Chapter 5

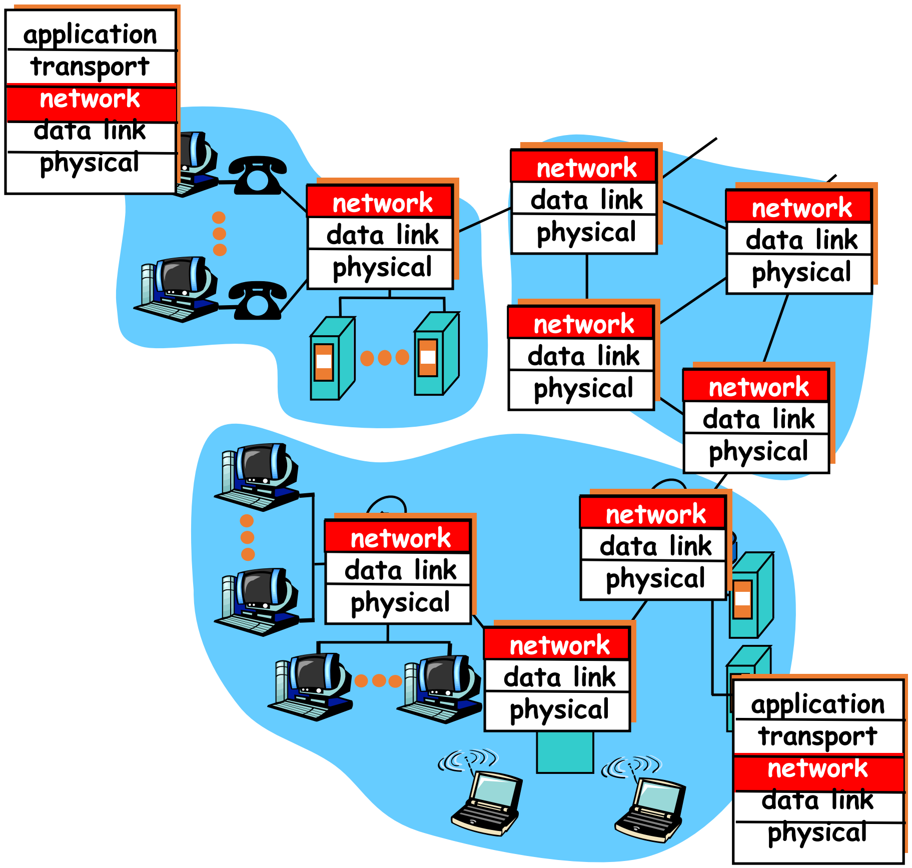
Network Layer

Prepared by :

Dr. Adel Soudani & Dr. Mznah Al-Rodhaan

Recall Layering

- Transport segment from sending to receiving host
- on sending side encapsulates segments into datagrams
- on receiving 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?
  - Each path has changing characteristics
    - Queuing time varies, losses happen, router down …
  - How do you broadcast (find where someone is)
  - How do you multicast (webTV, conference call)
  - 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

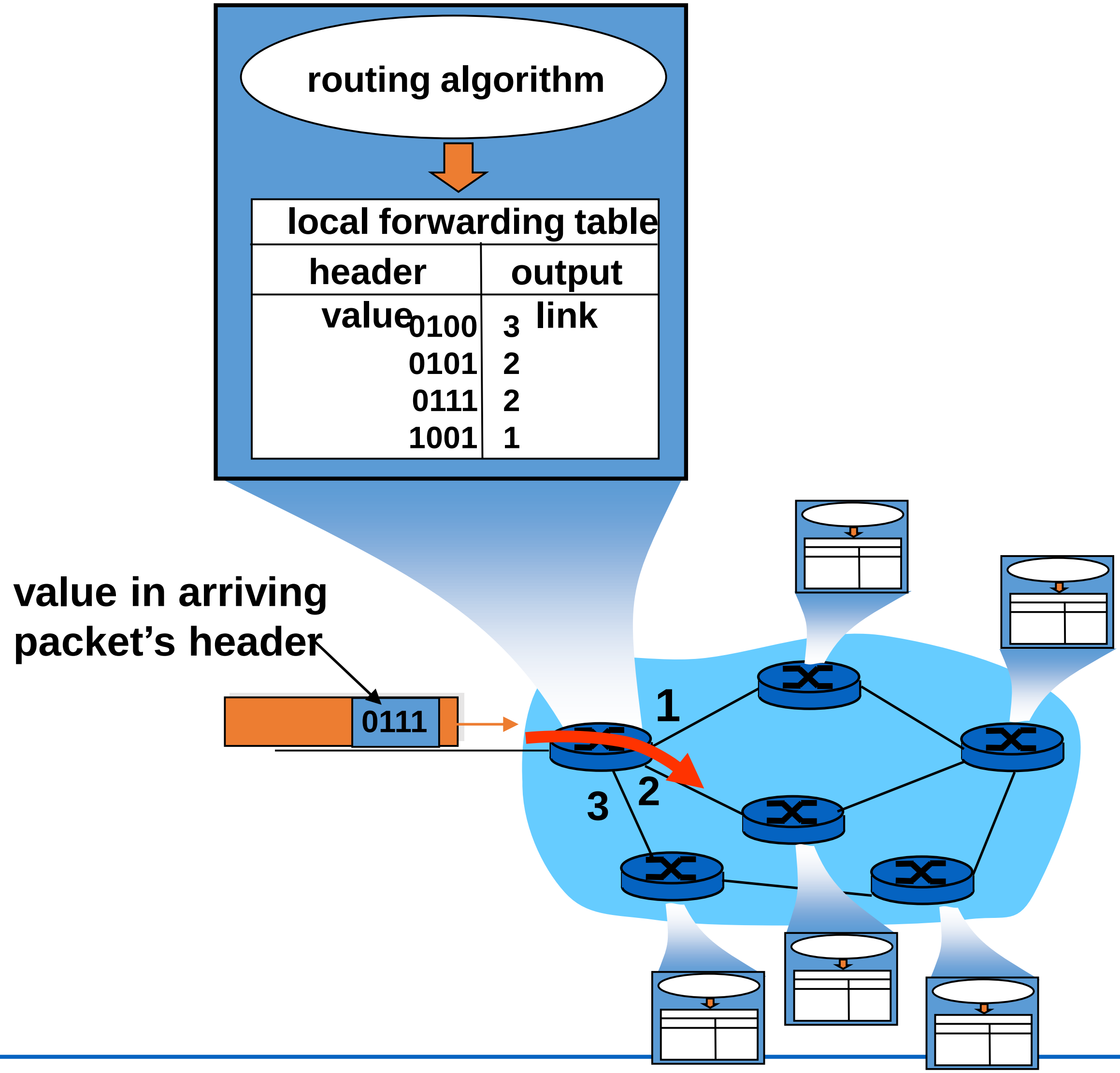
- *forwarding:* move packets from router's input to appropriate router output
- *routing:* determine route taken by packets from source to dest.

analogy:

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

forwarding*: process of getting through actual traffic intersections*

Interplay between routing and forwarding



Two types of Network Architecture

- *Connection-Oriented*

Connection-Less

Virtual Circuit Switching

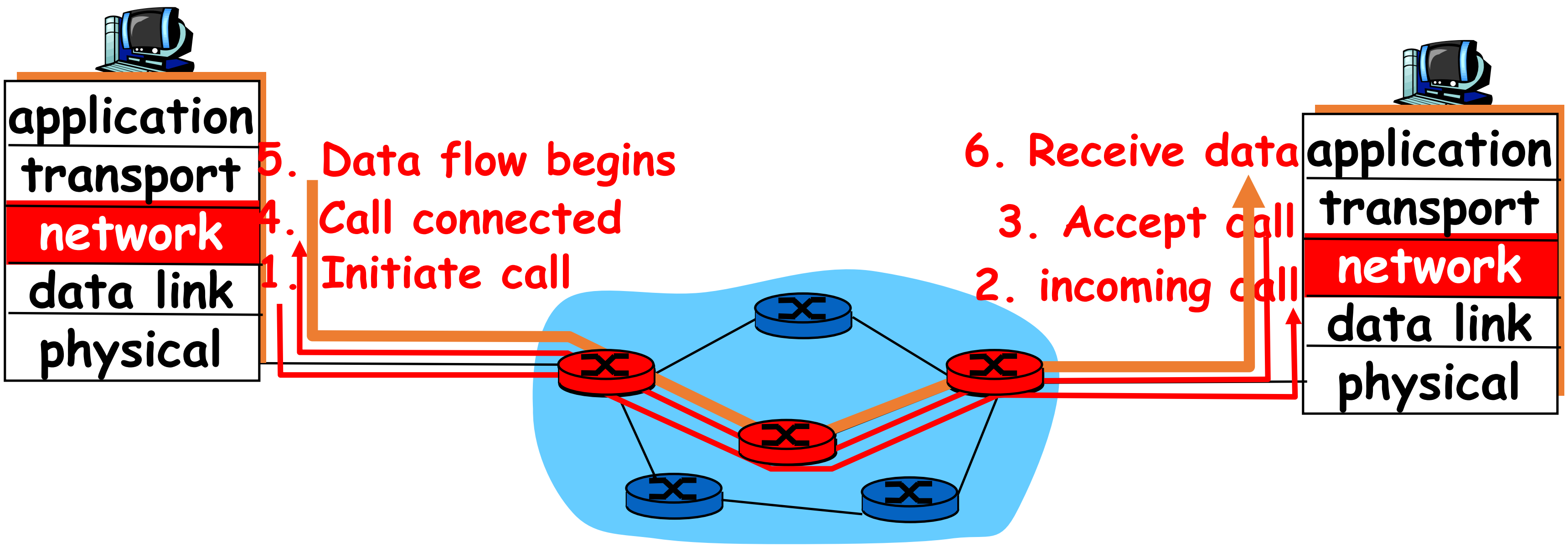
Example:ATM, X.25
Analogy: Telephone

Datagram forwarding

Example: IP networks
Analogy: Postal service

Virtual circuits: signaling protocols

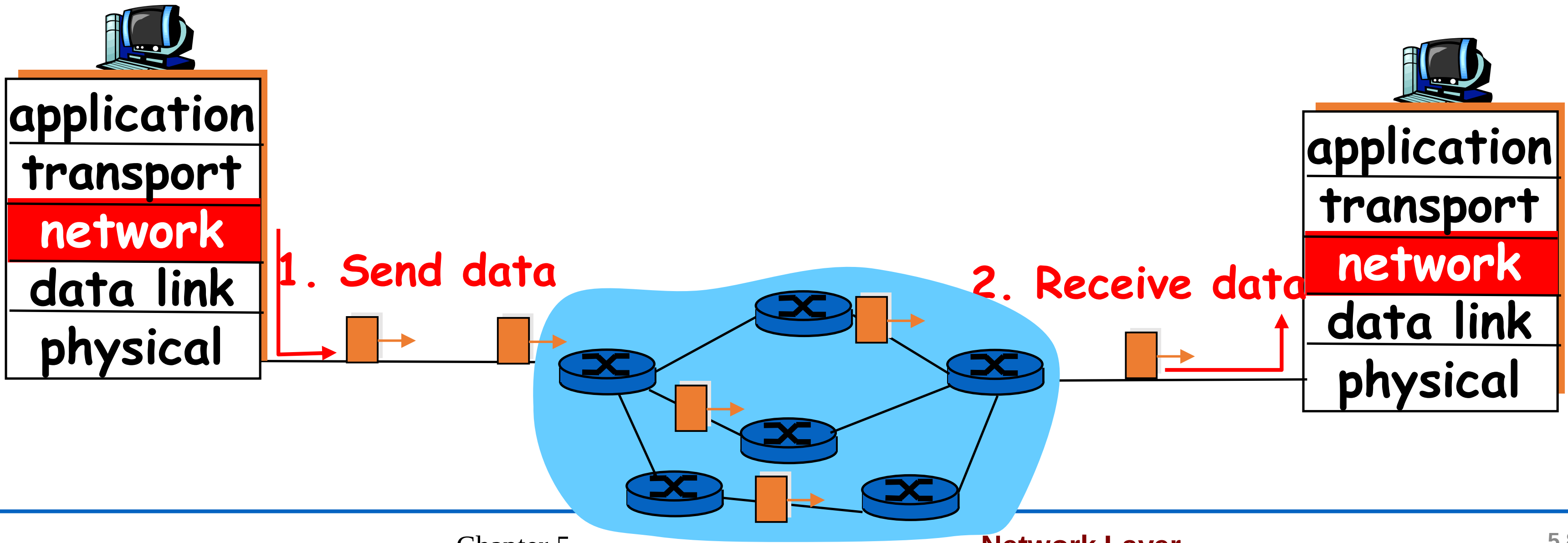
- used to setup, maintain teardown VC
- used in ATM, frame-relay, X.25



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

- 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

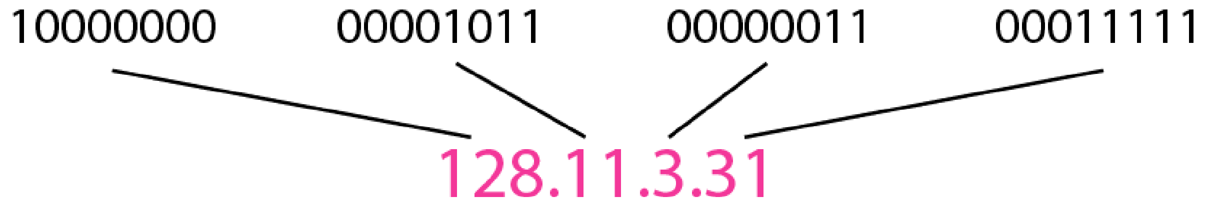
Chapter 5: Network Layer

IP Adressing

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

- 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 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.

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).

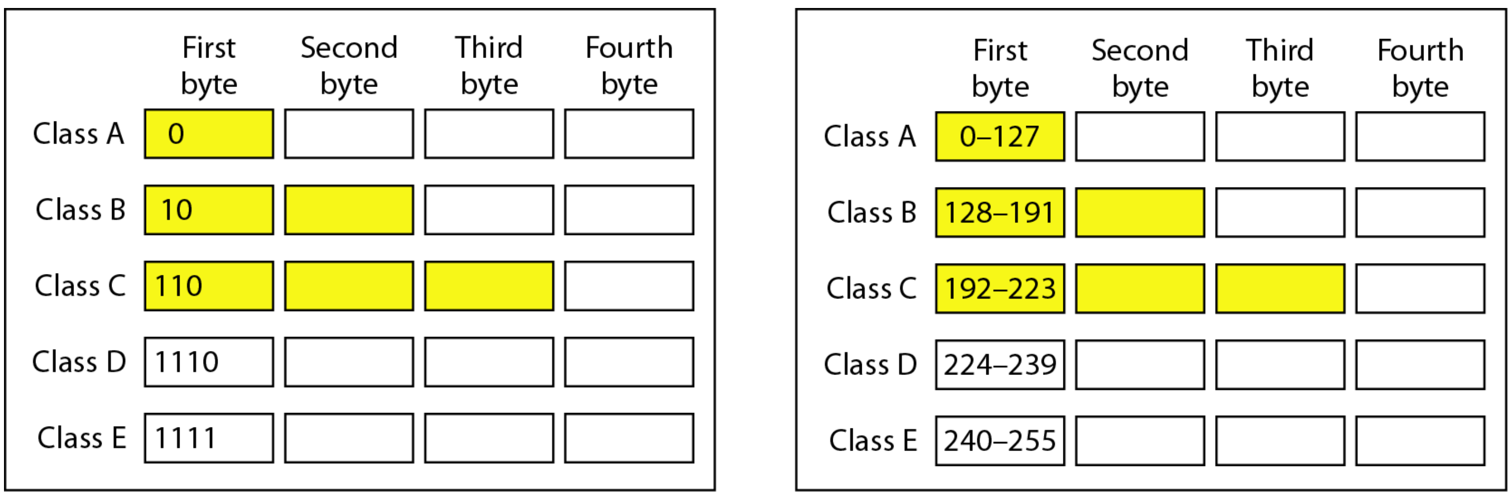
b. There can be no more than four numbers.

