

# Network Layer

Dr. Raja Vara Prasad,  
IIIT Sri City, Chittoor

# Inside a Router

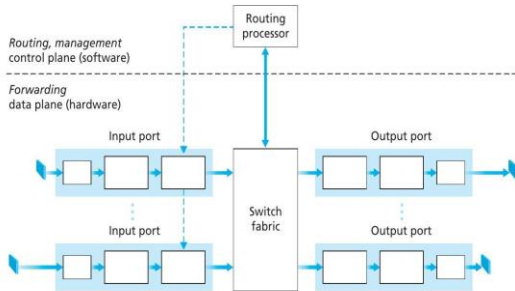


Figure 4.6 ♦ Router architecture

- 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

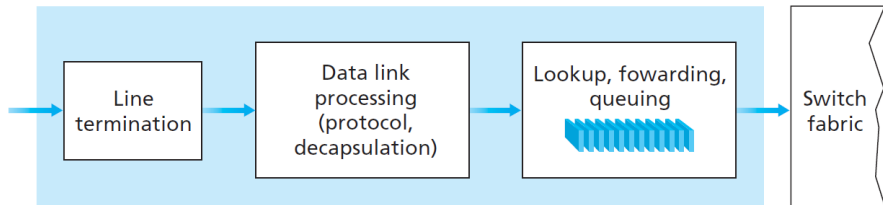
- executing the routing protocols, responding to attached links that go up or down, and performing management functions
- operate at the **millisecond** or second timescale
- **router control plane** functions are usually implemented in **software** and execute on the **routing processor**

# 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.



## 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*

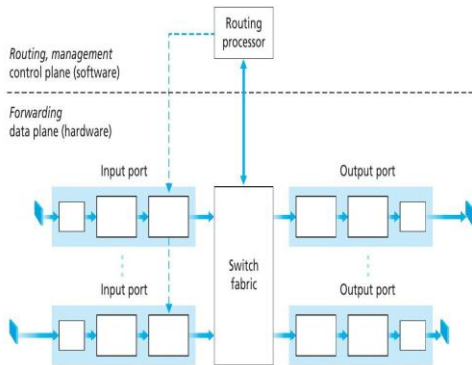


Figure 4.6 ♦ Router architecture

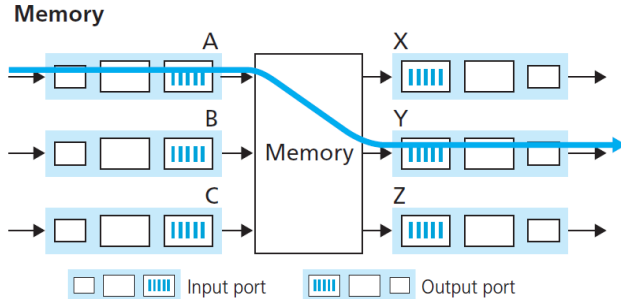


# Inside a Router: Switching

## 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

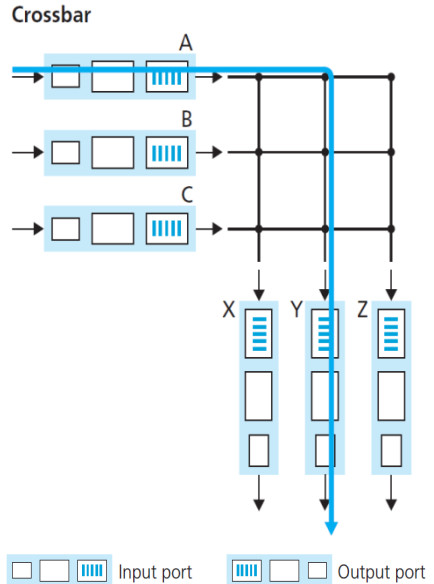




# Inside a Router: Switching

## Switching via an interconnection network:

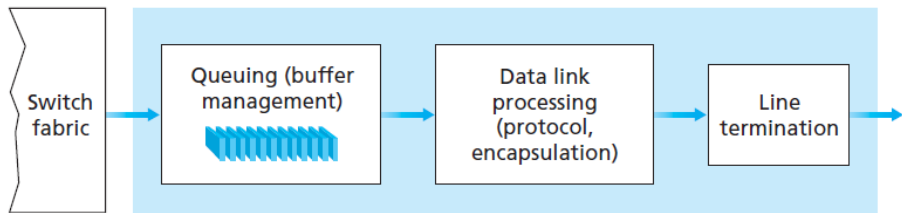
- more sophisticated interconnection network
- crossbar switch is an interconnection network consisting of  $2N$  buses that connect  $N$  input ports to  $N$  output ports
- Each vertical bus intersects each horizontal bus at a crosspoint  $\rightarrow$  can be opened or closed at any time by the switch fabric controller
- crossbar networks are capable of forwarding multiple packets in parallel
- if two packets from two different input ports  $\rightarrow$  to the same output port  $\rightarrow$  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  $R_{\text{line}}$  packets/sec
- $R_{\text{switch}}$  rate at which packets can be moved from input to output port
- if  $R_{\text{switch}} = N * R_{\text{line}}$  negligible queueing 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

