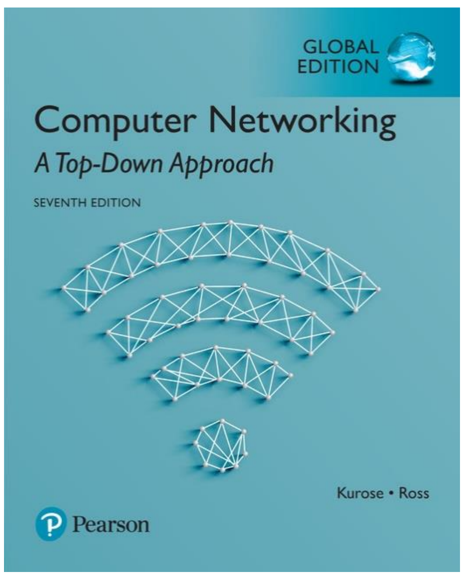
Slides Production: Prof. Yau Kok Lim

Dr. Saad Aslam

Office: Room AE-327, 3rd Floor, NUB

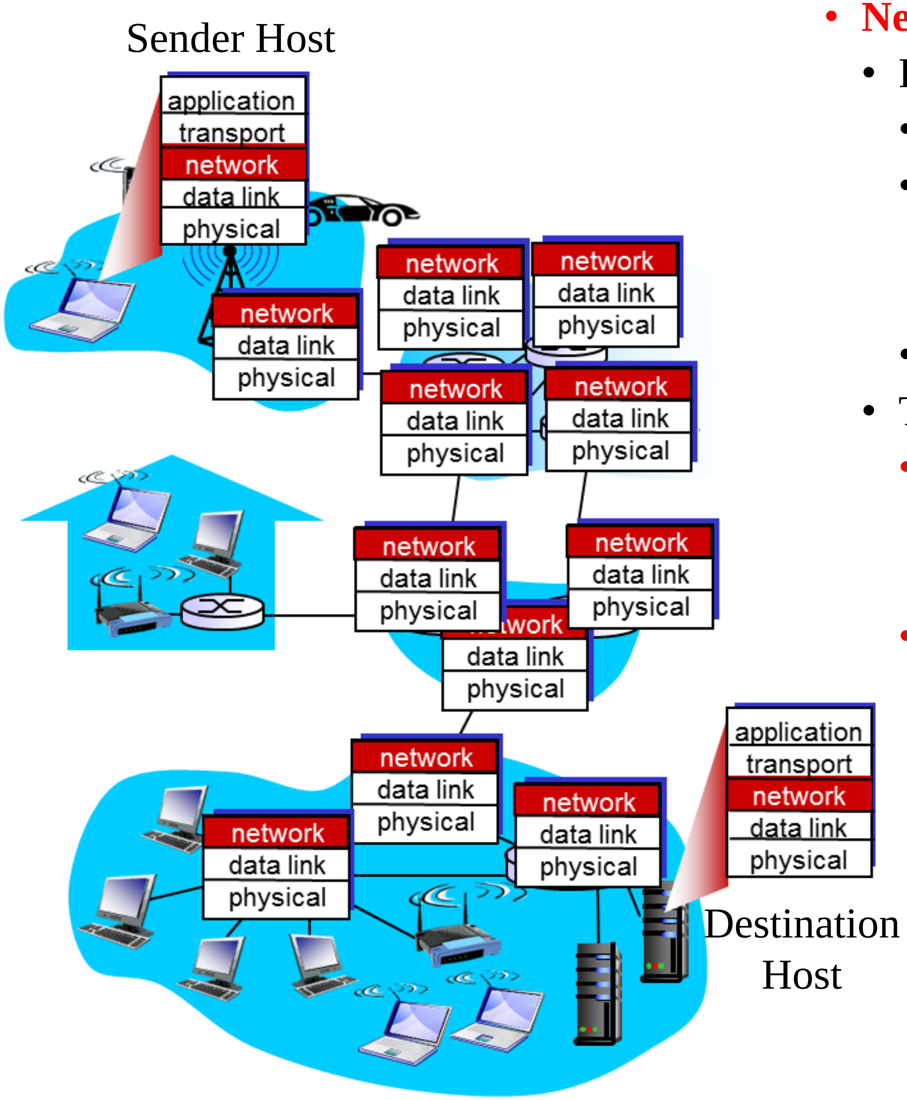
Contact: +603 74918622
(Ext: 7143)

Email: saada@sunway.edu.my



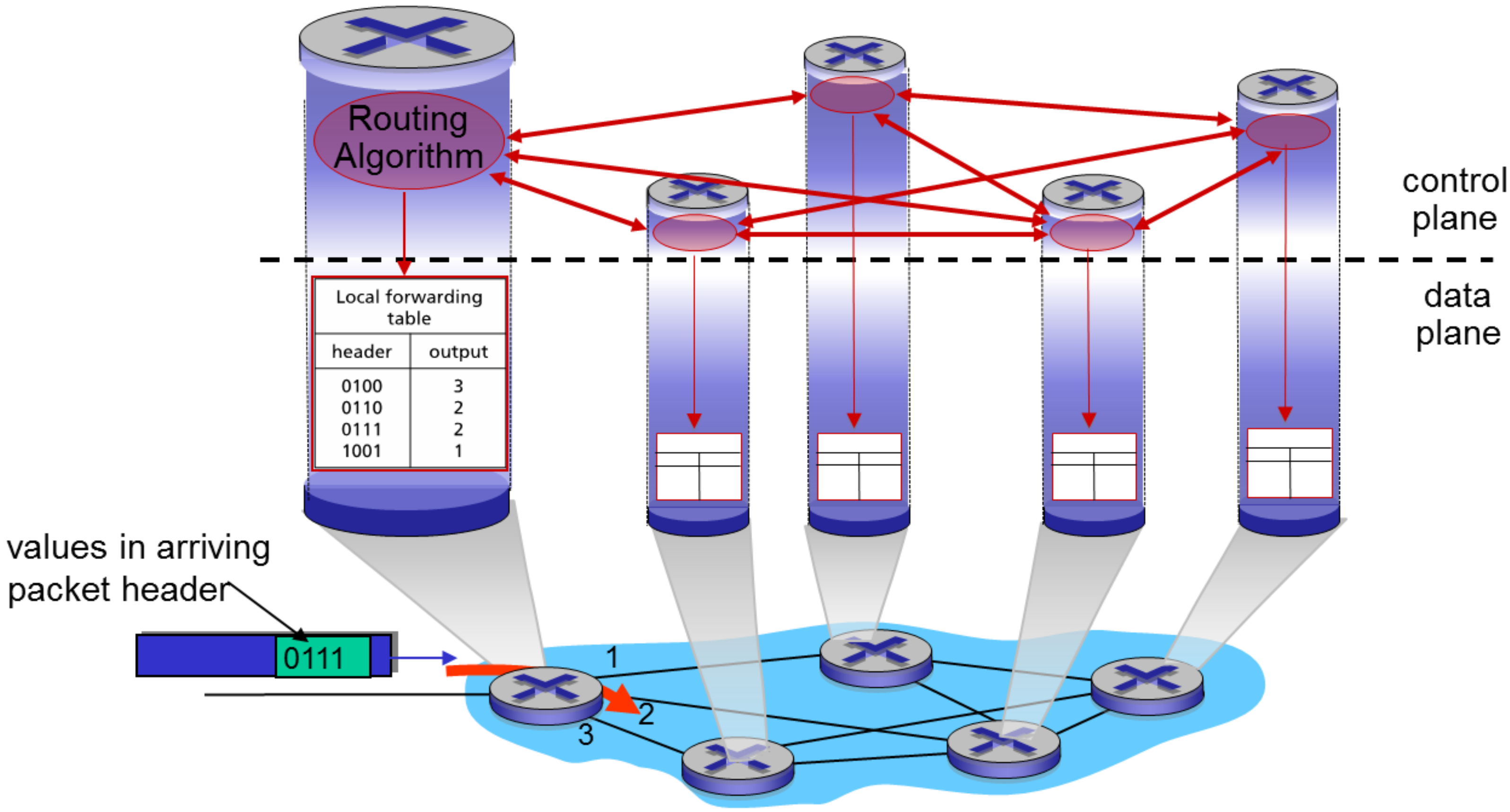
Computer Networking: A Top-down Approach, 7th edition.
Jim Kurose, Keith Ross
Pearson