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.



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 C:
  - Class D:

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

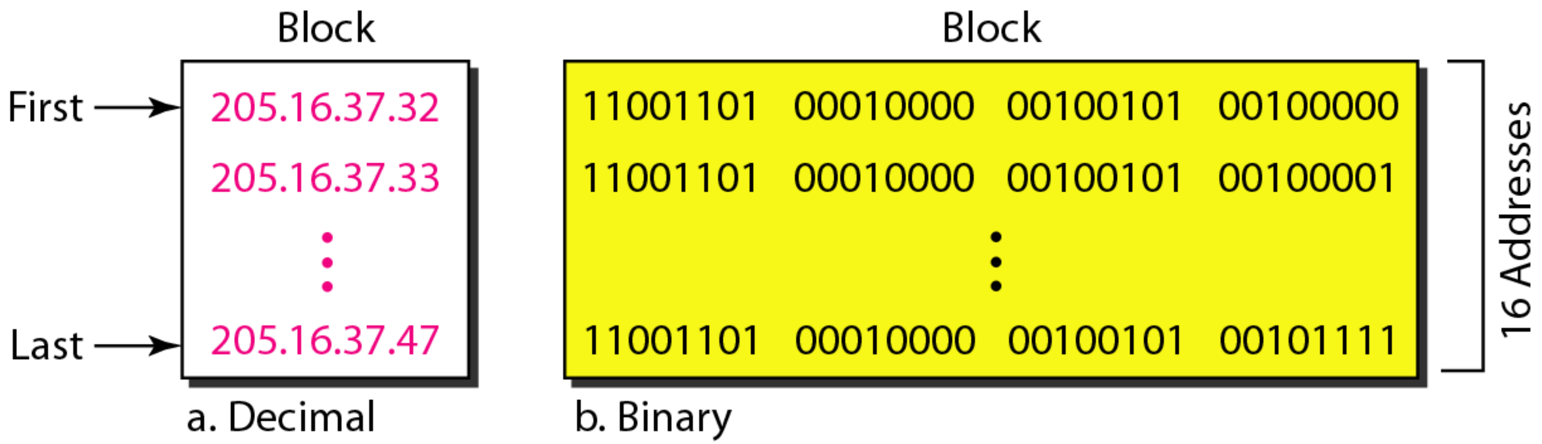
- Subnetting
  - Divide a large address block into smaller sub-groups.
  - Use of flexible net mask.
- Supernetting
  - Exhausted class A and B address space
  - Huge demand for class B address space
  - To combine several contiguous address spaces into a larger single address space

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}.

A block of 16 addresses granted to a small organization

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.

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 the previous slide.

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.

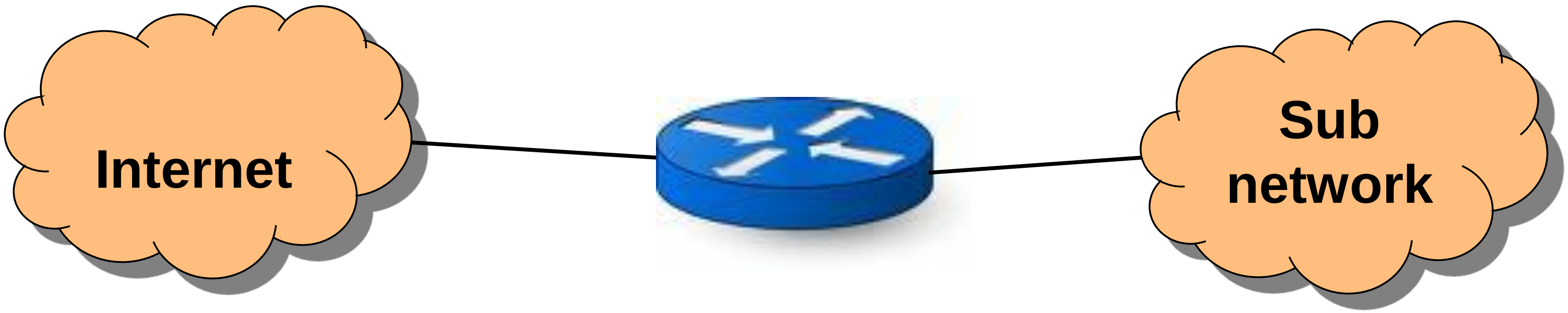
Mask complement:	**000000000 00000000 00000000 00001111**
Number of addresses:	$15 + 1 = 16$

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

Routing in IPv4

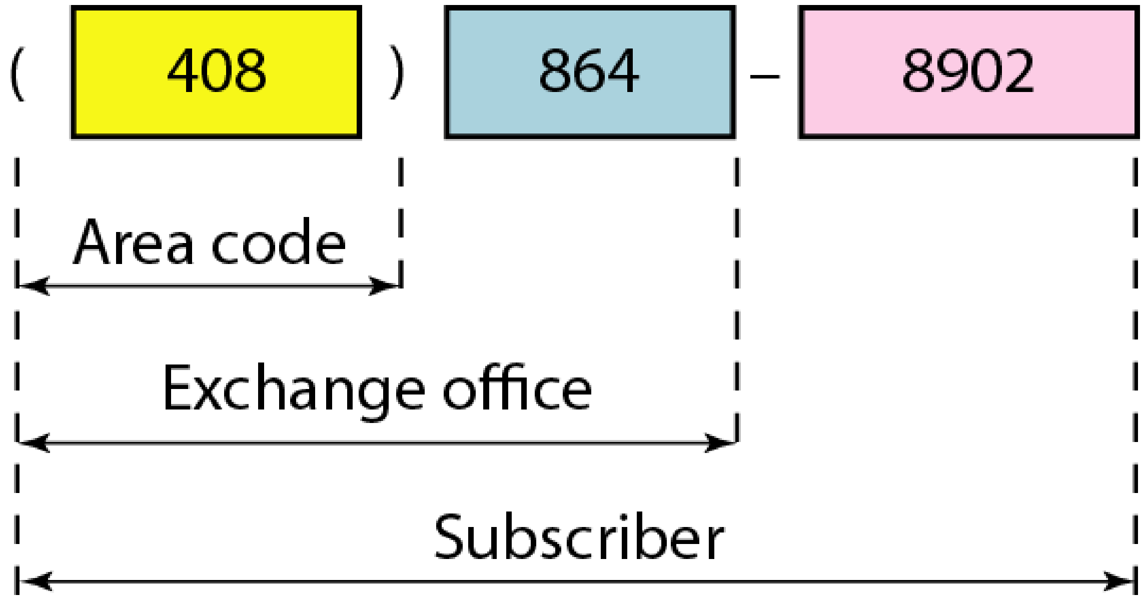
- A router has two addresses
  - An address through which the device inside of the router can be accessed.
  - Another address belongs to the granted block (sub-network).



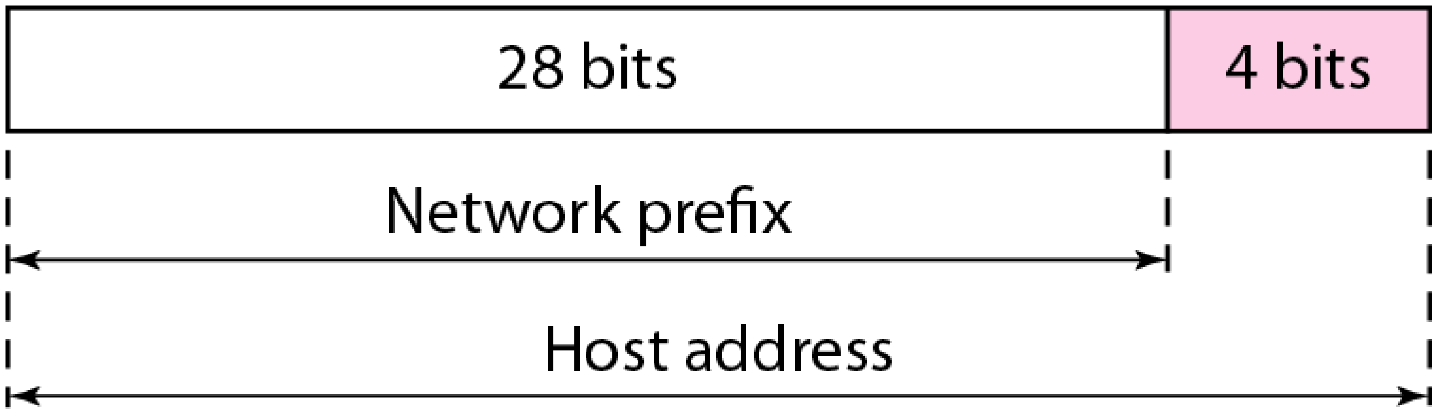
Hierarchy of IPv4 Addressing

- 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?

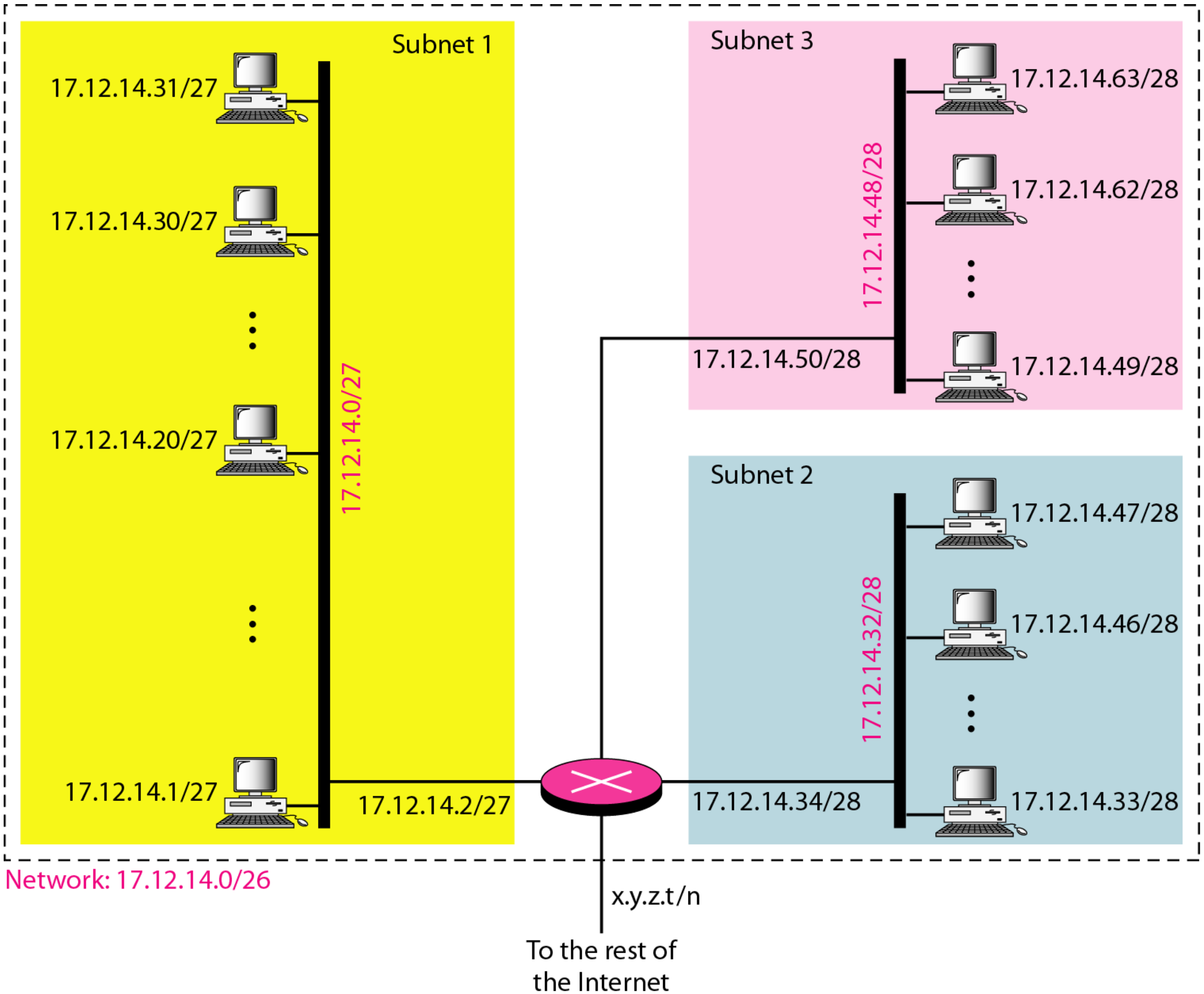
Figure 5 *Two levels of hierarchy in an IPv4 address*



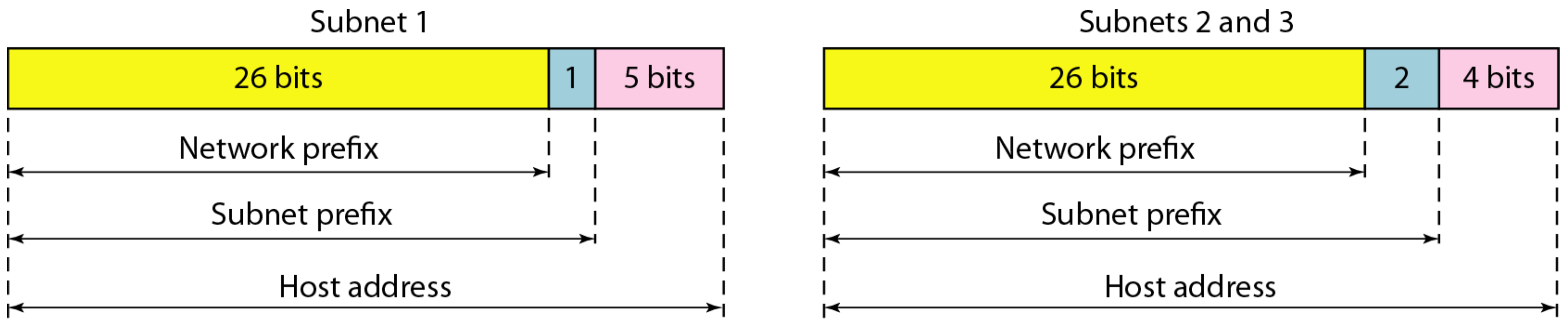
Two Level of Hierarchy



Three Level of Hierarchy



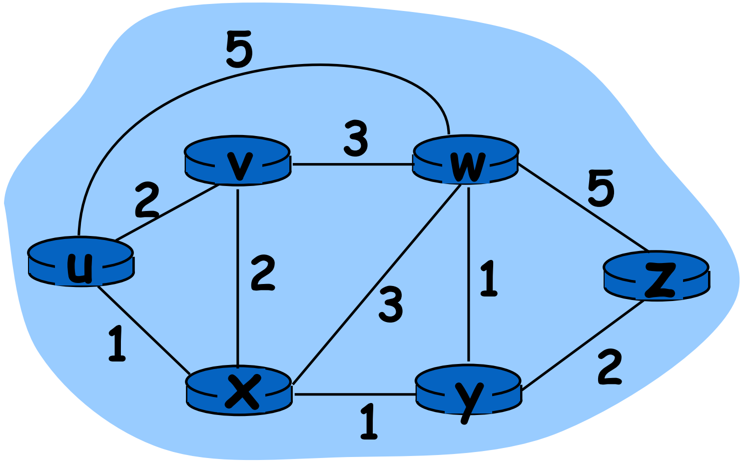
Three Level of Hierarchy



Chapter 5: Network Layer

Routing Algorithms

Graph abstraction



Graph: G = (N,E)