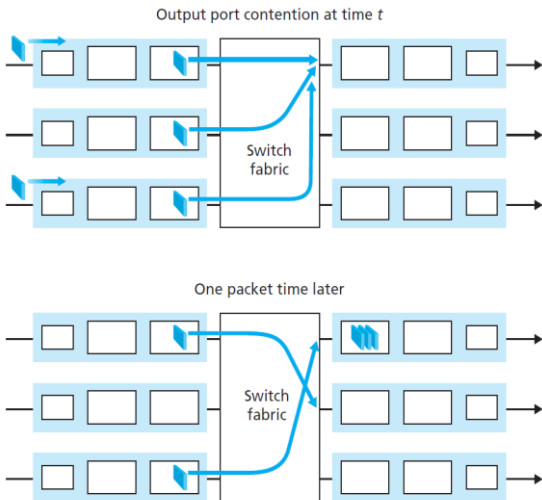
# Inside a Router: Output processing

## 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  $R_{\text{line}}$  packets/sec
- $R_{\text{switch}}$  rate at which packets can be moved from input to output port
- if  $R_{\text{switch}} = N * R_{\text{line}}$  negligible queueing 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

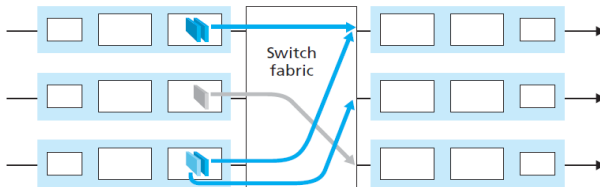
Assume:

- (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

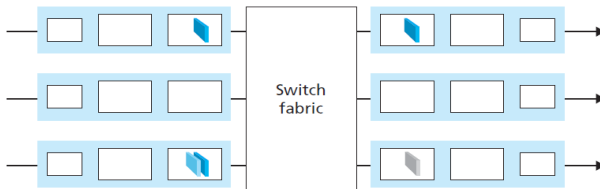


# Inside a Router: Output processing:


Output port contention at time  $t$  —  
one dark packet can be transferred





Light blue packet experiences HOL blocking



Key:

 destined for upper output port

 destined for middle output port

 destined for lower output port

# 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)**

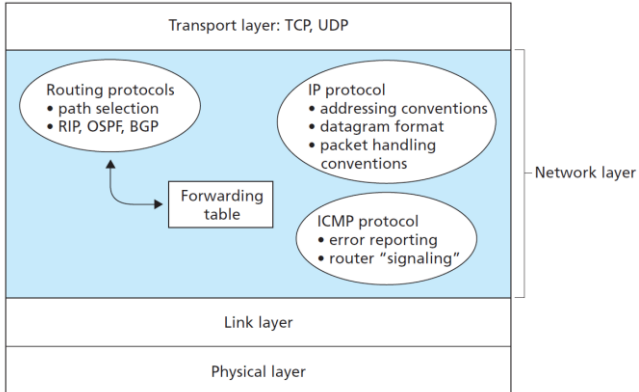
- separating the software control plane from the hardware data plane
- allowing different customized control planes to operate over fast hardware data planes

