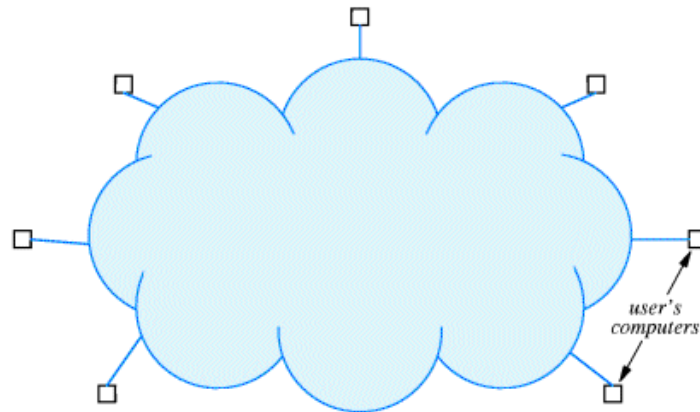


# 11. Network Layer

---

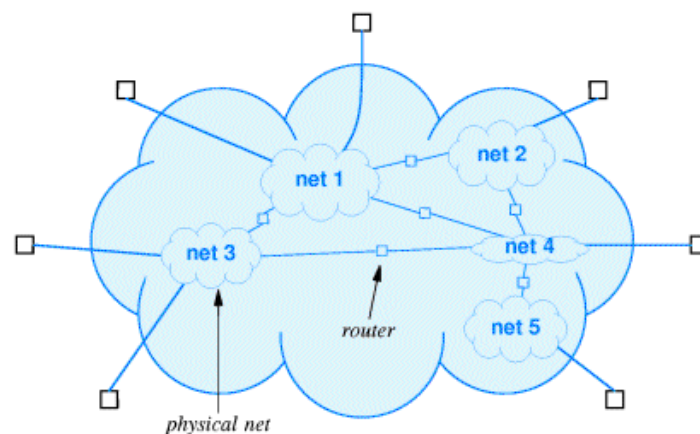
1. The Internet Concept (1)
  2. The Internet Concept (2)
  3. Routers
  4. Hosts and Routers
  5. Forwarding and Routing
  6. Data and Control Planes
  7. IP Addresses and Dotted-Decimal Notation
  8. IP Addressing
  9. Subnets
  10. Subnet Addressing
  11. Classful Addressing
  12. Obtaining a Block of Addresses
  13. Special IP Addresses
  14. IP Loopback Address
  15. Obtaining a Host Address
  16. Dynamic Host Configuration Protocol
  17. DHCP Client-Server Interaction
  18. Network Address Translation
  19. NAT Example
  20. Criticisms of NAT
  21. IP Datagrams
  22. Forwarding IP Datagrams
  23. Forwarding Table (1)
  24. Forwarding Table (2)
  25. Software-Defined Networking (SDN)
  26. IP Best-Effort Delivery
  27. IP Datagram Format (1)
  28. IP Datagram Format (2)
  29. Encapsulation
  30. Transmission across an Internet
  31. Example of IP Header
  32. Maximum Transmission Unit (MTU)
  33. Datagram Fragmentation (1)
  34. Datagram Fragmentation (2)
  35. Datagram Reassembly
  36. Fragment Loss
  37. Internet Control Message Protocol (ICMP)
  38. Some ICMP Message Types
  39. Some ICMP Message Descriptions
  40. Routing Algorithms
  41. Autonomous Systems
  42. Internet Routing Protocols
  43. Abstract Graph Model
  44. Open Shortest Path First
  45. OSPF Algorithm
  46. Example of OSPF Algorithm
  47. Links to more information
-

## 11.1. The Internet Concept (1)



- Internet software provides the *appearance* of a single, seamless communication system (above)
- each computer is assigned an address
- any computer can send a packet to any other computer
- Internet software provides a *virtual network* by hiding the details of
  - physical network connections
  - physical addresses
  - routing information

## 11.2. The Internet Concept (2)



- neither users nor applications are aware of the underlying physical networks or the routers that connect them
- diagram shows the underlying physical structure in which a computer attaches to one physical network, and routers interconnect the networks