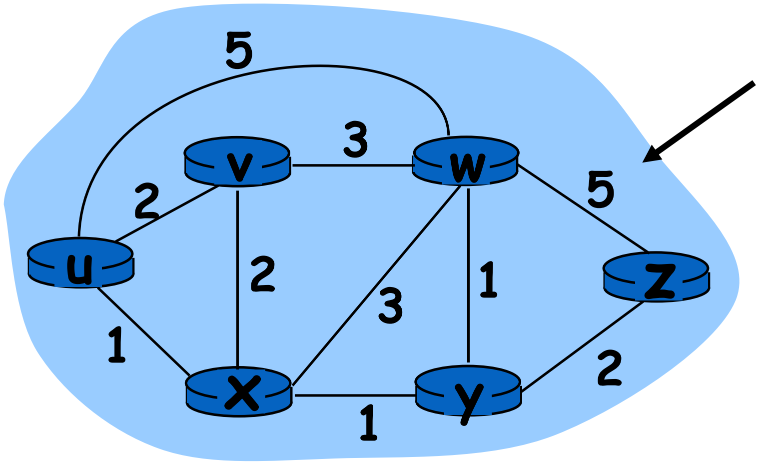
N = set of routers = { u, v, w, x, y, z }

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

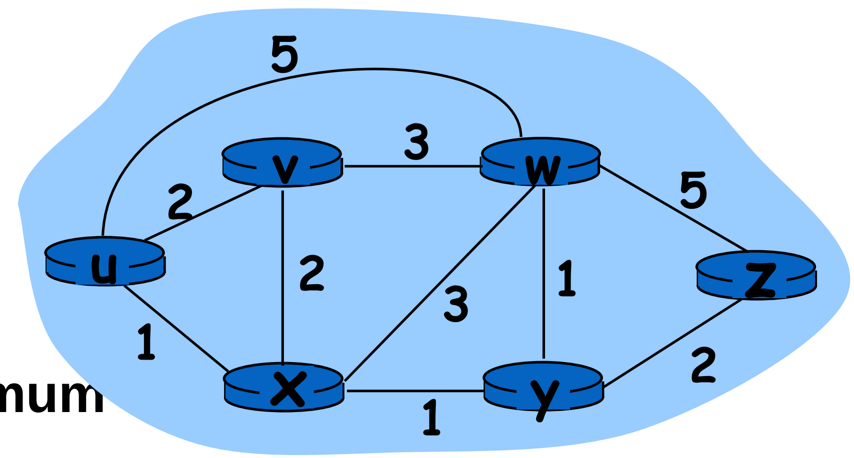
Dijkstra's Algorithm

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

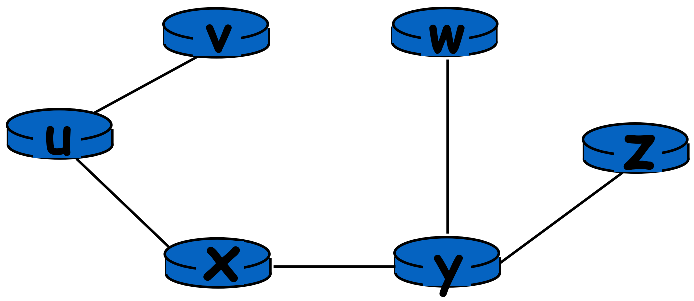
```
Initialization:
  N' = {u}
  for all nodes v
    if v adjacent to u
        then D(v) = c(u,v)
    else D(v) = ∞

Loop
  find w not in N' s.t. D(w) is a minimum
   add w to N'
   update D(v) for all v adjacent to w and not in N' :
      D(v) = min( D(v), D(w) + c(w,v) )
   /* new cost to v is either old cost to v or known
    shortest path cost to w plus cost from w to v */
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)

Generic Link State Routing

- Each node monitors neighbors/local links and advertises them to the network
  - Usually state of local links is sent periodically
  - Must be re-sent because of non-reliable delivery and possible joins/merges
- Each node maintains the full graph by collecting the updates from all other nodes
  - The set of all links forms the complete graph
  - Routing is performed using shortest path computations on the graph

Hello Protocol Description

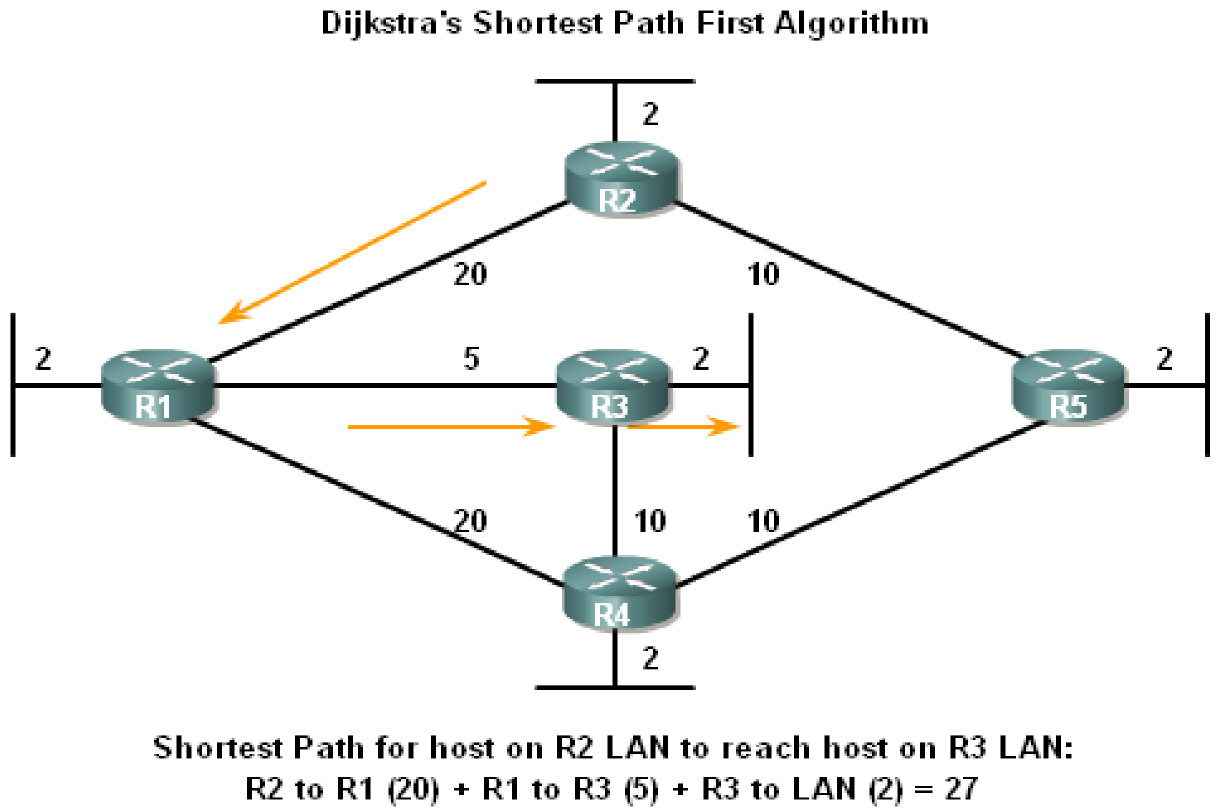
- Used for neighbor discovery
- Basic version
  - Each node sends a hello/beacon message periodically containing its id
  - Neighbors are discovered by hearing a hello from a previously un-heard from node
  - Neighbors are maintained by continuing to hear their periodic hello messages
  - Neighbors are considered lost if a number of their hello messages are no longer heard (typically two)

Hello Protocol Properties

- Very simple protocol agnostic method of discovering neighbors
- Generates constant overhead
  - Basic hello packets are very small
  - N/T packets per second (N=nodes, T=period)
  - Not scalable in very dense networks where nodes have a large number of neighbors
- Link failure only discovered after >2T
- May discover asymmetric links

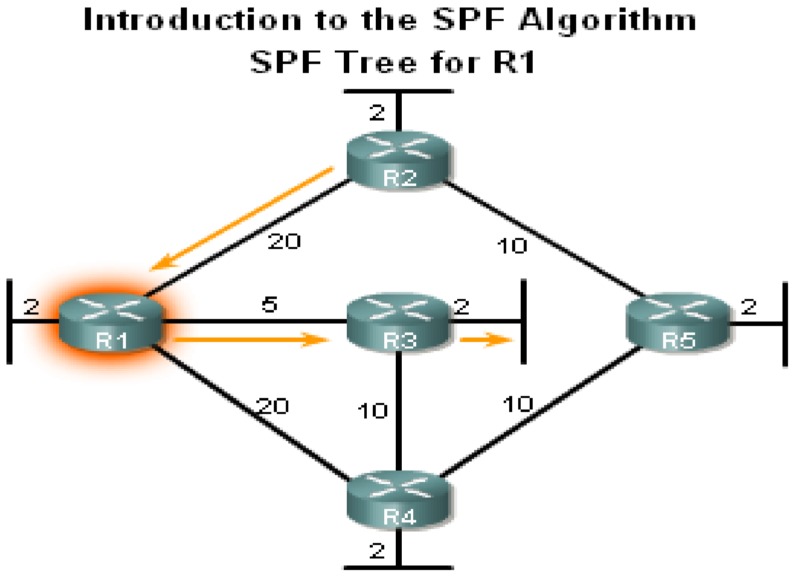
Link-State Routing

- Dikjstra’s algorithm also known as the shortest path first (SPF) algorithm



Link-State Routing

- The shortest path to a destination is not necessarily the path with the least number of hops



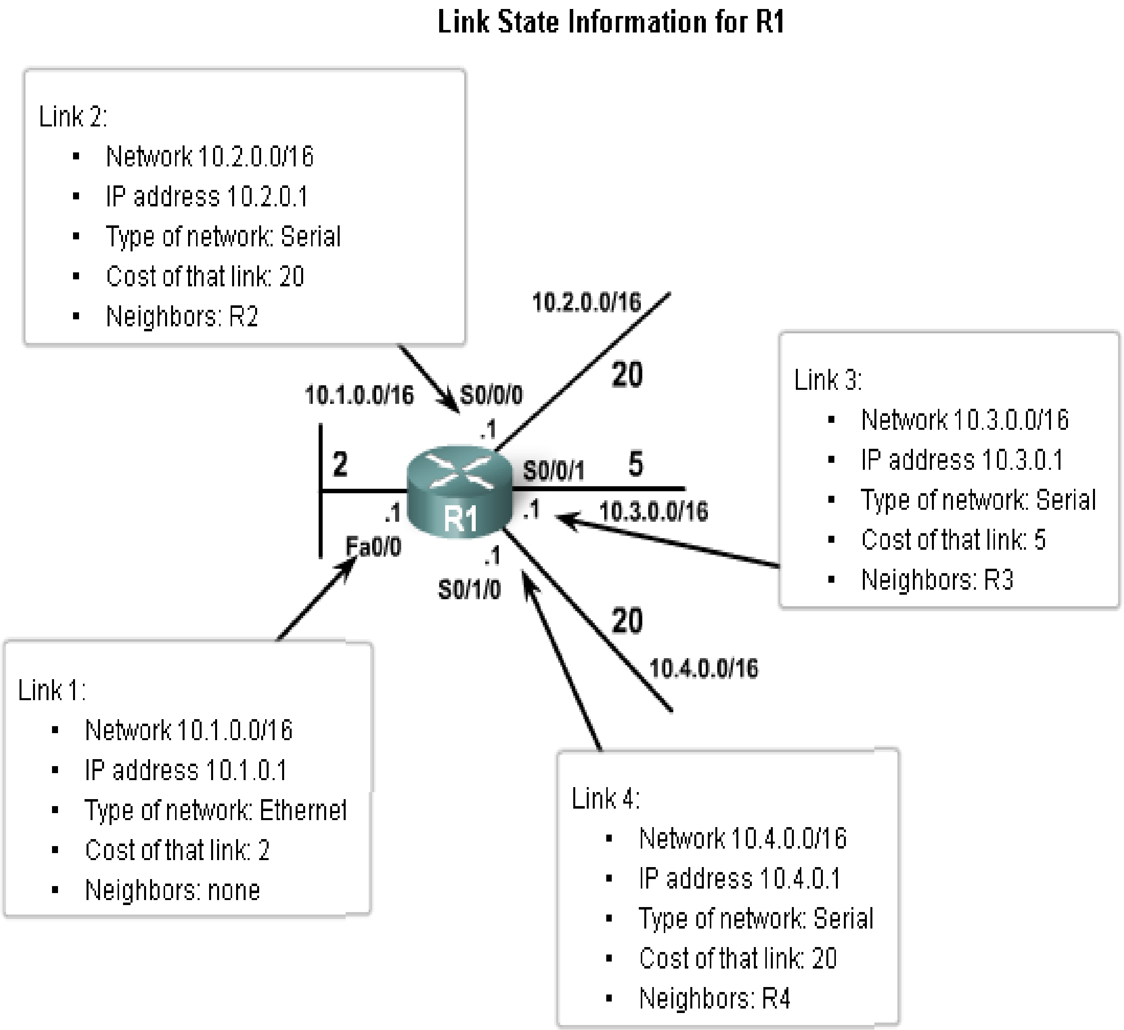
Destination	Shortest Path	Cost
R2 LAN	R1 to R2	22
R3 LAN	R1 to R3	7
R4 LAN	R1 to R3 to R4	17
R5 LAN	R1 to R3 to R4 to R5	27

Link-State Routing

- Link-State Routing Process
  - How routers using Link State Routing Protocols reach **convergence**
    - Each routers learns about its own directly connected networks
    - Link state routers exchange hello packet to "meet" other directly
    - Connected link state routers
    - Each router builds its own Link State Packet (LSP) which includes information about neighbors such as neighbor ID, link type, & bandwidth
    - After the LSP is created the router floods it to all neighbors who then store the information and then forward it until all routers have the same information
    - Once all the routers have received all the LSPs, the routers then construct a topological map of the network which is used to determine the best routes to a destination

Link-State Routing

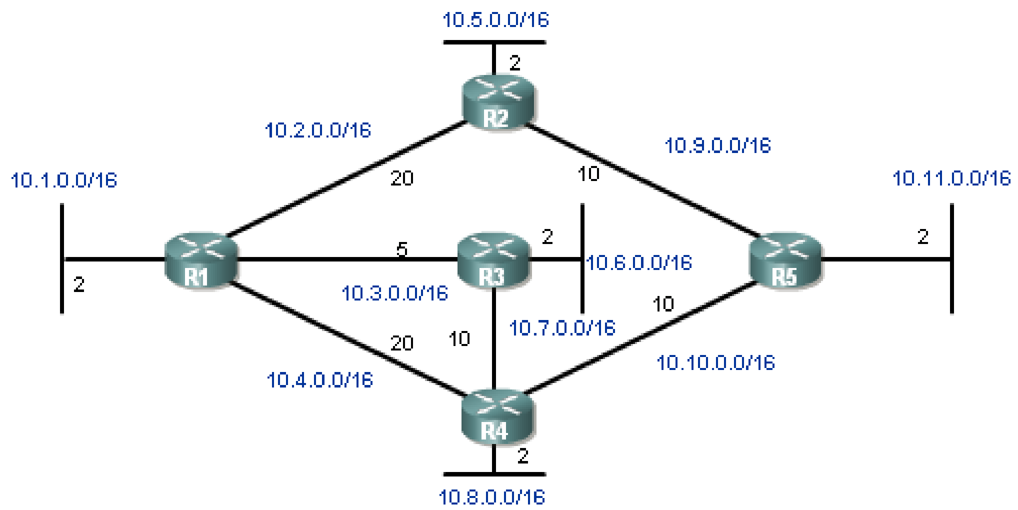