# Internet Protocol

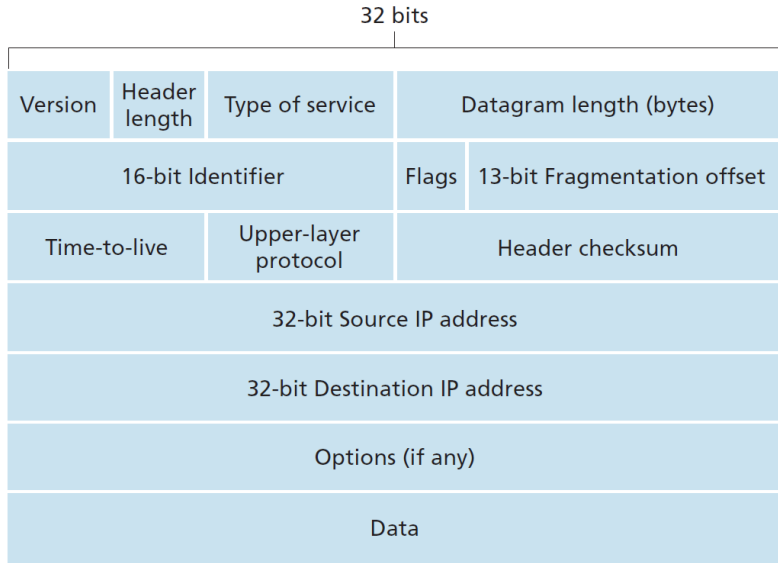
Important modules:

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

## ***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 576 bytes.
  - \*\*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 may 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:

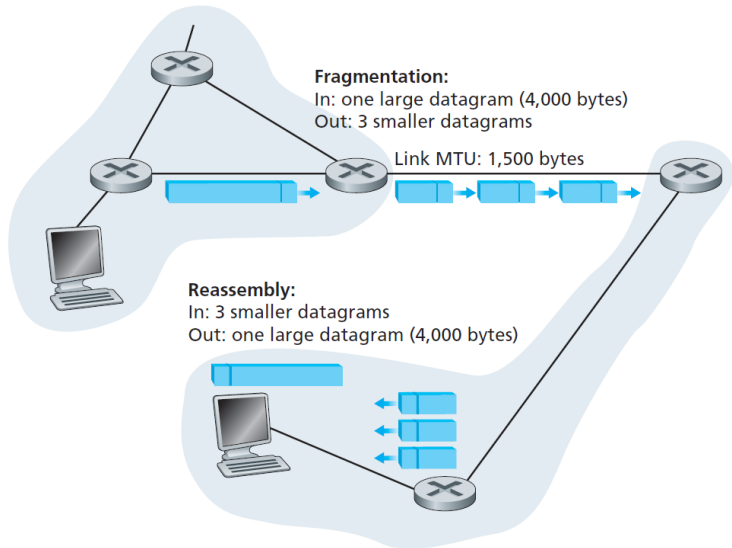
- IP datagram into two or more smaller IP datagrams
  - encapsulate each smaller IP datagrams in a separate link-layerframe;
  - send these frames over the outgoing link. Each smaller datagrams is a **fragment**.
- 
- Fragments need to be reassembled before they reach destination.
  - TCP and UDP: expecting 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 $185 \cdot 8 = 1,480$ )	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 $370 \cdot 8 = 2,960$ )	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.

# Internet Protocol - Addressing

- host typically has only a single link into the network  
boundary between the host and the physical link is called an **interface**
- Router to receive a datagram on one link and forward the datagram on some other link → two or more links
- Boundary between the router and any one of its links: interface: multiple interfaces
- IP requires each host and router interface to have its own IP address
- IP address is technically associated with an interface, rather than with the host or router containing that interface

IP address → 32 bits →  $2^{(32)}$  → approx 4 billion addresses

→ Address are in dotted decimal notation

Ex: 193.32.216.9

11000001 00100000 11011000 00001001

Each interface on every host and router in the global Internet must have an IP address that is globally unique

# Internet Protocol - Addressing

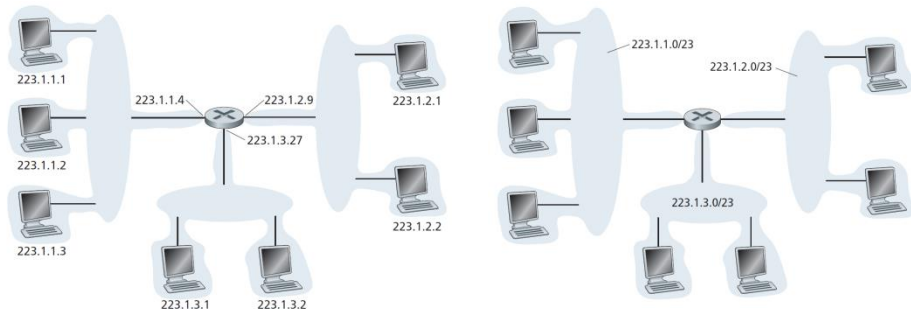
IP address of the form 223.1.1.xxx → same leftmost 24 bits in their IP address  
four interfaces are also interconnected to each other by a network *that contains no routers*