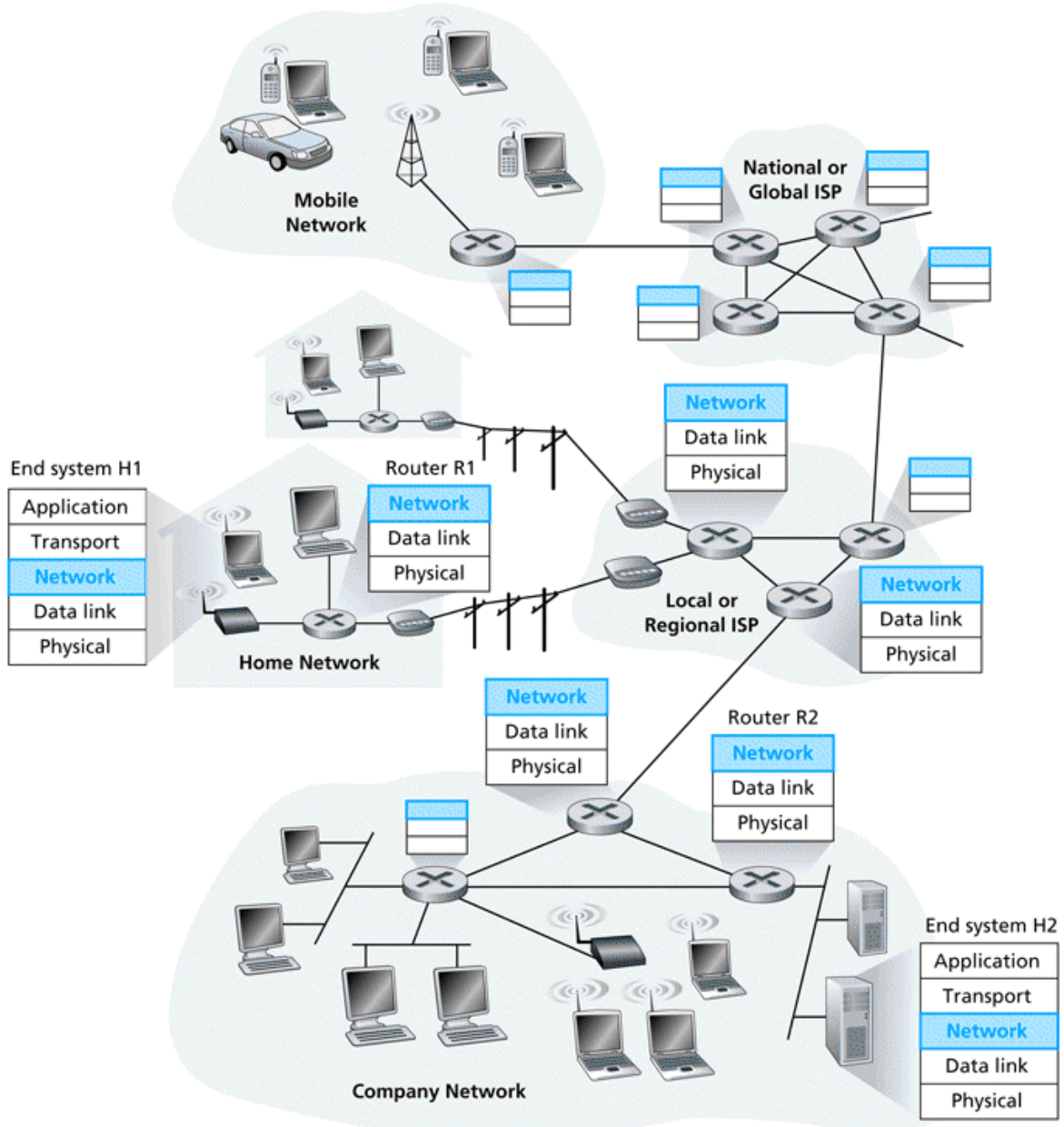
## 11.3. Routers



- the basic hardware used to connect **heterogeneous networks** is called a *router*
- network treats a connection to a router the same as a connection to any other computer
- the router has a separate I/O interface for each network it is connected to
- an organisation seldom uses a single router to connect all of its networks:
  - the processor in one router is insufficient to handle all the traffic
  - redundancy improves internet reliability

## 11.4. Hosts and Routers

- TCP/IP uses the term *host* to refer to any computer connected to an internet
- **hosts can be computers running applications or routers interconnecting the underlying physical networks**
- host computers will need all five layers of the TCP/IP protocol stack
- whereas routers only require the bottom three layers



## 11.5. Forwarding and Routing

- the role of the network layer is simple: move packets from a sending host to a receiving host
- this requires two network-layer functions:
  - *forwarding*
  - *routing*
- *forwarding* refers to the router's task of deciding which outgoing link an incoming packet should be forwarded on
- *routing* refers to the network-wide process of determining the best paths for packets to follow from a source to a destination
- **routers use *forwarding tables* which are determined and distributed using *routing algorithms***

## 11.6. Data and Control Planes

- the network layer can be decomposed into two interacting parts:
  - *data plane*
  - *control plane*
- the *data plane* refers to the function of each router (i.e. forwarding)
- the *control plane* refers to the network-wide logic associated with routing
- the above terminology is used particularly in *software-defined networking (SDN)*

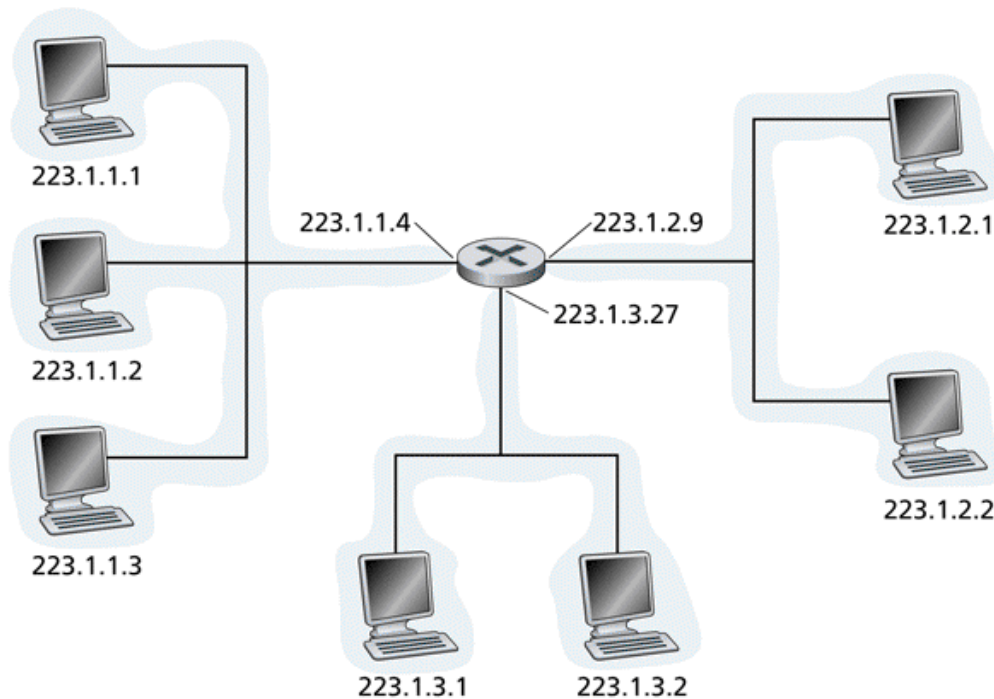
## 11.7. IP Addresses and Dotted-Decimal Notation

- recall that, in IPv4, each *IP address* is 32 bits long (128 bits in IPv6)
- although IP addresses are 32-bit numbers, humans use *dotted-decimal notation*
- each 8-bit byte is written in decimal and the four values have periods (dots) between them
- the range of possible dotted-decimal addresses goes from 0.0.0.0 through 255.255.255.255
- here are some example 32-bit addresses and their equivalent dotted-decimal forms:

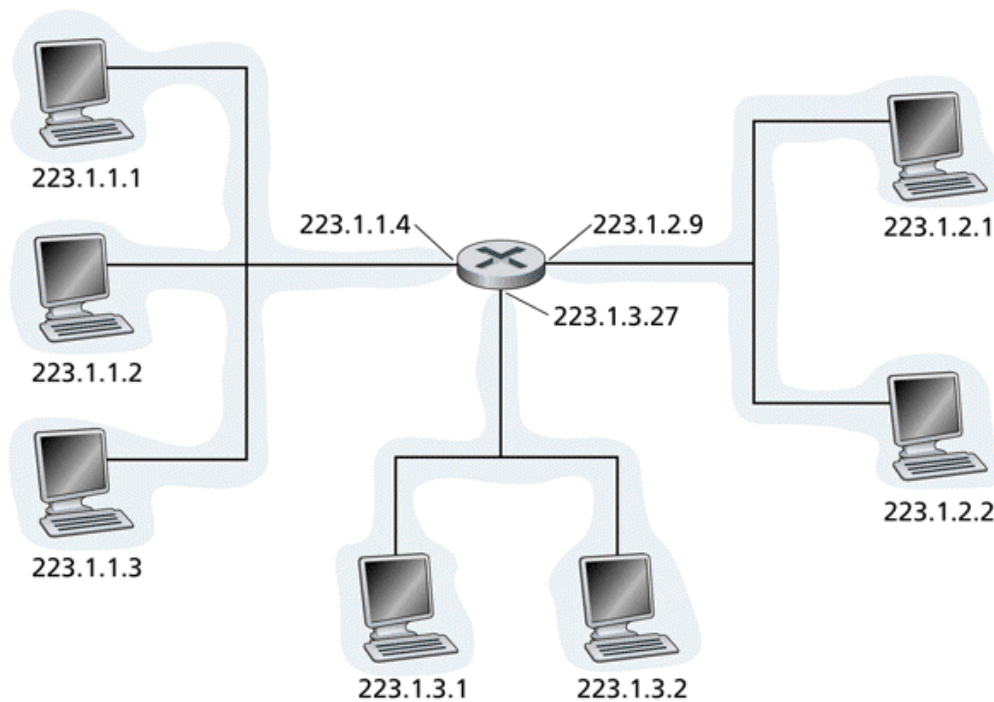
32-bit Binary Number				Equivalent Dotted Decimal
10000001	00110100	00000110	00000000	129.52.6.0
11000000	00000101	00110000	00000011	192.5.48.3
00001010	00000010	00000000	00100101	10.2.0.37
10000000	00001010	00000010	00000011	128.10.2.3
10000000	10000000	11111111	00000000	128.128.255.0

