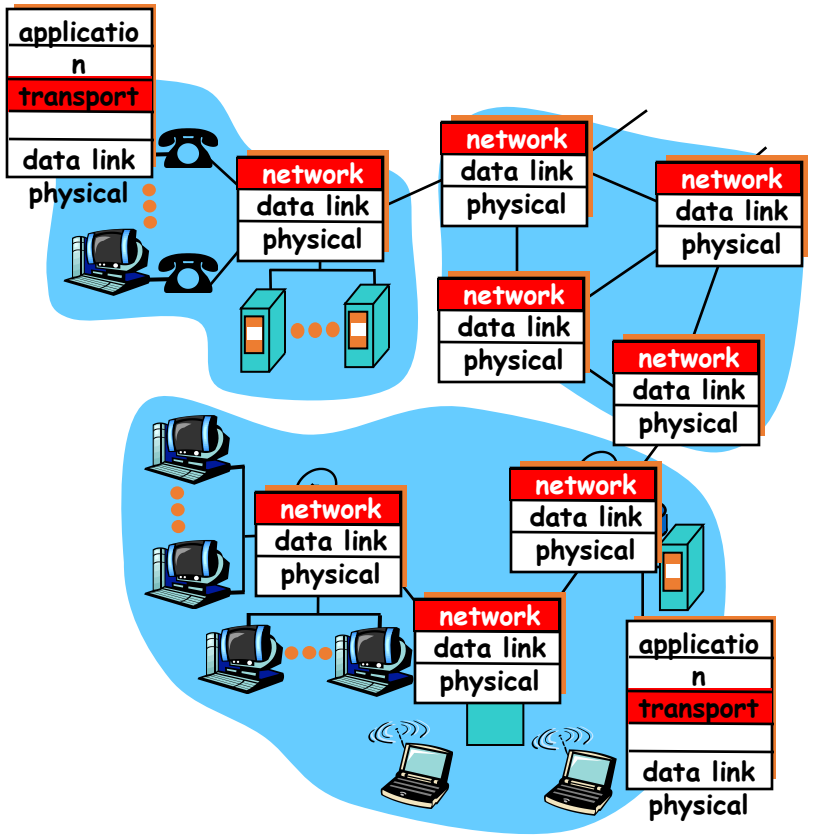
Chapter 5

Prepared by :

Dr. Mznah Al-Rodhaan & Dr. Adel Soudani

Recall Layering

- transport segment from sending to receiving host
- on sending side encapsulates segments into datagrams
- on rcving side, delivers segments to transport layer
- network layer protocols in *every* host, router
- Router examines header fields in all IP datagrams passing through it



Routing - Why Difficult ?

- Several algorithmic problems:
  - Many many paths - which is the best? ماهو أفضل طريق
  - Each path has changing characteristics
    - Queuing time varies, losses happen, router down …
  - How do you broadcast (find where someone is) إرسال خاص
  - How do you multicast (webTV, conference call) إرسال متعدد
  - How do routers perform routing at GBbps scale

- Several management problems:
  - How do you detect/diagnose faults
  - How do you do pricing, accounting

Key Network-Layer Functions

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

تحديد المسار الذي اتخذته الـ **packets** من المصدر إلى الوجهة

analogy:

عملية التخطيط لرحلة معينة من المصدر إلى الوجهة

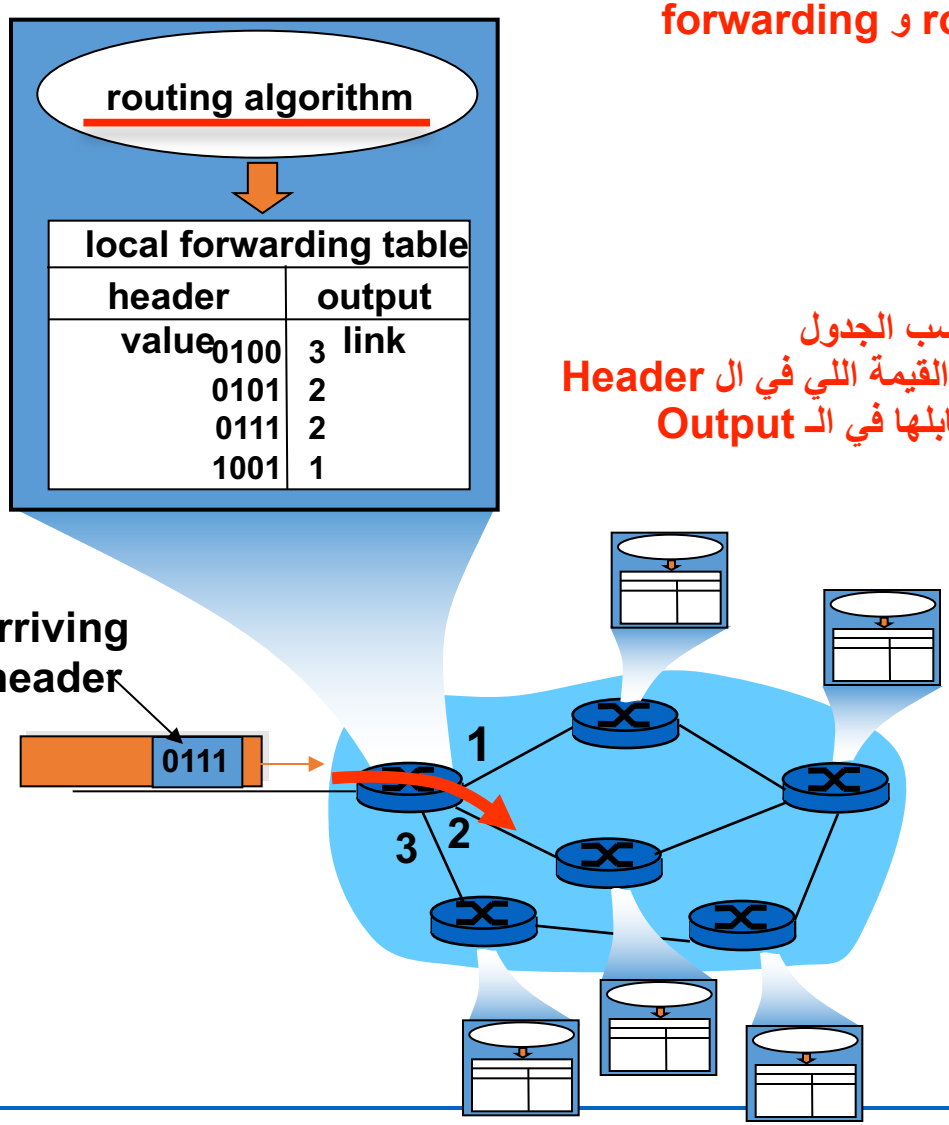
routing: *process of planning trip from source to dest*