- Directly Connected Networks
- Link
  - This is an interface on a router
- Link state
  - This is the information about the state of the links



Link-State Routing

- Sending Hello Packets to Neighbors
  - Link state routing protocols use a hello protocol
  - Purpose of a hello protocol:
    - To discover neighbors (that use the same link state routing protocol) on its link

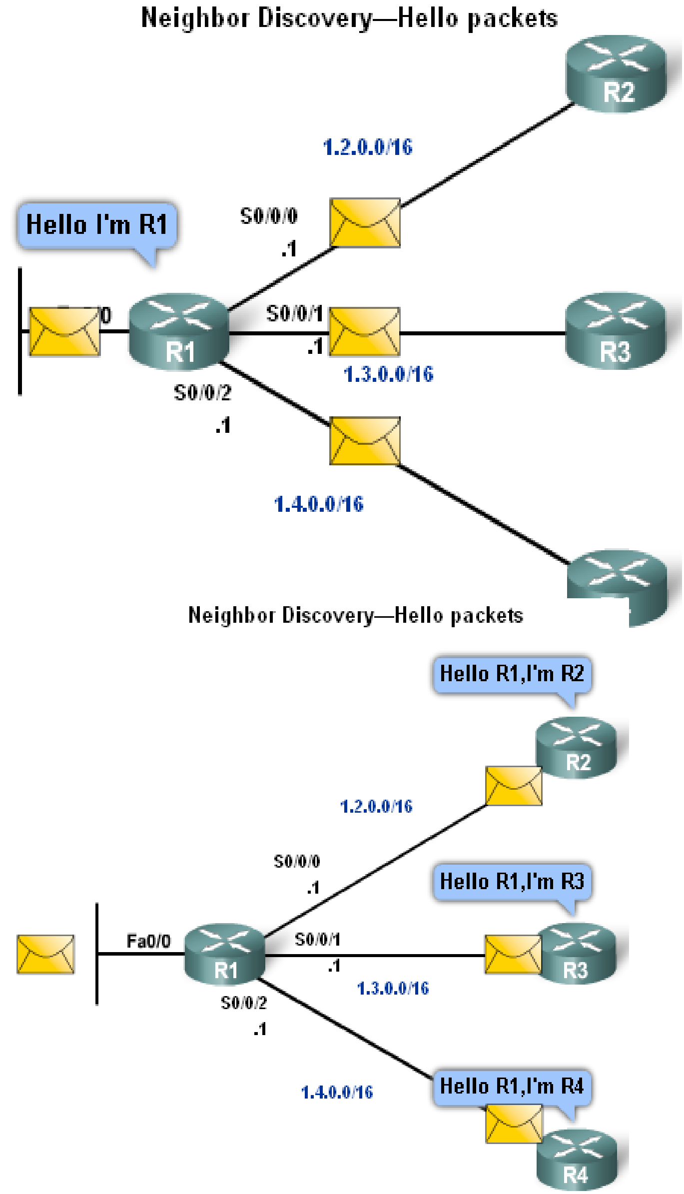
Link-State Routing Process



1. Each router learns about each of its own directly connected networks.
2. Each router is responsible for "saying hello" to its neighbors on directly connected networks.

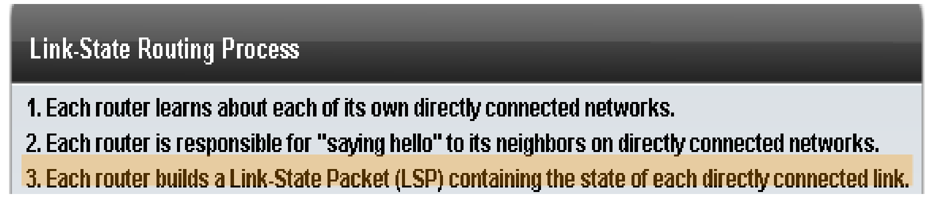
Link-State Routing

- Sending Hello Packets to Neighbors
  - Connected interfaces that are using the same link state routing protocols will exchange hello packets
  - Once routers learn it has neighbors they form an adjacency
    - 2 adjacent neighbors will exchange hello packets
    - These packets will serve as a keep alive function

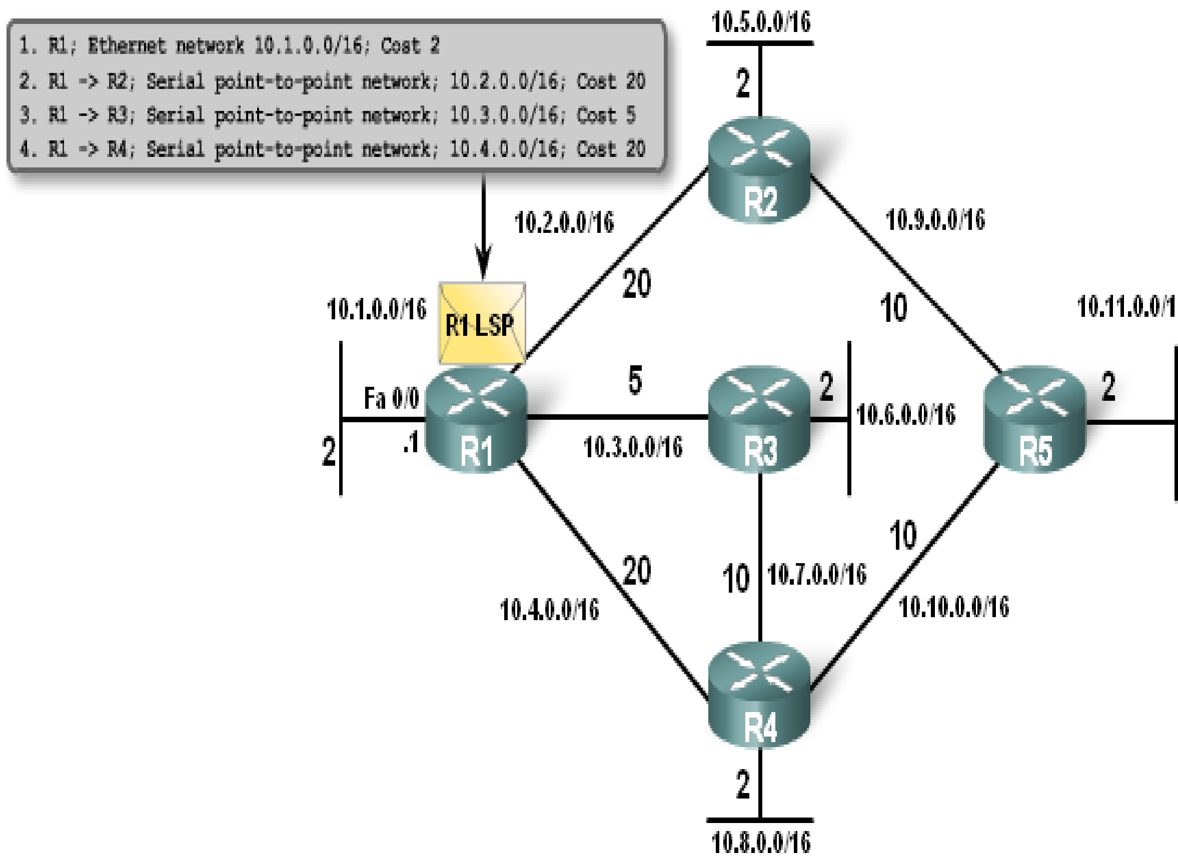


Link-State Routing

- Building the Link State Packet
  - Each router builds its own Link State Packet (LSP)
  - Contents of LSP:
    - State of each directly connected link
    - Includes information about neighbors such as neighbor ID, link type, & bandwidth

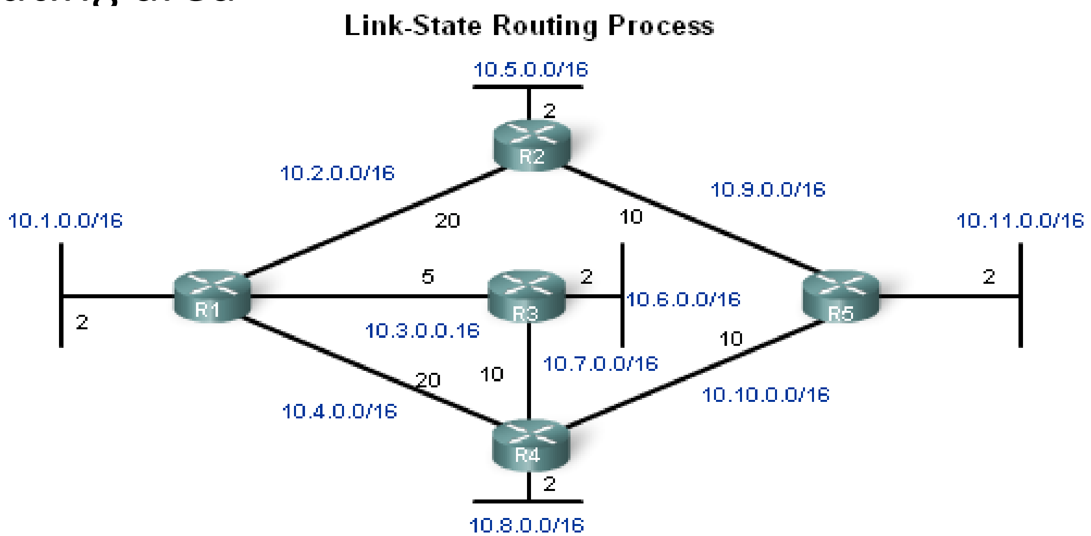


Link-State Routing Process



Link-State Routing

- Flooding LSPs to Neighbors
  - Once LSP are created they are forwarded out to neighbors
  - After receiving the LSP the neighbor continues to forward it throughout routing area



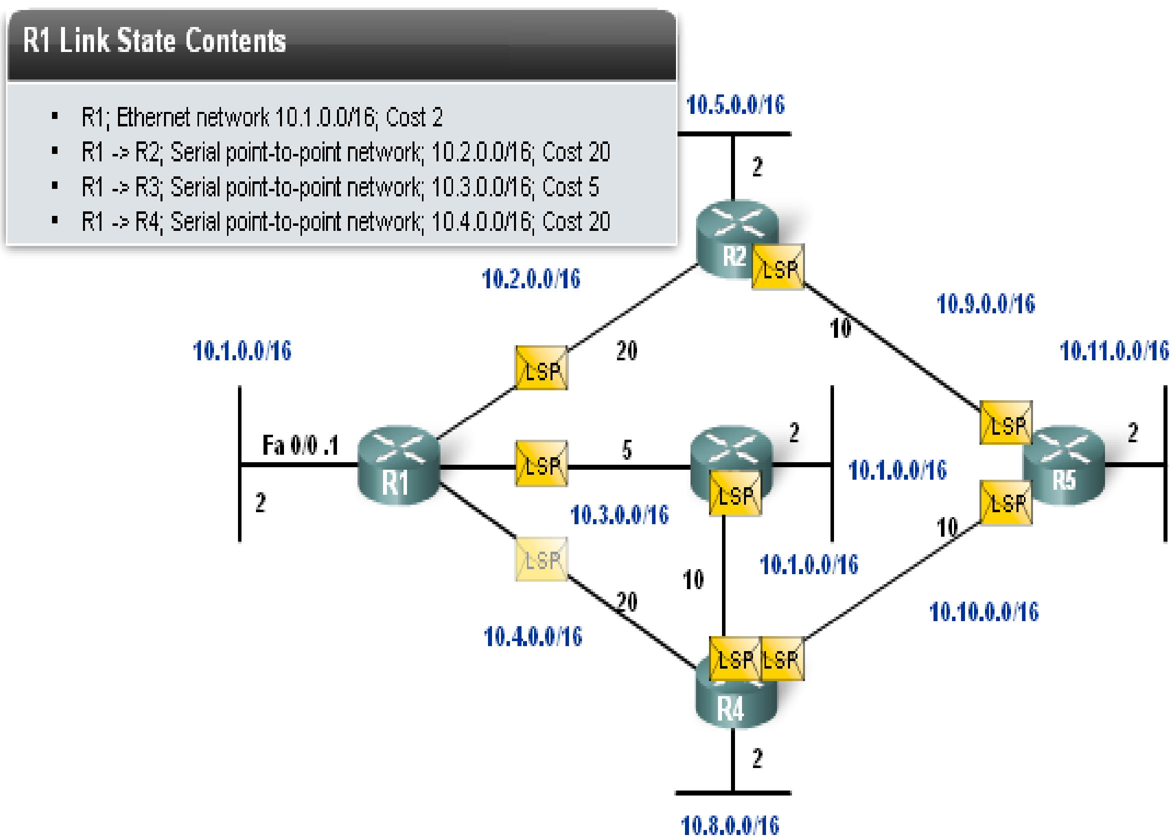
Link-State Routing Process

1. Each router learns about each of its own directly connected networks.
2. Each router is responsible for "saying hello" to its neighbors on directly connected networks.
3. Each router builds a Link-State Packet (LSP) containing the state of each directly connected link.
4. Each router floods the LSP to all neighbors, who then store all LSPs received in a database.

Link-State Routing

- LSPs are sent out under the following conditions:
  - Initial router start up or routing process
  - When there is a change in topology

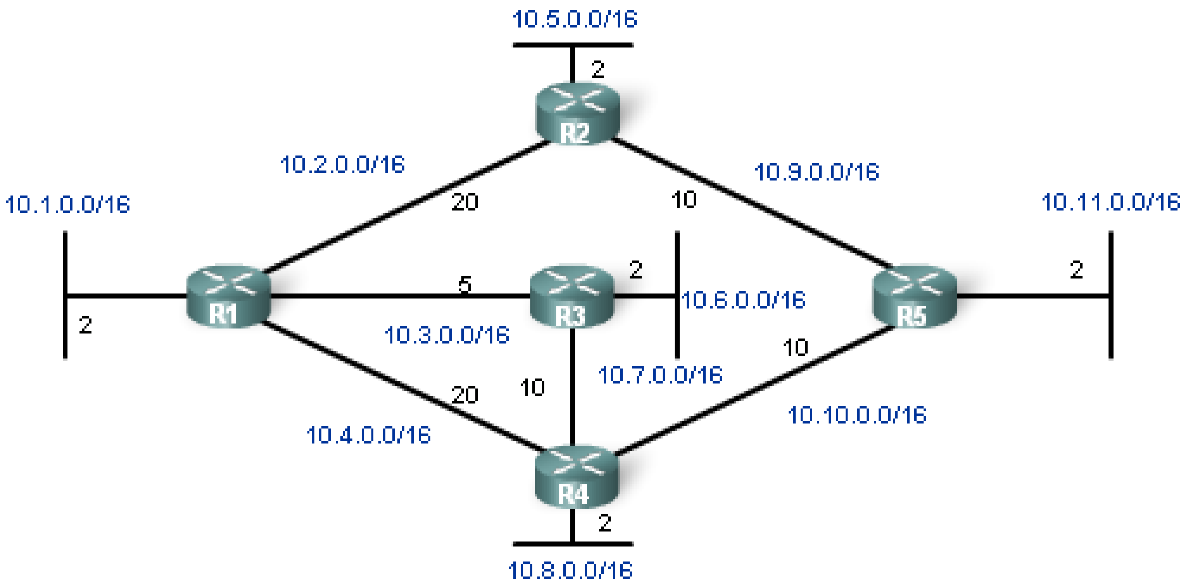
Flooding of the R1 LSP



Link-State Routing

- Constructing a link state data base
  - Routers use a database to construct a topology map of the network

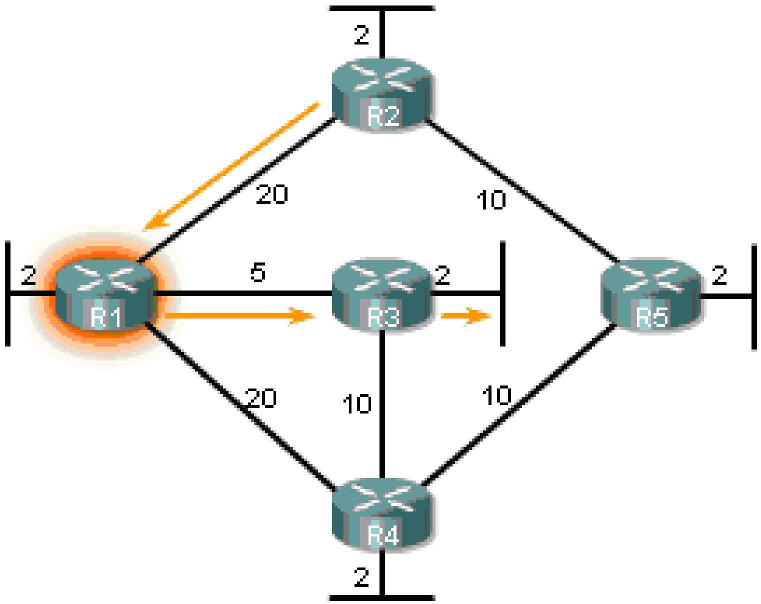
Link-State Routing Process



Link-State Routing Process

1. Each router learns about each of its own directly connected networks.
2. Each router is responsible for "saying hello" to its neighbors on directly connected networks.
3. Each router builds a Link-State Packet (LSP) containing the state of each directly connected link.
4. Each router floods the LSP to all neighbors, who then store all LSPs received in a database.
5. Each router uses the database to construct a complete map of the topology and computes the best path to each destination network.

Link-State Routing



Destination	Shortest Path	Cost
R2 LAN	R1 to R2	22
R3 LAN	R1 to R3	7
R4 LAN	R1 to R3 to R4	17
R5 LAN	R1 to R3 to R4 to R5	27

R1 Link-State Database

R1s Link-State DatabaseLSPs from R2:

- Connected to neighbor R1 on network 10.2.0.0/16, cost of 20