## 11.8. IP Addressing

- a host computer typically has only one link to the Internet
- the boundary between the host and the physical link is called an *interface*
- a router necessarily has two or more interfaces
- IP addresses are technically associated with *interfaces* rather than hosts or routers



## 11.9. Subnets



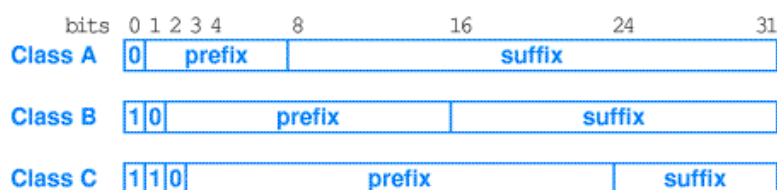
- consider the 4 interfaces whose addresses start with 223.1.1 in the above figure
- they are interconnected without a router and may comprise a local area network
- this network is called a *subnet* (or sometimes just *network*)

## 11.10. Subnet Addressing

- conceptually, each 32-bit address is divided into two parts: a *network prefix* and a *host suffix*
- this is called *subnet addressing* or *classless addressing*
- addresses are now written in the form a.b.c.d/n, where n indicates the number of bits in the network prefix
- this notation is called *Classless Inter-Domain Routing (CIDR)* notation
- for example, the top-left subnet in the previous figure would have the subnet address 223.1.1.0/24 to denote a 24-bit network prefix
- any additional host attached to this subnet would be *required* to have an address of the form 223.1.1.xxx

## 11.11. Classful Addressing

- IP addresses were originally divided into *classes*, known as *classful IP addressing*, as illustrated below



- the IP class scheme does not divide the 32-bit address space equally:

Address Class	Bits In Prefix	Maximum Number of Networks	Bits In Suffix	Maximum Number Of Hosts Per Network
A	7	128	24	16777216
B	14	16384	16	65536
C	21	2097152	8	256

- the total number of hosts for each class differ because of the number of bits required to identify the class
- class A can contain 2,147,483,648 hosts, class B half of that, and class C half again
- classless (subnet) addressing is more flexible

## 11.12. Obtaining a Block of Addresses

- IP addresses are managed by the *Internet Corporation for Assigned Names and Numbers* (ICANN) to avoid conflicts
- ICANN allocates addresses to regional Internet registries
- ISPs are allocated blocks of addresses by regional registries
- an organisation requests a block of addresses from its ISP
- in this example, an ISP divides its address block among 8 organisations:

```

ISP: 200.23.16.0/20 11001000 00010111 00010000 00000000
Org. 0: 200.23.16.0/23 11001000 00010111 00010000 00000000
Org. 1: 200.23.18.0/23 11001000 00010111 00010010 00000000
Org. 2: 200.23.20.0/23 11001000 00010111 00010100 00000000
...
Org. 7: 200.23.30.0/23 11001000 00010111 00011110 00000000
  
```

## 11.13. Special IP Addresses

Prefix	Suffix	Type Of Address	Purpose
all-0s	all-0s	this computer	used during bootstrap
network	all-0s	network	identifies a network
network	all-1s	directed broadcast	broadcast on specified net
all-1s	all-1s	limited broadcast	broadcast on local net
127	any	loopback	testing

- IP defines the above set of special addresses that are *reserved* and never assigned to hosts
- the following three blocks of the IP address space are reserved for *private* internets:

```

10.0.0.0 - 10.255.255.255 (10/8 prefix)
172.16.0.0 - 172.31.255.255 (172.16/12 prefix)
192.168.0.0 - 192.168.255.255 (192.168/16 prefix)
  
```

## 11.14. IP Loopback Address

- IP reserves 127/8 for *loopback*
- the host address used with loopback is irrelevant
- programmers often use 127.0.0.1 as the *loopback address*
- it enables a user to run both a client and a server on the same machine
- packets sent to the loopback address never appear on the network