forwarding*: process of getting through actual traffic intersections*

عملية الحصول على تقاطعات المرور الفعلية

Interplay between routing and forwarding

التفاعل بين الـ routing و forwarding



على حسب الجدول
نشوف القيمة اللي في ال Header
وش يقابلها في الـ Output

Two types of Network Architecture

- *Connection-Oriented* and *Connection-Less*

Virtual Circuit Switching

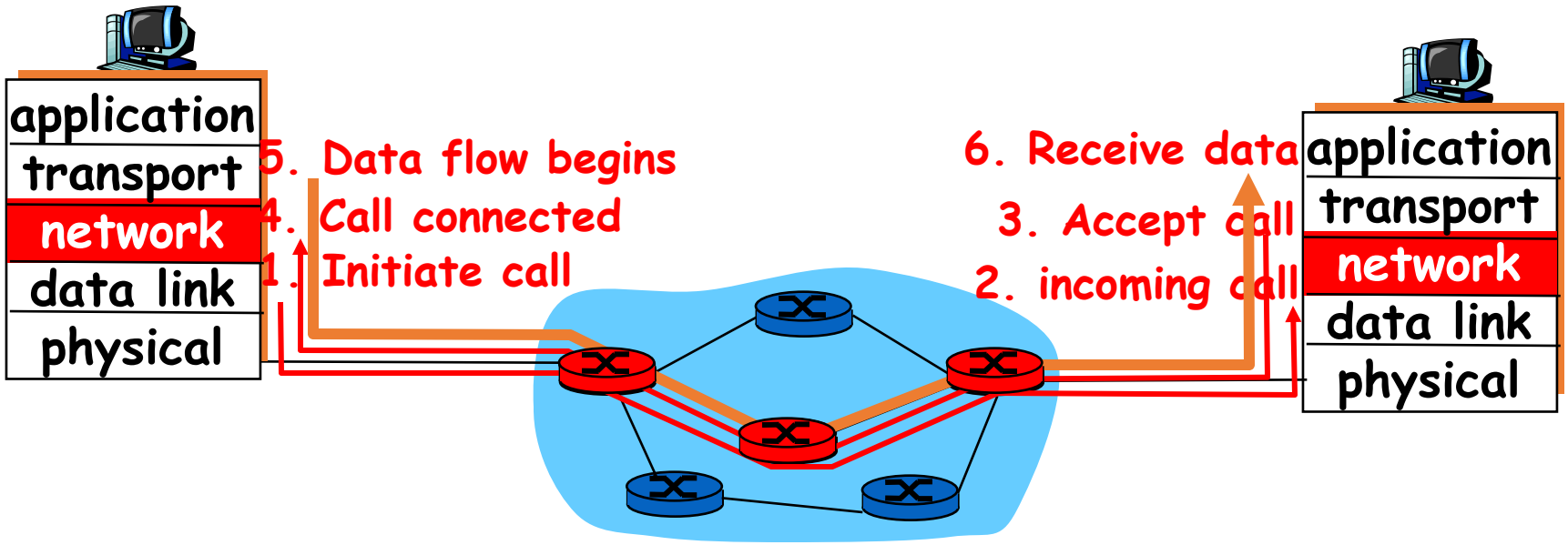
Example:ATM, X.25
Analogy: Telephone

Datagram forwarding

Example: IP networks
Analogy: Postal service

Virtual circuits: signaling protocols

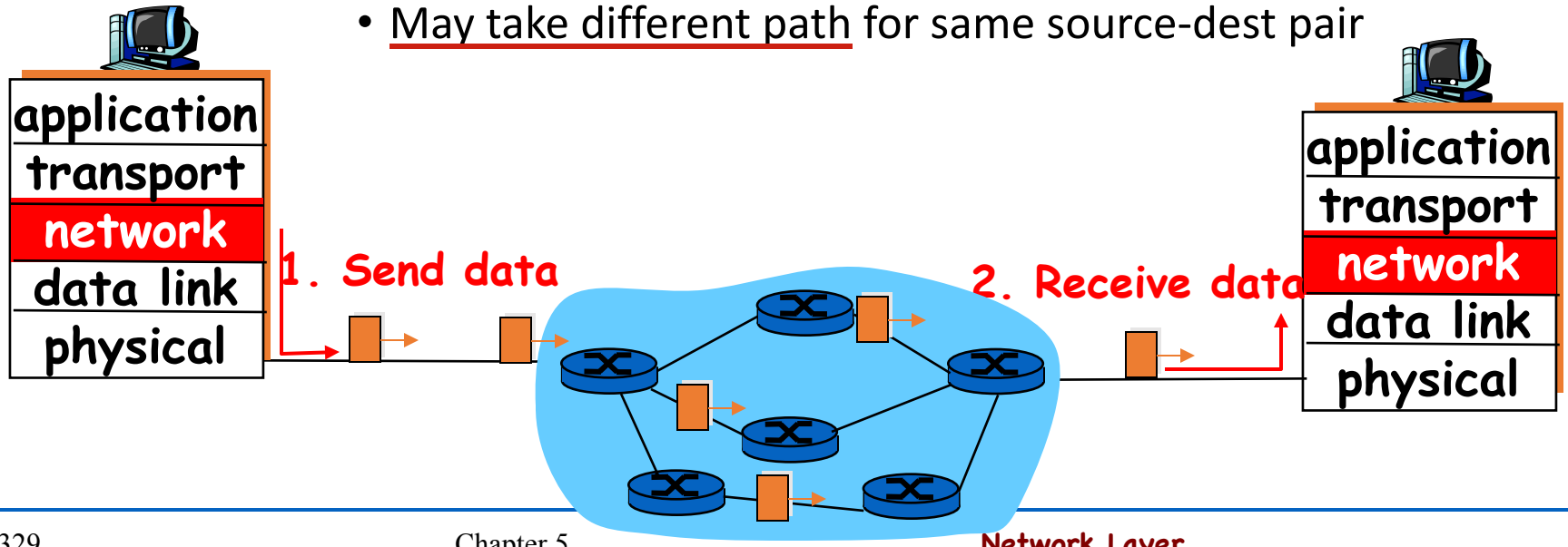
- used to setup, maintain teardown VC
- used in ATM, frame-relay, X.25
- not used in today's Internet



Datagram networks

- No call setup at network layer
- @ routers: no state about end-to-end connections
  - no concept of "connection"

- packets forwarded using destination host address
  - May take different path for same source-dest pair



Design Decisions

Virtual circuits

- Thoughts on why VC isn't great?

- Thoughts on why dataram may not be great?
  - Think of an application that's better with VC

Datagram or VC network: why?

Internet

- data traffic
  - "elastic" service, no strict timing req.
- "smart" end computers
  - simple network
  - complexity at "edge"
- many link types
  - different characteristics
  - uniform service difficult

ATM

- evolved from telephony
  - Call admission control
- human conversation:
  - strict timing, reliability requirements
  - need for guaranteed service
- "dumb" end systems
  - telephones
  - complexity inside network

Chapter 5: Network Layer

IP Adressing

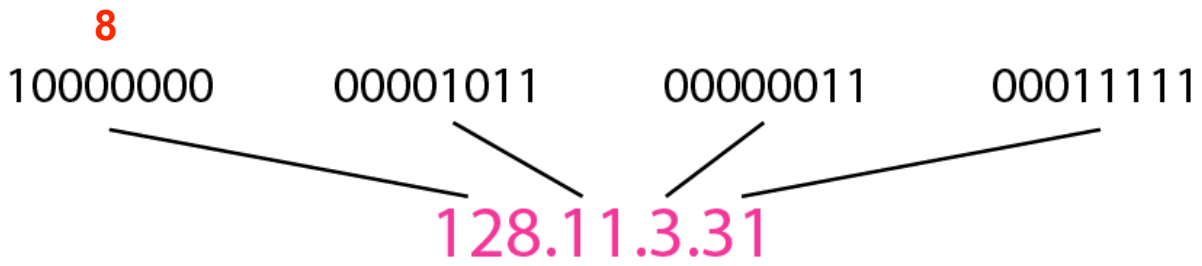
يعرِف بشكل فريد وعالمي اتصال الجهاز

An IPv4 address is a 32-bit address that uniquely and universally defines the connection of a device (for example, a computer or a router) to the Internet.