- Connected to neighbor R5 on network 10.9.0.0/16, cost of 10
- Has a network 10.5.0.0/16, cost of 2

LSPs from R3:

- Connected to neighbor R1 on network 10.3.0.0/16, cost of 5
- Connected to neighbor R4 on network 10.7.0.0/16, cost of 10
- Has a network 10.6.0.0/16, cost of 2

LSPs from R4:

- Connected to neighbor R1 on network 10.4.0.0/16, cost of 20
- Connected to neighbor R3 on network 10.7.0.0/16, cost of 10
- Connected to neighbor R5 on network 10.10.0.0/16, cost of 10
- Has a network 10.8.0.0/16, cost of 2

LSPs from R5:

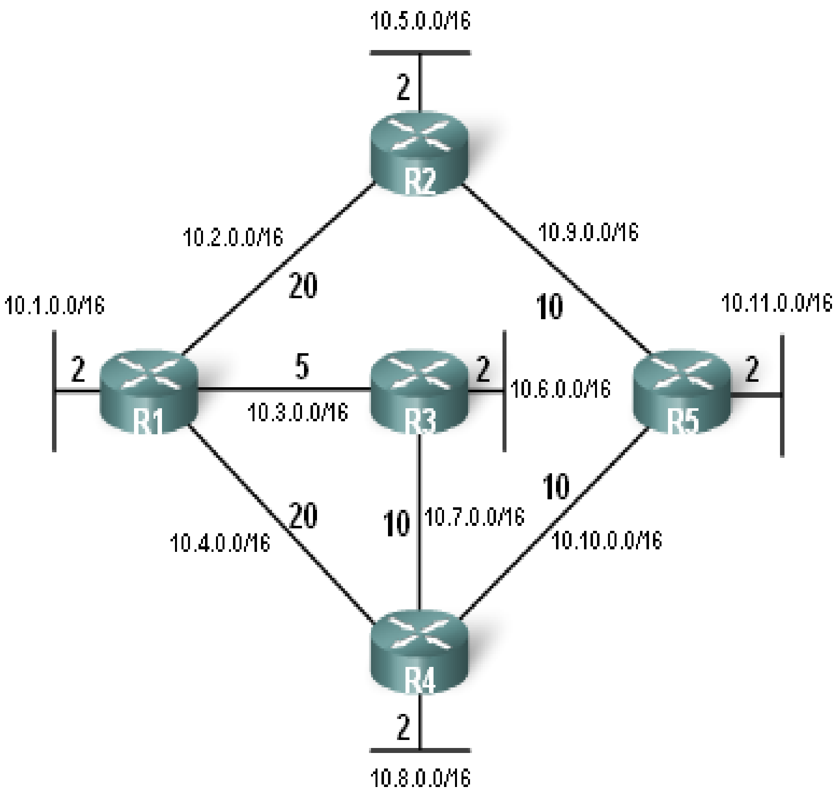
- Connected to neighbor R2 on network 10.9.0.0/16, cost of 10
- Connected to neighbor R4 on network 10.10.0.0/16, cost of 10
- Has a network 10.11.0.0/16, cost of 2

R1 Link-states:

- Connected to neighbor R2 on network 10.2.0.0/16, cost of 20
- Connected to neighbor R3 on network 10.3.0.0/16, cost of 5
- Connected to neighbor R4 on network 10.4.0.0/16, cost of 20

Link-State Routing

- Shortest Path First (SPF) Tree
  - Building a **portion** of the SPF tree



R1s Link State Database

R1 Links-states:

- Connected to neighbor R2 on network 10.2.0.0/16, cost of 20
- Connected to neighbor R3 on network 10.3.0.0/16, cost of 5
- Connected to neighbor R4 on network 10.4.0.0/16, cost of 20
- Has a network 10.1.0.0/16, cost of 2

LSPs from R2:

- Connected to neighbor R1 on network 10.2.0.0/16, cost of 20
- Connected to neighbor R5 on network 10.9.0.0/16, cost of 10
- Has a network 10.5.0.0/16, cost of 2

LSPs from R3:

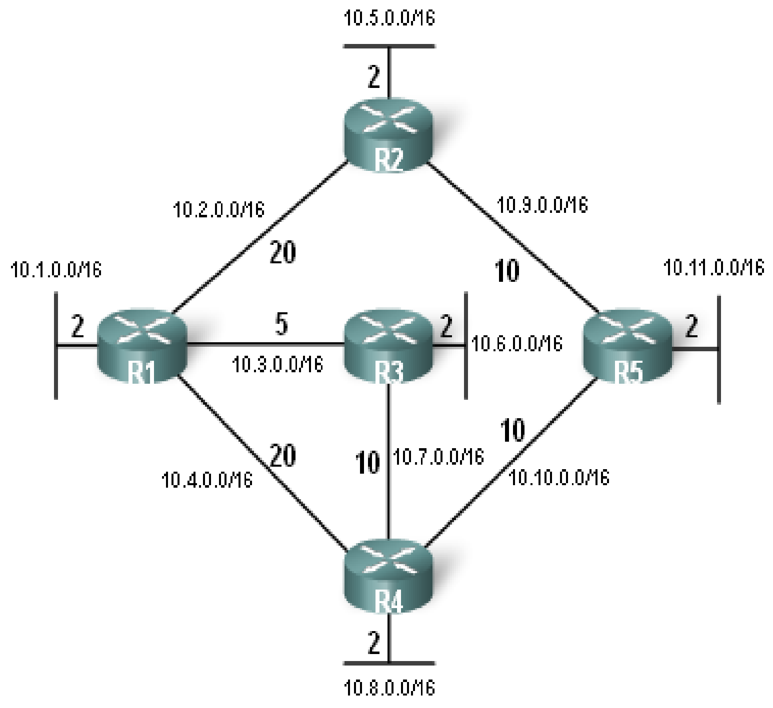
- Connected to neighbor R1 on network 10.3.0.0/16, cost of 5
- Connected to neighbor R4 on network 10.7.0.0/16, cost of 10
- Has a network 10.6.0.0/16, cost of 2
- LSPs from R4:
- Connected to neighbor R1 on network 10.4.0.0/16, cost of 20
- Connected to neighbor R3 on network 10.7.0.0/16, cost of 10
- Connected to neighbor R5 on network 10.10.0.0/16, cost of 10
- Has a network 10.8.0.0/16, cost of 2

LSPs from R5:

- Connected to neighbor R2 on network 10.9.0.0/16, cost of 10
- Connected to neighbor R4 on network 10.10.0.0/16, cost of 10
- Has a network 10.11.0.0/16, cost of 2

Link-State Routing

- Building a **portion** of the SPF tree
  - R1 uses 2nd LSP
    - Reason: R1 can create a link from R2 to R5 - this information is added to R1's SPF tree



R1s Link State Database

R1 Links-states:
- Connected to neighbor R2 on network 10.2.0.0/16, cost of 20
- Connected to neighbor R3 on network 10.3.0.0/16, cost of 5
- Connected to neighbor R4 on network 10.4.0.0/16, cost of 20
- Has a network 10.1.0.0/16, cost of 2

LSPs from R2:
- Connected to neighbor R1 on network 10.2.0.0/16, cost of 20
- Connected to neighbor R5 on network 10.9.0.0/16, cost of 10
- Has a network 10.5.0.0/16, cost of 2

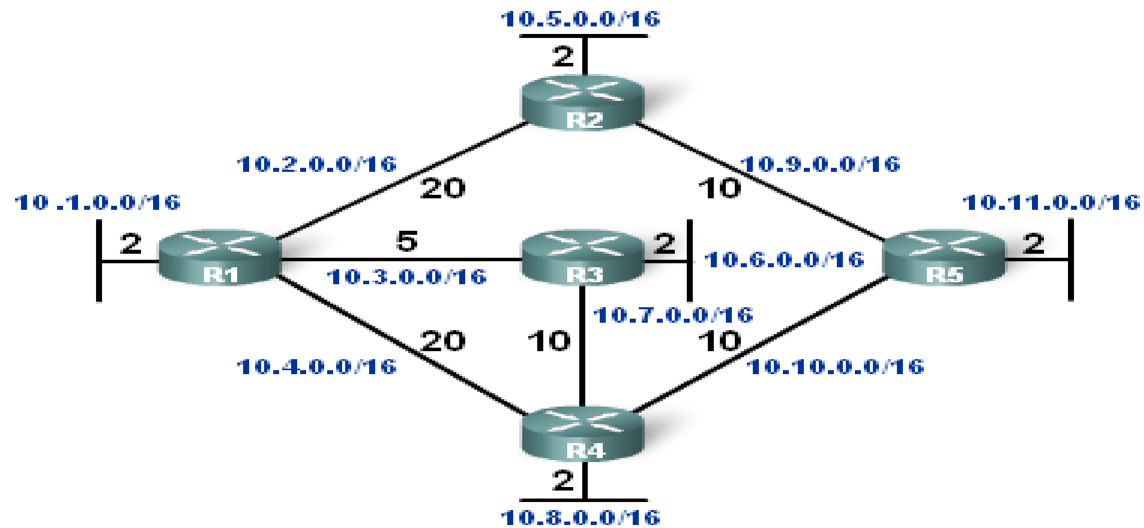
LSPs from R3:
- Connected to neighbor R1 on network 10.3.0.0/16, cost of 5
- Connected to neighbor R4 on network 10.7.0.0/16, cost of 10
- Has a network 10.6.0.0/16, cost of 2
- LSPs from R4:
- Connected to neighbor R1 on network 10.4.0.0/16, cost of 20
- Connected to neighbor R3 on network 10.7.0.0/16, cost of 10
- Connected to neighbor R5 on network 10.10.0.0/16, cost of 10
- Has a network 10.8.0.0/16, cost of 2

LSPs from R5:
- Connected to neighbor R2 on network 10.9.0.0/16, cost of 10
