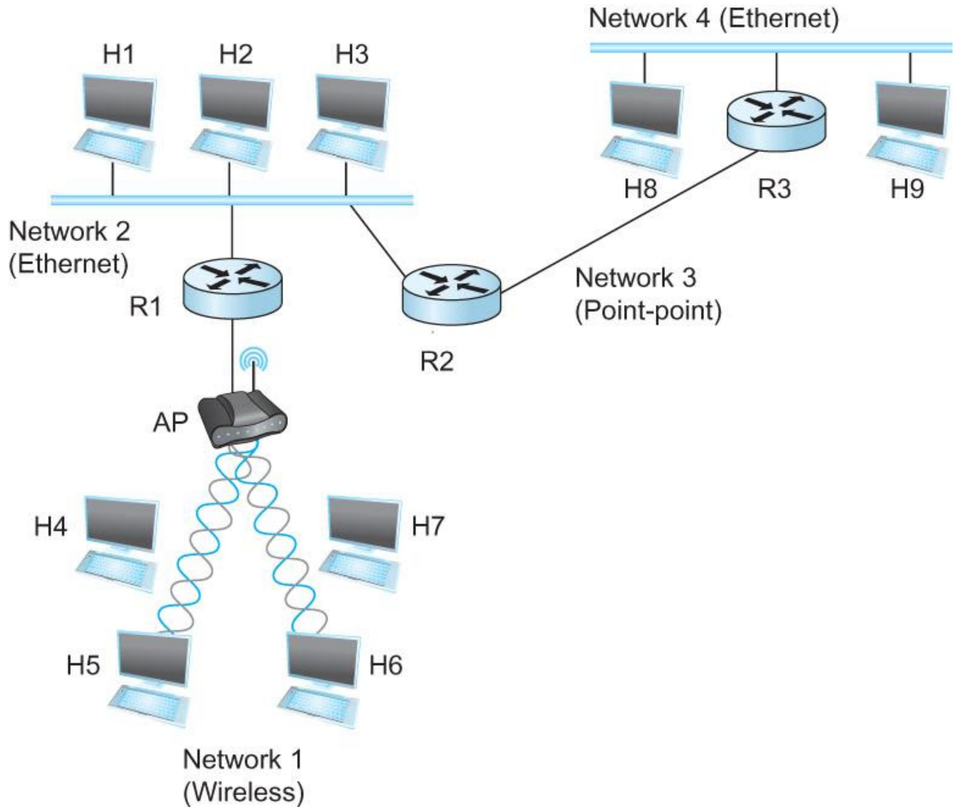
- So far, we are able to build a small-scaled switched network



How can we interconnect them together and enable the host-to-host communication at scale?

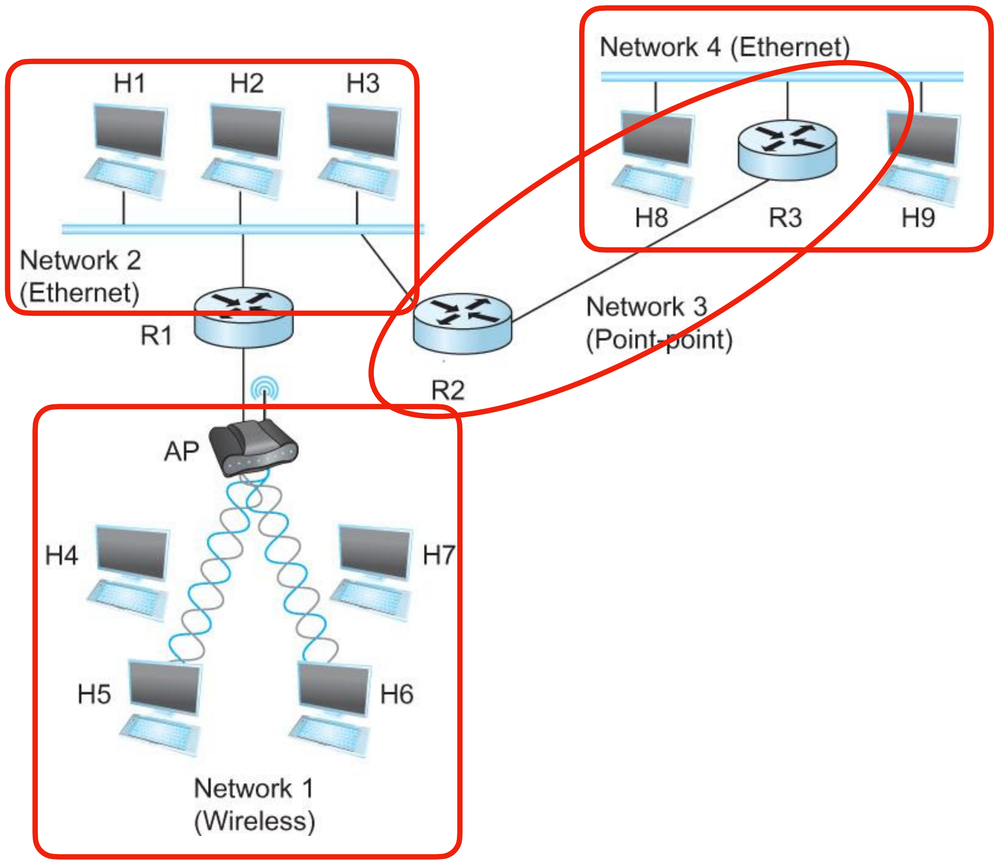
Inter-Networking

- An arbitrary collection of networks interconnected
  - Provide some sort of host-host delivery service

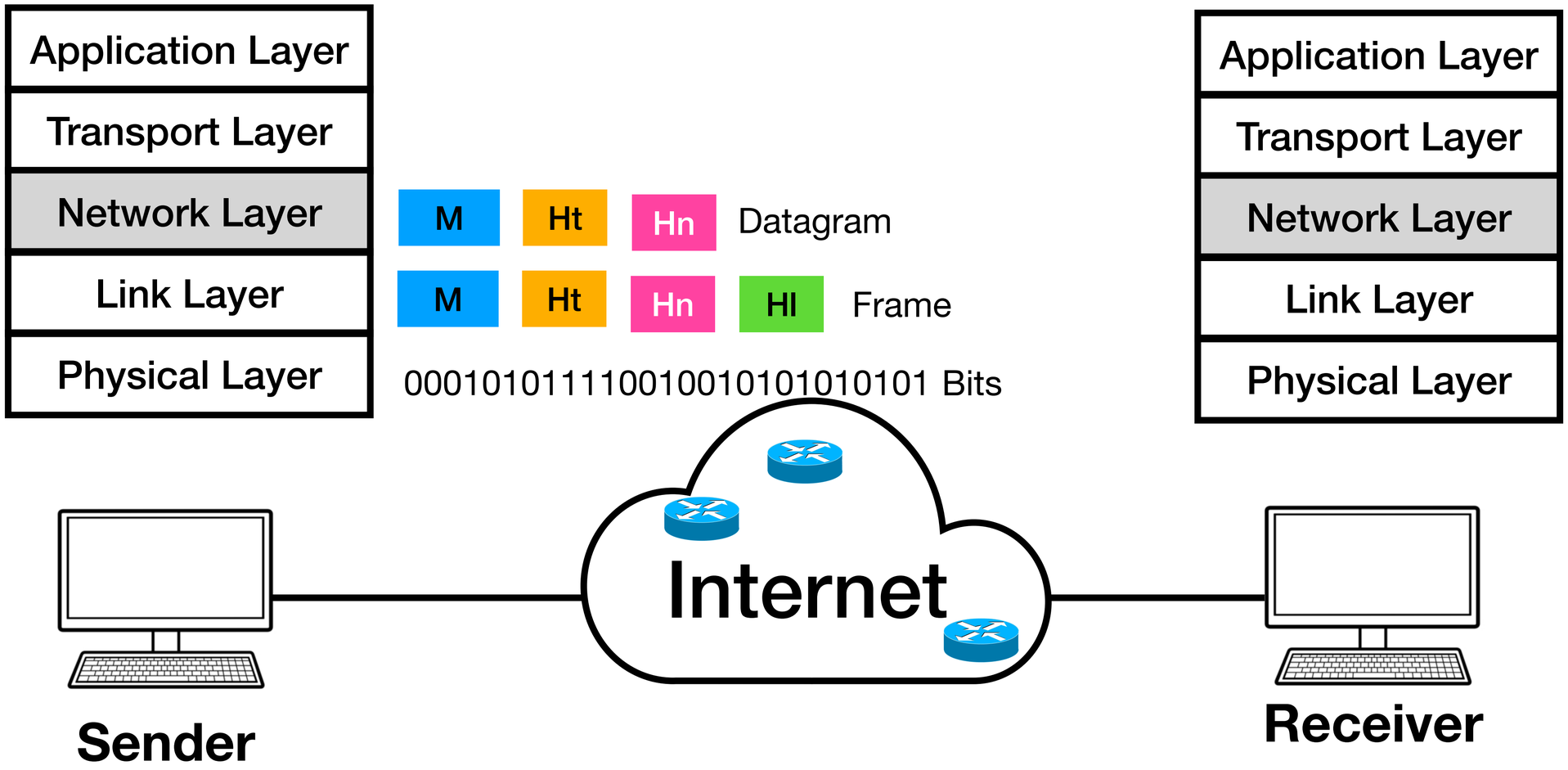


Inter-Networking

- An arbitrary collection of networks interconnected
  - Provide some sort of host-host delivery service