4.1 Introduction



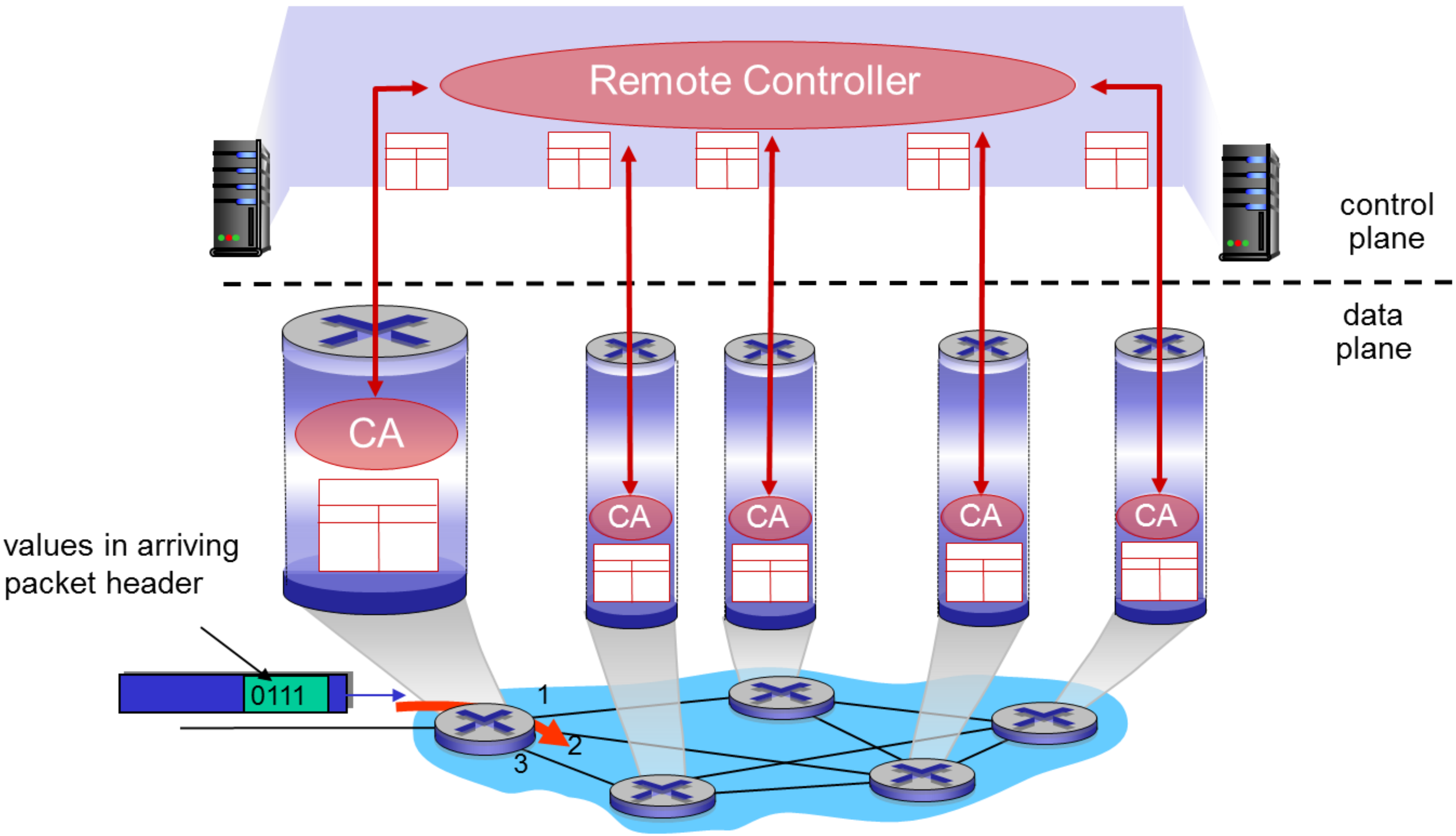
- **Network Layer**
  - Provide logical communication between two hosts
    - Sender host: Encapsulate segments into datagrams
    - Intermediate host (Router):
      - Examine datagram header
      - Perform routing and forwarding
    - Destination host: Pass datagrams to transport layer
  - Two planes
    - **Data plane**
      - Move packet from router's input to a router's output
        - Local function
    - **Control plane**
      - Determine a route from source to destination (routing)
        - Network-wide function
      - Two approaches
        - **Per-router control plane**
          - Each router perform routing separately to compute forwarding table
        - **Logically centralized control plane**
          - A centralized remote controller interact with local control agent in routers to compute forwarding table

4.1 Per-router Control Plane



- A traditional approach implemented in all routers.
- Each router computes its own forwarding table in a distributed manner.

4.1 Logically Centralized Control Plane
(also called Software-defined Networking, SDN)



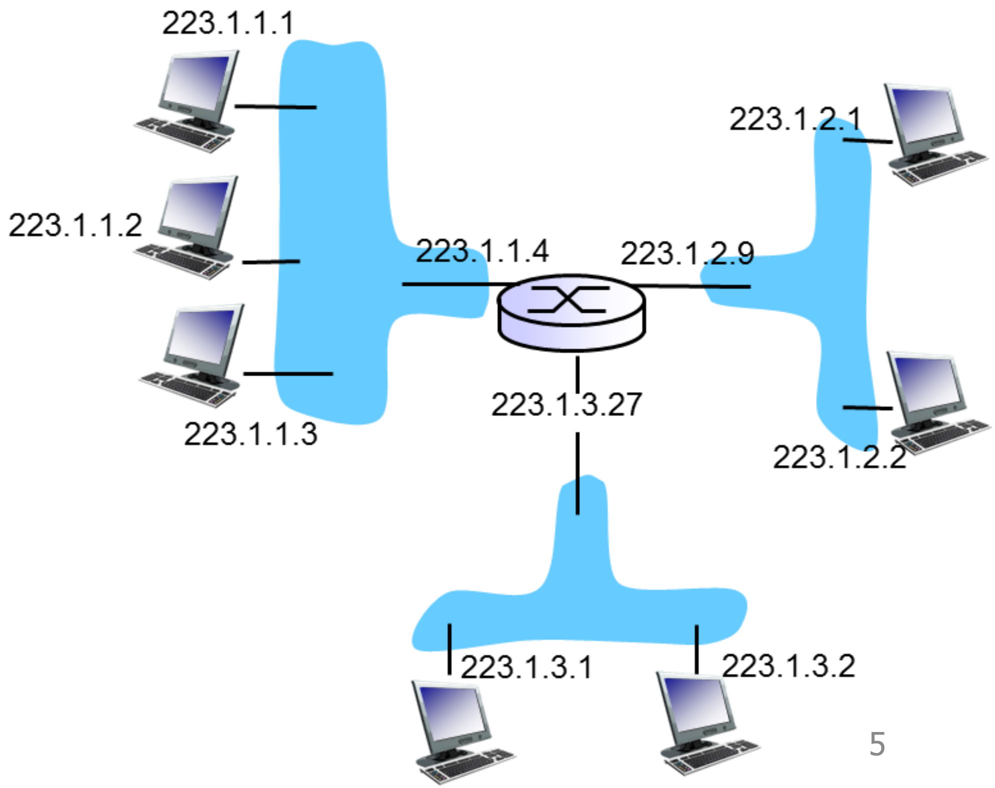
- A newer approach implemented in a remote controller.
- Each router contains a forwarding table that is computed and distributed by a remote controller.

4.3 IPv4 Addressing

- IPv4 address
  - 32 bit
    - 2^{32} possible IP addresses
  - Problem
    - 2^{32} IP addresses is used up
    - Solution
      - Use IPv6
        - 128 bit

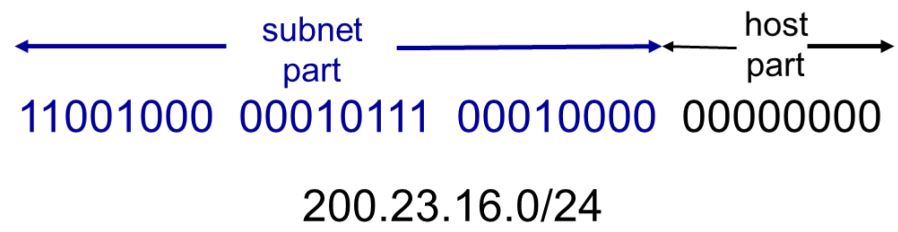
$$223.1.1.1 = \underbrace{11011111}_{223}\ \underbrace{00000001}_{1}\ \underbrace{00000001}_{1}\ \underbrace{00000001}_{1}$$