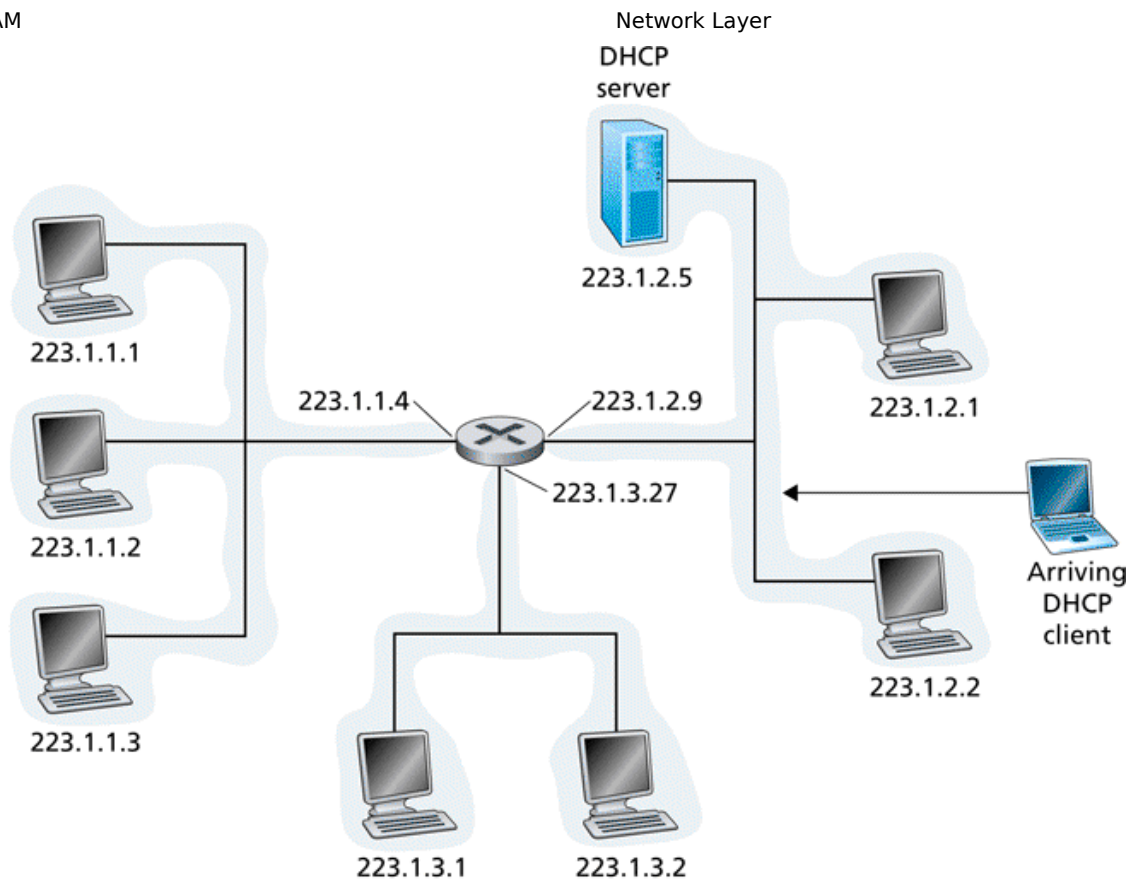
## 11.15. Obtaining a Host Address

- host IP addresses can be configured manually
- more likely that an IP address will be obtained *automatically* when a computer boots
- this is particularly the case for *mobile* computers
- address allocation is done using the *Dynamic Host Configuration Protocol* (DHCP)
- DHCP also allows a host to learn additional information, such as
  - its network prefix
  - the address of its first-hop router (often called the *default gateway*)
  - its local DNS server

## 11.16. Dynamic Host Configuration Protocol

- DHCP is a client-server protocol
- it uses UDP and the server runs on port 67
- in the simplest case, a subnet will have its own DHCP server
- in a home network, the router usually acts as the DHCP server





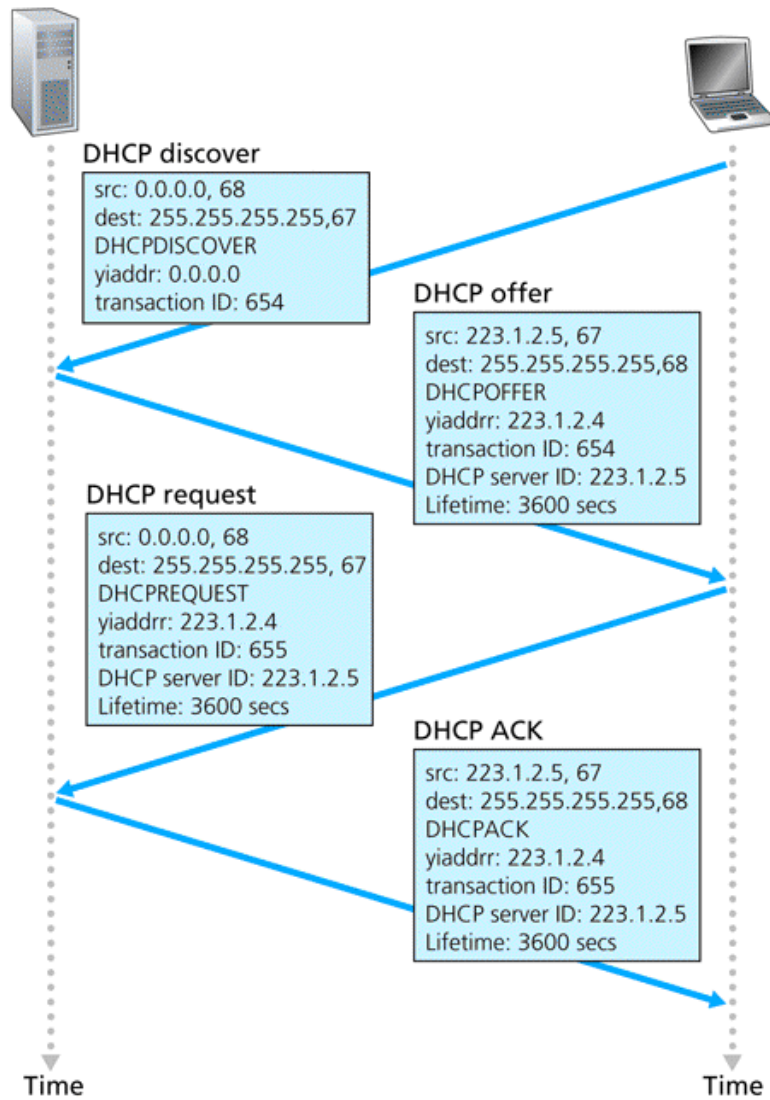
## 11.17. DHCP Client-Server Interaction

For a newly arriving host, DHCP is a four-step process:

- *DHCP server discovery*: note use of broadcast address (dest) and "this host" (src) for yiaddr ("Your Internet address")
- *DHCP server offer(s)*: note use of broadcast address and IP address lease time (Lifetime)
- *DHCP request*: used once client has chosen an offer
- *DHCP ACK*: confirmation from server