Recap: The Networking Layer

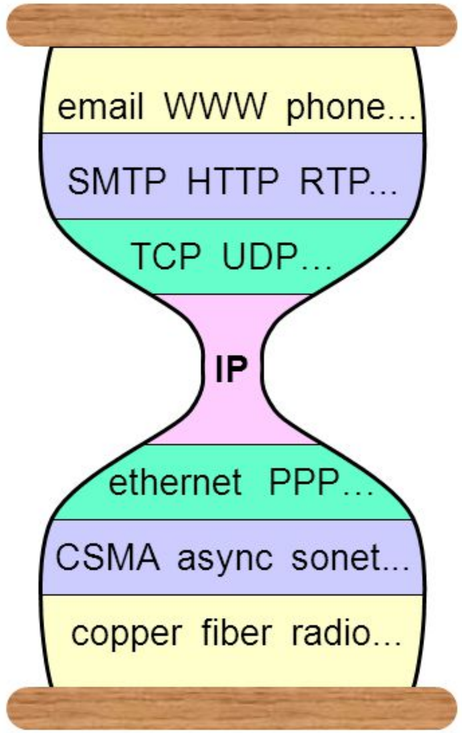


Internet Protocol (IP)

- Run over all the entities in a collection of networks

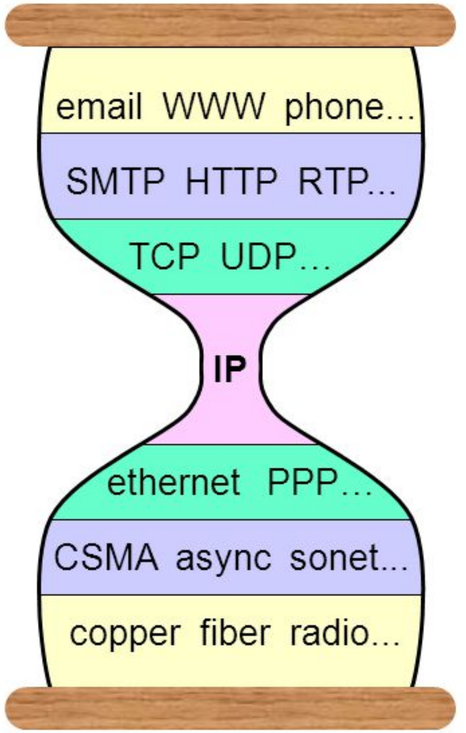
- Define the infrastructure that allows all networks to work as a single logical network!

The Hour Glass Model



The Hour Glass Model

- Hide underlying L2 technologies
  - E.g., Ethernet, Wi-Fi, etc.
- Support many different types of apps
  - E.g., Email, browsing, streaming, etc.
- Provide minimal functionalities
  - Addressing, forwarding, and routing



Why is IP so powerful?

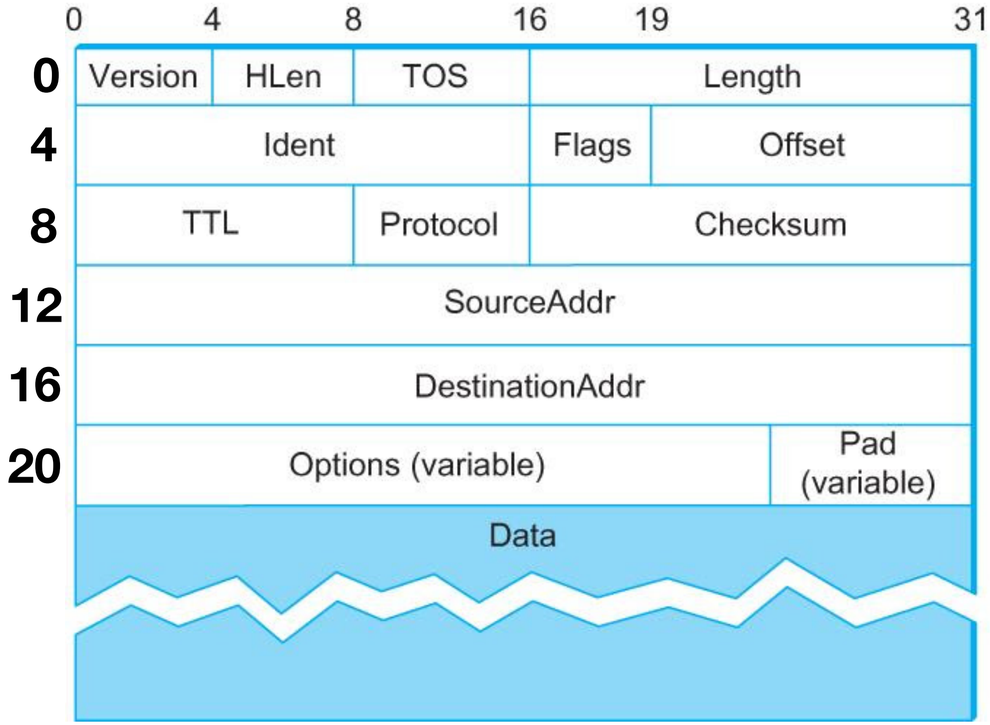
Best-effort host-to-host service model

Best-Effort Host-to-Host Service Model

- #1: use a unified header format
- #2: support heterogeneous networks
- #3: provide unreliable packet delivery

IP Packet Format

- #1: Version (4 bits)
  - IP version number, default: 4
- #2: HLen (4 bits)
  - The number of 32-bit words in the header
- #3: TOS (8 bits)
  - Type of service
  - 6-bit DSCP (differentiated service)
  - 2-bit ECN (Explicit Congestion Notification)

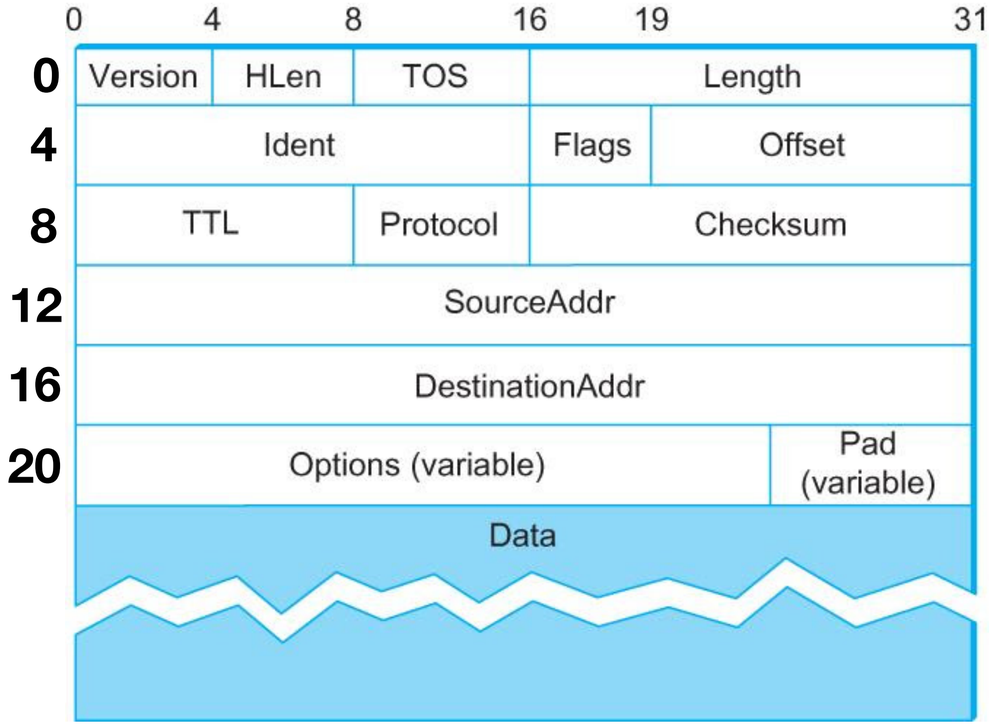


IP Packet Format (cont'd)

- #4: Length (16 bits)
  - #bytes in this datagram
- #5: Identification (16 bits)
  - Sequence number
  - Used by fragmentation
- #6: Flags + Offset (3 + 13 bits)
  - Used by fragmentation
  - In the unit of 8 bytes