IPv4 Address

فريد وعالمي

- The IPv4 addresses are unique and universal.
- An IPv4 address is 32 bits long.
  - The address space of IPv4 is 2^{32} (4,294,967,296)
  - Notation.
    - Binary notation
    - Dotted-decimal notation



Example 1

Change the following IPv4 addresses from binary notation to dotted-decimal notation.

a. 10000001 00001011 00001011 11101111

b. 11000001 10000011 00011011 11111111

Solution

We replace each group of 8 bits with its equivalent decimal number (see Appendix B) and add dots for separation.

a. 129.11.11.239

b. 193.131.27.255

Example 2

Change the following IPv4 addresses from dotted-decimal notation to binary notation.

a. 111.56.45.78

b. 221.34.7.82

Solution

We replace each decimal number with its binary equivalent (see Appendix B).

a. 01101111 00111000 00101101 01001110

b. 11011101 00100010 00000111 01010010

Example 3

Find the error, if any, in the following IPv4

a. 111.56.045.78

b. 221.34.7.8.20

c. 75.45.301.14

d. 11100010.23.14.67

Solution

a. There must be no leading zero (045).

b. There can be no more than four numbers.

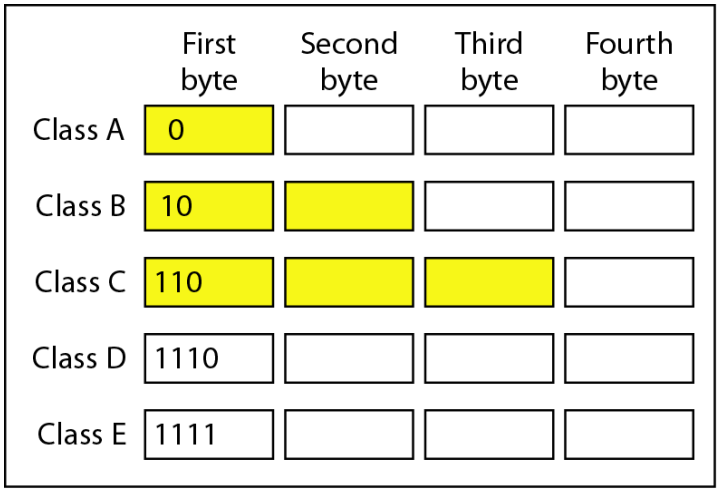
c. Each number needs to be less than or equal to 255.

d. A mixture of binary notation and dotted-decimal notation is not allowed.

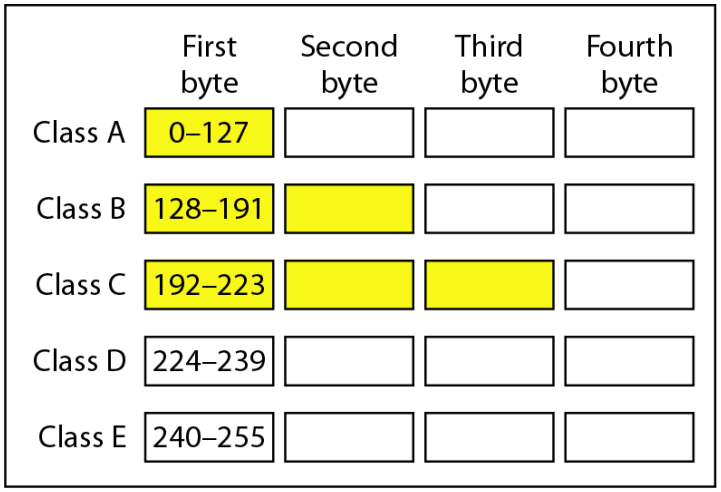
Classful Addressing

- In classful addressing, the address space is divided into five classes: A, B, C, D, and E.

الباقي في كلاس بي 6 bit
٦ + ٨ = ١٤ بالنسبة للـ ID



a. Binary notation



b. Dotted-decimal notation

Example 4

Find the class of each address.

a. 0**0000001 00001011 00001011 11101111**

b. 110**00001 10000011 00011011 11111111**

c. 14**.23.120.8**

d. 252**.5.15.111**

Solution

a. The first bit is 0. This is a class A address.

b. The first 2 bits are 1; the third bit is 0. This is a class C address.

c. The first byte is 14; the class is A.

d. The first byte is 252; the class is E.

Classes and Blocks

- The classful addressing wastes a large part of the address space.
  - Class A:
  - Class B:
  -
  -

Class	*Number of Blocks*	*Block Size*	*Application*
A	128	16,777,216	Unicast
B	16,384	65,536	Unicast
C	2,097,152	256	Unicast
D	1	268,435,456	Multicast
E	1	268,435,456	Reserved

Structure of IPv4 Address

- Consists of Net ID and Host ID.

Class	*Binary*	*Dotted-Decimal*	*CIDR*
A	**11111111** 00000000 00000000 00000000	**255**.0.0.0	/8
B	**11111111 11111111** 00000000 00000000	**255.255**.0.0	/16
C	**11111111 11111111 11111111** 00000000	**255.255.255**.0	/24

- Mask
  - 32-bit number of contiguous 1's followed by contiguous 0's.
  - To help to find the net ID and the host ID.

Use of IPv4 Address

- Subnetting
  - Divide a large address block into smaller sub-groups.
  - Use of flexible net mask.
- Supernetting
  - Exhausted class A and B address space
  - Huge demand for class B address space
  - To combine several contiguous address spaces into a larger single address space

Classless Addressing

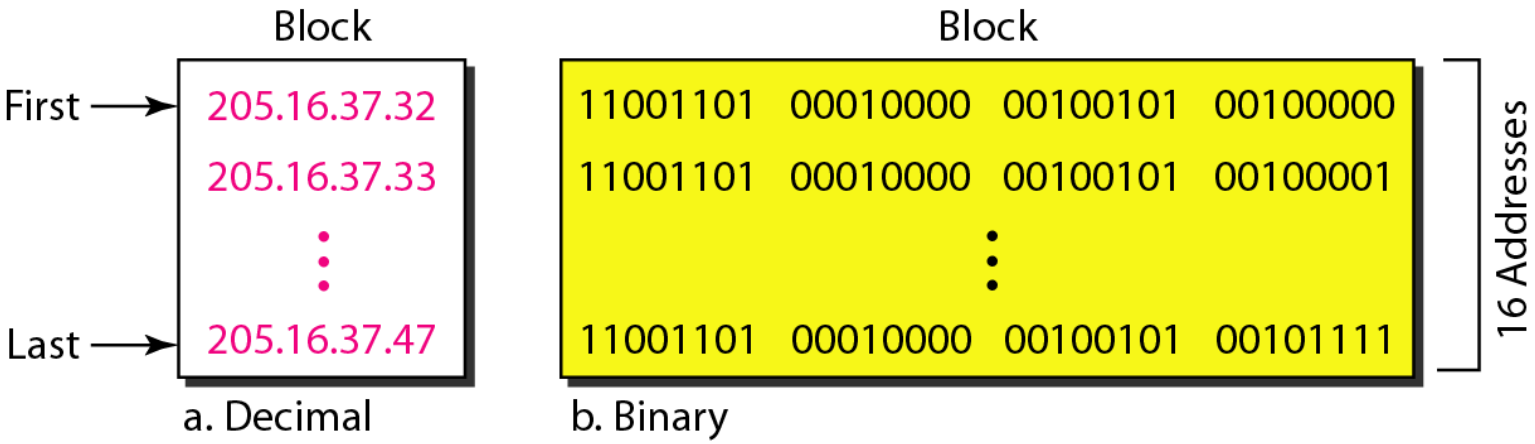
- To overcome the depletion of address space.
- Restriction
  - The addresses in a block must be contiguous.
  - The number of addresses in a block must be a power of 2.
  - The first address must be evenly divisible by the number of address.
- Mask
  - Consists of *n* consecutive 1's followed by zeros.
  - *n* can be any number b/w 0 and 32.
- Tips:
  - In IPv4 addressing, a block of addresses can be defined as x.y.z.t /n, in which x.y.z.t defines one of the addresses and the /n defines the mask.
  - The first address in the block can be found by setting the rightmost 32 − n bits to 0s.
  - The last address in the block can be found by setting the rightmost 32 − n bits to 1s.
  - The number of addresses in the block can be found by using the formula 2^{32-n}.