**DHCP server:**  
223.1.2.5

**Arriving client**

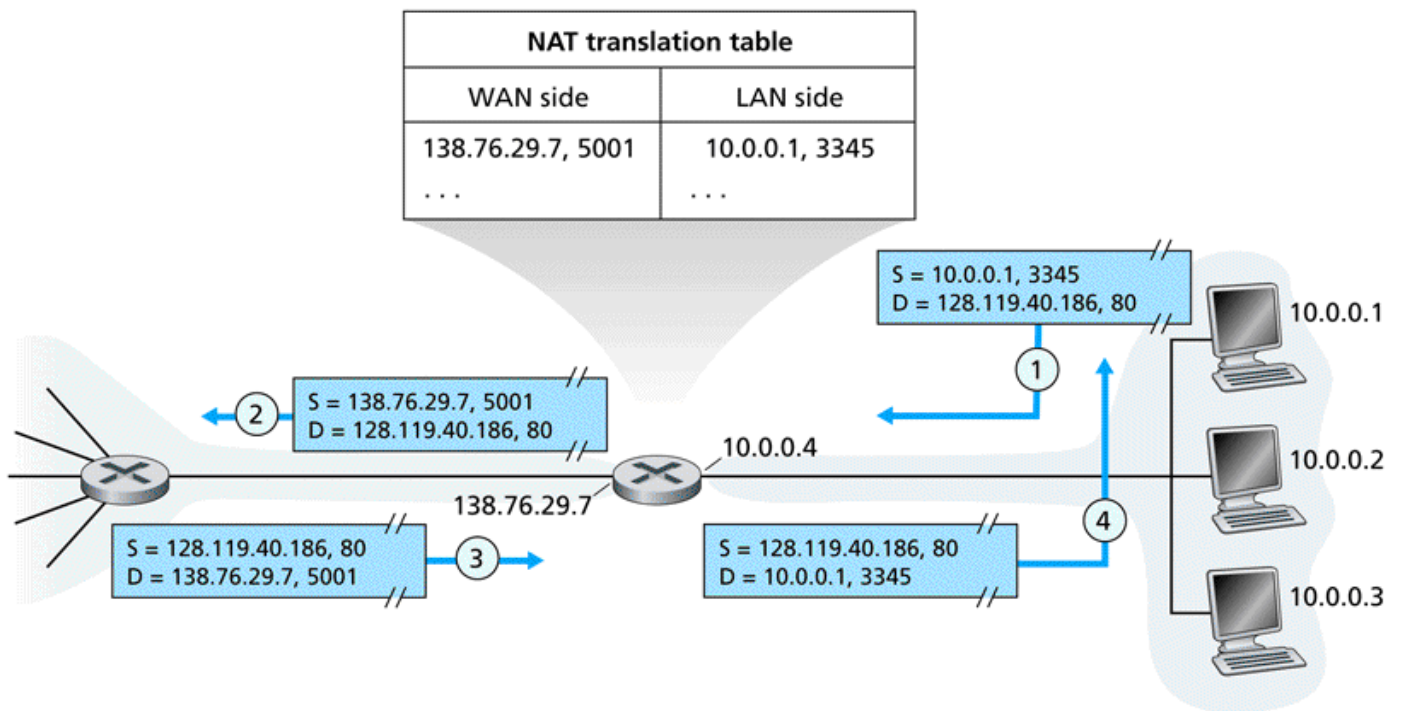


## 11.18. Network Address Translation

- every IP-capable device needs an IP address
- implies that every small office, home office (SOHO) subnet needs addresses allocated by ISP
- what if SOHO subnet needs more addresses, but ISP cannot allocate contiguous addresses?
- solution is to use the blocks of IP addresses reserved for [private internets](#)
- but these will not be unique, as required for successful communication
- one solution is to use *Network Address Translation* (NAT)
- NAT is usually enabled in home routers

## 11.19. NAT Example





- NAT-enabled router looks (to the outside world) like a single device with a single IP address
- all traffic leaving the home router has a source IP address of 138.76.29.7
- all traffic entering the home router has destination address 138.76.29.7
- how are the hosts within the home network distinguished?
- router uses a *NAT translation table* which includes port numbers
- the NAT router replaces the source IP address and source port number of a LAN datagram with its own IP address and an arbitrary port number (which is recorded in the table)
- the NAT router replaces the destination IP address and destination port number of a WAN datagram with the LAN IP address and port number from the table

## 11.20. Criticisms of NAT

- port numbers should be used for addressing *processes*, not hosts
- routers are supposed to process packets only up to the network layer
- NAT violates end-to-end principle: hosts should be communicating directly
- IPv6 should be used to solve problem of shortage of addresses
- nevertheless, NAT is widely deployed

## 11.21. IP Datagrams

- the network layer is a *connectionless* service
- packets created by a source travel from router to router until they reach a router that can deliver them to the destination
- these packets are called *IP datagrams*, whose format is illustrated below:



- the *header* contains information that controls where and how the datagram is sent
- the *data area* is also known as the *payload*
- the size of a datagram is determined by the application that uses it

## 11.22. Forwarding IP Datagrams

- datagrams follow a path from the source, through routers, to the destination
- each router examines the IP destination address (in the header) and uses a *forwarding table* (also called a *routing table*) to determine the next hop:



Destination	Next Hop
net 1	R <sub>1</sub>
net 2	deliver direct
net 3	deliver direct
net 4	R <sub>3</sub>

(b)

- (a) shows an internet with three routers connecting four physical networks
- (b) shows the conceptual forwarding table for router R<sub>2</sub>
- each entry in the table lists a destination network and the next hop to that network

### 11.23. Forwarding Table (1)

- each range of addresses in the table can be represented using CIDR notation
- the next hop can be represented by the interface on which to forward the datagram
- a typical forwarding table would also contain a *default route*
- the router will first scan the table in order to find a matching address range
- failing that, it will use the default route

Destination Address Range	Next Hop Interface
200.23.16.0/21	0
200.23.28.0/24	1
200.23.24.0/21	2
Default	3

### 11.24. Forwarding Table (2)

- what if a destination address matches more than one entry in the table?
- e.g., the address 11001000 00010111 00011100 10101010 matches
  - the first 24 bits of the second entry
  - the first 21 bits of the third entry
 in the following table:

Destination Address Range	Next Hop Interface
200.23.16.0/21 11001000 00010111 00010000 00000000	0
200.23.28.0/24 11001000 00010111 00011100 00000000	1
200.23.24.0/21 11001000 00010111 00011000 00000000	2
Default	3

- the router uses the *longest matching prefix rule*
- so the datagram is forwarded to interface 1

### 11.25. Software-Defined Networking (SDN)

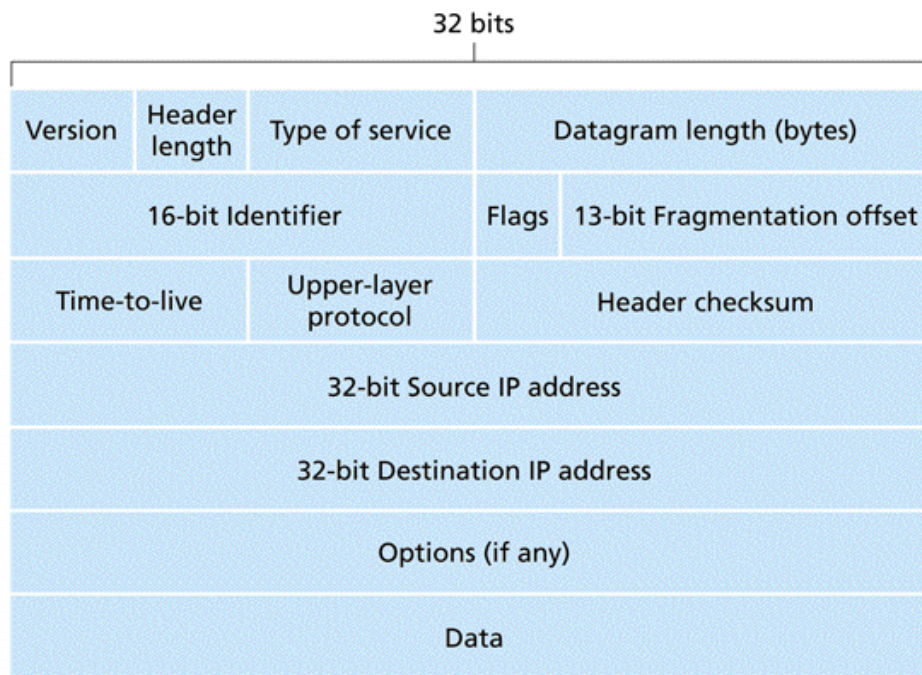
- traditional forwarding is based only on the datagram's destination address
- in SDN, a (software) *controller* (rather than the IP routing algorithms) determines the forwarding tables for the routers under its control
- forwarding tables have more complex "match-plus-action" behaviour
- *match* can be based on multiple header fields, from link, network and transport layers
- *action* can be to forward, drop or copy the datagram, possibly rewriting header fields
- this can be useful in, e.g.:
  - network address translation