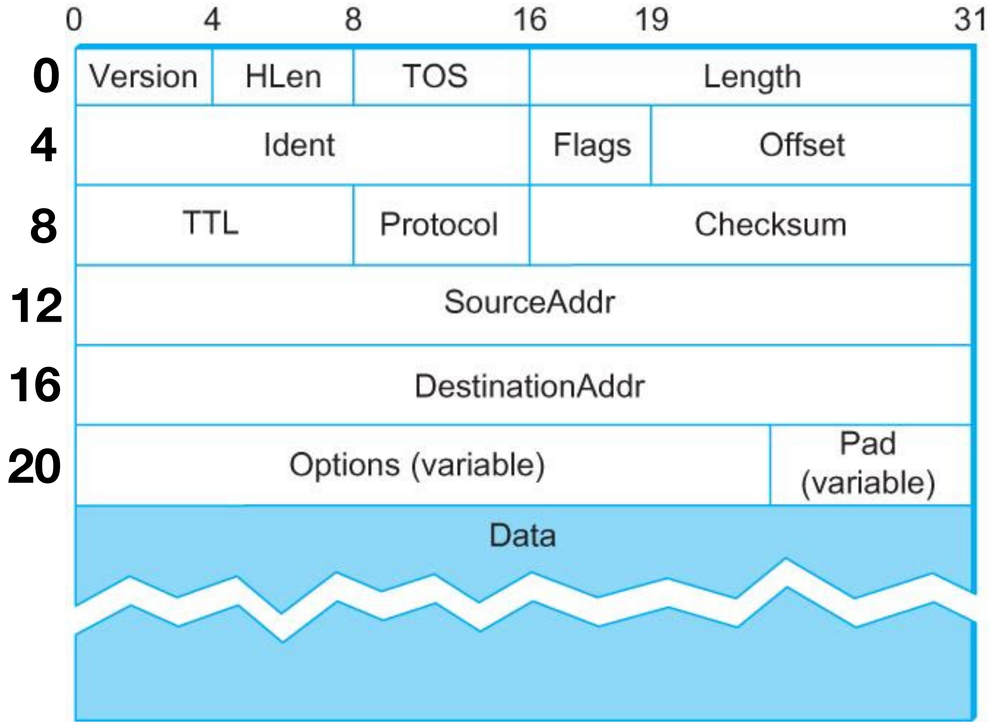
IP Packet Format (cont'd)

- #7: TTL (time-to-live) (8 bits)
  - # hops this datagram has traversed
  - Decremented at every router
- #8: Protocol (8 bits)
  - Demultiplex key (e.g., TCP=6, UDP=17)
- #9: Checksum (16 bits)
  - The checksum of the IP header in terms of 16-bit words



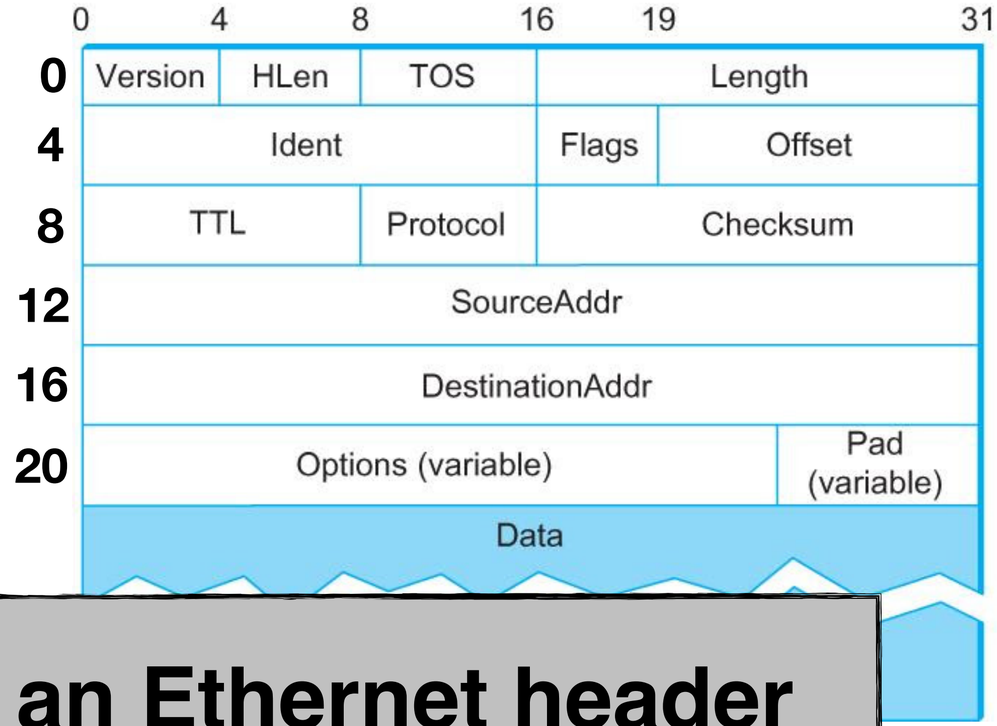
IP Packet Format (cont'd)

- #10: SourceAddr (32 bits)
  - The address of the source host
- #11: DestinationAddr (32 bits)
  - The address of the destination host



IP Packet Format (cont'd)

- #10: SourceAddr (32 bits)
  - The address of the source host
- #11: DestinationAddr (32 bits)
  - The address of the destination host



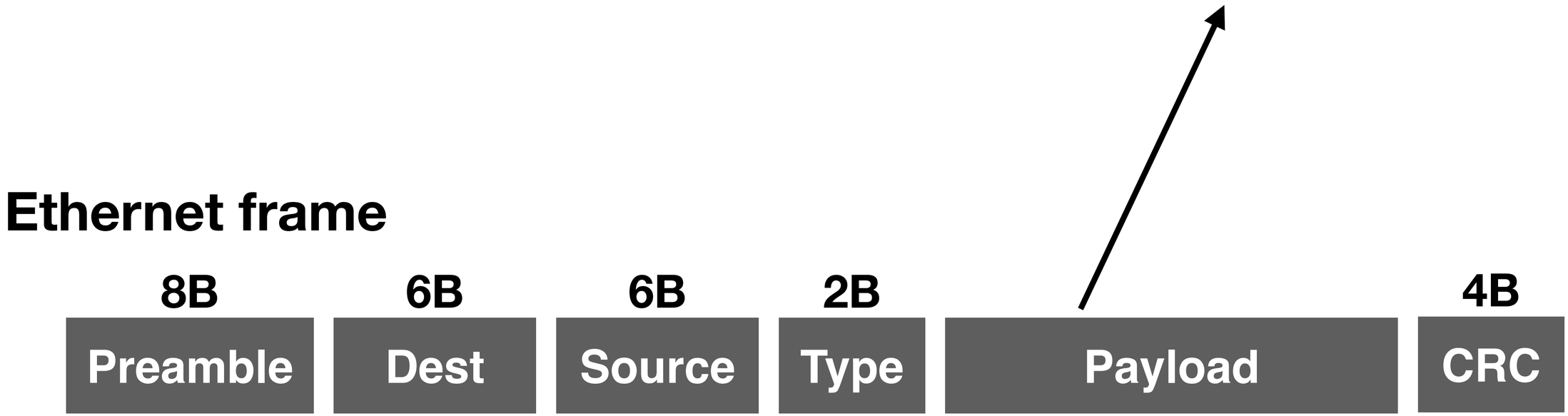
What are the differences between an Ethernet header and an IP header? What can we learn from the header?