interfaces would be interconnected by an **Ethernet switch**

223.1.1.0/24, where the /24 notation → **subnet mask**

→ three host interfaces 223.1.1.1, 223.1.1.2, & 223.1.1.3

subnet-1: 223.1.1.0/24 , subnet-2: 223.1.2.0/24, subnet-3: 223.1.3.0/24



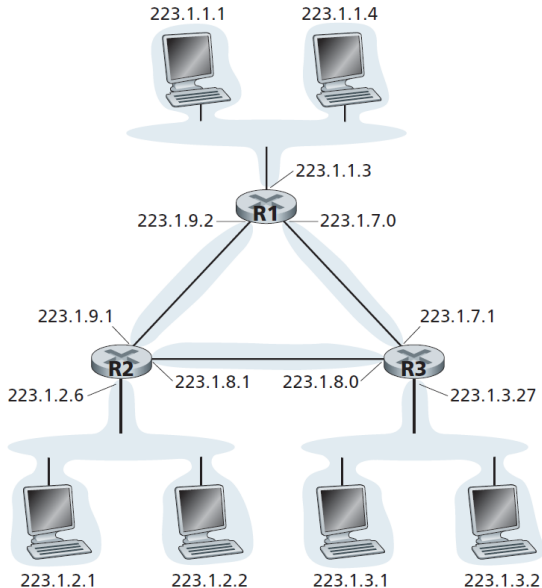
# Internet Protocol - Addressing

To determine the subnets,  
*detach each interface from its  
host or router,*

Creating *islands of isolated  
networks,*  
with *interfaces terminating the  
end points of the isolated  
networks.*

Each of these isolated networks  
is called a **subnet**.

How many subnets ? In the  
given example network





# Internet Protocol - Addressing

Internet's address assignment strategy: **Classless Interdomain Routing (CIDR)**

generalizes the notion of subnet addressing  
the 32-bit IP address is divided into two parts

→ dotted-decimal form  $a.b.c.d/x$ , where  $x$  indicates the number of bits in the first part of the address.

The  $x$  most significant bits of an address of the form  $a.b.c.d/x$  constitute the network portion of the IP address, and are often referred to as the **prefix** (or *network prefix*) of the address

organization is typically assigned a block of contiguous addresses, that is, a range of addresses with a common prefix

IP addresses of devices within the organization will share the common prefix  
 $x$  leading prefix bits are considered by routers outside the organization's network

# Internet Protocol - Addressing

a router outside the organization → forwards a datagram whose destination address is inside the organization, **only the leading x bits of the address need be considered**

considerably **reduces the size of the forwarding table** in these routers

→ a **single entry** of the form *a.b.c.d/x* will be sufficient to forward packets to **any destination** within the organization.

→ remaining  $32-x$  bits of an address can be thought of as distinguishing among the devices *within* the organization → with **same network prefix**

→ Remaining bits that will be considered when forwarding packets at routers *within* the organization

→ lower-order bits may or may not have an additional subnetting structure

## **Classful addressing:**

the network portions of an IP address constrained to 8, 16, or 24 bits

subnets with 8, 16, and 24-bit subnet addresses → class A, B, and C networks

class C (/24) subnet accommodates:  $2^8 - 2 = 254$  hosts

class B (/16) subnet  $2^{16} - 2 = 65,534$  hosts

**\*\*\*\*\* broadcast address 255.255.255.255 → message from a host is delivered to all hosts on the same subnet.**

# Internet Protocol - Addressing

- ability to use a single prefix to advertise multiple networks is often referred to as **address aggregation** or **route aggregation** or **route summarization**.
- works extremely well when addresses are allocated in blocks to ISPs and then from ISPs to client organizations
- what happens when addresses are not allocated in such a hierarchical manner ?

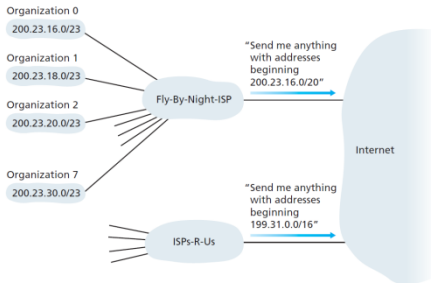


Figure 4.18 ♦ Hierarchical addressing and route aggregation

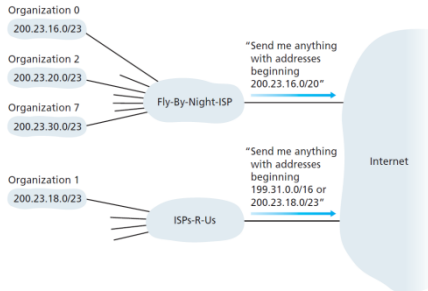


Figure 4.19 ♦ ISPs-R-Us has a more specific route to Organization 1

# Internet Protocol - Addressing

## Obtaining a Block of Addresses:

network administrator might first contact its ISP

provide addresses from a larger block of addresses that had already been allocated