- firewalls
- load balancing

## 11.26. IP Best-Effort Delivery

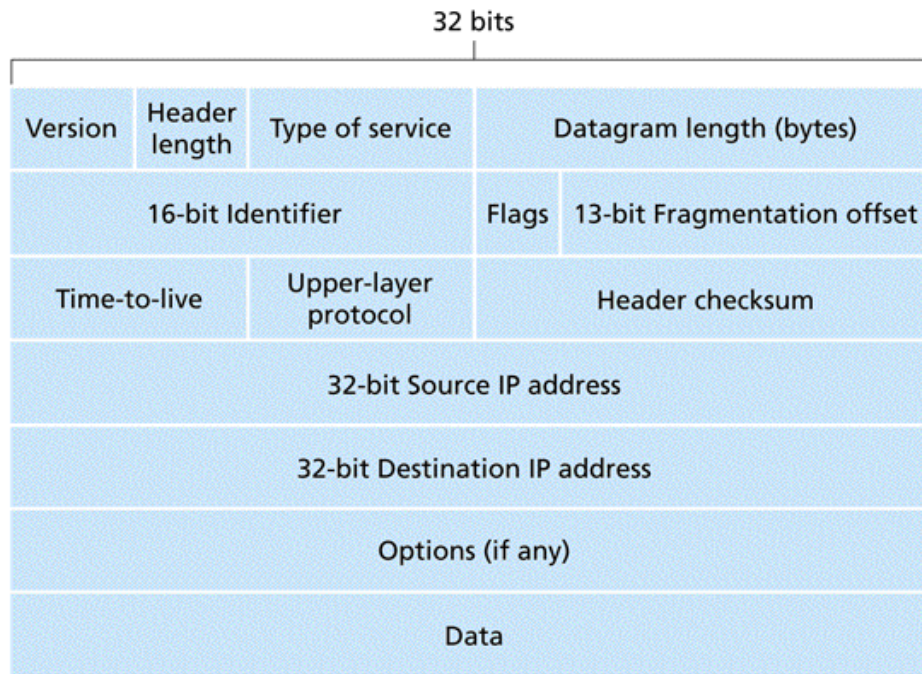
- the standard states that IP will make a *best-effort* attempt to deliver a datagram
- this is because there are a number of problems out of its control:
  - datagram duplication
  - delayed or out-of-order delivery
  - corruption of data
  - datagram loss
- the underlying physical networks can cause such problems
- the higher protocol layers (e.g., TCP) are required to handle these errors

## 11.27. IP Datagram Format (1)



- *Version*: the IP protocol version number; figure uses version 4 header
- *Header length* (in 32-bit words); required because options may be present in the header
- *Type of service*: 8-bit field for differentiated service (seldom used)
- *Datagram length* (16 bits): number of bytes in header and data
- *Identifier*, *Flags* and *Fragmentation offset* are for IP fragmentation (see later)

## 11.28. IP Datagram Format (2)



- *Time-to-live* (TTL)
  - initialised by the sender
  - decremented by one at each router that handles the datagram
  - when it reaches zero, the datagram is discarded and the sender notified with an ICMP message (see later)
- *Protocol* identifies which transport-level protocol should receive the data (e.g. ICMP, TCP or UDP)
- *Header checksum* is calculated over the header only; checked by routers which discard datagram if wrong
- *Source IP address* identifies original sender
- *Destination IP address* identifies final recipient
- a variable-length list of options may also be present
- padding is required to expand the options to a 32-bit word boundary

## 11.29. Encapsulation

- datagrams are passed down to the link layer in order to be sent over a physical network
- recall that "packets" at the link layer are called *frames*
- so an IP datagram is *encapsulated* in the data area of the frame as illustrated below

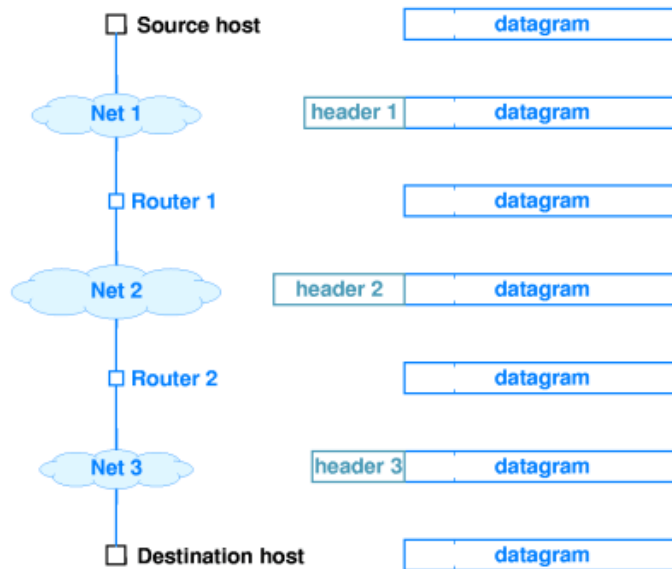


- in practice, some hardware technologies include a frame trailer as well as a frame header
- the receiver of an incoming frame knows that the payload is an IP datagram because of the value of the *frame type* field in the *frame header* (see later)

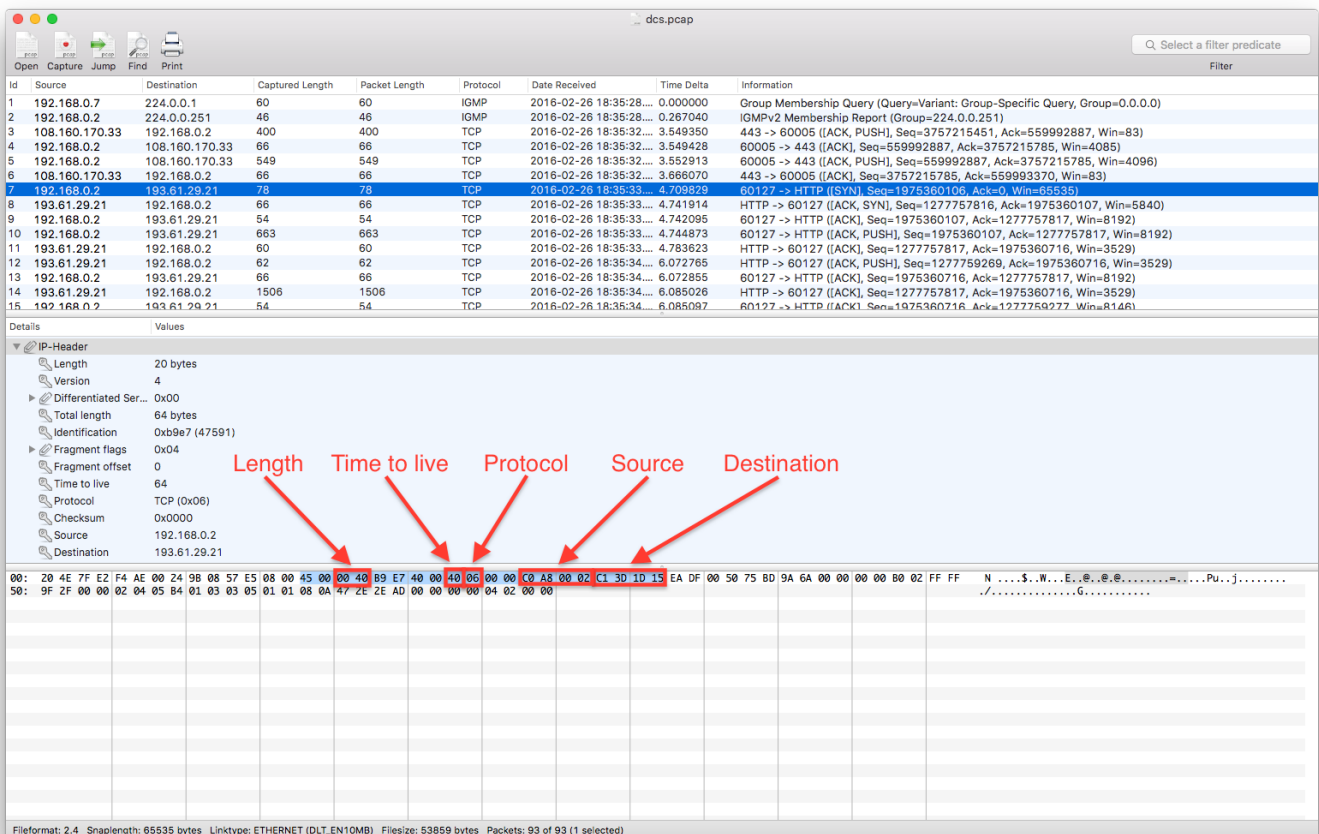
## 11.30. Transmission across an Internet