Data Transformation across Layers

Signals ⟷ Bitstreams ⟷ Frames ⟷ IP datagrams

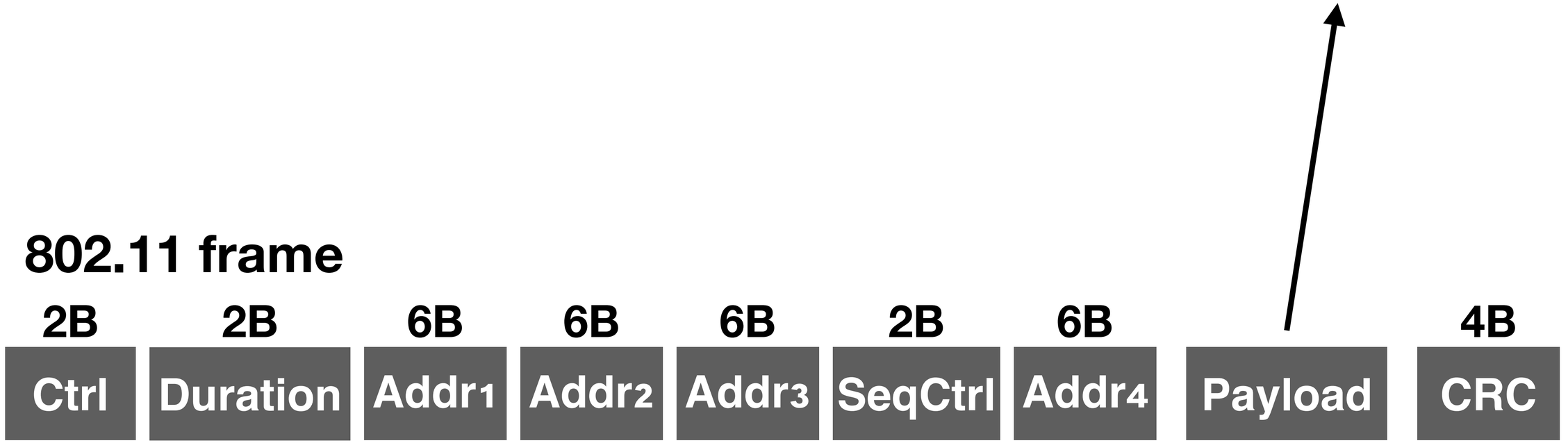
Data Transformation across Layers

Signals ⟷ Bitstreams ⟷ Frames ⟷ IP datagrams



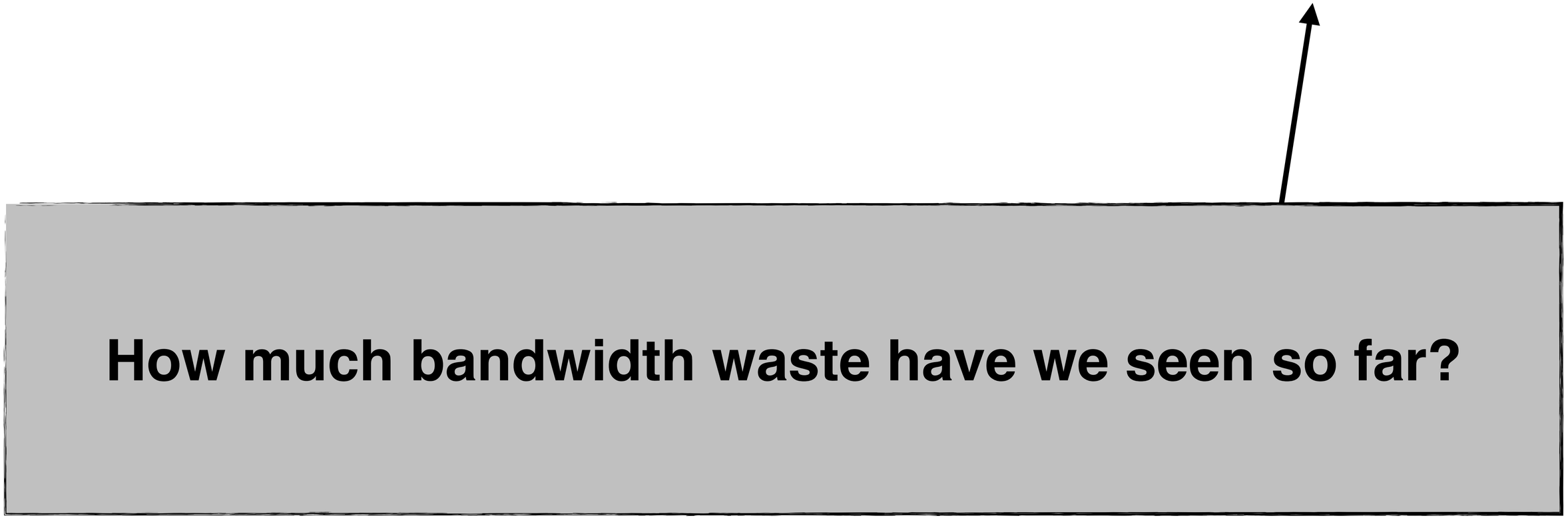
Data Transformation across Layers

Signals ⟷ Bitstreams ⟷ Frames ⟷ IP datagrams



Data Transformation across Layers

Signals ⟷ Bitstreams ⟷ Frames ⟷ IP datagrams



Best-Effort Host-to-Host Service Model

- *#1: use a unified header format*

- #2: support heterogeneous networks

- #3: provide unreliable packet delivery

Heterogeneous L2 Networks

- Different L2 networks define their own frame size limit
  - Maximum Transmission Unity (MTU)
  - Ethernet: 1500B, 9KB, etc.

- Adapt the IP datagram to the underlying L2 frame
  - #1: Fragmentation and reassembly
  - #2: Synchronize the MTU

Fragmentation and Reassembly

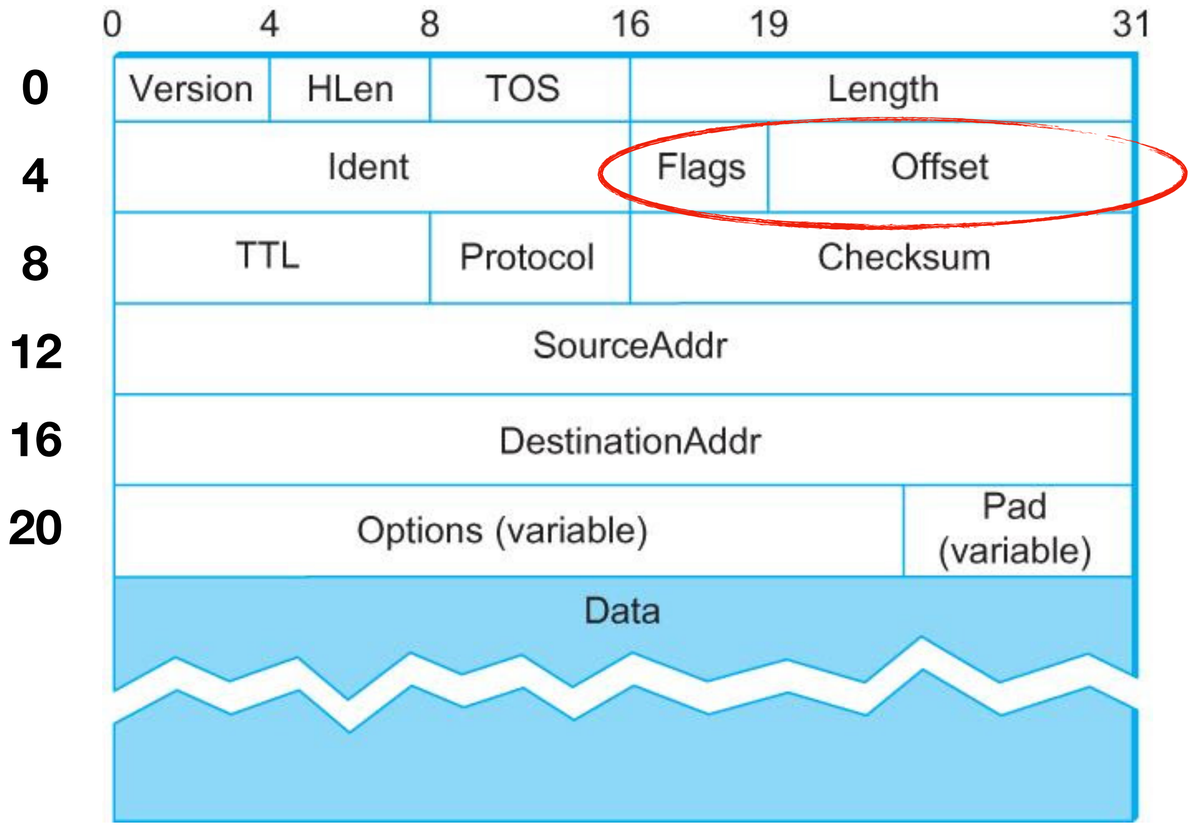
- Idea: Breakdown the IP datagram when traversing a link based on the small-sized MTU

Fragmentation and Reassembly

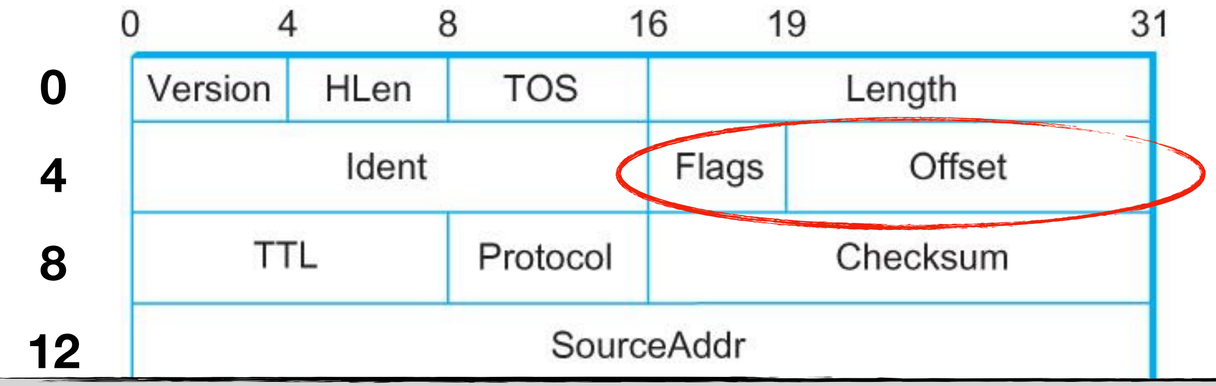
- Idea: Breakdown the IP datagram when traversing a link based on the small-sized MTU

- Strategy
  - #1: Fragment the datagram when necessary (MTU < Datagram)
  - #2: Avoid fragmentation at the source host
  - #3: Re-fragment the datagram is possible
  - #4: Delay reassembly until the destination host
  - #5: Do not recover from lost fragments