ISP may itself have been allocated the address block 200.23.16.0/20

IP addresses are managed under the authority

\*\*\*\*\*Internet Corporation for Assigned Names and Numbers (ICANN)

→ to allocate IP addresses, to manage the DNSroot servers. It also has the very

→ assigning domain names and resolving domain name disputes

ISP's block	200.23.16.0/20	<u>11001000</u>	<u>00010111</u>	<u>00010000</u>	00000000
Organization 0	200.23.16.0/23	<u>11001000</u>	<u>00010111</u>	<u>00010000</u>	00000000
Organization 1	200.23.18.0/23	<u>11001000</u>	<u>00010111</u>	<u>00010010</u>	00000000
Organization 2	200.23.20.0/23	<u>11001000</u>	<u>00010111</u>	<u>00010100</u>	00000000
...	...				...
Organization 7	200.23.30.0/23	<u>11001000</u>	<u>00010111</u>	<u>00011110</u>	00000000

# Internet Protocol - Addressing

Host addresses can also be configured  
Manually

## Dynamic Host Configuration Protocol

**(DHCP):** allows a host to obtain an IP  
address automatically

network admin can configure DHCP

→ host receives the same IP address each  
time it connects to the network,  
or

→ A host may be assigned a **temporary IP  
address** that will be different each time the  
host connects to the network.

→ also allows a host to learn additional information → subnet mask, the address  
of its first-hop router and the address of its local DNS server.

→ **plug-and-play protocol**

→ in residential Internet access networks and in wireless LANs, where hosts join  
and leave the network frequently

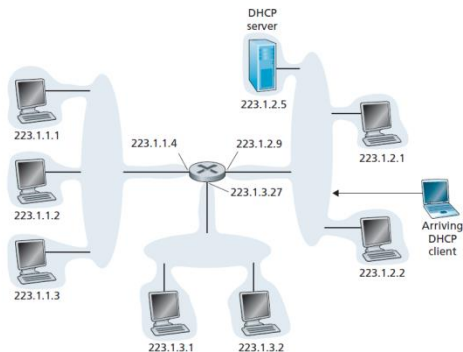


Figure 4.20 ♦ DHCP client-server scenario

# Internet Protocol - Addressing

Suited for many users coming and going, and addresses are needed for only a limited amount of time

Ex:

In Campus moving from Library to classroom to labs → one subnet to other

residential ISP access networks:

Ex: 2,000 customers, but no more than 400 customers are ever online at the same time

Instead of a block of 2,048 addresses, a DHCP server assigns dynamically needs only 512 addresses : block → a.b.c.d/23.

As the hosts join and leave,

→ DHCP server update its list of available IP addresses.

→ Each time a host joins, the DHCP server allocates an arbitrary address from its current pool of available addresses; each time a host leaves, its address is returned to the pool.

# Internet Protocol - Addressing

DHCP is a client-server protocol

- client is typically a newly arriving host wanting to obtain network configuration information
- each subnet will have a DHCP server or DHCP relay agent : typically a router

a four-step process:

- **DHCP server discovery**
- **DHCP server offer(s)**
- **DHCP request.**
- **DHCP ACK.**

Once the client receives the DHCP ACK, client can use the DHCP-allocated IP address for the lease duration.

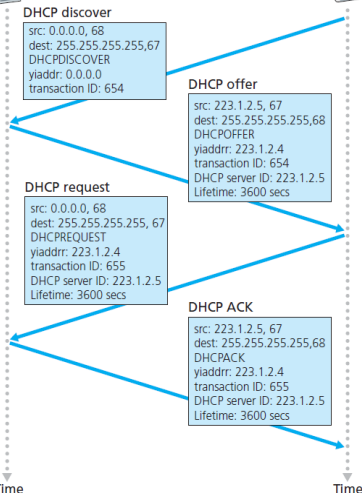
## Challenges:

- mobility aspect: each time a node connects to a new subnet, a TCP connection to a remote application cannot be maintained as mobile node moves between subnets
- single permanent address as it moves between subnets

DHCP server:  
223.1.2.5



Arriving client



# Internet Protocol - NAT

## Network Address Translation (NAT)

- small office, home office (SOHO) subnets
- what if the ISP had already allocated the contiguous portions of the SOHO network's current address range?
- for a private network or a **realm** with private addresses
- A *realm with private addresses* refers to a network whose addresses only have meaning to devices within that network
- Devices within a given home network can send packets to each other using 10.0.0.0/24 addressing
- packets forwarded *beyond* the home network into the larger global Internet clearly cannot use these addresses
- if private addresses only have meaning within a given network, how is addressing handled when packets are sent to or received from the global Internet, where addresses are necessarily unique?