Example 5

Figure 3 shows a block of addresses, in both binary and dotted-decimal notation, granted to a small business that needs 16 addresses.

We can see that the restrictions are applied to this block. The addresses are contiguous. The number of addresses is a power of 2 ($16 = 2^4$), and the first address is divisible by 16. The first address, when converted to a decimal number, is 3,440,387,360, which when divided by 16 results in 215,024,210.

A block of 16 addresses granted to a small organization



Example 6

A block of addresses is granted to a small organization. We know that one of the addresses is 205.16.37.39/28. What is the first address in the block?

Solution

The binary representation of the given address is

11001101 00010000 00100101 00100111

If we set 32−28 rightmost bits to 0, we get

11001101 00010000 00100101 0010000

or

205.16.37.32.

This is actually the block shown in Figure 19.3.

Example 7

Find the last address for the block in Example 6.

Solution

The binary representation of the given address is

11001101 00010000 00100101 00100111

If we set 32 − 28 rightmost bits to 1, we get

11001101 00010000 00100101 00101111

or

205.16.37.47

This is actually the block shown in the previous figure .

Example 8

Find the number of addresses in Example 6.

Solution

The value of n is 28, which means that number of addresses is 2^{32-28} or 16.

Example 9

Another way to find the first address, the last address, and the number of addresses is to represent the mask as a 32-bit binary (or 8-digit hexadecimal) number. This is particularly useful when we are writing a program to find these pieces of information. In Example 5 the /28 can be represented as

11111111 11111111 11111111 11110000

(twenty-eight 1s and four 0s).

Find

a. The first address
b. The last address
c. The number of addresses.

Example 9 (continued)

Solution

a. The first address can be found by ANDing the given addresses with the mask. ANDing here is done bit by bit. The result of ANDing 2 bits is 1 if both bits are 1s; the result is 0 otherwise.

Address:	11001101	00010000	00100101	00100111
Mask:	**11111111**	**11111111**	**11111111**	**11110000**
First address:	11001101	00010000	00100101	00100000

Example 9 (continued)

b. The last address can be found by ORing the given addresses with the complement of the mask. Oring here is done bit by bit. The result of ORing 2 bits is 0 if both bits are 0s; the result is 1 otherwise. The complement of a number is found by changing each 1 to 0 and each 0 to 1.

Address:	11001101	00010000	00100101	00100111
Mask complement:	**00000000**	**00000000**	**00000000**	**00001111**
Last address:	11001101	00010000	00100101	00101111

Example 9 (continued)

c. The number of addresses can be found by complementing the mask, interpreting it as a decimal number, and adding 1 to it.

Mask complement:	00000000 00000000 00000000 00001111
Number of addresses:	15 + 1 = 16

Special Addresses

- Network address
  - The first address in a block is normally not assigned to any device; it is used as the network address that represents the organization to the rest of the world.

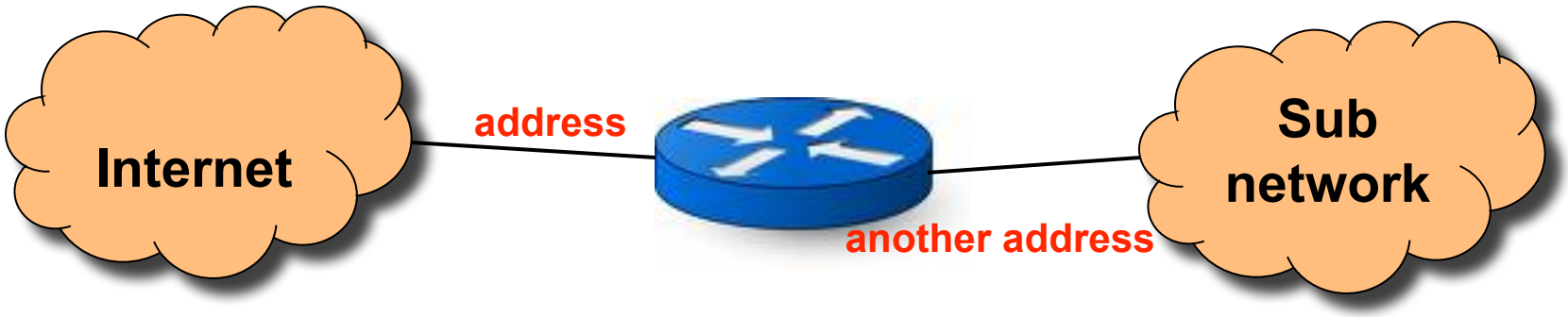
in class A : 10.0.0.0 is network address

- Broadcast address
  - The last address in a block is used for broadcasting to all devices under the network.

in class A : 10.255.255.255 is Broadcast address

Routing in IPv4

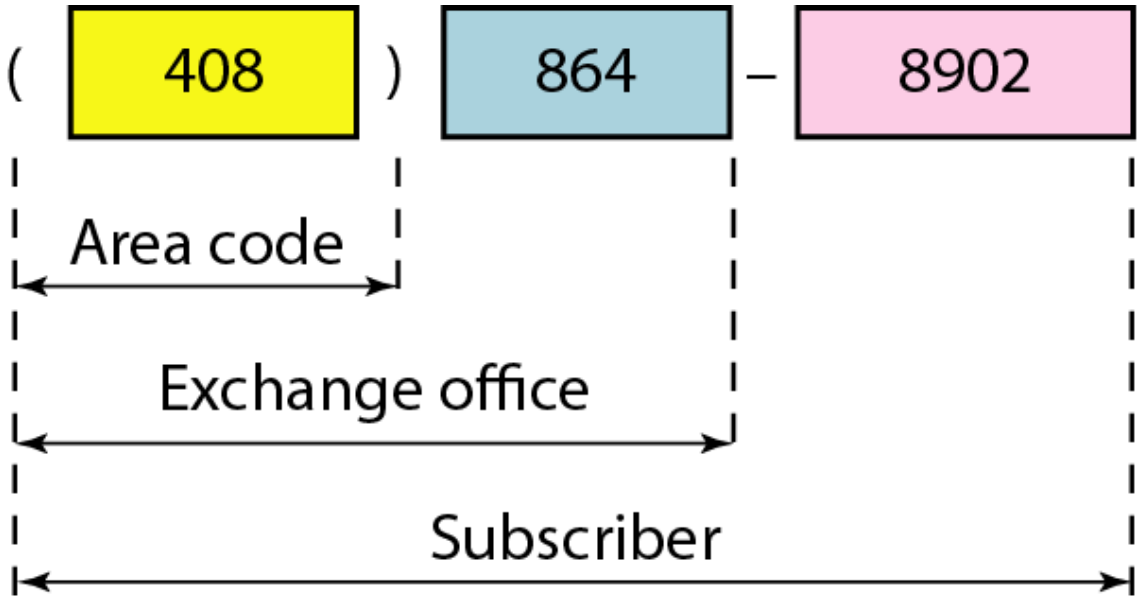
- A router has two addresses
  - An address through which the device inside of the router can be accessed.
  - Another address belongs to the granted block (sub-network).