Header Support for Fragmentation and Reassembly

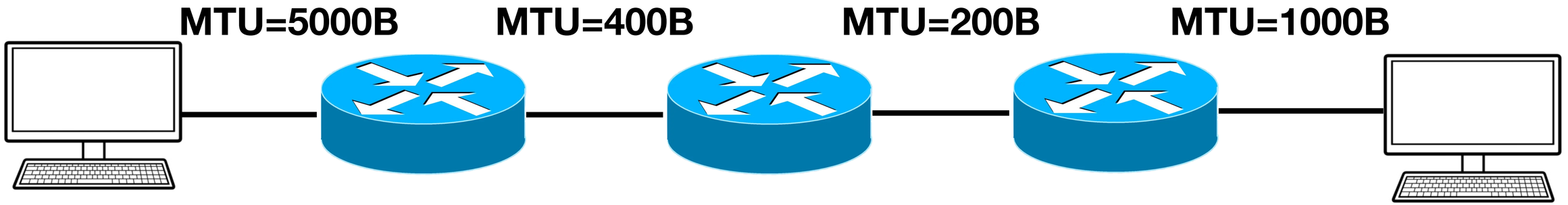


Header Support for Fragmentation and Reassembly



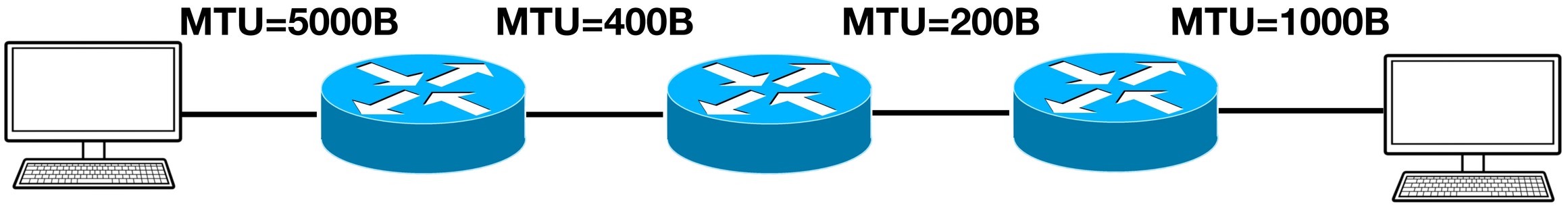
- Three-bit flags
  - bit 0: reserved; must be zero
  - bit 1: Don’t Fragment (DF)
  - bit 2: More Fragments (MF)
- Offset (13 bits)
  - Specify the offset of a particular fragment relative to the beginning of the original unfragmented IP datagram in units of eight-byte blocks

An Example



1000B IP Datagram

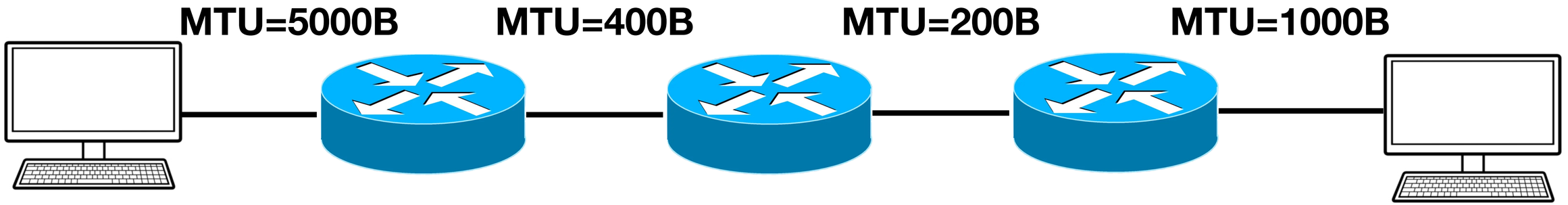
An Example



1000B IP Datagram

After R1	MF	Offset	Size
Datagram1	0	0	1000

An Example

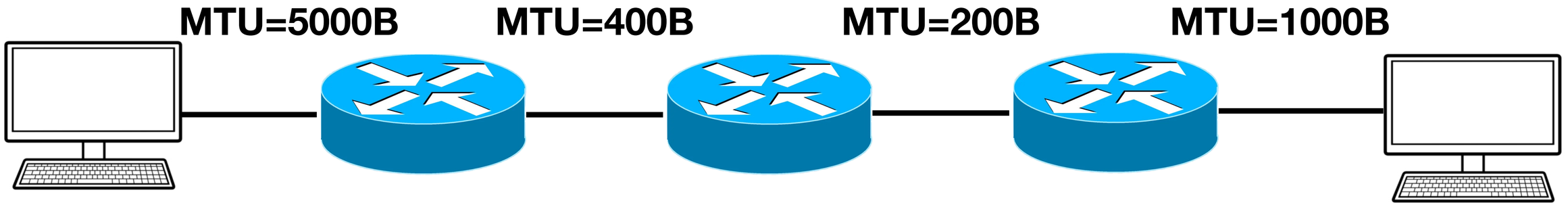


1000B IP Datagram

After R1	MF	Offset	Size
Datagram1	0	0	1000

After R2	MF	Offset	Size
Datagram1	1	0	400
Datagram2	1	50	400
Datagram3	0	100	200

An Example



1000B IP Datagram

After R1	MF	Offset	Size
Datagram1	0	0	1000

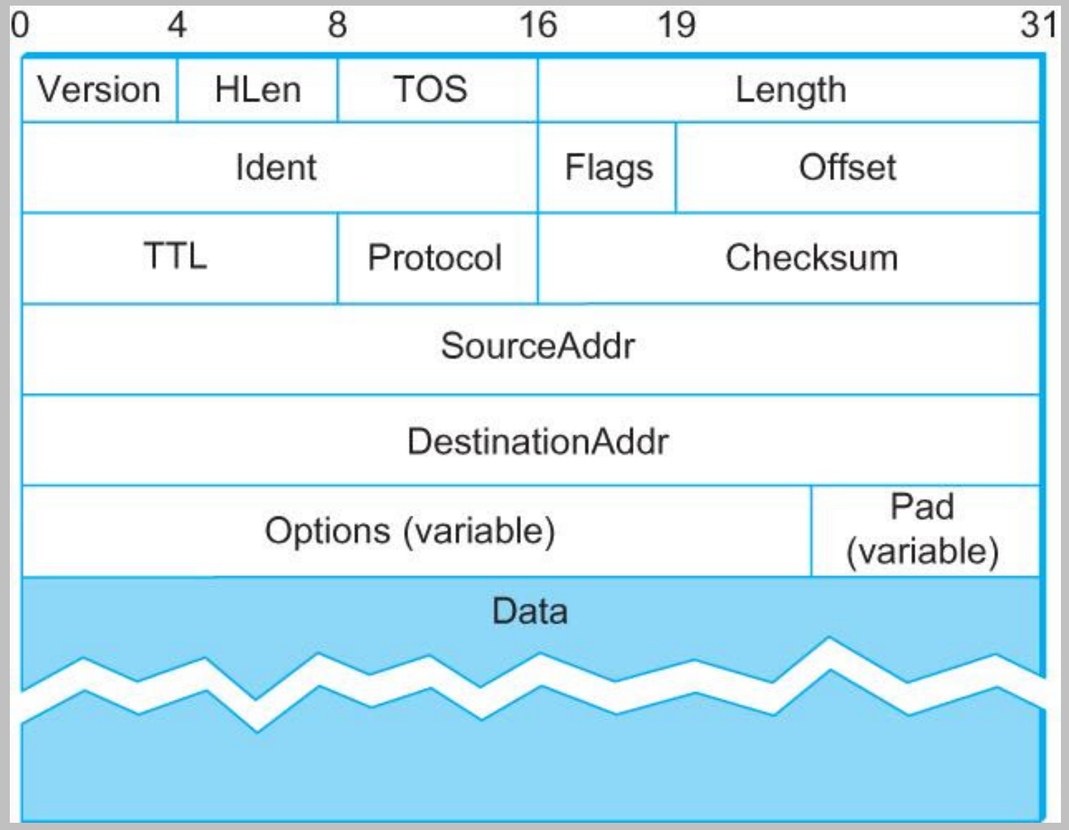
After R2	MF	Offset	Size
Datagram1	1	0	400
Datagram2	1	50	400
Datagram3	0	100	200

After R3	MF	Offset	Size
Datagram1	1	0	200
Datagram2	1	25	200
Datagram3	1	50	200
Datagram4	1	75	200
Datagram5	0	100	200