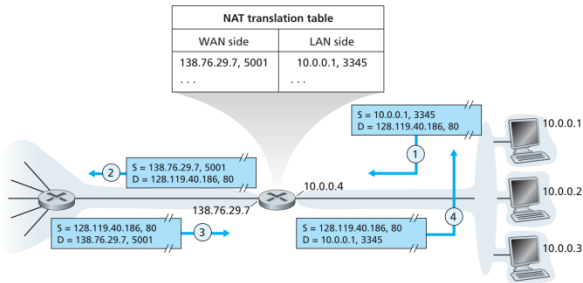


Figure 4.22 ♦ Network address translation



# Internet Protocol - NAT

## Network Address Translation (NAT)

- NAT router behaves to the outside world as a *single* device with a *single* IP address
- NAT-enabled router is hiding the details of the home network from the outside world
- The router gets its address from the ISP's DHCP server
- the router runs a DHCP server to provide addresses to computers within the NAT-DHCP-router-controlled home network's address space.
- **NAT translation table** at the NAT router → to include port numbers as well as IP addresses in the table entries

Example: Host in Home network requests a web server (port 80) with IP – 128.119.40.186 assigns source port: 3345 and sends to datagram.

- NAT router receives the datagram, generates a new source port number 5001 for the datagram, replaces the source IP address with its WAN-side IP address 138.76.29.7 --
- replaces the original source port number 3345 with the new source port number 5001
- NAT router can select any source port number that is not currently in the NAT translation table
- Web server unaware → the arriving datagram containing the HTTP request has been manipulated by the NAT router
- responds with a datagram whose destination address is the IP address of the NAT router, and whose destination port number is 5001

# Internet Protocol - NAT

## Network Address Translation (NAT)

- port numbers are meant to be used for addressing processes, not for addressing hosts → violation can indeed cause problems for servers running on the home network
- NAT protocol violates the so-called end-to-end argument; that is, hosts should be talking directly with each other, without interfering nodes modifying IP addresses and port numbers
- we should use IPv6 (see Section 4.4.4) to solve the shortage of IP addresses, rather than recklessly patching up the problem with a stopgap solution like NAT. But like it or not, NAT has become an important component of the Internet
- NAT is that it interferes with P2P applications, including P2P file-sharing applications and P2P Voice-over-IP applications

# Internet Protocol - ICMP

## Internet Control Message Protocol (ICMP)

- used by hosts and routers to communicate network-layer information to each other
- most typical use of ICMP is for error reporting.
- For example, when running a Telnet, FTP, or HTTP session, you may have encountered an error message such as “Destination network unreachable.” → origins in ICMP
- IP router was unable to find a path to the host specified in your Telnet, FTP, or HTTP application. That router created and sent a type-3 ICMP message to your host indicating the error
- ICMP is often considered part of IP → lies just above IP,
  - as ICMP messages are carried inside IP datagrams. ICMP messages are carried as IP payload, just as TCP or UDP segments are carried as IP payload.
  - when a host receives an IP datagram with ICMP specified as the upper-layer protocol, it demultiplexes the datagram’s contents to ICMP, just as it would demultiplex a datagram’s content to TCP or UDP.
- ICMP messages have a type and a code field, and contain the header and the first 8 bytes of the IP datagram that caused the ICMP message to be generated in the first place
- well-known ping program sends an ICMP type 8 code 0 message to the specified host. The destination host, seeing the echo request, sends back a type 0 code 0 ICMP echo reply.

# Internet Protocol - ICMP

ICMP Type	Code	Description
0	0	echo reply (to ping)
3	0	destination network unreachable
3	1	destination host unreachable
3	2	destination protocol unreachable
3	3	destination port unreachable
3	6	destination network unknown
3	7	destination host unknown
4	0	source quench (congestion control)
8	0	echo request
9	0	router advertisement
10	0	router discovery
11	0	TTL expired
12	0	IP header bad

**Figure 4.23** ♦ ICMP message types

# Internet Protocol - ICMP

## Internet Control Message Protocol (ICMP)

- ICMP : **source quench message**

- to perform congestion control—to allow a congested router to send an ICMP source quench message to a host to force that host to reduce its transmission rate. TCP has its own congestion-control mechanism that operates at the transport layer, without the use of network-layer feedback such as the ICMP source quench message.