- encapsulation applies to one transmission at a time
- when the frame arrives at the next hop, the IP datagram is removed and the frame discarded
- if the datagram must be forwarded across another network, a new frame is created and the encapsulation is repeated
- each network can use a different hardware technology, so frame formats can differ
- figure below illustrates how a datagram appears as it is encapsulated and unencapsulated from source to destination





### 11.31. Example of IP Header



### 11.32. Maximum Transmission Unit (MTU)

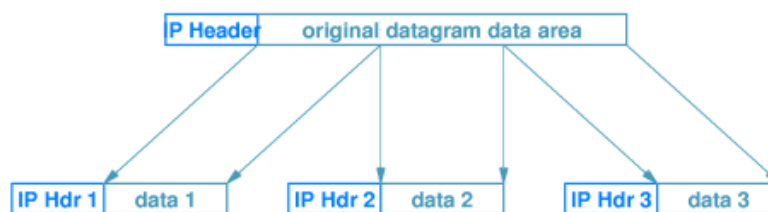
- each hardware technology specifies the maximum amount of data that a frame can carry
- this is called the *Maximum Transmission Unit* (MTU)
- figure below illustrates a router connecting two networks with different MTUs



- host  $H_2$  can only transmit datagrams containing 1,000 bytes or less, which router  $R$  can forward across network 1
- however, if host  $H_1$  transmits a 1,500-byte datagram, router  $R$  cannot send it across network 2

### 11.33. Datagram Fragmentation (1)

- IP uses a technique called *fragmentation* to solve the problem of differing MTUs
- if a datagram is larger than the network MTU, it is divided into smaller *fragments* which are each sent separately
- this process is illustrated below



### 11.34. Datagram Fragmentation (2)

- a host or router uses the MTU and the datagram header size to calculate how many fragments are required (they must be in multiples of 8 bytes)
- then the header of the original datagram is copied into the headers of each of the fragments
- the following fields change
  - *Datagram length* to reflect the smaller size
  - *Flags* to indicate this is a fragment
  - *Fragmentation offset* to reflect the position of the fragment within the *original* datagram
  - *Header checksum*
- crucially, the *Identification* field does not change
- each fragment becomes its own datagram and is routed independently of any other datagrams
- this makes it possible for the fragments of the original datagram to arrive at the final destination out of order

### 11.35. Datagram Reassembly

- at the final destination, the process of re-constructing the original datagram is called *reassembly*
- the unique *Identification* field groups fragments together
- the *Fragmentation offset* field tells the receiver how to order the fragments
- a bit in *Flags* signals the last fragment
- an example is illustrated below



- assume host  $H_1$  sends a 1,500 byte datagram (20-byte header and 1,480 bytes of data) to host  $H_2$
- so router  $R_1$  will fragment the datagram into two fragments



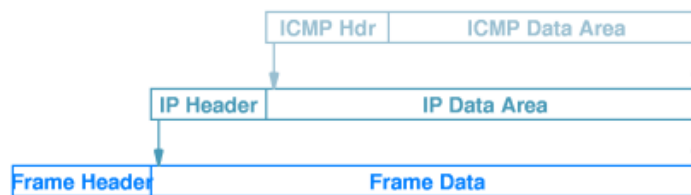
- the first fragment will contain a 20-byte header (*Fragmentation offset* is zero) and 976 bytes of data (four bytes are wasted)
- the second fragment will contain a 20-byte header (*Fragmentation offset* is  $976/8 = 122$ ) and the remaining 504 bytes of data

## 11.36. Fragment Loss

- with IP's best-effort delivery, it is possible for one or more fragments to be lost
- when the first fragment arrives, the receiver starts a timer
- if this timer expires before all the fragments have been received, the receiver *discards* those fragments it has received
- TCP can recover from the loss of the corresponding datagram
- there is no mechanism for a receiver to tell a sender which fragments arrived
- this makes sense because
  - the sender does not know about fragmentation
  - if it retransmits, the route and hence fragmentation may be different
- it is possible for a fragment to be fragmented
- the information in the fragments always refers to the *original* datagram

## 11.37. Internet Control Message Protocol (ICMP)

- if the IP software detects a transmission error, via an invalid header checksum, there is nothing that can be done - datagram is discarded
- however, less drastic errors can be reported back to the sender
- one example is a router decrementing the TTL field in the IP header to zero
- it discards the datagram, but reports the fact to the original source
- this is done using the *Internet Control Message Protocol* (ICMP)
- note that ICMP messages are carried in the payload area of an IP datagram, as illustrated below



## 11.38. Some ICMP Message Types

- the first two fields in the ICMP header are Type and Code

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	bad IP header

## 11.39. Some ICMP Message Descriptions

- *Destination Unreachable*
  - a router sends this error back to the source if it is unable to forward a datagram
- *Source Quench*
  - when a router runs out of buffer space, it starts discarding datagrams
  - when it discards a datagram, it sends a source quench to the sender requiring it to reduce the rate at which it is transmitting datagrams
  - in fact, this is seldom used because TCP performs congestion control
- *TTL Expired*
  - a router has decremented the TTL field in the IP header to zero
  - this error message is used by the `traceroute` application to construct a list of routers to a given destination
- *Bad IP Header*
  - one of the values in the IP header is wrong
- ICMP also includes message types for passing information between hosts, in particular between routers
  - e.g., *Echo Request/Echo Reply*, is used by the `ping` application