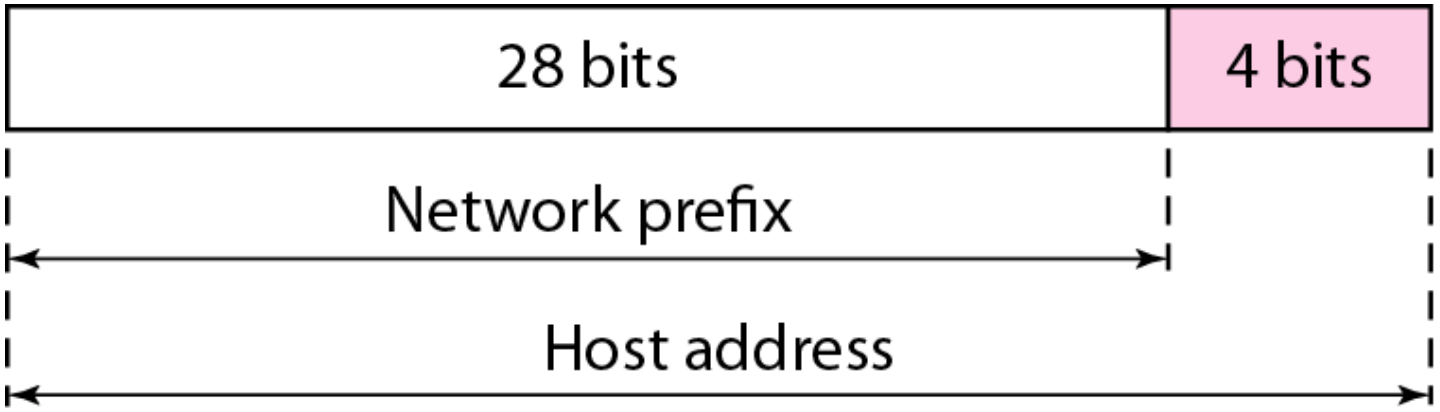
Hierarchy of IPv4 Addressing

- Each address in the block can be considered as a two-level hierarchical structure: the leftmost *n* bits (prefix) define the network; the rightmost 32 − *n* bits define the host.
- Why Hierarchy?

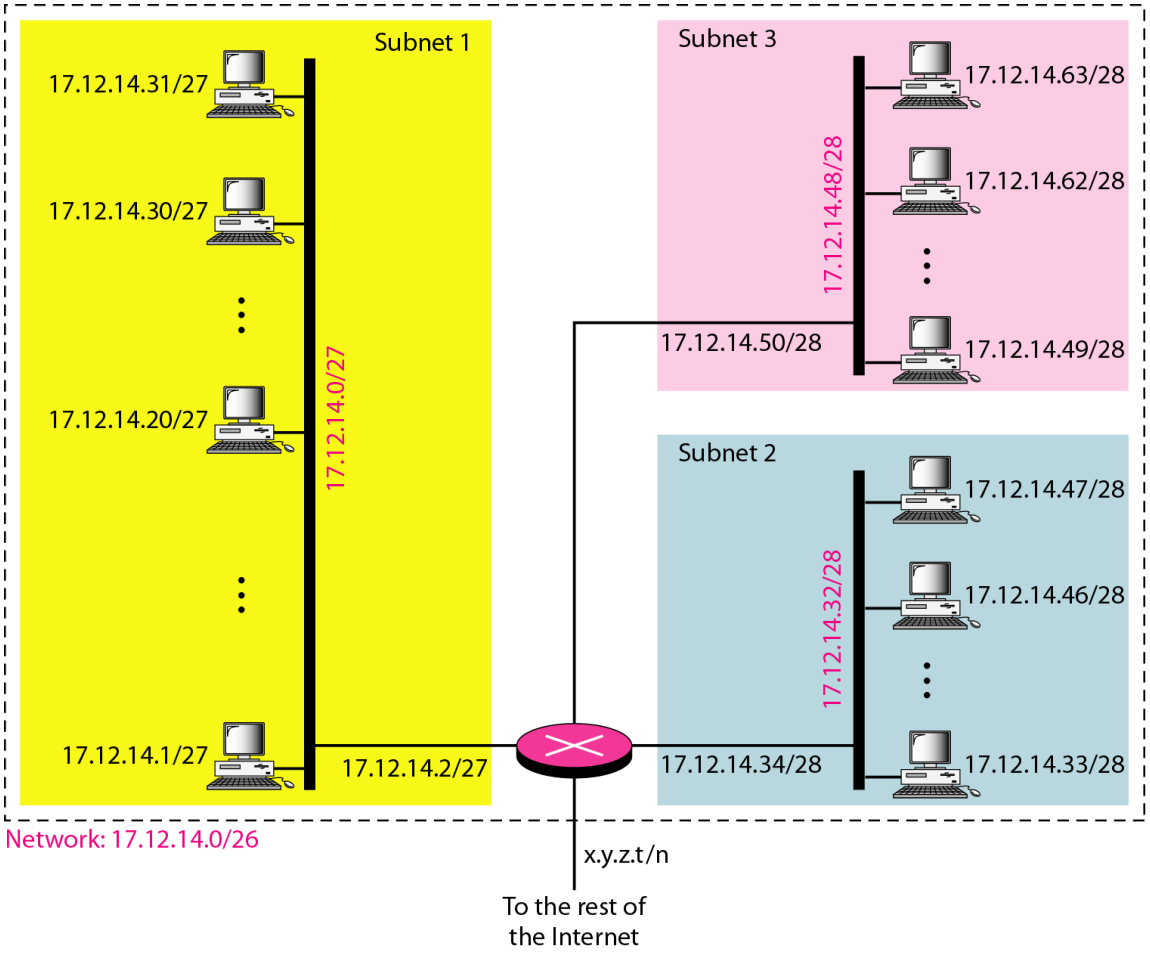
Figure 5 *Two levels of hierarchy in an IPv4 address*