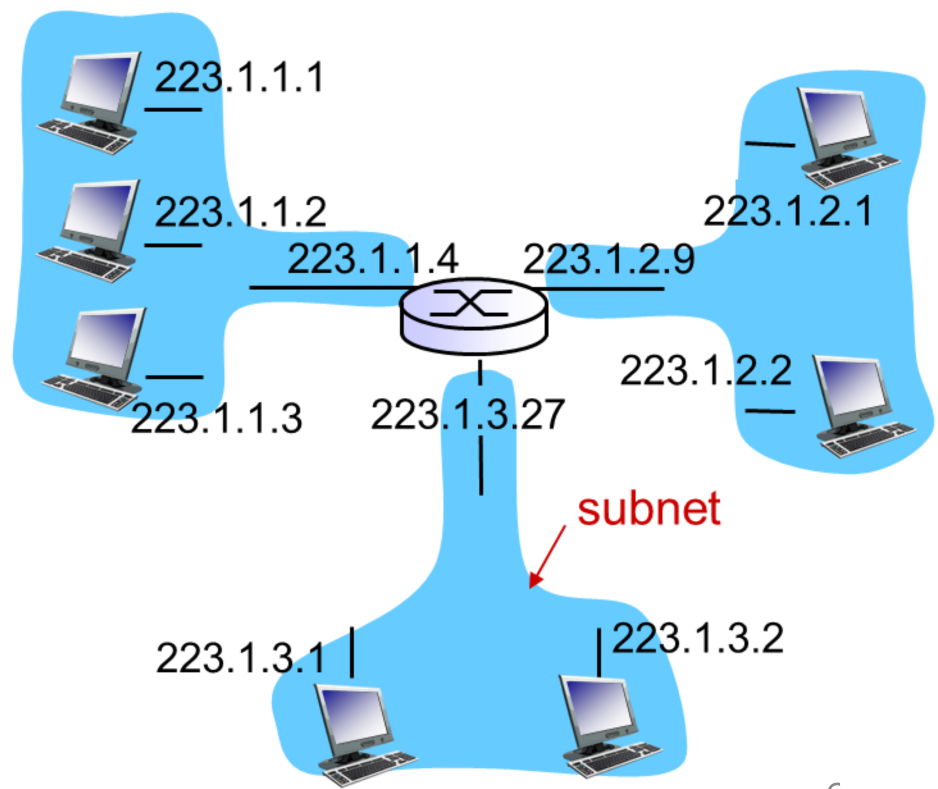
- Each IPv4 address identify an interface (connection between a host/ router and a link
  - Each IP address is globally unique
  - Router has multiple interface
  - E.g.: This router has 3 interfaces
      - Each interface has an IP address



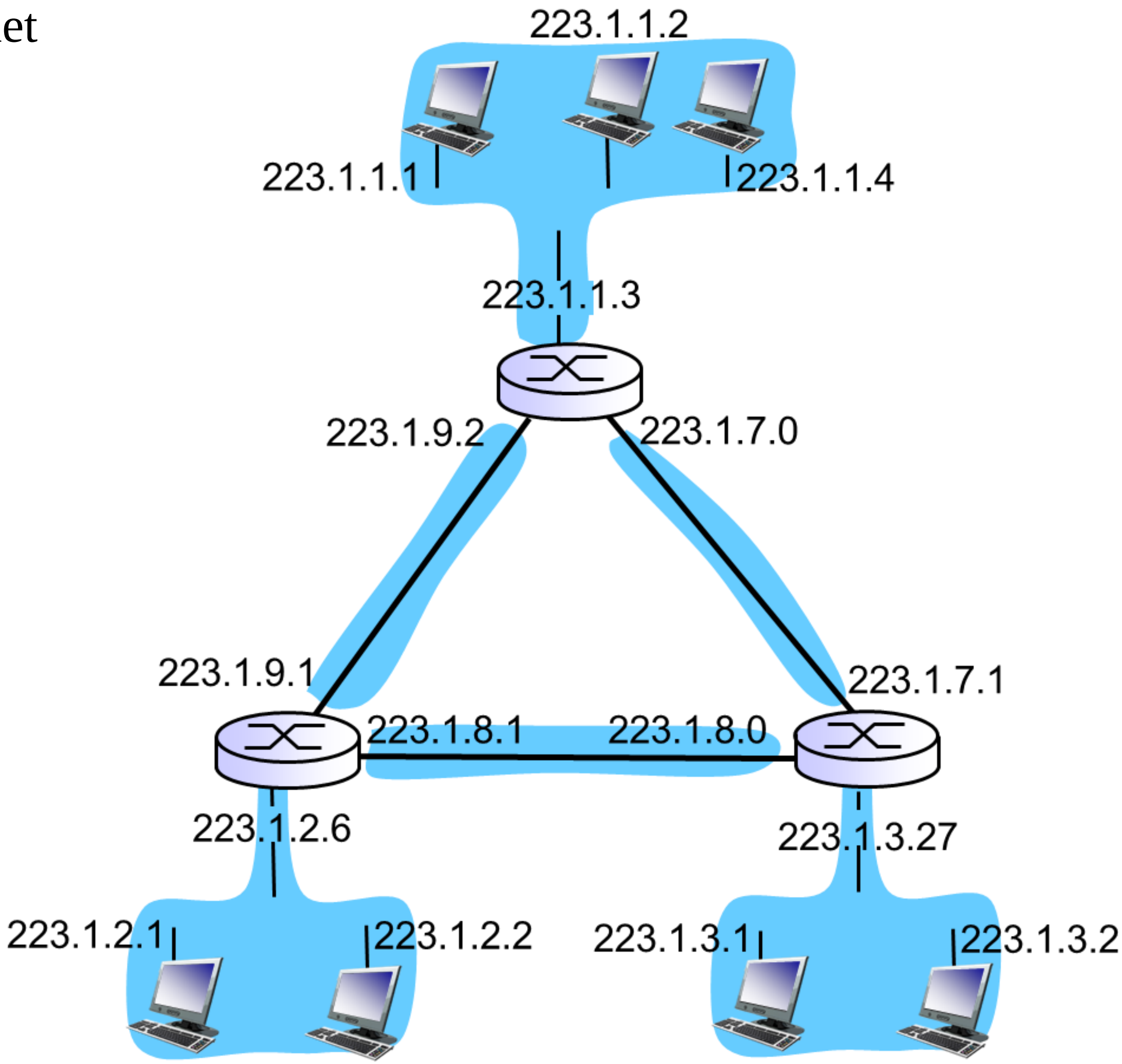
4.3 IPv4 Addressing

- IPv4 address
  - High-order bits
    - Indicate subnet
  - Low-order bits
    - Indicate host
- Each subnet
  - Hosts have similar high-order bits
  - Hosts are physically relevant to each other (e.g.: an organization)
  - E.g.: This router has 3 subnet
    - 223.1.1.0/24
      - /24 is subnet mask
        - Indicate 24 high-order bits
      - Has 3 host interface
        - 223.1.1.1
        - 223.1.1.2
        - 223.1.1.3
      - Has 1 router interface
        - 223.1.1.4
    - 223.1.3.0/24
    - 223.1.2.0/24



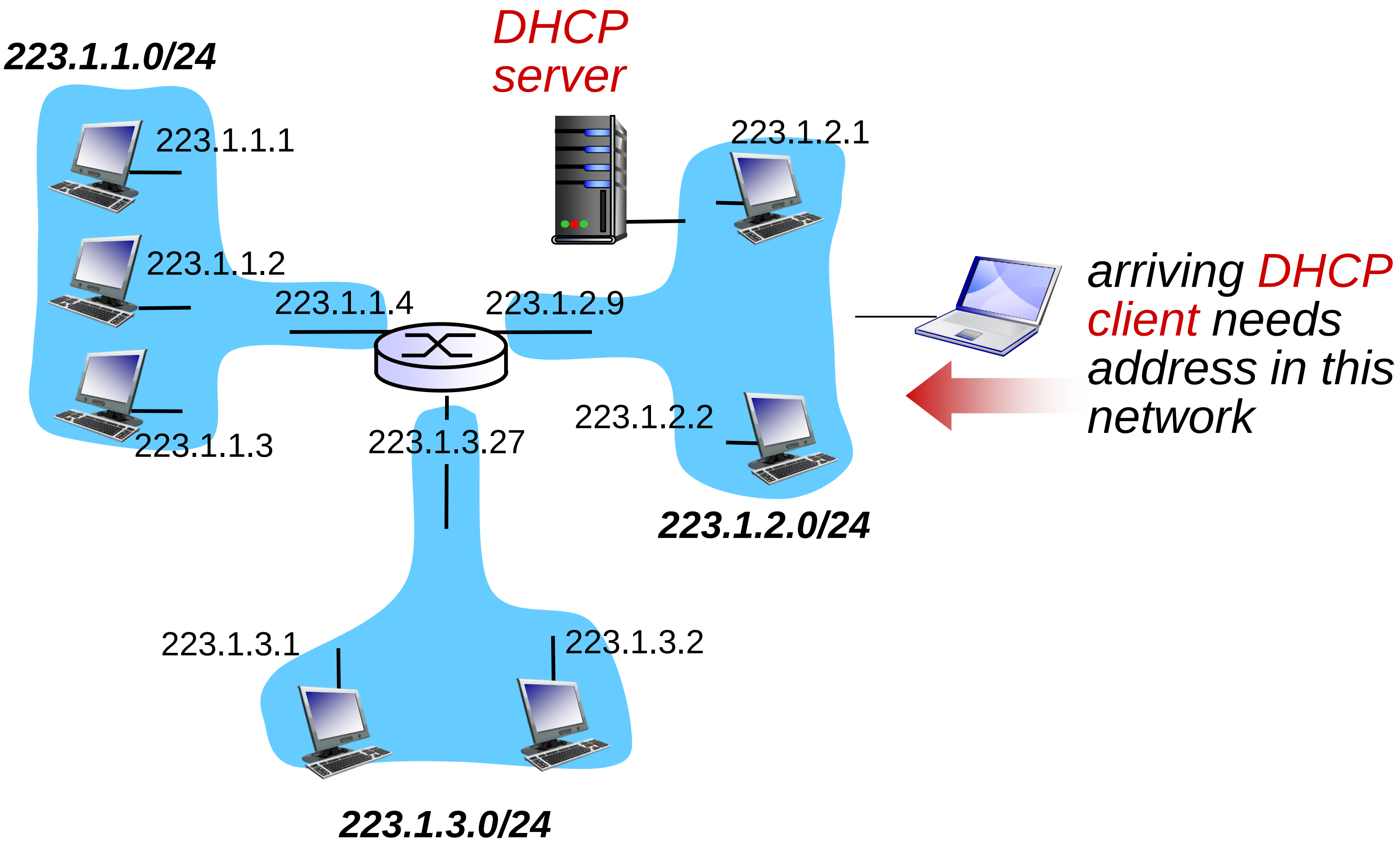


- How many subnets? 6 subnet
  - 223.1.1.0/24
  - 223.1.2.0/24
  - 223.1.3.0/24
  - 223.1.7.0/24
  - 223.1.8.0/24
  - 223.1.9.0/24