- Connected to neighbor R4 on network 10.10.0.0/16, cost of 10
- Has a network 10.11.0.0/16, cost of 2

Link-State Routing

- Determining the shortest path
  - The shortest path to a destination determined by adding the costs & finding the lowest cost

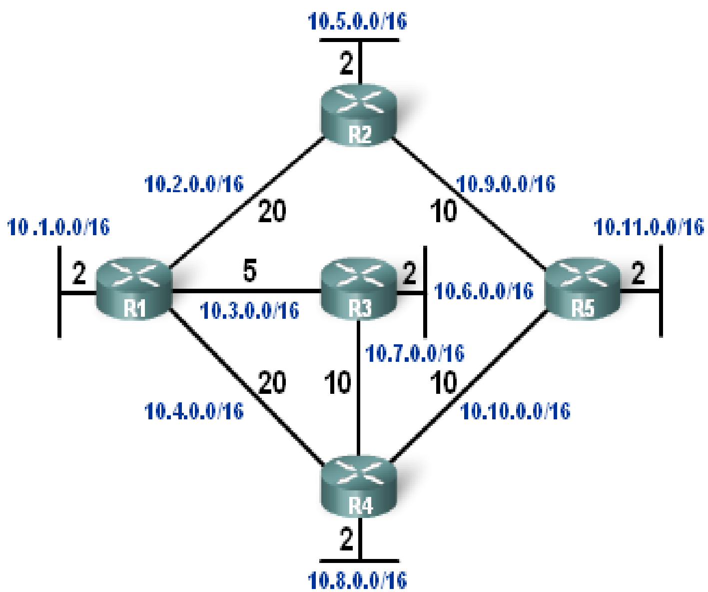
SPF Tree for R1



Destination	Shortest Path	Cost
R2 LAN	R1 to R2	22
R3 LAN	R1 to R3	7
R4 LAN	R1 to R3 to R4	17
R5 LAN	R1 to R3 to R4 to R5	27

Link-State Routing

- Once the SPF algorithm has determined the shortest path routes, these routes are placed in the routing table



R1 Routing Table

SPF Information

- Network 10.5.0.0/16 via R2 serial 0/0/0 at a cost of 22
- Network 10.6.0.0/16 via R3 serial 0/0/1 at a cost of 7
- Network 10.7.0.0/16 via R3 serial 0/0/1 at a cost of 15
- Network 10.8.0.0/16 via R3 serial 0/0/1 at a cost of 17
- Network 10.9.0.0/16 via R2 serial 0/0/0 at a cost of 30
- Network 10.10.0.0/16 via R3 serial 0/0/1 at a cost of 25
- Network 10.11.0.0/16 via R3 serial 0/0/1 at a cost of 27

R1 Routing Table

Directly Connected Networks

- 10.1.0.0/16 Directly Connected Network
- 10.2.0.0/16 Directly Connected Network
- 10.3.0.0/16 Directly Connected Network
- 10.4.0.0/16 Directly Connected Network

Remote Networks

- 10.5.0.0/16 via R2 serial 0/0/0, cost = 22
- 10.6.0.0/16 via R3 serial 0/0/1, cost = 7
- 10.7.0.0/16 via R3 serial 0/0/1, cost = 15
- 10.8.0.0/16 via R3 serial 0/0/1, cost = 17
- 10.9.0.0/16 via R2 serial 0/0/0, cost = 30
- 10.10.0.0/16 via R3 serial 0/0/1, cost = 25
- 10.11.0.0/16 via R3 serial 0/0/1, cost = 27

Link-State Routing Protocols

- Requirements for using a link state routing protocol
  - Memory requirements
    - Typically link state routing protocols use more memory
  - Processing Requirements
    - More CPU processing is required of link state routing protocols
  - Bandwidth Requirements
    - Initial startup of link state routing protocols can consume lots of bandwidth

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)

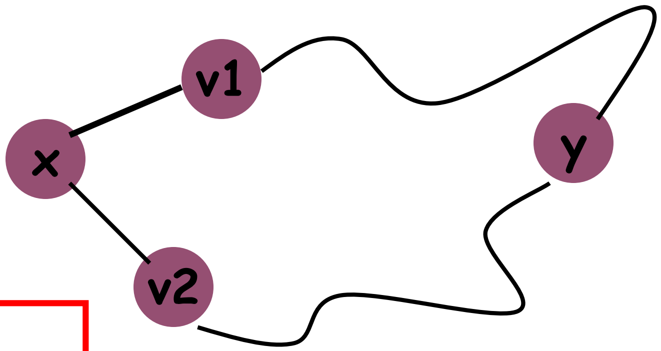
Define

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

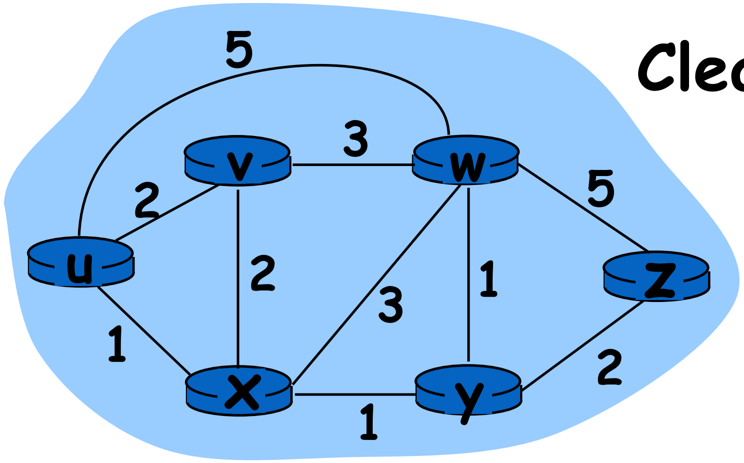
Then

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

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:

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

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

Distance Vector Algorithm

- $D_x(y)$ = estimate of least cost from x to y
- Distance vector: $\mathbf{D}_x = [D_x(y)\text{: } y \in N\ ]$
- Node x knows cost to each neighbor v: $c(x,v)$
- Node x maintains $\mathbf{D}_x = [D_x(y)\text{: } y \in N\ ]$
- Node x also maintains its neighbors' distance vectors
  - For each neighbor v, x maintains
    $\mathbf{D}_v = [D_v(y)\text{: } y \in N\ ]$

Distance vector algorithm

Basic idea:

- Each node periodically sends its own distance vector estimate to neighbors
- When a node x receives new DV estimate from neighbor, it updates its own DV using B-F equation:

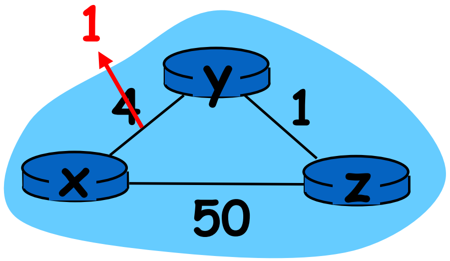
$$D_x(y) \leftarrow \min_v\{c(x,v) + D_v(y)\} \quad \text{for each node } y \in N$$

- **Under minor, natural conditions, the estimate** ***$D_x(y)$ converge to the actual least cost*** **$d_x(y)$**