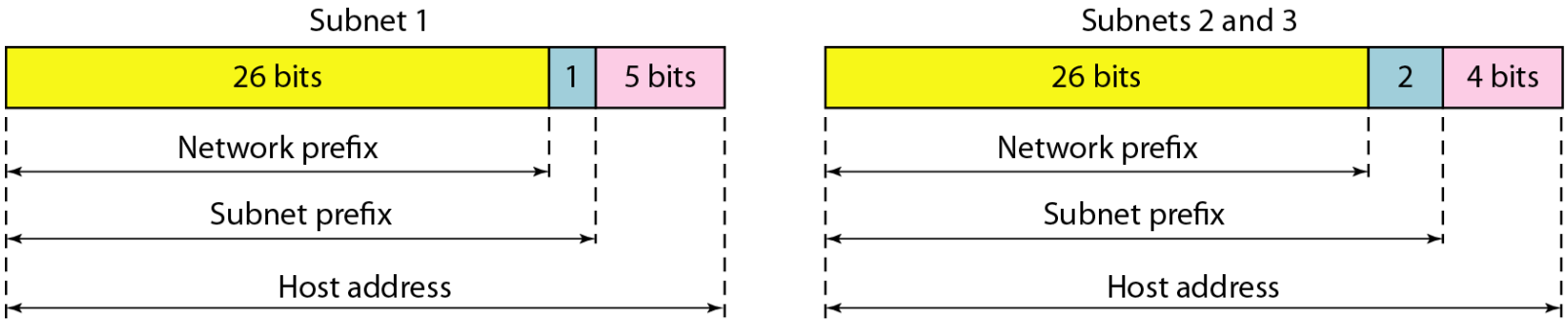
Two Level of Hierarchy



Three Level of Hierarchy



Three Level of Hierarchy



Example 10

An ISP is granted a block of addresses starting with 190.100.0.0/16 (65,536 addresses). The ISP needs to distribute these addresses to three groups of customers as follows:

a. The first group has 64 customers; each needs 256 addresses.

b. The second group has 128 customers; each needs 128 addresses.

c. The third group has 128 customers; each needs 64 addresses.

Design the subblocks and find out how many addresses are still available after these allocations.

Example 10 (continued)

Solution

Figure 9 shows the situation.

Group 1

For this group, each customer needs 256 addresses. This means that 8 (log2 256) bits are needed to define each host. The prefix length is then 32 − 8 = 24. The addresses are

1st Customer:	*190.100.0.0/24*	*190.100.0.255/24*
2nd Customer:	*190.100.1.0/24*	*190.100.1.255/24*
. . .		
64th Customer:	*190.100.63.0/24*	*190.100.63.255/24*

Total = 64 × 256 = 16,384

Example 10 (continued)

Group 2

For this group, each customer needs 128 addresses. This means that 7 (log2 128) bits are needed to define each host. The prefix length is then 32 − 7 = 25. The addresses are

1st Customer:	*190.100.64.0/25*	*190.100.64.127/25*
2nd Customer:	*190.100.64.128/25*	*190.100.64.255/25*
. . .		
128th Customer:	*190.100.127.128/25*	*190.100.127.255/25*

Total = 128 × 128 = 16,384

Example 10 (continued)

Group 3

For this group, each customer needs 64 addresses. This means that 6 ($log_2 64$) bits are needed to each host. The prefix length is then 32 − 6 = 26. The addresses are

1st Customer:	*190.100.128.0/26*	*190.100.128.63/26*
2nd Customer:	*190.100.128.64/26*	*190.100.128.127/26*
. . .		
128th Customer:	*190.100.159.192/26*	*190.100.159.255/26*

Total = 128 × 64 = 8192

Number of granted addresses to the ISP: 65,536
Number of allocated addresses by the ISP: 40,960
Number of available addresses: 24,576

Network Address Translation (NAT)

- Benefits
  - Use of a single IP address among many devices in a network
  - Use of a dynamic IP address for home user for sharing
- Private Addresses

Range			*Total*
10.0.0.0	to	10.255.255.255	2^{24}
172.16.0.0	to	172.31.255.255	2^{20}
192.168.0.0	to	192.168.255.255	2^{16}