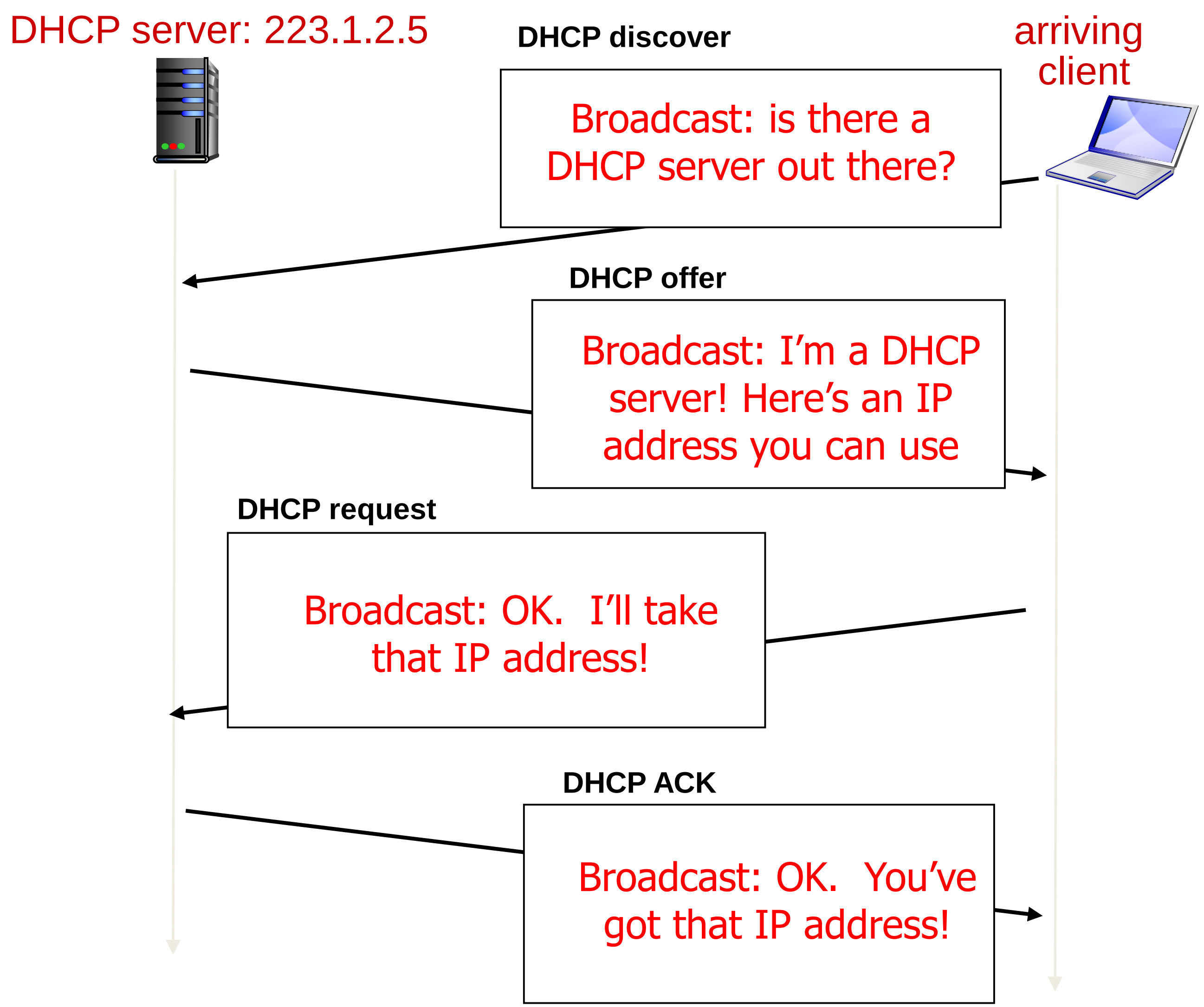


- Hard-coded by system admin in a file
  - Windows
    - control-panel->network->configuration->tcp/ip->properties
  - UNIX
    - /etc/rc.config
- Dynamic Host Configuration Protocol (DHCP)
  - Allows a host to dynamically obtain an IP address from a server when it joins a network
    - “plug-and-play”

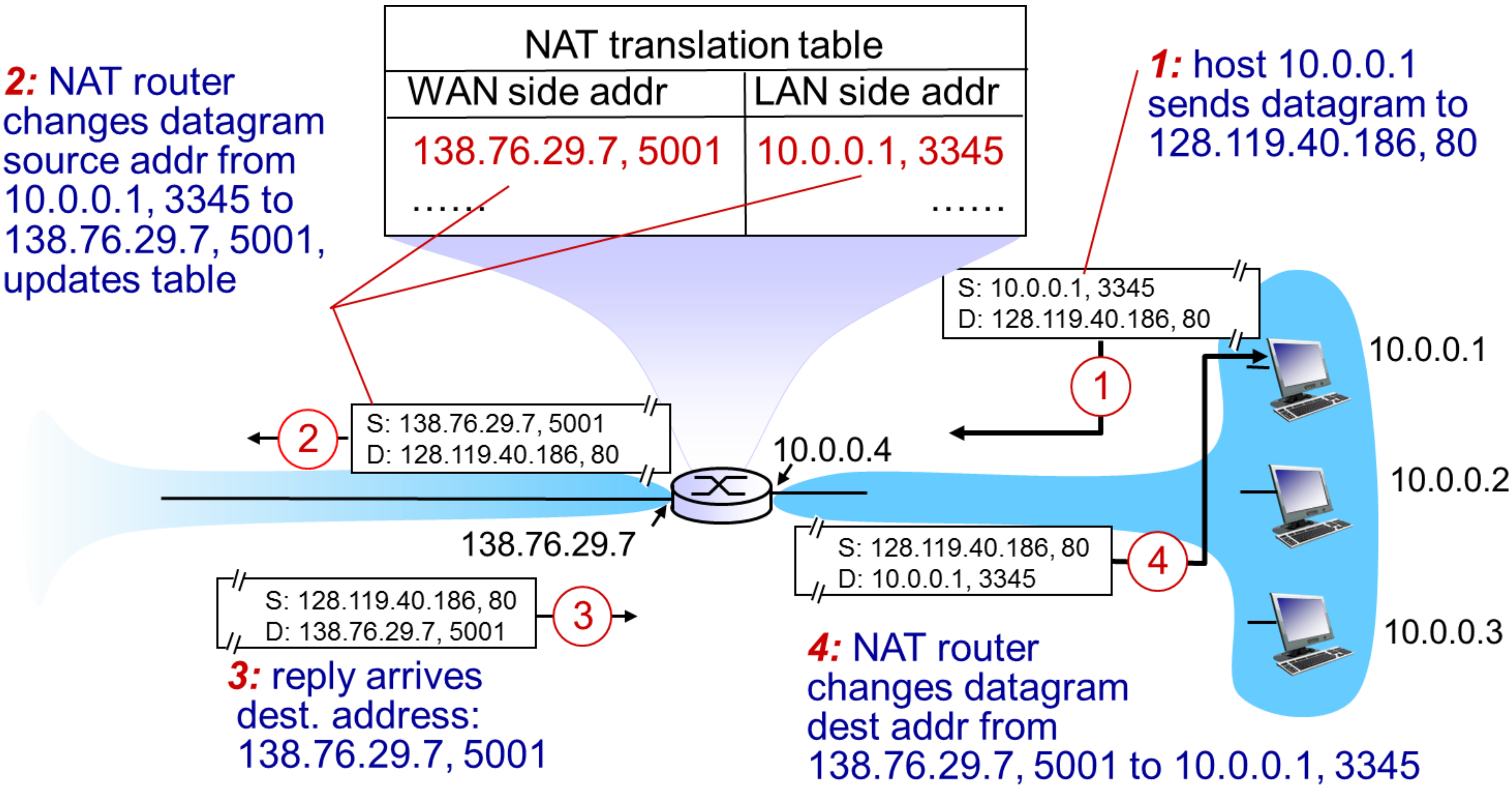
4.3 How does a host get IP address?
Dynamic Host Configuration Protocol