Distance Vector: link cost changes

Link cost changes:

- if DV changes, notify neighbors

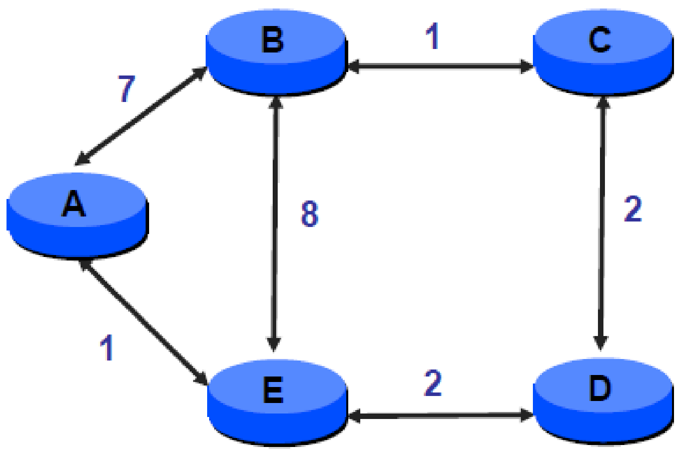


At time t_0, *y* detects the link-cost change, updates its DV, and informs its neighbors.

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.

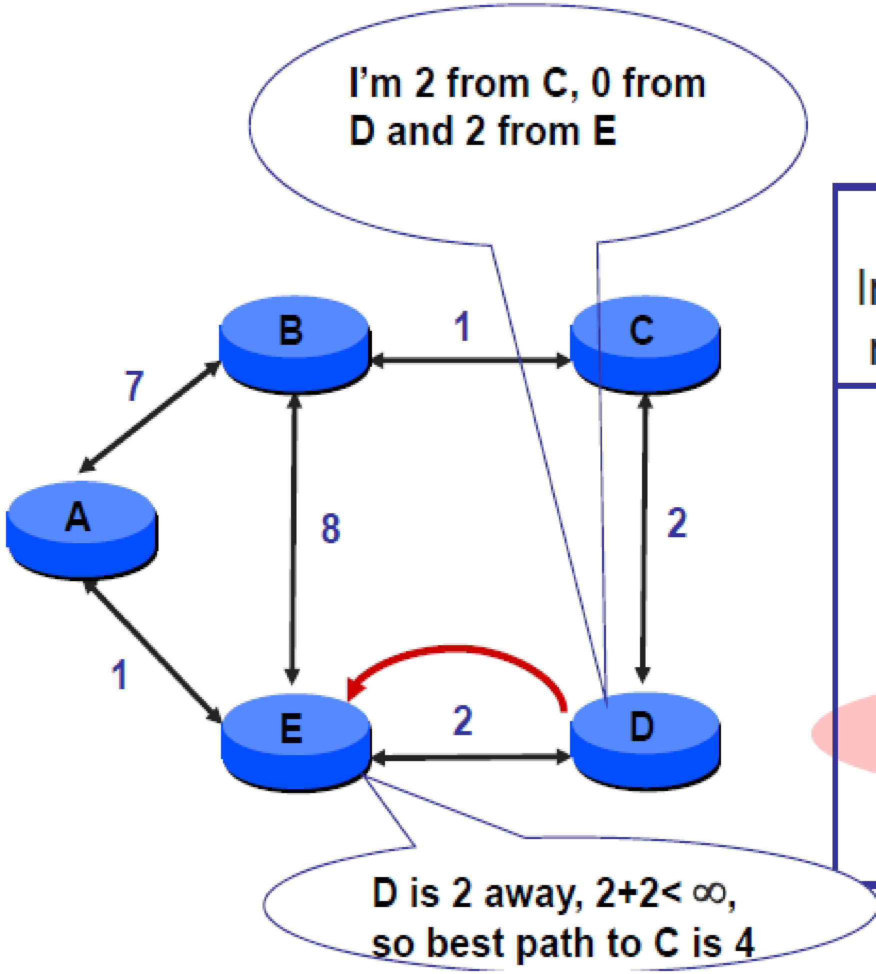
At time t_2, *y* receives *z*'s update and updates its distance table. *y*'s least costs do not change and hence *y* does *not* send any message to *z*.

Example: Initial State



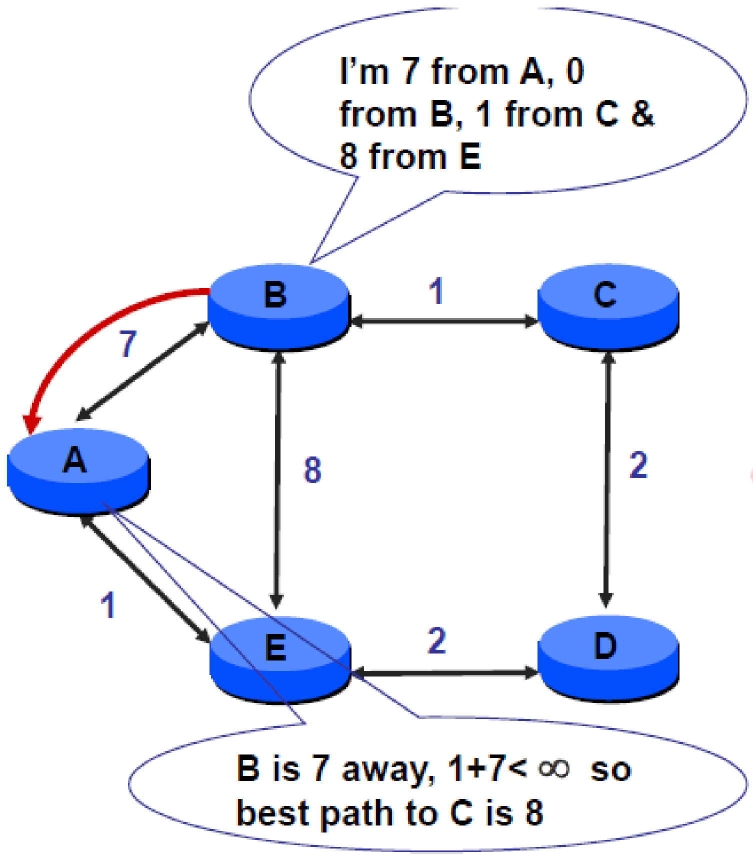
Info at node	Distance to Node				
	A	**B**	**C**	**D**	**E**
A	0	7	∞	∞	1
B	7	0	1	∞	8
C	∞	1	0	2	∞
D	∞	∞	2	0	2
E	1	8	∞	2	0

D sends vector to *E*



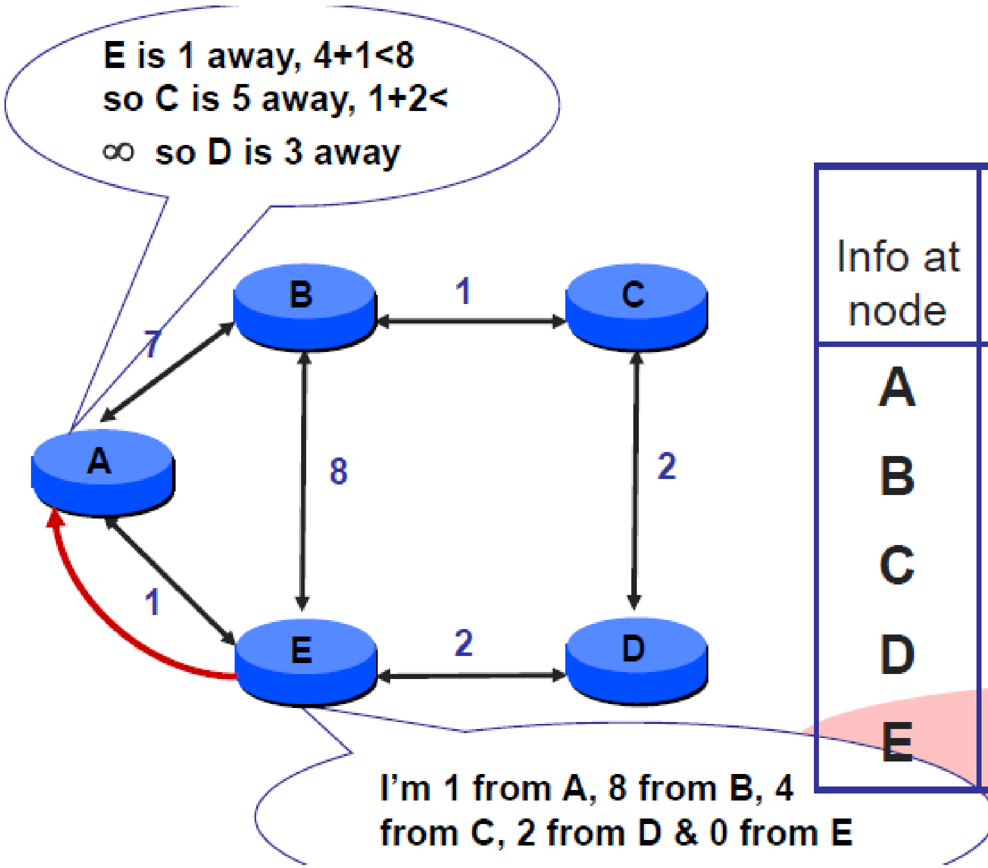
Info at node	Distance to Node				
	A	**B**	**C**	**D**	**E**
A	0	7	∞	∞	1
B	7	0	1	∞	8
C	∞	1	0	2	∞
D	∞	∞	2	0	2
E	1	8	**4**	2	0

B sends vector to *A*



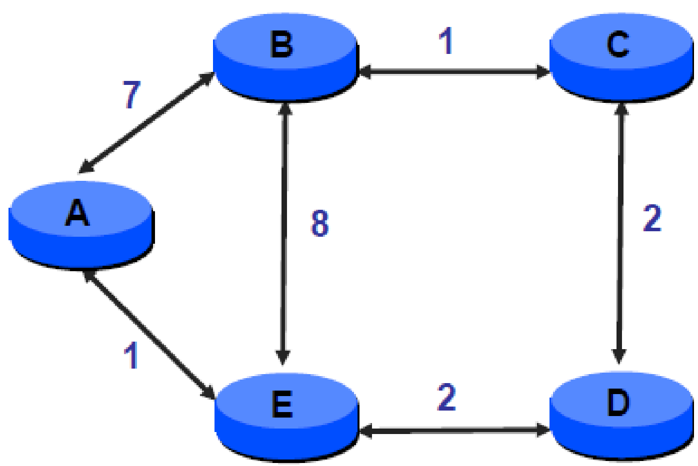
Info at node	Distance to Node				
	A	**B**	**C**	**D**	**E**
A	0	7	8	∞	1
B	7	0	1	∞	8
C	∞	1	0	2	∞
D	∞	∞	2	0	2
E	1	8	4	2	0

E sends vector to *A*



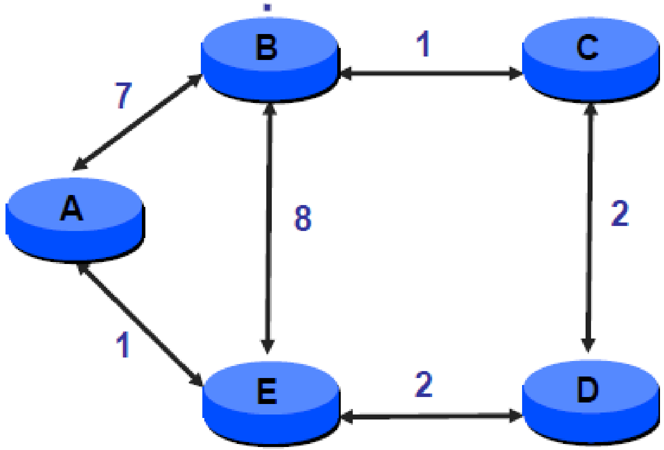
Info at node	Distance to Node				
	A	**B**	**C**	**D**	**E**
A	0	7	**5**	**3**	1
B	7	0	1	∞	8
C	∞	1	0	2	∞
D	∞	∞	2	0	2
E	1	8	4	2	0

…until Convergence



Info at node	Distance to Node				
	A	**B**	**C**	**D**	**E**
A	0	6	5	3	1
B	6	0	1	3	5
C	5	1	0	2	4
D	3	3	2	0	2
E	1	5	4	2	0

Node *B*’s distance vector



Dest	Next hop		
	A	**E**	**C**
A	7	9	**6**
C	12	12	**1**
D	10	10	**3**
E	8	8	**5**

Internet Routing

- The link state and DV routing protocols used in internet routing
  - RIP (routing information protocol)
  - OSPF (Open shortest path first)
  - BGP (Border gateway protocol)

Comparison of LS and DV algorithms

Message complexity

- LS: with n nodes, E links, O(nE) msgs sent
- DV: exchange between neighbors only
  - convergence time varies

Speed of Convergence

- LS: $O(n^2)$ algorithm requires O(nE) msgs
- DV: convergence time varies
  - may be routing loops
  - count-to-infinity problem

Robustness: what happens if router malfunctions?

LS:

- node can advertise incorrect *link* cost
- each node computes only its *own* table

DV:

- DV node can advertise incorrect *path* cost
- each node's table used by others
  - error propagate in the network

Hierarchical Routing

- **all routers identical**
- **network "flat"**

***… not* true in practice**

scale: with 200 million destinations:

- can't store all dest's in routing tables!
- routing table exchange would swamp links!

administrative autonomy

- internet = network of networks
- each network admin may want to control routing in its own network

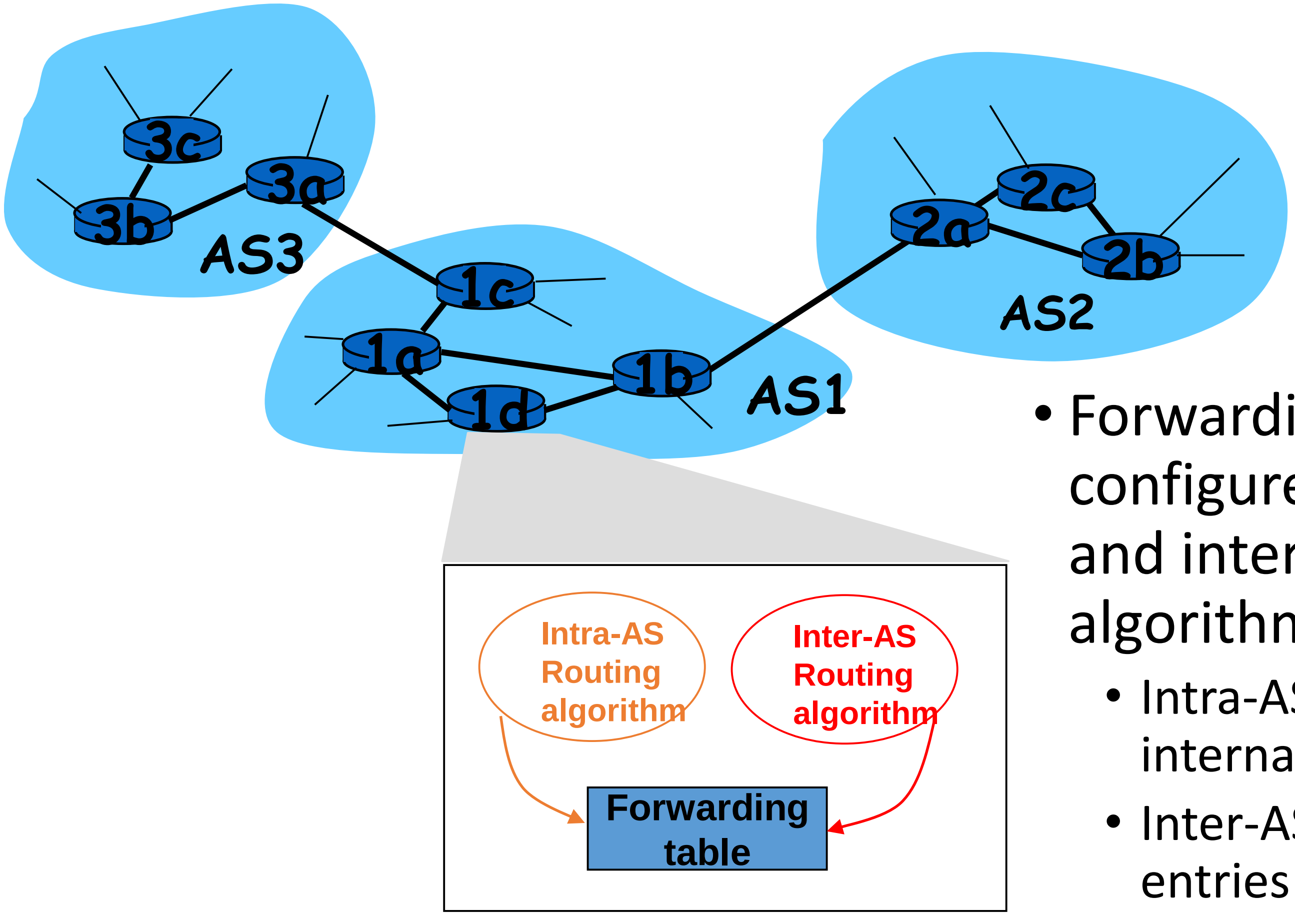
Hierarchical Routing

- aggregate routers into regions, "autonomous systems" (AS)
- routers in same AS run same routing protocol
  - "intra-AS" routing protocol
  - routers in different AS can run different intra-AS routing protocol

Gateway router

- Direct link to router in another AS

Interconnected ASes



- Forwarding table is configured by both intra- and inter-AS routing algorithm
  - Intra-AS sets entries for internal dests
  - Inter-AS & Intra-As sets entries for external dests

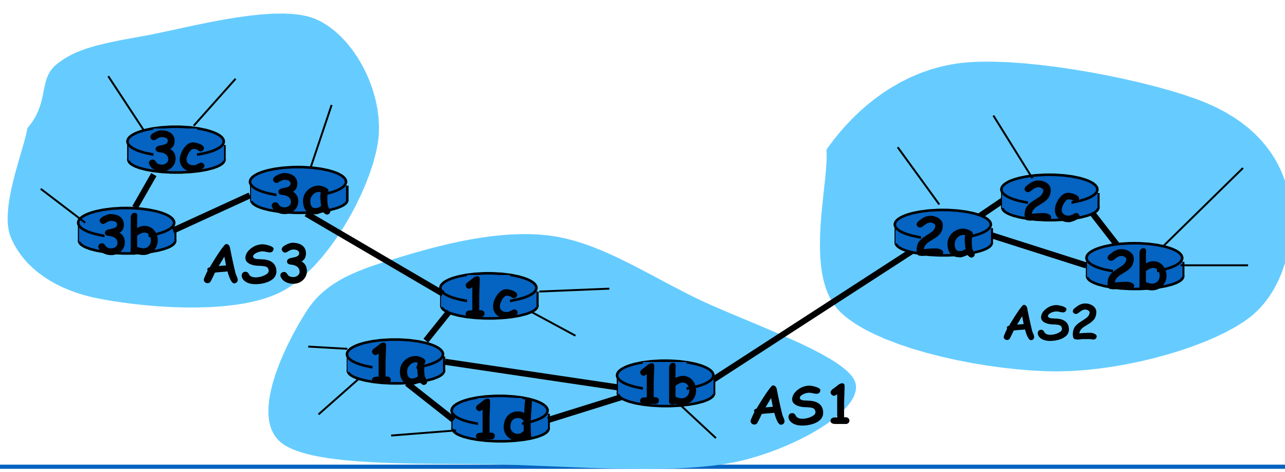
Inter-AS tasks

- Suppose router in AS1 receives datagram for which dest is outside of AS1