### Traceroute program:

- to trace a route from a host to any other host in the world is implemented with ICMP messages
- source sends a series of ordinary IP datagrams to the destination. Each of these datagrams carries a UDP segment with an unlikely UDP port number
- The first of these datagrams has a TTL of 1, the second of 2, the third of 3, and so on. The source also starts timers for each of the datagrams.
- When the  $n$ th datagram arrives at the  $n$ th router, the  $n$ th router observes that the TTL of the datagram has just expired. Rules of the IP protocol, the router discards the datagram and sends an ICMP warning message to the source (type 11 code 0).
- warning message includes the name of the router and its IP address. When this ICMP message arrives back at the source, the source obtains the round-trip time from the timer and the name and IP address of the  $n$ th router from the ICMP message.

# Internet Protocol - ICMP

## Internet Control Message Protocol (ICMP)

### Traceroute program:

How does a Traceroute source know when to stop sending UDP segments?

- source increments the TTL field for each datagram it sends.
- one of the datagrams will eventually make it all the way to the destination host.
- this datagram contains a UDP segment with an unlikely port number, the destination host sends a port unreachable ICMP message (type 3 code 3) back to the source.
- the source host receives this particular ICMP message, it knows it does not need to send additional probe packets.
- source host learns the number and the identities of routers that lie between it and the destination host and the round-trip time between the two hosts.

# Internet Protocol – IPV6

- 32-bit IP address space was beginning to be used up
- need for a large IP address space, a new IP protocol: IPv6

## IPv6 Datagram Format

### *streamlined 40-byte*

#### *header*

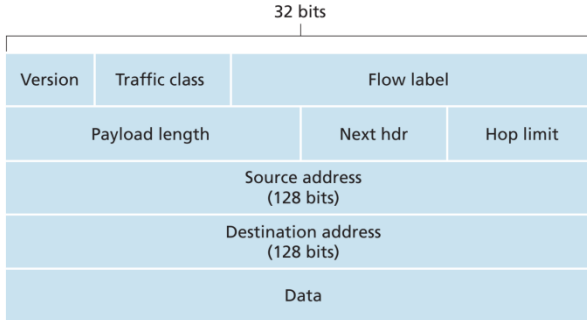
→ 40-byte fixed-length header for faster processing of the IP datagram

- → 4-bit field identifies the IP version number

**Traffic class** : 8-bit field similar to the TOS field in IPv4

**Payload length**: 16-bit value → number of bytes in the IPv6 datagram + 40-byte datagram header.

**Next header**: field identifies the protocol to which the contents  
example: to TCP or UDP, uses same fields as IPV4 header



# Internet Protocol – IPV6

**Hop limit:** field decremented by one by each router that forwards the datagram.  
→ hop limit count reaches zero, the datagram is discarded.

**Source and destination addresses:** The various formats of the IPv6 128-bit address

**Data:** This is the payload portion of the IPv6 datagram.

**Unique features of IPV6:**

**Expanded addressing capabilities:** size of the IP address from 32 to 128 bits

→ IPv6 new type of address → **anycast address** → allows a datagram to be delivered to any one of a group of hosts.

**Flow labeling and priority:** IPv6 definition of a **flow**.

- labeling of packets belonging to particular flows for which the sender requests special handling → non-default quality of service or real-time service.  
example, audio and video transmission might likely be treated as a flow. On the traditional applications → file transfer and e-mail, might not be treated as flows.
- traffic carried by a high-priority user might also be treated as a flow.
- The IPv6 header also has an 8-bit traffic class field.
- This field, like the TOS field in IPv4, can be used to give priority to certain datagrams within a flow
- it can be used to give priority to datagrams from certain applications  
ex: ICMP over datagrams from other applications



# Internet Protocol – IPV6

## Fields dropped from IPV4 :

### Fragmentation/Reassembly:

- does not allow for fragmentation and reassembly at intermediate routers;
- these operations can be performed only by the source and destination.
- If an IPv6 datagram received by a router is too large to be forwarded over the outgoing link, the router simply drops the datagram
- sends a “Packet Too Big” ICMP error message back to the sender.
- The sender can then resend the data, using a smaller IP datagram size.
- removing this functionality from the routers and placing it squarely in the end systems considerably speeds up IP forwarding within the network.