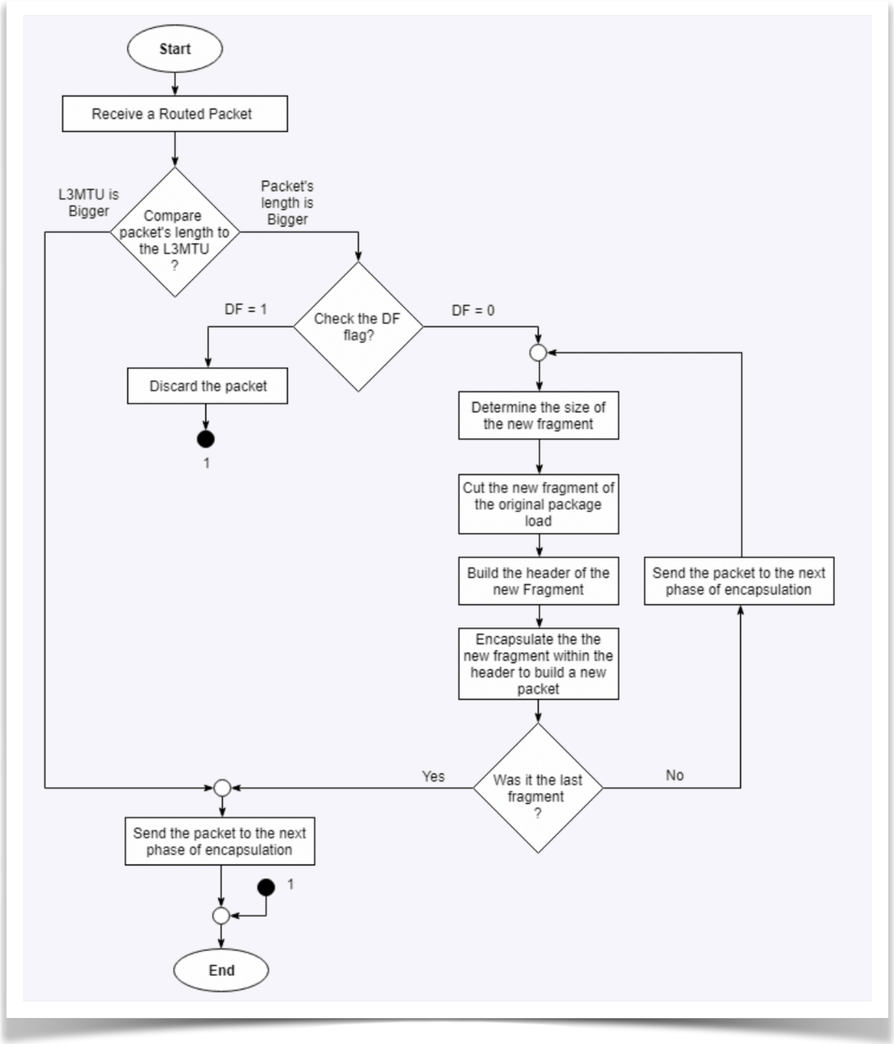
To simplify the discussion, we ignore the header size.

An Example

Why is the offset in units of 8-byte blocks?



The Algorithm and Implementation



net/ipv4/ip_fragment.c

```c
/* Process an incoming IP datagram fragment. */
int ip_defrag(struct sk_buff *skb, u32 user)
{
        struct ipq *qp;
        struct net *net;

        net = skb->dev ? dev_net(skb->dev) : dev_net(skb_dst(skb)->dev);
        IP_INC_STATS_BH(net, IPSTATS_MIB_REASMREQDS);

        /* Start by cleaning up the memory. */
        if (atomic_read(&net->ipv4.frags.mem) > net->ipv4.frags.high_thresh)
                ip_evictor(net);

        /* Lookup (or create) queue header */
        if ((qp = ip_find(net, ip_hdr(skb), user)) != NULL) {
                int ret;

                spin_lock(&qp->q.lock);

                ret = ip_frag_queue(qp, skb);

                spin_unlock(&qp->q.lock);
                ipq_put(qp);
                return ret;
        }

        IP_INC_STATS_BH(net, IPSTATS_MIB_REASMFAILS);
        kfree_skb(skb);
        return -ENOMEM;
}
```

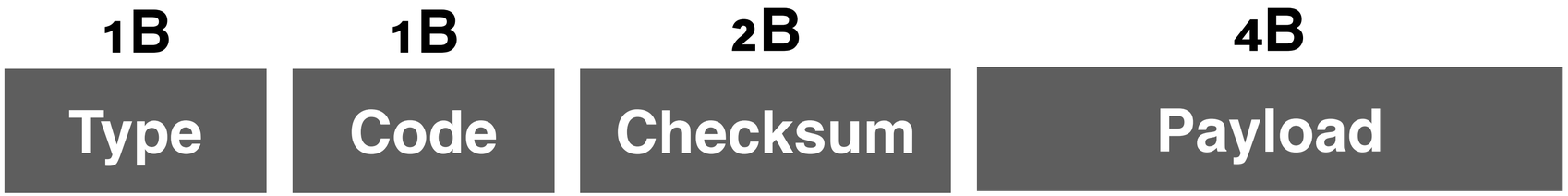
Synchronize the MTU

- Path MTU discovery
  - Originally introduced in IPv4
  - IPv6 delegates it to the end-hosts

- Key idea:
  - Set the Don’t Fragment (DF) flag bit in the IP header
  - Any on-path devices along the path whose MTU is smaller than the packet would (a) drop the packet and (b) send back an Interconnect Control Message Protocol (ICMP) message containing its MTU
  - The source host then reduces their path MTU accordingly

Internet Control Message Protocol (ICMP)

- A supporting protocol for the IP to handle errors
  - Report the error status to the source host so that it can react accordingly
  - Datagrams are not dropped blindly



Best-Effort Host-to-Host Service Model

- *#1: use a unified header format*
- *#2: support heterogeneous networks*
- #3: provide unreliable packet delivery

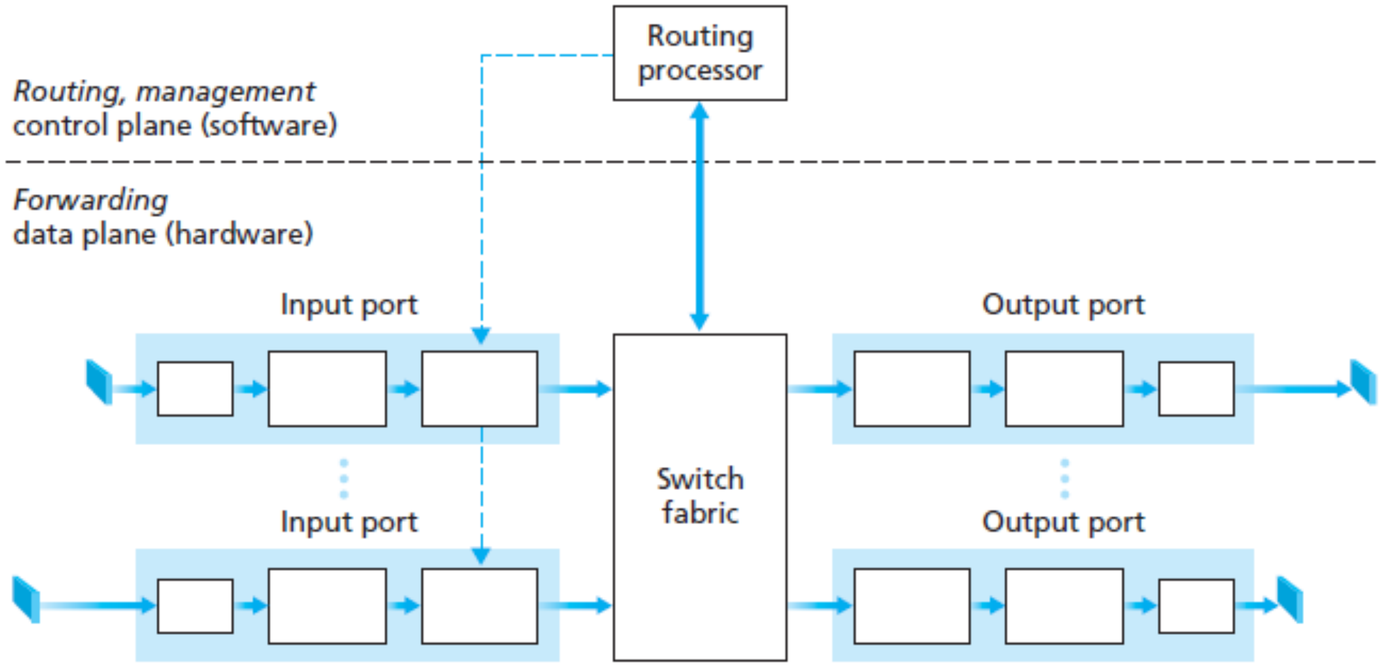
Unreliable Packet Delivery

- No guarantee of how the data is transmitted
  - Datagrams can be lost
  - Datagrams can be delivered out of order
  - Datagrams can be duplicated
  - Datagrams can be delayed for a long time