  - Router should forward packet towards one of the gateway routers, but which one?

AS1 needs:

1. to learn which dests are reachable through AS2 and which through AS3
2. to propagate this reachability info to all routers in AS1

Job of inter-AS routing!



Example: Setting forwarding table in router 1d

- Suppose AS1 learns from the inter-AS protocol that subnet *x* is reachable from AS3 (gateway 1c) but not from AS2.
- Inter-AS protocol propagates reachability info to all internal routers.
- Router 1d determines from intra-AS routing info that its interface *I* is on the least cost path to 1c.
- Puts in forwarding table entry *(x,I)*.

Intra-AS Routing

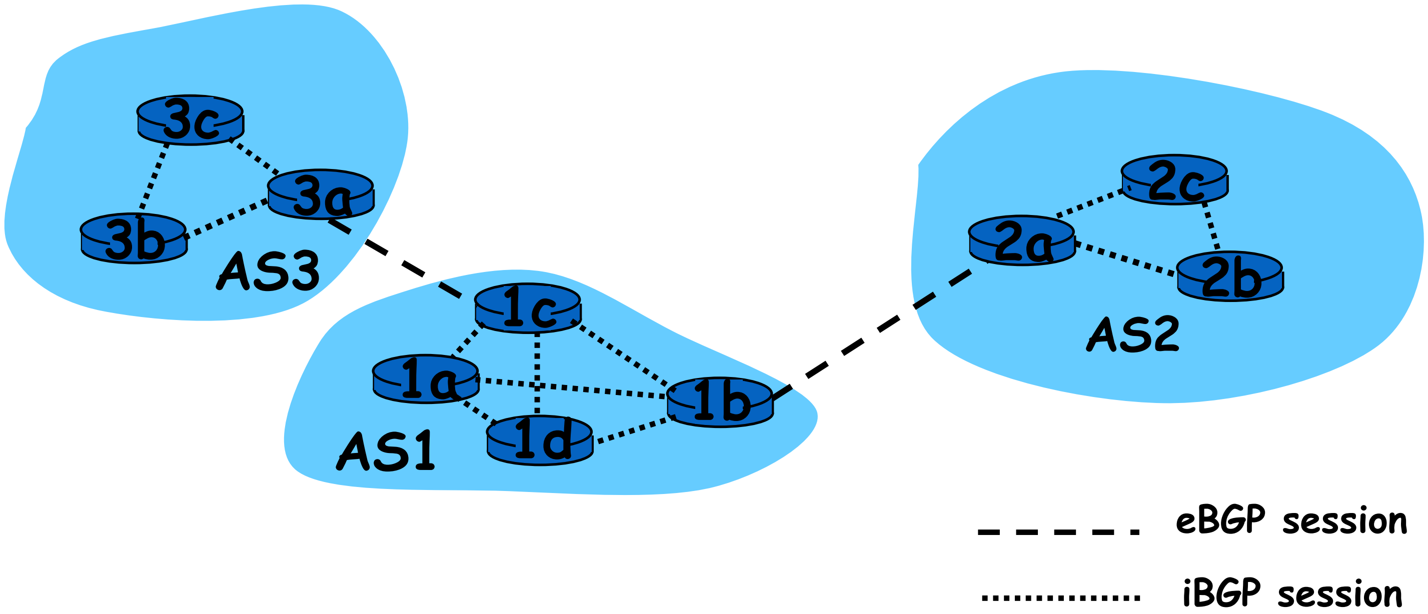
- Also known as Interior Gateway Protocols (IGP)
- Most common Intra-AS routing protocols:
  - RIP: Routing Information Protocol
  - OSPF: Open Shortest Path First
  - IGRP: Interior Gateway Routing Protocol (Cisco proprietary)

Internet inter-AS routing: BGP

- BGP (Border Gateway Protocol): *the* de facto standard
- BGP provides each AS a means to:
  1. Obtain subnet reachability information from neighboring ASs.
  2. Propagate the reachability information to all routers internal to the AS.
  3. Determine "good" routes to subnets based on reachability information and policy.
- Allows a subnet to advertise its existence to rest of the Internet: *"I am here"*

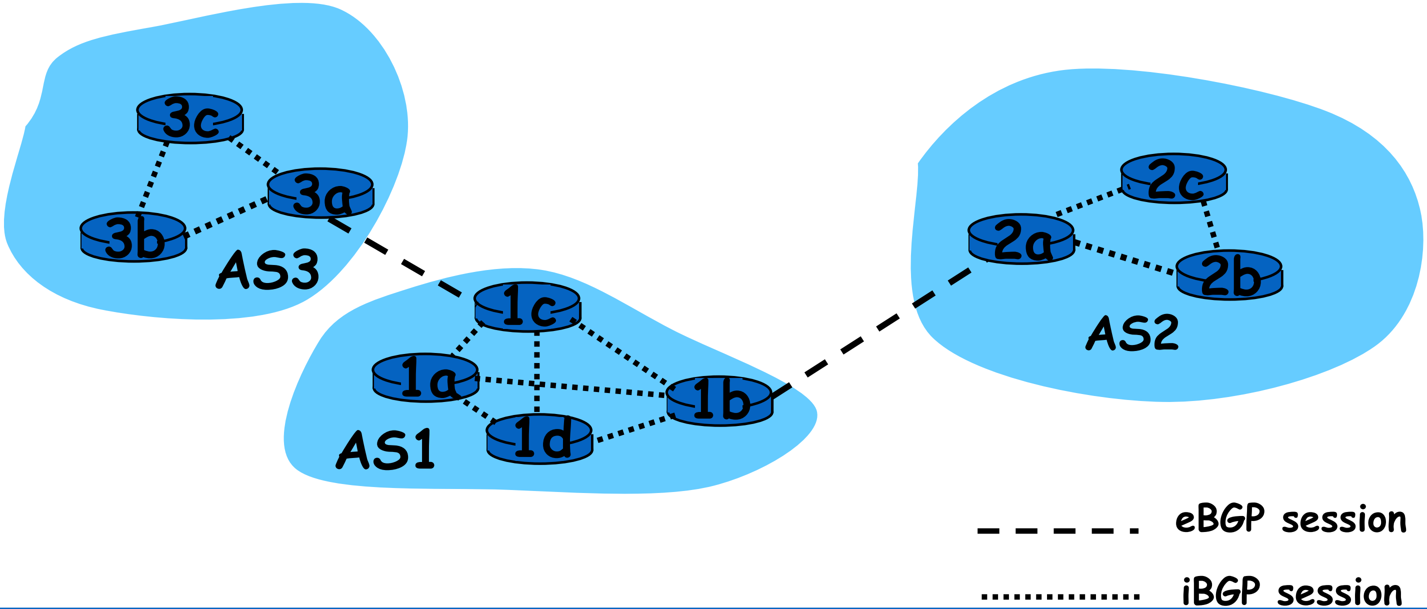
BGP basics

- Pairs of routers (BGP peers) exchange routing info over semi-permanent TCP connection: BGP sessions
- Note that BGP sessions do not correspond to physical links.
- When AS2 advertises a prefix to AS1, AS2 is *promising* it will forward any datagrams destined to that prefix towards the prefix.
  - AS2 can aggregate prefixes in its advertisement



Distributing reachability info

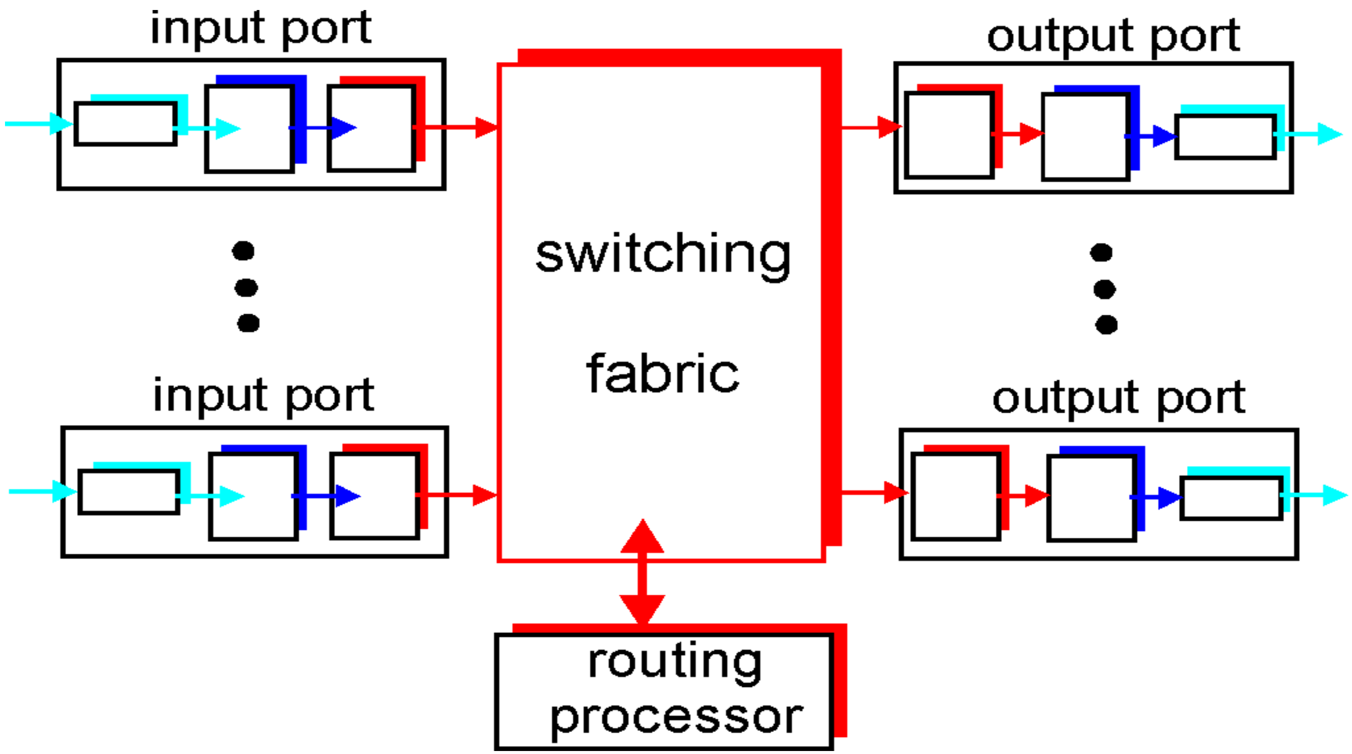
- With eBGP session between 3a and 1c, AS3 sends prefix reachability info to AS1.
- 1c can then use iBGP do distribute this new prefix reach info to all routers in AS1
- 1b can then re-advertise the new reach info to AS2 over the 1b-to-2a eBGP session
- When router learns about a new prefix, it creates an entry for the prefix in its forwarding table.



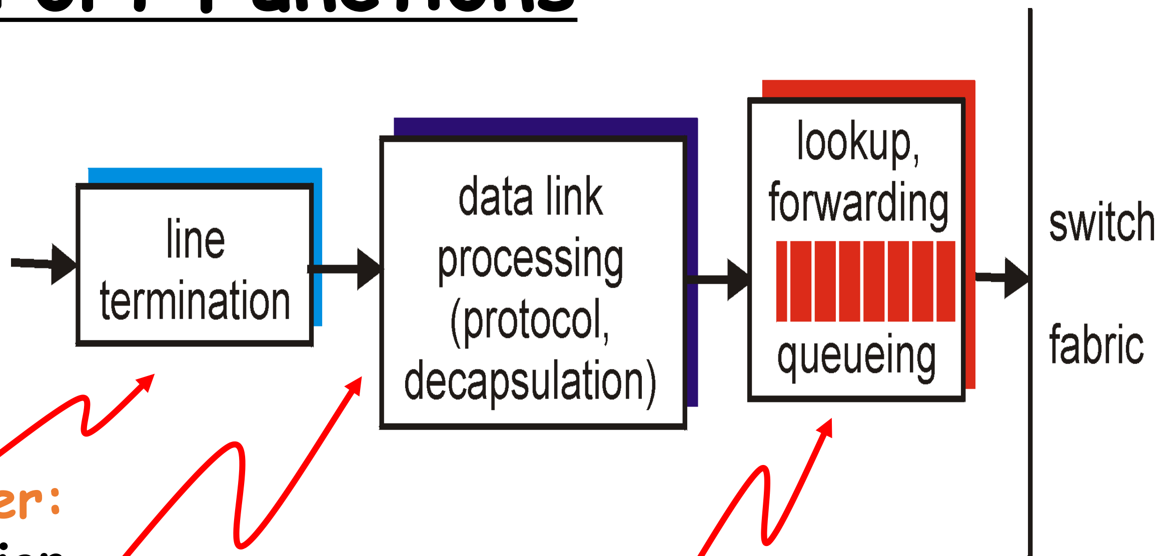
Router Architecture Overview

Two key router functions:

- run routing algorithms/protocol (RIP, OSPF, BGP)
- *forwarding* datagrams from incoming to outgoing link



Input Port Functions



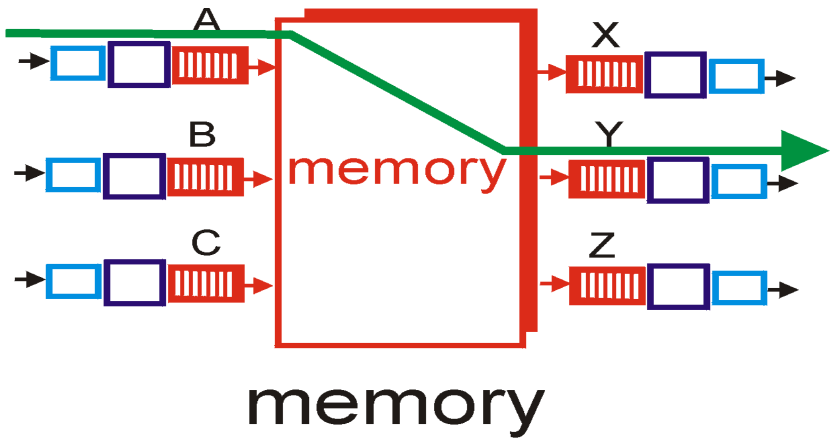
Physical layer:
bit-level reception

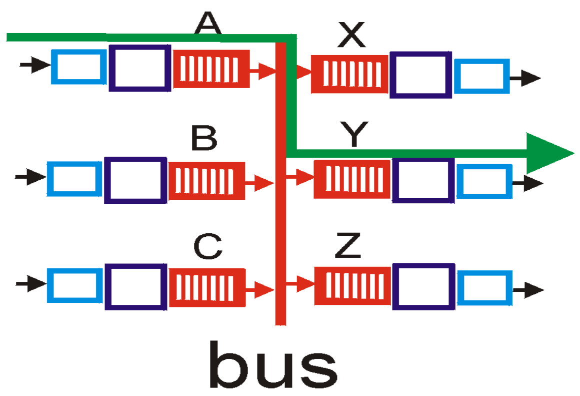
Data link layer:
e.g., Ethernet
see chapter 5

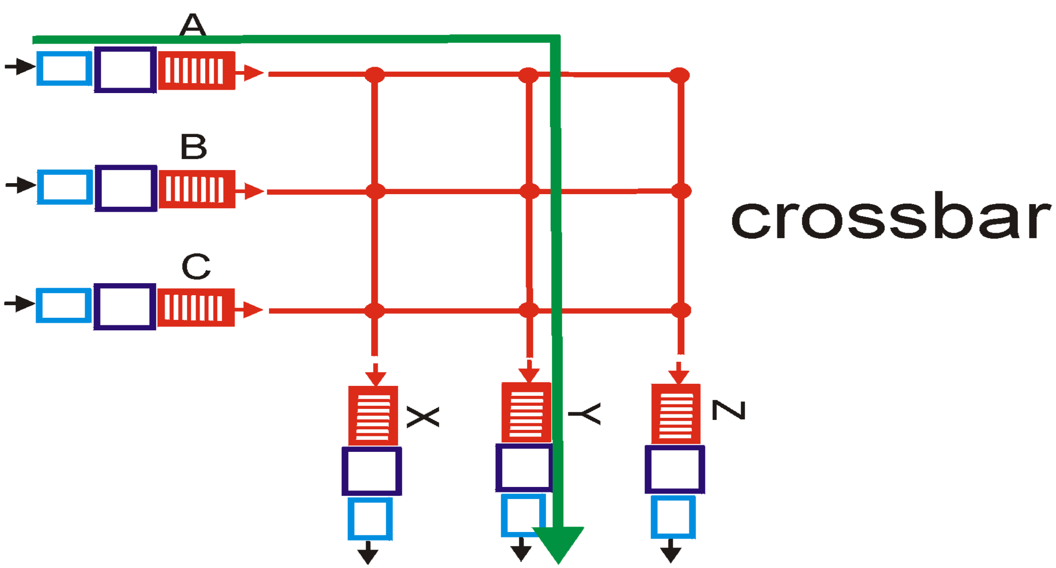
Decentralized switching:

- given datagram dest., lookup output port using forwarding table
- goal: complete input port processing at ‘line speed’
- queuing: if datagrams arrive faster than forwarding rate into switch fabric

Three types of switching fabrics

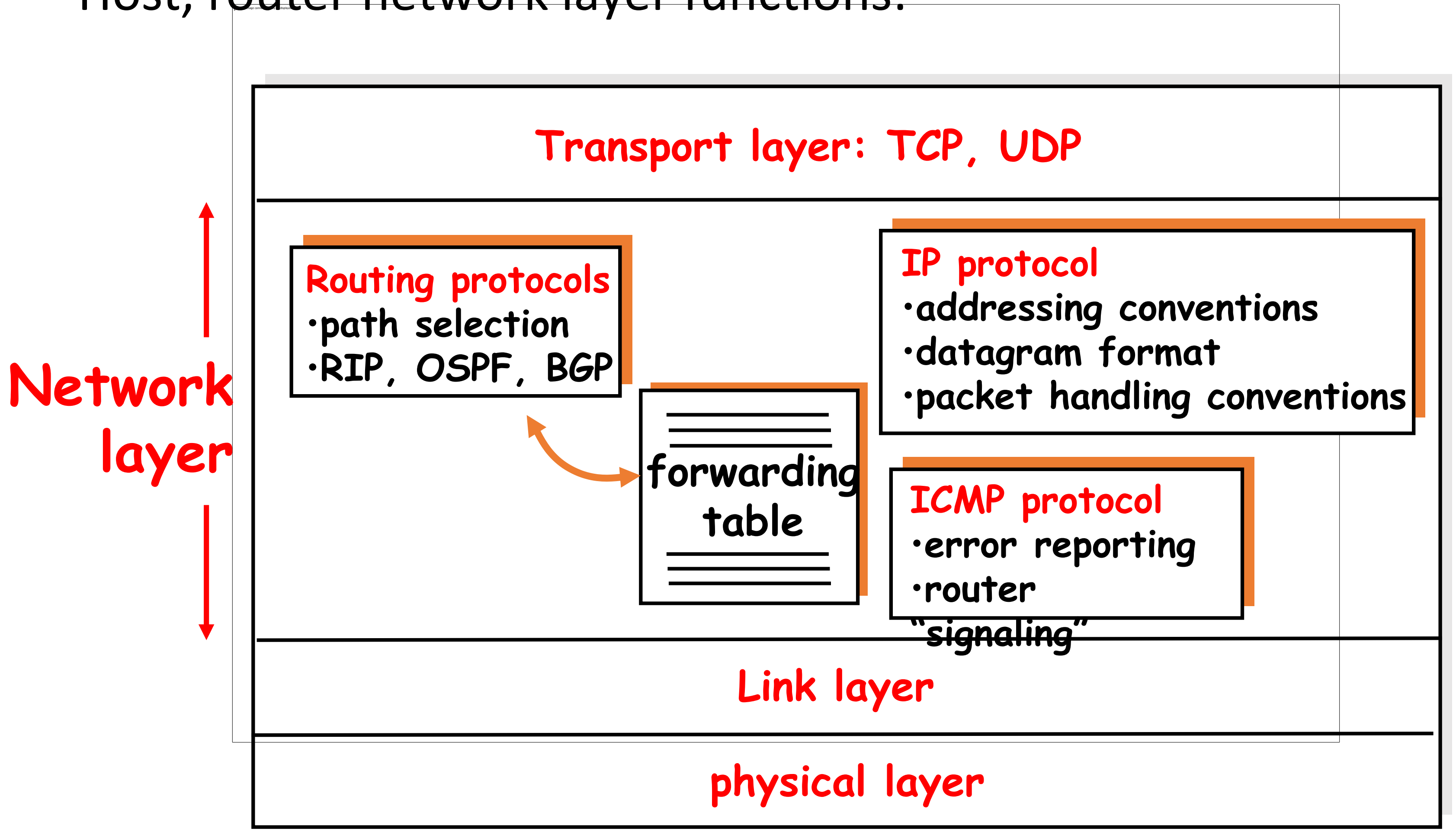






The Internet Network layer

Host, router network layer functions:



IP addressing: the last word...

Q: How does an ISP get block of addresses?

A: ICANN: Internet Corporation for Assigned Names and Numbers

- allocates addresses
- manages DNS
- assigns domain names, resolves disputes