## 11.40. Routing Algorithms

- recall that routers use forwarding tables to determine the next hop for a datagram
- how are the forwarding tables calculated?
- typically a host is directly connected (e.g. on a LAN) to a single (*default*) router
- communication with this default router is done by the link layer (see later)
- so the problem is finding a route (or *path*) from a source router to a destination router
- this is done by *routing algorithms* executing in the routers and exchanging information
- or by SDN (software-defined networking) controllers

## 11.41. Autonomous Systems

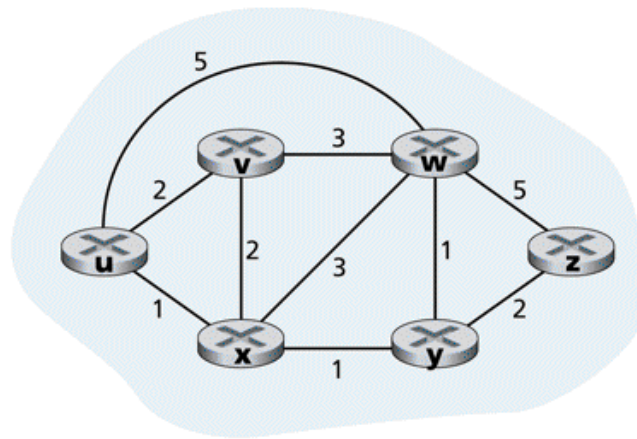
- in fact, Internet routers are grouped into *autonomous systems* (ASs)
- each autonomous system is assigned an Autonomous System Number (ASN)
- there are 5 Regional Internet Registries (RIRs) worldwide
- each RIR assigns ASNs to network operators
- the routers within each AS are under the same administrative control and run the same (*intra-AS*) routing algorithm
- ASs are connected by *gateway routers* running an (*inter-AS*) routing algorithm

## 11.42. Internet Routing Protocols

- there are a number of Internet routing protocols, e.g.:
  - *Routing Information Protocol* (RIP)
  - *Open Shortest Path First* (OSPF)
  - *Border Gateway Protocol* (BGP)
- RIP and OSPF are intra-AS protocols
- BGP is an inter-AS protocol
- we will consider only OSPF

## 11.43. Abstract Graph Model

- the routers and the connections between them can be represented as a *graph*
- a graph  $G = (N, E)$  consists of a set  $N$  of *nodes* and a set  $E$  of *edges*
- each edge in  $E$  is a pair of nodes  $(x, y)$
- so the nodes represent routers and the edges represent connections:



- each edge  $(x,y)$  also has a *cost*  $c(x,y)$  associated with it
- the cost might reflect physical length, link speed or monetary cost
- a *path* from  $x_1$  to  $x_n$  is a sequence of nodes  $(x_1, x_2, \dots, x_n)$  such that each  $(x_i, x_{i+1})$  is an edge in  $E$
- the *cost* of a path is the sum of the costs of the edges in it
- routing is concerned with finding *least-cost paths* between pairs of nodes
- what is the least-cost path between  $u$  and  $w$ ?

## 11.44. Open Shortest Path First

- OSPF assumes that all link costs are known
- this is accomplished by nodes broadcasting packets to all other nodes
- the routing algorithm uses Dijkstra's algorithm for computing "shortest" paths in a graph
- Dijkstra's algorithm computes the least-cost path from a source node ( $u$  below) to all other nodes
- the following notation is used in the algorithm:
  - $D(v)$ : the current cost of the least-cost path from  $u$  to  $v$
  - $p(v)$ : previous node on current least-cost path to  $v$
  - $N'$ : subset of nodes to which least-cost path is known

## 11.45. OSPF Algorithm

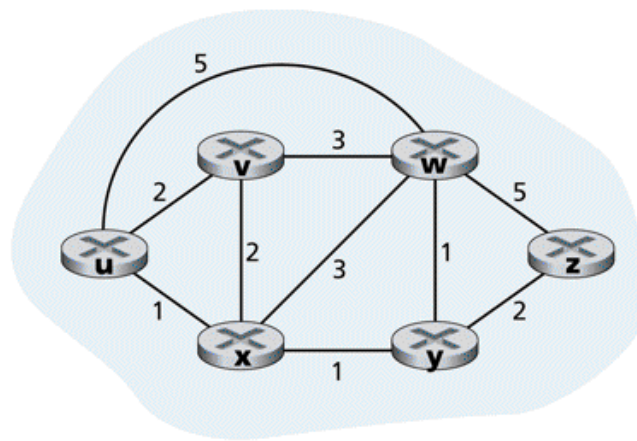
### Initialisation:

1.  $N' = \{u\}$
2. for all nodes  $v$
3.   if  $v$  is a neighbour of  $u$
4.     then  $D(v) = c(u,v)$
5.     else  $D(v) = \text{infinity}$
6. **Loop**
7.   find  $w$  not in  $N'$  such that  $D(w)$  is a minimum
8.   add  $w$  to  $N'$
9.   update  $D(v)$  for each neighbour  $v$  of  $w$  not in  $N'$ :
10.    $D(v) = \min( D(v), D(w) + c(w,v) )$
11. **until**  $N' = N$

- in line 10, the new cost to  $v$  is either the old cost to  $v$  or the cost of the known least-cost path to  $w$  plus the cost from  $w$  to  $v$

## 11.46. Example of OSPF Algorithm

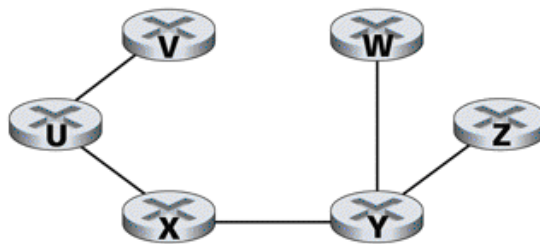
- running the OSPF algorithm on the graph



- gives rise to the following values:

step	$N'$	$D(v), p(v)$	$D(w), p(w)$	$D(x), p(x)$	$D(y), p(y)$	$D(z), p(z)$
0	u	2,u	5,u	1,u	$\infty$	$\infty$
1	ux	2,u	4,x		2,x	$\infty$
2	uxy	2,u	3,y			4,y
3	uxyv		3,y			4,y
4	uxyvw					4,y
5	uxyvwz					

- and the following least-cost paths and forwarding table for u:



Destination	Link
v	(u, v)
w	(u, x)
x	(u, x)
y	(u, x)
z	(u, x)

## 11.47. Links to more information

- The companion web site for [Tanenbaum's book](#), Chapter 5.
- [IANA IPv4 Address Space Registry](#).
- Network Address Translation is specified in [RFC3022](#) (2001)
- Address allocation for private Internets is specified in [RFC1918](#) (1996)
- Internet Control Message Protocol is specified in [RFC792](#) (1981)

See Chapters 4 and 5 of [Kurose and Ross], Chapters 20, 21, 22 and parts of 23 of [Comer] and parts of Chapter 5 of [Tanenbaum].