Router

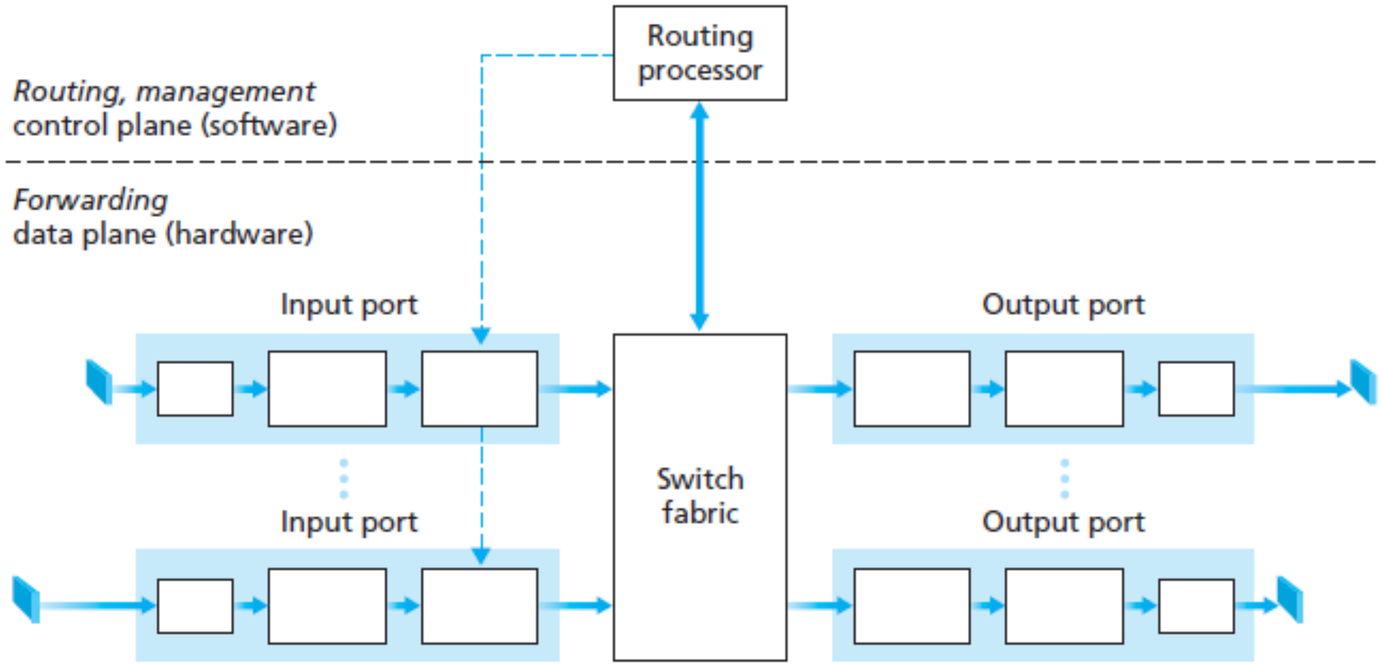
- A multiple-input multiple-output I/O device that performs forwarding based on the IP address
  - Physical appearance is similar to an L2 switch
  - Key functionalities are different

The Router Architecture



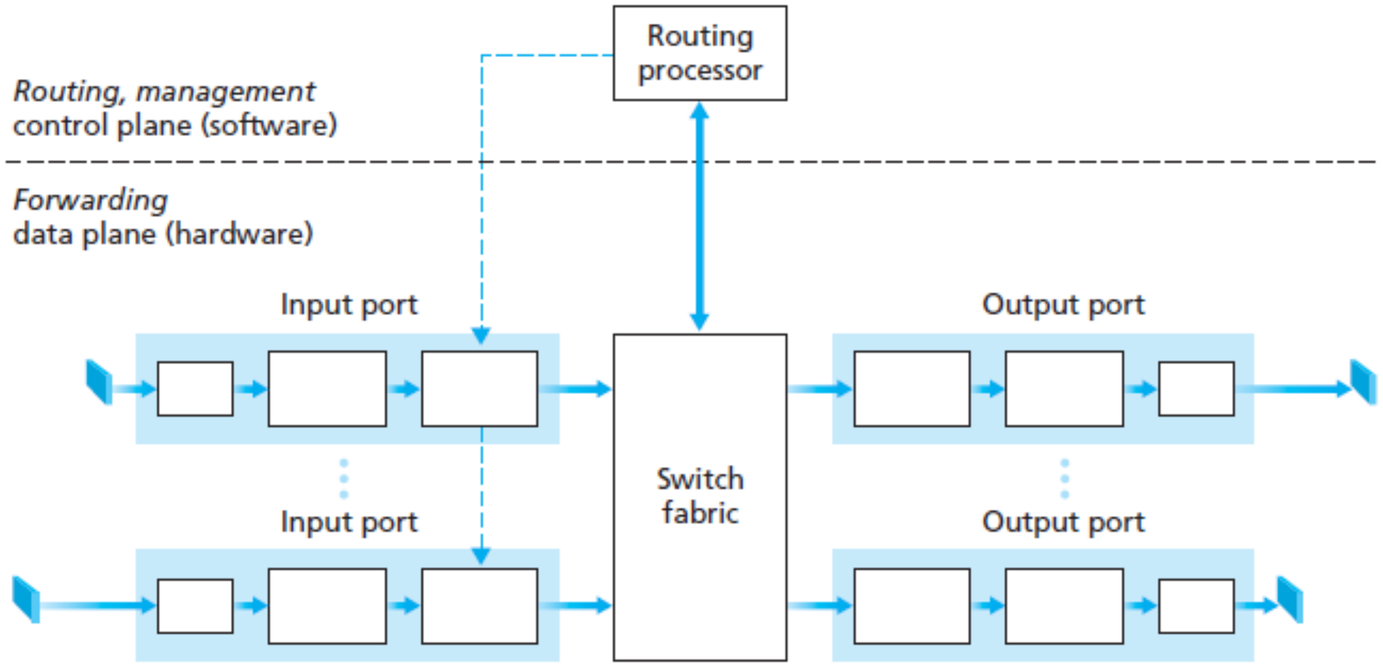
The Router Architecture—Input Ports

- Input port (Ingress Pipeline):
  - Perform the physical layer function of terminating the physical link
  - Perform the link layer functions
  - Look up the forwarding table to decide the output port



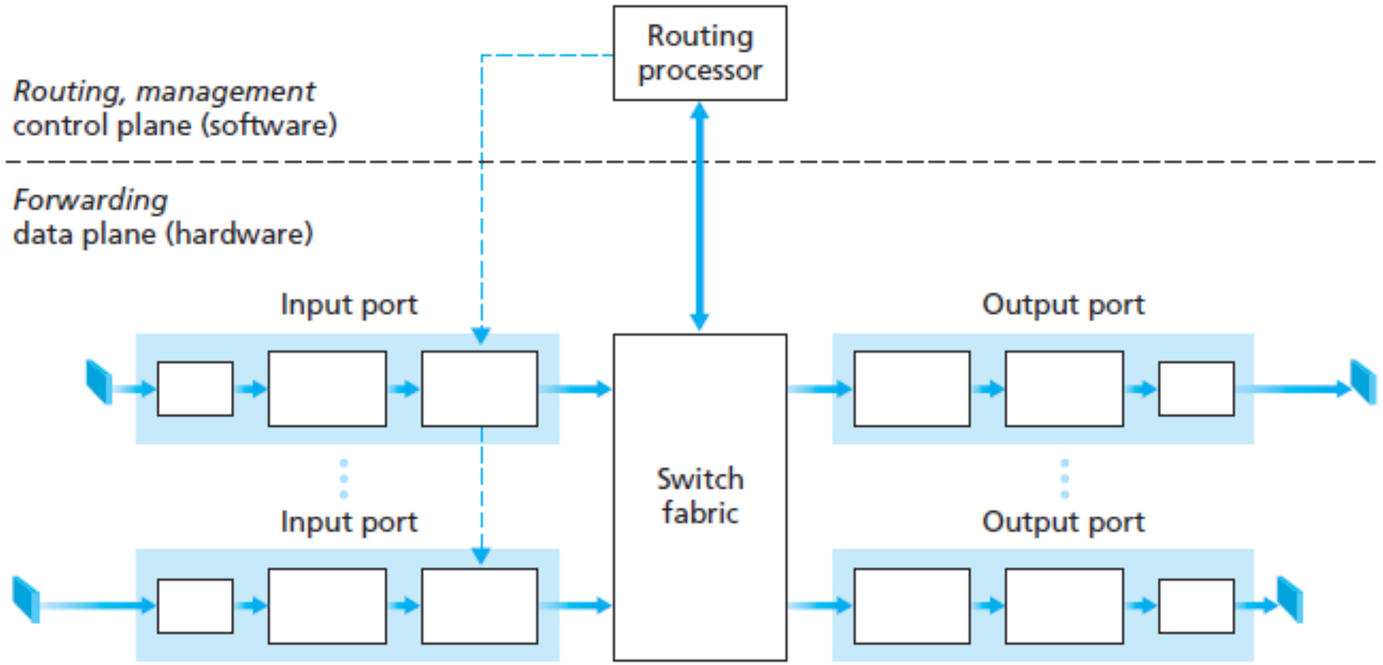
The Router Architecture—Output Ports

- Output port (Egress Pipeline):
  - Perform the physical/link layer functionalities
  - Under bidirectional communication, output and input are paired on the same line card