### Header checksum.

- Transport-layer and link-layer protocols in the Internet layers perform check sum
- functionality is redundant in the network layer → removed.
- fast processing of IP packets was a central concern → the IPv4 header contains a TTL field (similar to the hop limit field in IPv6), the IPv4 header checksum needed to be recomputed at every router.

### Options: no longer a part of the standard IP header

New version of ICMP for IPv6 : ICMPv6: added new types and codes required by IPv6

- “Packet Too Big” type, and an “unrecognized IPv6 options” error code.
- ICMPv6 subsumes the functionality of the Internet Group Management Protocol (IGMP)

# Internet Protocol – IPV6

## Transitioning from IPv4 to IPv6 :

- public Internet, which is based on IPv4, be transitioned to IPv6
- new IPv6-capable systems can be made backward compatible
- can send, route, and receive IPv4 datagrams, already deployed IPv4-capable systems are not capable of handling IPv6 datagrams

Two ways :

### **Dual-stack** approach:

- IPv6 nodes also have a complete IPv4 implementation
- an IPv6/IPv4 → node ability to send and receive both IPv4 and IPv6 datagrams
- interoperating with an IPv4 node → IPv6/IPv4 node can use IPv4 datagrams → when interoperating with an IPv6 node, it can speak IPv6
- if either the sender or the receiver is only IPv4- capable, an IPv4 datagram must be used.
- As a result, it is possible that two IPv6- capable nodes can end up, in essence, sending IPv4 datagrams to each other.
- in performing the conversion from IPv6 to IPv4, there will be IPv6-specific fields in the IPv6 datagram that have no counterpart in IPv4 → information in these fields will be lost.

# Internet Protocol – IPV6

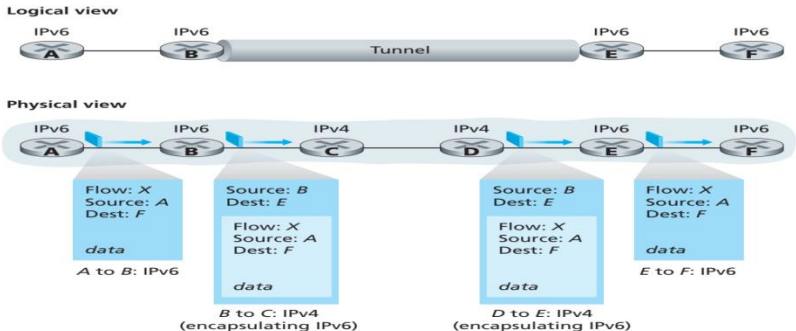
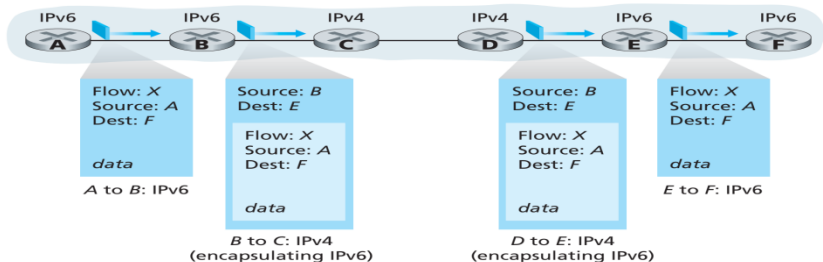


Figure 4.26 ♦ Tunneling

## Tunneling:

- two IPv6 nodes want to interoperate using IPv6 datagrams but are connected to each other by intervening IPv4 routers
- the intervening set of IPv4 routers between two IPv6 routers as a **tunnel**
- IPv6 node on the sending side of the tunnel takes the *entire* IPv6 datagram and puts it in the data field of an IPv4 datagram.
- This IPv4 datagram is then addressed to the IPv6 node on the receiving side of the tunnel and sent to the first node in the tunnel.

# Internet Protocol – IPv6



**Figure 4.26** ♦ Tunneling

## Tunneling:

- The intervening IPv4 routers in the tunnel route this IPv4 datagram as any other datagram
- IPv4 datagram itself contains a complete IPv6 datagram.
- The IPv6 node on the receiving side of the tunnel eventually receives the IPv4 datagram
- determines that the IPv4 datagram contains an IPv6 datagram → extracts the IPv6 datagram
- routes the IPv6 datagram exactly as it would if it had received the IPv6 datagram from a directly connected IPv6 neighbor.