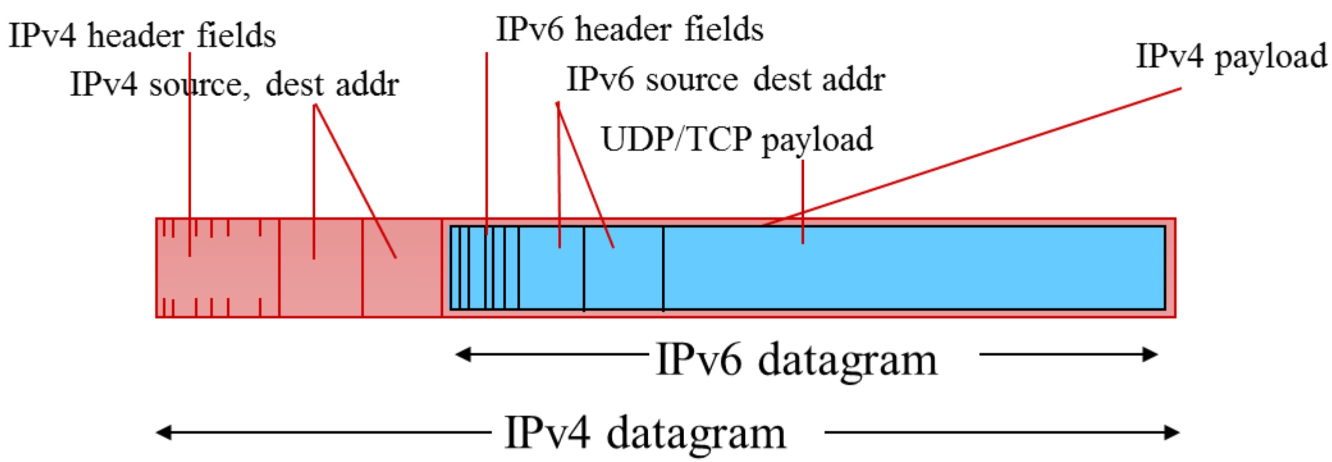
4.3 How does a host get IP address?
Dynamic Host Configuration Protocol

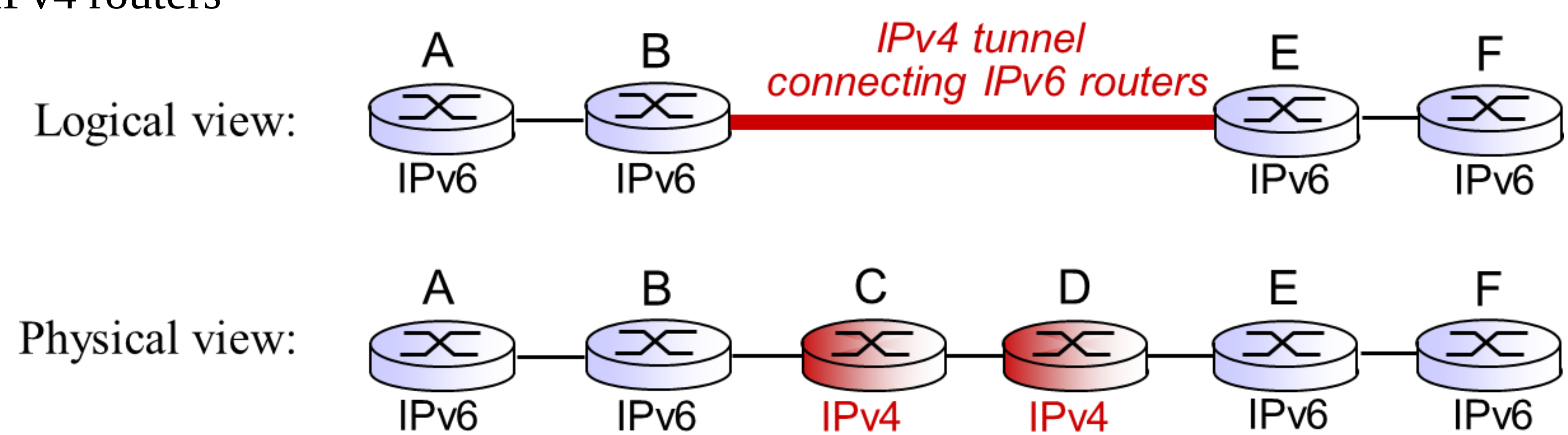


- Network address translation (NAT)
  - local network uses just one IP address as far as outside world is concerned
  - Advantages
    - range of addresses not needed from ISP
      - just one IP address for all devices
    - can change addresses of devices in local network without notifying outside world
    - can change ISP without changing addresses of devices in local network
    - devices inside local network not explicitly addressable, visible by outside world (a security plus)

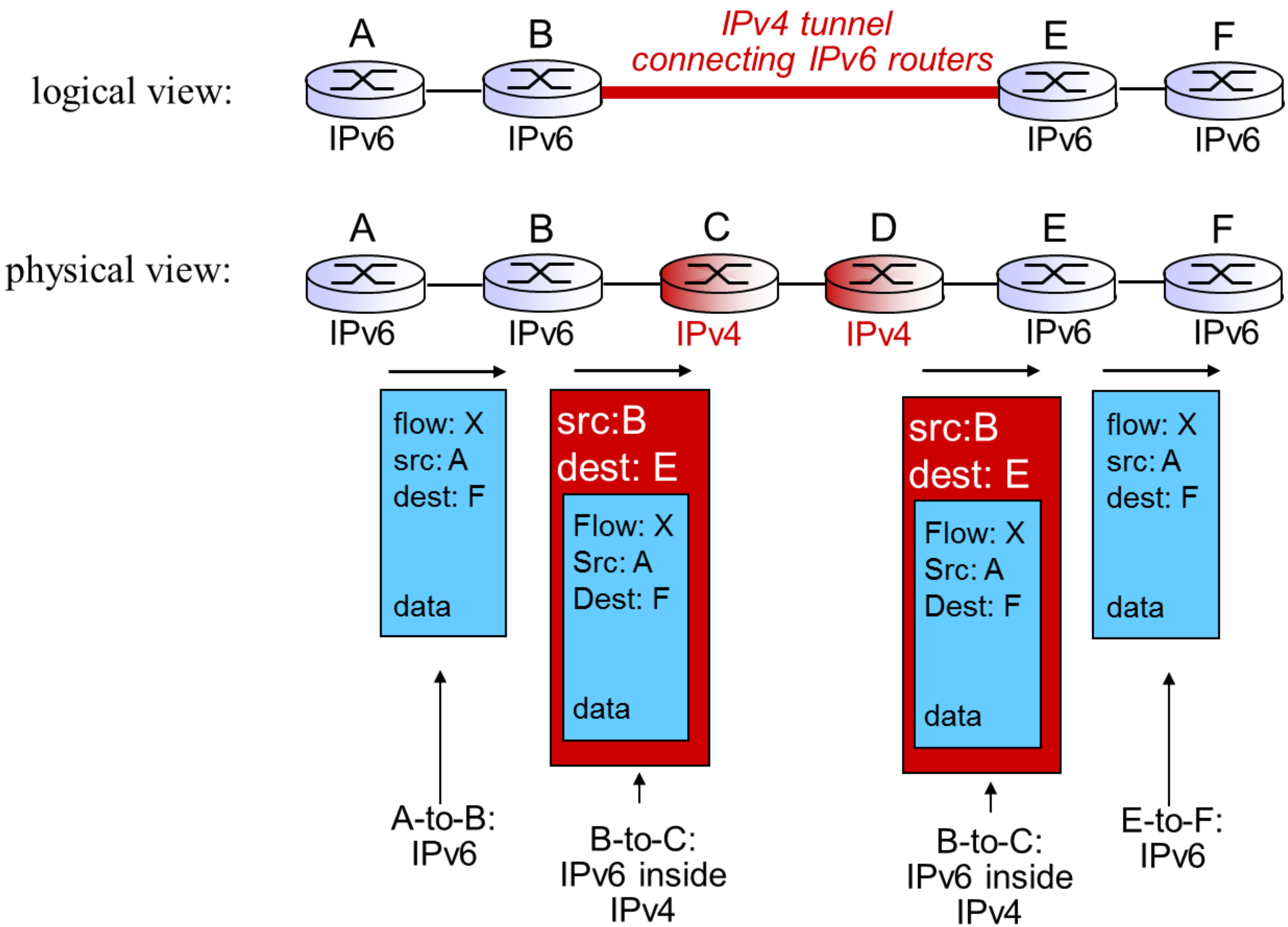

NAT translation table
WAN side addr
LAN side addr
138.76.29.7, 5001
10.0.0.1, 3345
……
……
1: host 10.0.0.1 sends datagram to 128.119.40.186, 80
2: NAT router changes datagram source addr from 10.0.0.1, 3345 to 138.76.29.7, 5001, updates table
S: 10.0.0.1, 3345
D: 128.119.40.186, 80
1
10.0.0.1
10.0.0.2
10.0.0.3
2
S: 138.76.29.7, 5001
D: 128.119.40.186, 80
10.0.0.4
138.76.29.7
S: 128.119.40.186, 80
D: 10.0.0.1, 3345
4
S: 128.119.40.186, 80
D: 138.76.29.7, 5001
3
3: reply arrives dest. address: 138.76.29.7, 5001
4: NAT router changes datagram dest addr from 138.76.29.7, 5001 to 10.0.0.1, 3345