Figure 10 *A NAT implementation*

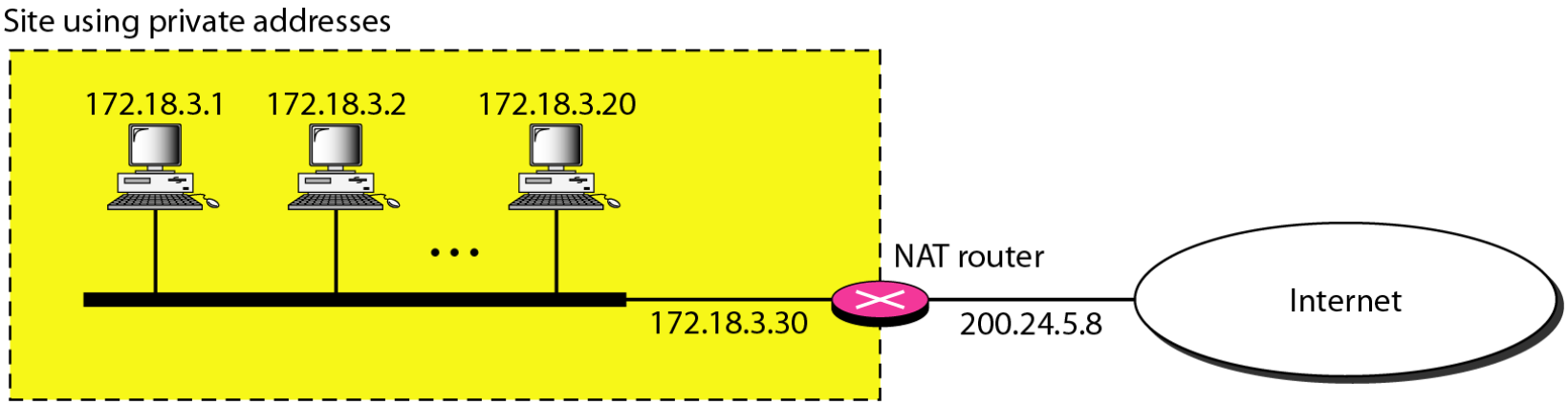


Figure 11 *Addresses in a NAT*

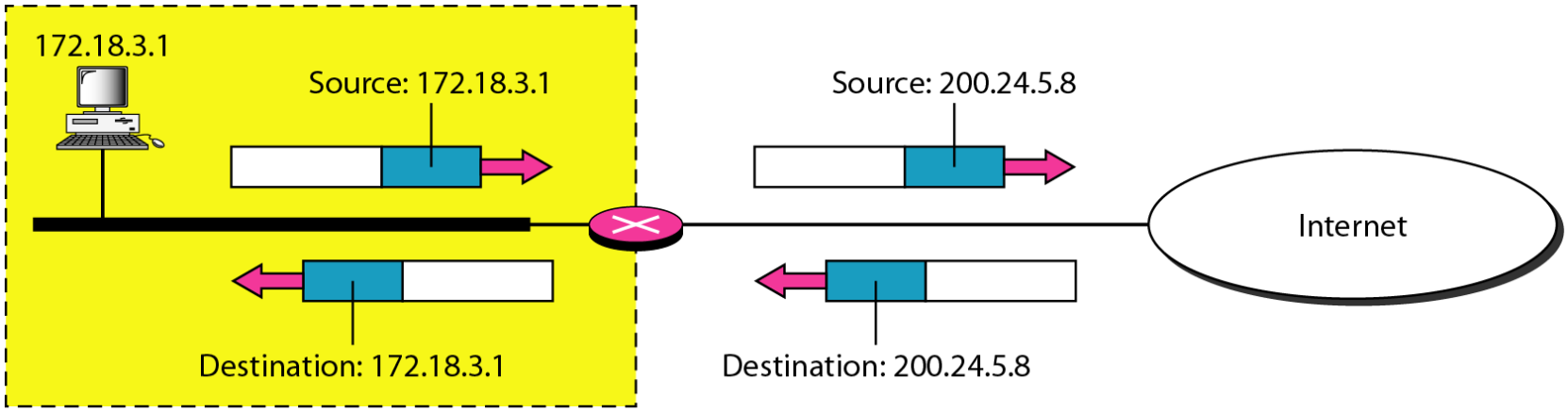


Figure 12 *NAT address translation*

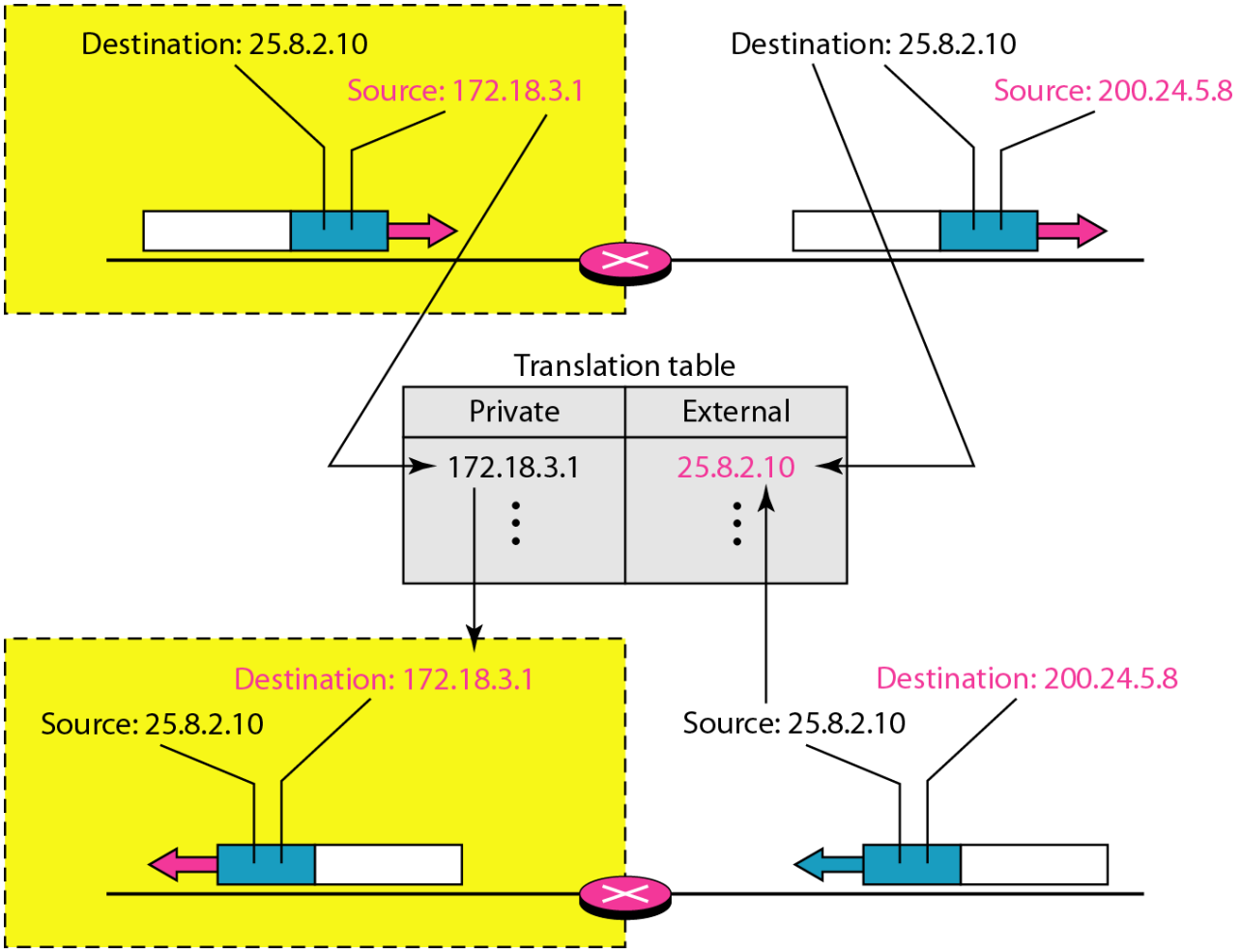


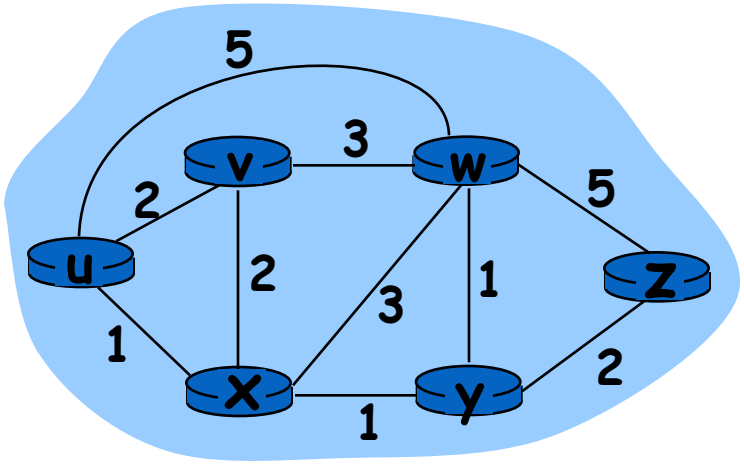
Table 4 *Five-column translation table*

Private Address	*Private Port*	*External Address*	*External Port*	*Transport Protocol*
172.18.3.1	1400	25.8.3.2	80	TCP
172.18.3.2	1401	25.8.3.2	80	TCP
. . .	. . .	. . .	. . .	. . .

Chapter 5: Network Layer

Routing Algorithms

Graph abstraction



Graph: G = (N,E)

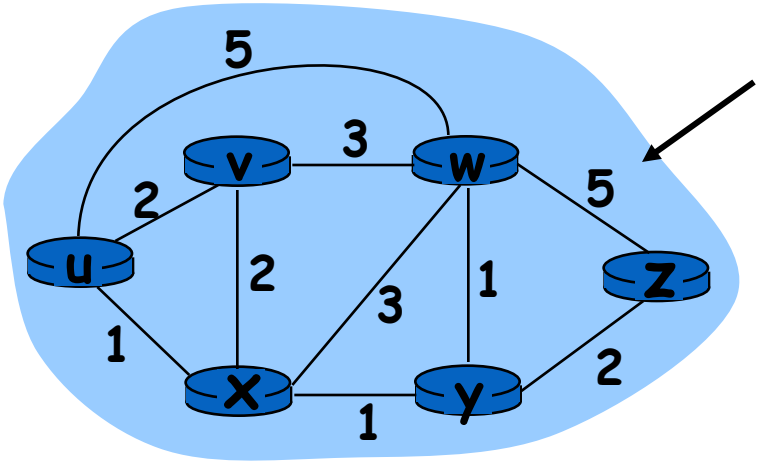
N = set of routers = { u, v, w, x, y, z }

E = set of links ={ (u,v), (u,x), (v,x), (v,w), (x,w), (x,y), (w,y), (w,z), (y,z) }

Remark: Graph abstraction is useful in other network contexts

Example: P2P, where N is set of peers and E is set of TCP connections

Graph abstraction: costs



What factors influence this cost ?

Should costs be only on links ?