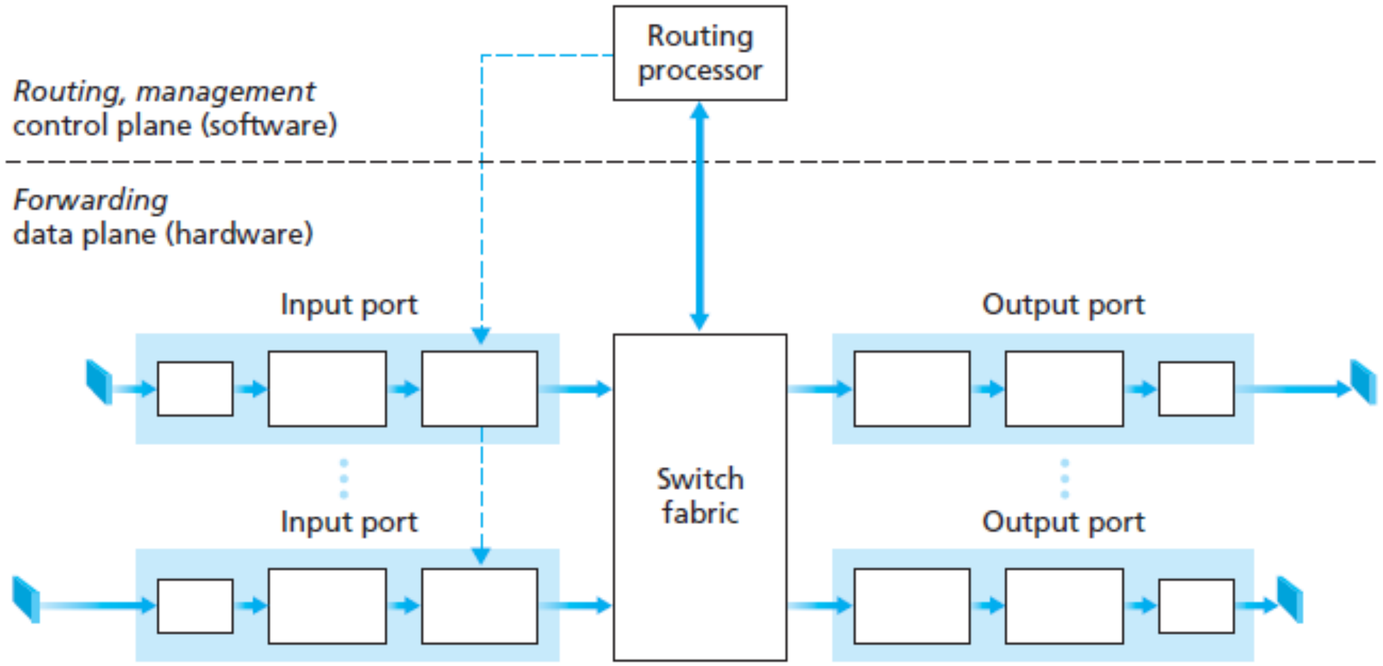
The Router Architecture—Switching Fabric

- Switching fabric:
  - Connect the input and output ports
  - Akin to an L2 switch, traffic manager and buffers are here



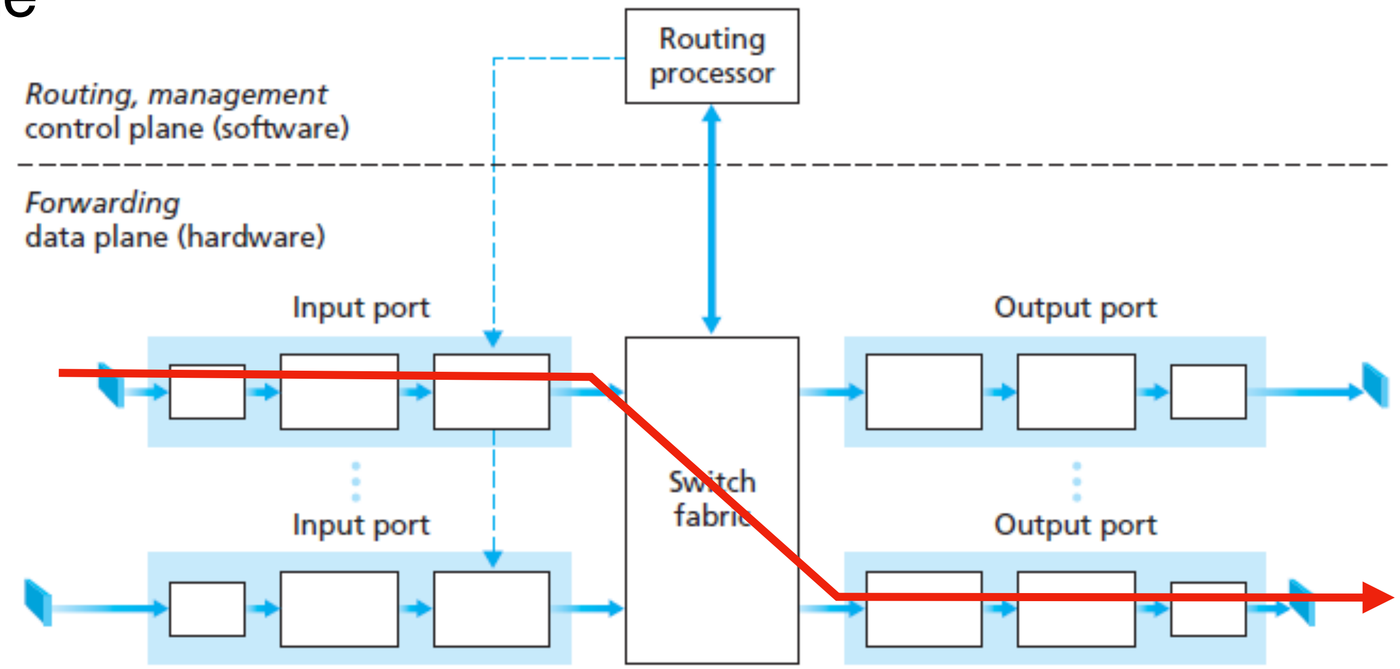
The Router Architecture—Routing Processor

- Routing Processor:
  - Execute the routing protocols
  - Maintain routable tables and attached link state information
  - Compute the forwarding table for the router



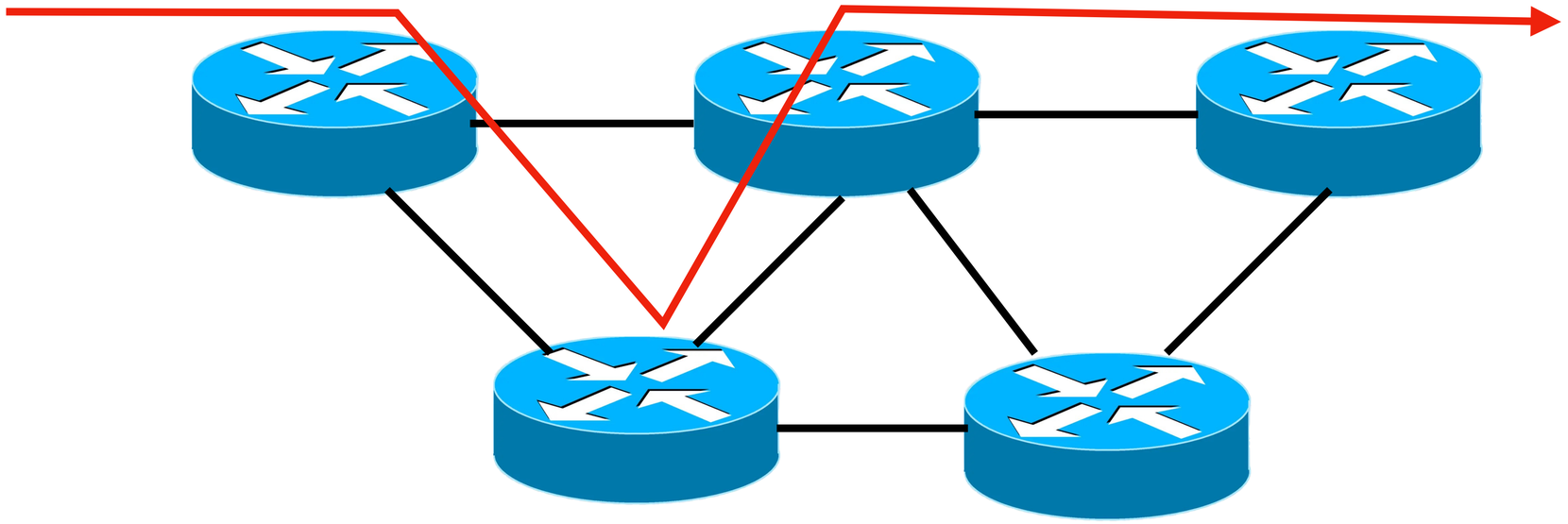
Forwarding v.s. Routing

- Forwarding refers to the router-local action of transferring a packet from an input interface to the appropriate output interface
  - Usually implemented in the hardware
  - O(nanosecond)
  - Data plane



Forwarding v.s. Routing

- Routing refers to the network-wide process that determines the end-to-end paths that packets take from source to destination
  - Usually implemented in the software (and hardware)
  - O(second)
  - Control plane



IP Provides Best-Effort Host-to-Host Service Model

- #1: use a unified header format
  - Encode software states to instruct how datagrams are delivered
- #2: support heterogeneous networks
  - Tolerate L2 networks with different transmission capabilities
- #3: provide unreliable packet delivery
  - Use a router to connect different networks

Summary

- Today
  - IP Introduction

- Next lecture
  - Efficient Addressing