- IPv6
  - not all routers can be upgraded simultaneously
  - How to provide downward compatible (compatible with IPv4)?
    - **Tunneling**
      - Carry IPv6 datagram as payload in IPv4 datagram among IPv4 routers





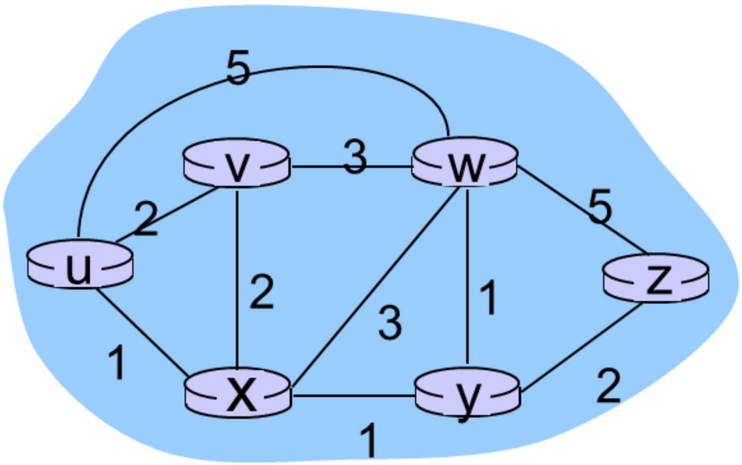
4.3 IPv6: Transitioning from IPv4 to IPv6



- IPv6 adoption
  - Google: 8% of clients access services via IPv6
  - Takes long time (up to 20 years) for adoption

5.2 Routing Protocols: Overview

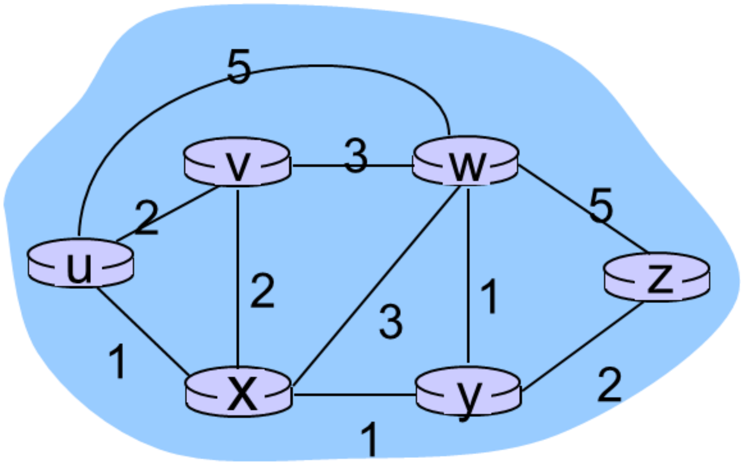
- Routing algorithm
  - Centralized or Decentralized
    - **Centralized**
      - Also called Link-State Algorithm
      - Router has complete network-wide information including
        - Network topology (Routers and Links)
        - Link cost
    - **Decentralized / Distributed**
      - Also called Distance Vector Algorithm
      - Router has neighbor nodes' information including
        - Network topology
        - Link cost
      - Neighbor nodes exchange information (i.e. route cost) among themselves
  - Static or Dynamic
    - **Static**
      - Route change slowly
        - Route change due to changes in network topology and link cost
    - **Dynamic**
      - Route change faster
      - Router update link changes frequently



- Network representation
  - Network, $G = (N, E)$
  - Routers, $N = \{u, v, w, x, y, z\}$
  - Links,
    $E = \{(u,v), (u,w), (u,x), (v,w), (v,x),$
    $(w,x), (w,y), (w,z), (x,y), (y,z)\}$
  - Link cost, $c(x, x')$
    - E.g.: $c(u, w) = 5$
    - Related to network condition, such as bandwidth, network congestion level, delay
  - Cost of path $(x_1, x_2, x_3, \dots, x_p)$
    $= c(x_1, x_2) + c(x_2, x_3) + \cdots + c(x_{p-1}, x_p)$
    - E.g.: Cost of path (u, v, w, z)
      $= c(u, v) + c(v, w) + c(w, z)$
      $= 2 + 3 + 5 = 10$

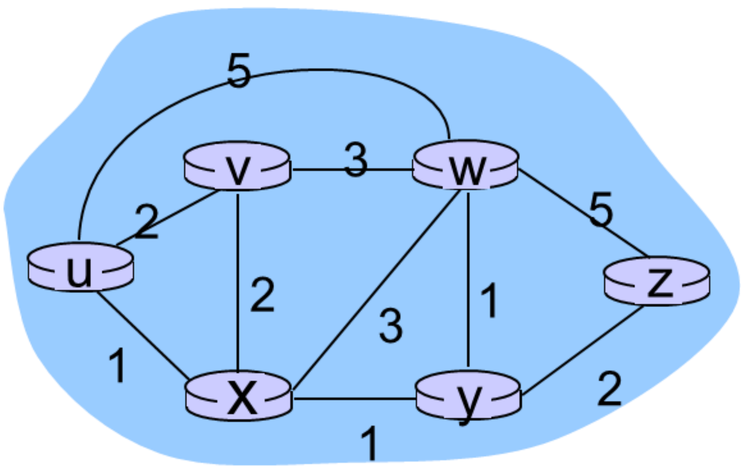
- Routing algorithm
  - Calculate least-cost path (route) between u and z

5.2 The Link-State (LS) Routing Algorithm



- Routing representation
  - $c(x, x')$: Link cost
  - $D(v)$: Current cost of path from source to destination v
  - $p(v)$: Previous node (neighbor of node v) along the current least-cost path from source to node v
  - N': Subset of nodes. Node v is in N' if the least-cost path from the source to v is known

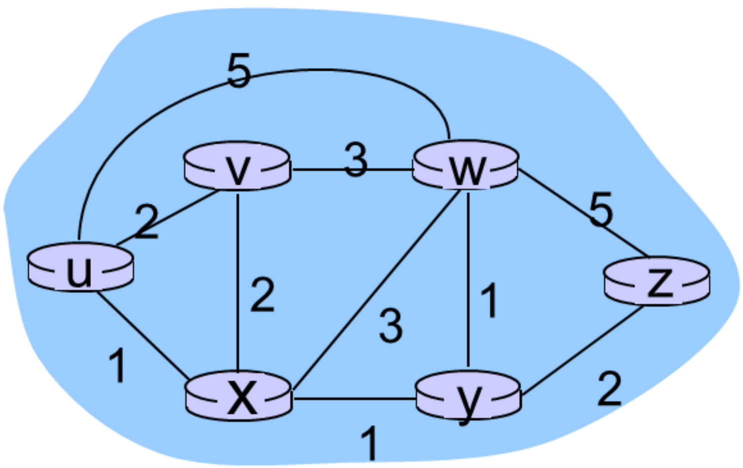
- Dijkstra's algorithm
  - Determine the least cost path (route) between u and z
  - Node (Source)
    - Calculate least-cost path (route) from itself to all other nodes in the network
    - Use least-cost path to update its forwarding table
  - After k iterations, node (source) know least-cost paths to k destination node



- Routing representation
  - $c(x, x')$: Link cost
  - $D(v)$: Current cost of path from source to destination v
  - $p(v)$: Previous node (neighbor of node v) along the current least-cost path from source to node v
  - N': Subset of nodes. Node v is in N' if the least-cost path from the source to v is known

- Dijkstra's algorithm

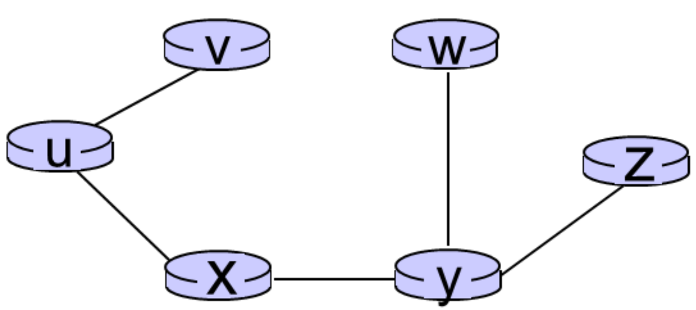
Step	N'	D(v),p(v)	D(w),p(w)	D(x),p(x)	D(y),p(y)	D(z),p(z)
0	u	2,u	5,u	1,u	∞	∞
1	ux	2,u	4,x		2,x	∞
2	uxy	2,u	3,y			4,y
3	uxyv		3,y			4,y
4	uxyvw					4,y
5	uxyvwz					



- Routing representation
  - $c(x, x')$: Link cost
  - $D(v)$: Current cost of path from source to destination v
  - $p(v)$: Previous node (neighbor of node v) along the current least-cost path from source to node v
  - N': Subset of nodes. Node v is in N' if the least-cost path from the source to v is known

Dijkstra's algorithm outcome:

- Least-cost path tree

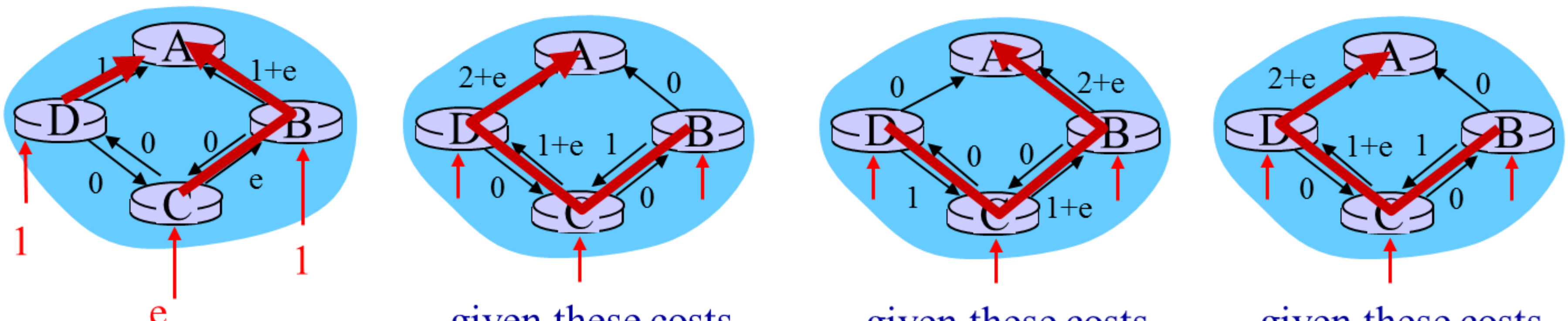


Dijkstra's algorithm outcome:

- Forwarding table for node u

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

- Problem
  - Oscillation
    - Happen when link cost is related to network congestion level (or traffic load)
    - Example: Node *B*, *C* and *D* send to node *A*



initially

given these costs, find new routing…. resulting in new costs

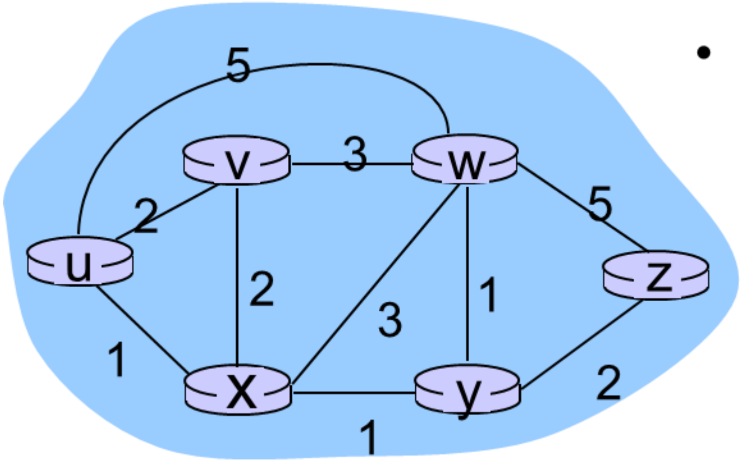
given these costs, find new routing…. resulting in new costs

given these costs, find new routing…. resulting in new costs

Path for	Path for	Path for	Path for
$B:(B,A)$	$B:(B,C,D,A)$	$B:(B,A)$	$B:(B,C,D,A)$
$C:(C,B,A)$	$C:(C,D,A)$	$C:(C,B,A)$	$C:(C,D,A)$
$D:(D,A)$	$D:(D,A)$	$D:(D,C,B,A)$	$D:(D,A)$

- Solution
  - Randomize the time neighbor nodes exchange link cost among themselves

5.2 The Distance-Vector (DV) Routing Algorithm



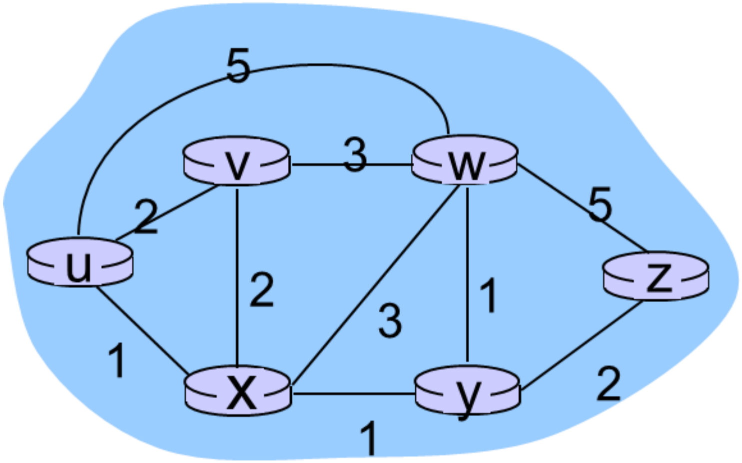
- Routing representation
  - $c(x, x')$: Link cost
  - $D_x(y)$: Distance vector
    - Cost of the least-cost path from node x and node y
  - v: Node x's neighbor nodes

- Bellman-Ford equation:

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

- $\min_v$ means that, among node x's neighbor nodes, node x choose a neighbor node with minimum $c(x, v) + D_v(y)$
- Neighbor nodes exchange information (i.e. distance vector) among themselves
  - Node x
    - Calculate $D_x(y)$
    - Send $D_x(y)$ to all neighbor nodes v only if there is changes on $D_x(y)$
  - Each neighbor node receive $D_x(y)$
    - Keep track and update the distance vector $D_x(y)$ using Bellman-Ford equation

5.2 The Distance-Vector (DV) Routing Algorithm



- Routing representation
  - $c(x, x')$: Link cost
  - $D_x(y)$: Distance vector
    - Cost of the least-cost path from node x and node y
  - v: Node x's neighbor nodes
  - Bellman-Ford equation

$$D_x(y) = \min_v\{c(x, v) + D_v(y)\}$$

- Example on Bellman-Ford equation:
  - Calculate least-cost path (route) between u and z
    - $D_v(z) = 5$, $D_w(z) = 3$, $D_x(z) = 3$
    - $c(u, v) = 2$, $c(u, w) = 5$, $c(u, x) = 1$
    - $D_u(z) = \min\{c(u, v) + D_v(z), c(u, w) + D_w(z), c(u, x) + D_x(z)\}$
      $= \min\{7, 8, 4\}$
      $= 4$

5.2 The Distance-Vector (DV) Routing Algorithm: An Example

$$D_x(y) = \min\{c(x,y) + D_y(y),\ c(x,z) + D_z(y)\}$$
$$= \min\{2+0\ ,\ 7+1\} = 2$$

$$D_x(z) = \min\{c(x,y) + D_y(z),\ c(x,z) + D_z(z)\}$$
$$= \min\{2+1\ ,\ 7+0\} = 3$$

node x table

from \ cost to	x	y	z
x	0	2	7
y	∞	∞	∞
z	∞	∞	∞

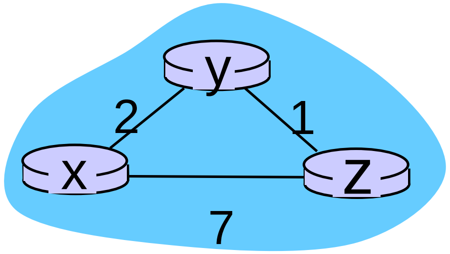
from \ cost to	x	y	z
x	0	2	3
y	2	0	1
z	7	1	0

node y table

from \ cost to	x	y	z
x	∞	∞	∞
y	2	0	1
z	∞	∞	∞

node z table

from \ cost to	x	y	z
x	∞	∞	∞
y	∞	∞	∞
z	7	1	0



time

5.2 The Distance-Vector (DV) Routing Algorithm: An Example

$$D_x(y) = \min\{c(x,y) + D_y(y),\ c(x,z) + D_z(y)\} = \min\{2+0\ ,\ 7+1\} = 2$$

$$D_x(z) = \min\{c(x,y) + D_y(z),\ c(x,z) + D_z(z)\} = \min\{2+1\ ,\ 7+0\} = 3$$

node x table

from \ cost to	x	y	z
x	0	2	7
y	∞	∞	∞
z	∞	∞	∞

from \ cost to	x	y	z
x	0	2	3
y	2	0	1
z	7	1	0

from \ cost to	x	y	z
x	0	2	3
y	2	0	1
z	3	1	0

node y table

from \ cost to	x	y	z
x	∞	∞	∞
y	2	0	1
z	∞	∞	∞

from \ cost to	x	y	z
x	0	2	7
y	2	0	1
z	7	1	0

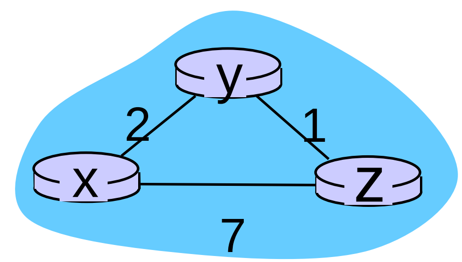
from \ cost to	x	y	z
x	0	2	3
y	2	0	1
z	3	1	0