Cost of path $(x_1, x_2, x_3, \ldots, x_p) = c(x_1,x_2) + c(x_2,x_3) + \ldots + c(x_{p-1},x_p)$

Question: What's the least-cost path between u and z ?

Routing algorithm: algorithm that finds least-cost path

Routing Algorithm classification

2 main classes:

Centralized

- all routers have complete topology, link cost info
- "link state" algorithms

Distributed:

- Each router knows link costs to neighbor routers only
- "distance vector" algorithms

A Link-State Routing Algorithm

Dijkstra's algorithm

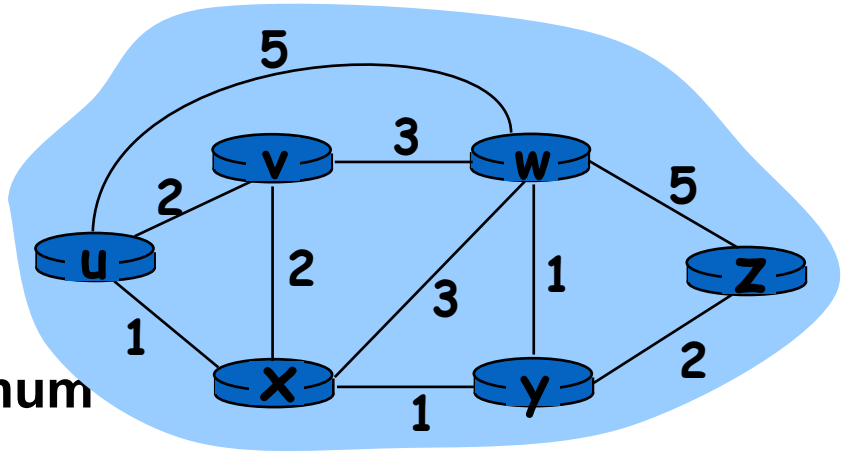
- Link costs known to all nodes
- computes least cost paths from one node ('source") to all other nodes
  - gives forwarding table for that node
- iterative: after k iterations, know least cost path to k dest.'s

Dijkstra's Algorithm

Notation:

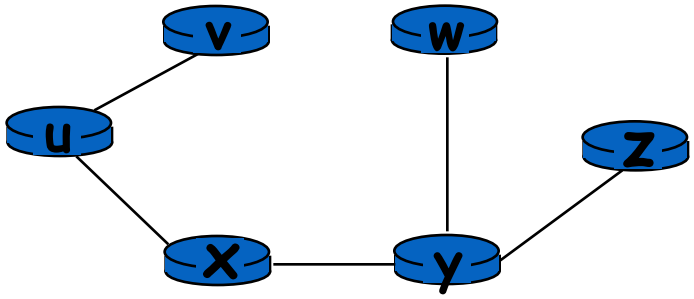
- c(x,y): link cost from node x to y; = ∞ if not direct neighbors
- D(v): current value of cost of path from source to dest. v

```
1  Initialization:
2    N' = {u}
3    for all nodes v
4      if v adjacent to u
5          then D(v) = c(u,v)
6      else D(v) = ∞
7
8  Loop
9     find w not in N' s.t. D(w) is a minimum
10    add w to N'
11    update D(v) for all v adjacent to w and not in N' :
12       D(v) = min( D(v), D(w) + c(w,v) )
13    /* new cost to v is either old cost to v or known
14     shortest path cost to w plus cost from w to v */
15  until all nodes in N'
```



Dijkstra's algorithm: example (2)

Resulting shortest-path tree from u:



Resulting forwarding table in u:

destination	link
v	(u,v)
x	(u,x)
y	(u,x)
w	(u,x)
z	(u,x)

Distributed: Distance Vector

- To find D, node S asks each neighbor X
  - How far X is from D
  - X asks its neighbors … comes back and says C(X,D)
  - Node S deduces C(S,D) = C(S,X) + C(X,D)
  - S chooses neighbor X_i that provides min C(S,D)

  - Later, X_j may find better route to D
  - X_j advertizes C(X_j,D)
  - All nodes update their cost to D if new min found

Distance Vector Algorithm

Bellman-Ford Equation (dynamic programming)

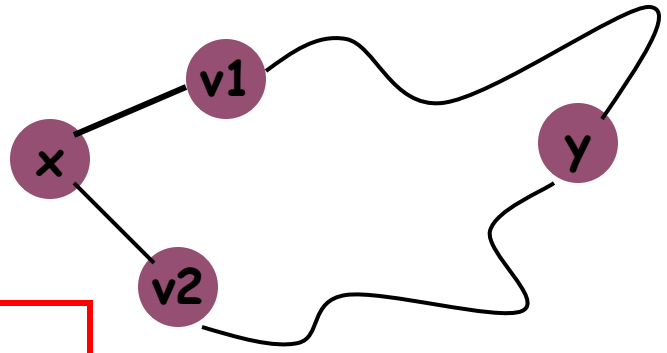
Define

$d_x(y)$:= cost of least-cost path from x to y

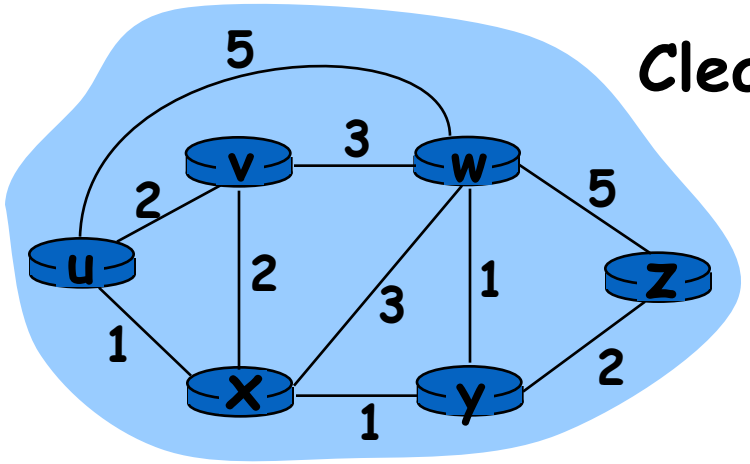
Then

$d_x(y) = \min \{c(x,v) + d_v(y) \}$

where min is taken over all neighbors v of x



Bellman-Ford example



Clearly, $d_v(z) = 5$, $d_x(z) = 3$, $d_w(z) = 3$

B-F equation says:

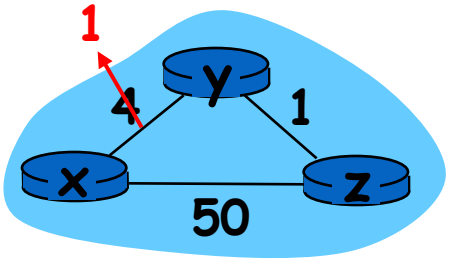
$$d_u(z) = \min \{ c(u,v) + d_v(z), \; c(u,x) + d_x(z), \; c(u,w) + d_w(z) \}$$
$$= \min \{2 + 5, \; 1 + 3, \; 5 + 3\} = 4$$

Node that achieves minimum is next hop in shortest path ➜ forwarding table

Distance Vector: link cost changes

Link cost changes:

- if DV changes, notify neighbors



At time t_0, y detects the link-cost change, updates its DV, and informs its neighbors.

At time t_1, z receives the update from y and updates its table. It computes a new least cost to x and sends its neighbors its DV.

At time t_2, y receives z's update and updates its distance table. y's least costs do not change and hence y does *not* send any message to z.