node z table

from \ cost to	x	y	z
x	∞	∞	∞
y	∞	∞	∞
z	7	1	0

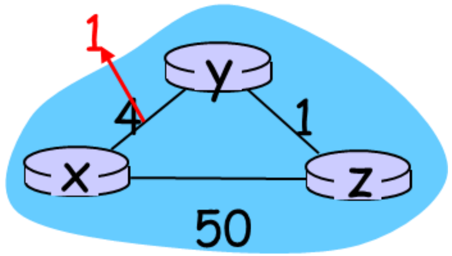
from \ cost to	x	y	z
x	0	2	7
y	2	0	1
z	3	1	0

from \ cost to	x	y	z
x	0	2	3
y	2	0	1
z	3	1	0

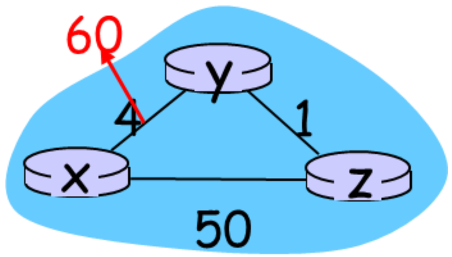


time

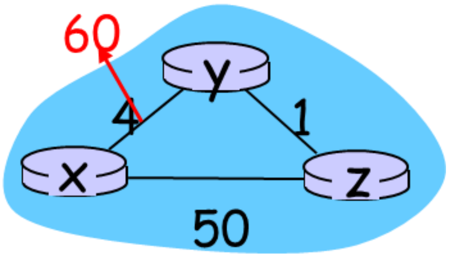
- Problem
  - **Good news** travel **Fast**, **Bad news** travel **Slow**
    - *Good news travel fast*
      - t_0: y detect link-cost change from 4 to 1
      - t_1: z receive $D_y(x) = 1$ from y, so z update $D_z(x) = 2$
      - t_2: y receive $D_z(x) = 2$ from z
        - Since $D_y(x) < c(y, z) + D_z(x)$
          - No update for $D_y(x)$ at z



    - *Bad news travel slow*
      - t_0: y detect link-cost change from 4 to 60
        - Since $D_y(x) > c(y, z) + D_z(x)$
          - $D_y(x) = 60; c(y, z) + D_z(x) = 1 + 5 = 6$
          - y update its own $D_y(x) = 6$
      - t_1: z receive $D_y(x) = 6$ from y, so z update $D_z(x) = 7$
      - t_2: y receive $D_z(x) = 7$ from z, so y update $D_y(x) = 8$
      - t_3: z receive $D_y(x) = 8$ from y, so z update $D_z(x) = 9$
      - **Routing loop** between y and z until $D_z(x) > 50$
        - So, routing loop has 44 iteration!

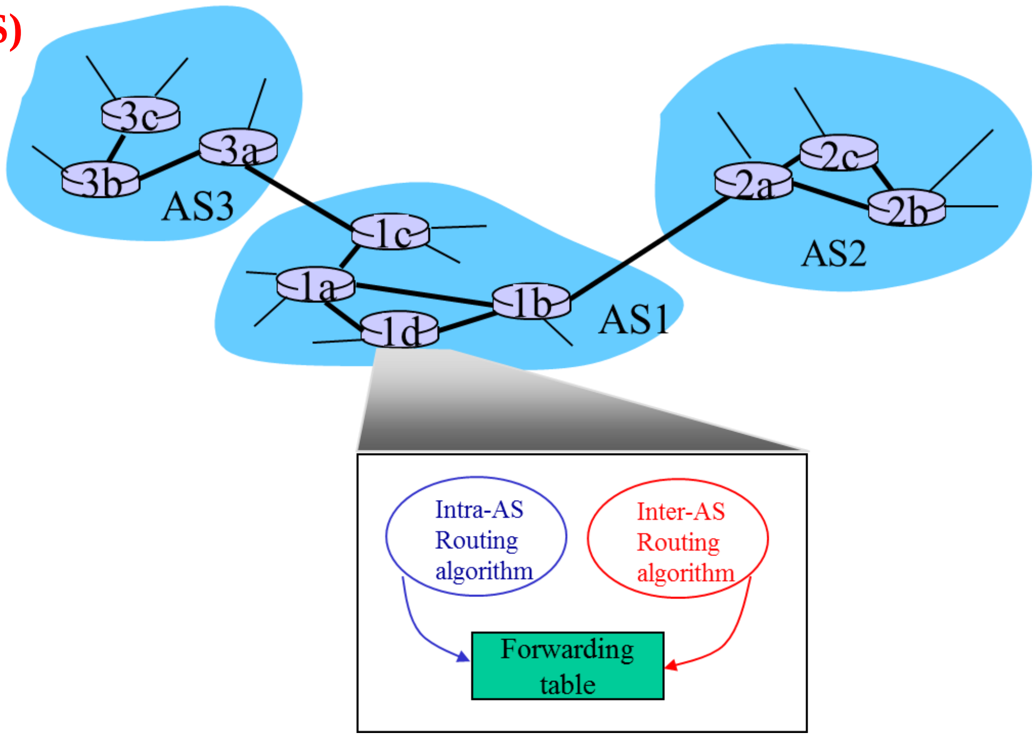


- Solution
  - **Poisoned Reverse**
    - Solve "*Bad news travel slow*"
      - How it works?
        - Since Z route through Y to get to X
          - Z tell Y the distance to X is infinite $D_z(x) = \infty$
            - So, Y do not route to X via Z
      - t_0: y detect link-cost change from 4 to 60
        - Since $D_y(x) > c(y, z) + D_z(x)$
          - $D_y(x) = 60$; $c(y, z) + D_z(x) = 1 + 5 = 6$
          - y update its own $D_y(x) = 6$
      - t_1: z receive $D_y(x) = 6$ from y
        - Using poisoned reverse, z tell lie that $D_z(x) = \infty$
        - z route to x directly and update its $D_z(x) = 50$
      - t_2: y receive $D_z(x) = \infty$ from z, so y update $D_y(x) = 60$
      - t_3: z receive $D_y(x) = 60$ from y, so z update $D_z(x) = 50$
      - t_4: y receive $D_z(x) = 50$ from y, so y update $D_y(x) = 51$



5.3 Hierarchical Routing

- Group routers into **Autonomous Systems (AS)**
  - Advantage
    - Scalability
      - Reduce routing info as the number of routers increase
    - Administrative autonomy
      - A single company can own and manage AS
  - Figure show three AS
    - AS1, AS2, AS3
  - Figure show AS1 has four router
    - 1a, 1b, 1c, 1d

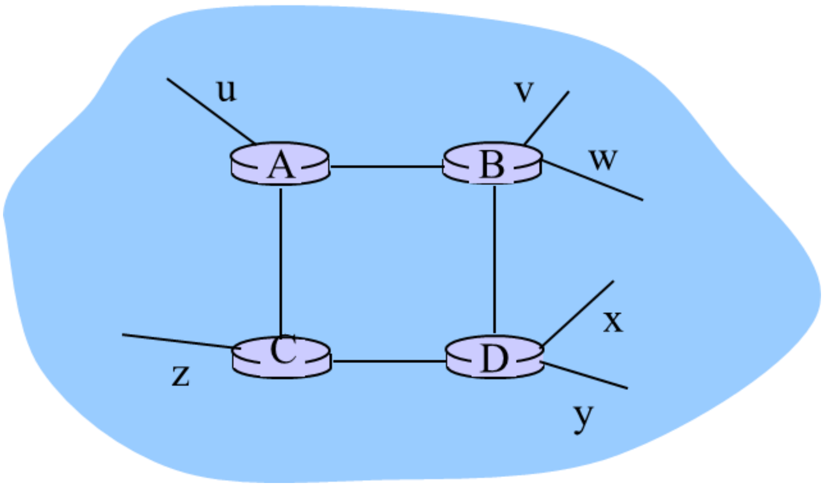


- Hierarchical Routing
  - Two types
    - **Intra-AS routing**
      - Forward packet within AS
      - Example
        - Routing Information Protocol (RIP)
    - **Inter-AS routing**
      - Forward packet between AS
      - Example
        - Border Gateway Protocol (BGP)

- **Gateway router**
  - Characteristic
    - Located at the edge of AS
    - Has link to router in the other AS
  - Used by inter-AS routing to forward packet between AS
  - Figure show three gateway router
    - 1b, 1c, 2a, 3a

5.3 Routing Information Protocol (RIP)

- RIP
  - Distance vector algorithm
    - $D_x(y)$: Distance vector
      - Cost of the least-cost path from node x and node y
      - $D_x(y)$ = Number of hops
      - Maximum $D_x(y)$ is 15 hops
    - $c(x,x')$: Link cost
      - $c(x,x') = 1$
  - Node exchange **RIP Response Message** with neighbors every 30s
    - If no RIP Response Message heard after 180s, it indicates a link failure
  - Route Recovery
    - New RIP Response Message sent to neighbour if routing table changes, so link failure is propagated to entire network quickly
    - Poison reverse prevent loops
- Note:
  - In the Figure
    - Node indicate router
      - Router A, B, C, D
    - Link indicate subnet
      - Subnet u, v, w, x, y, z

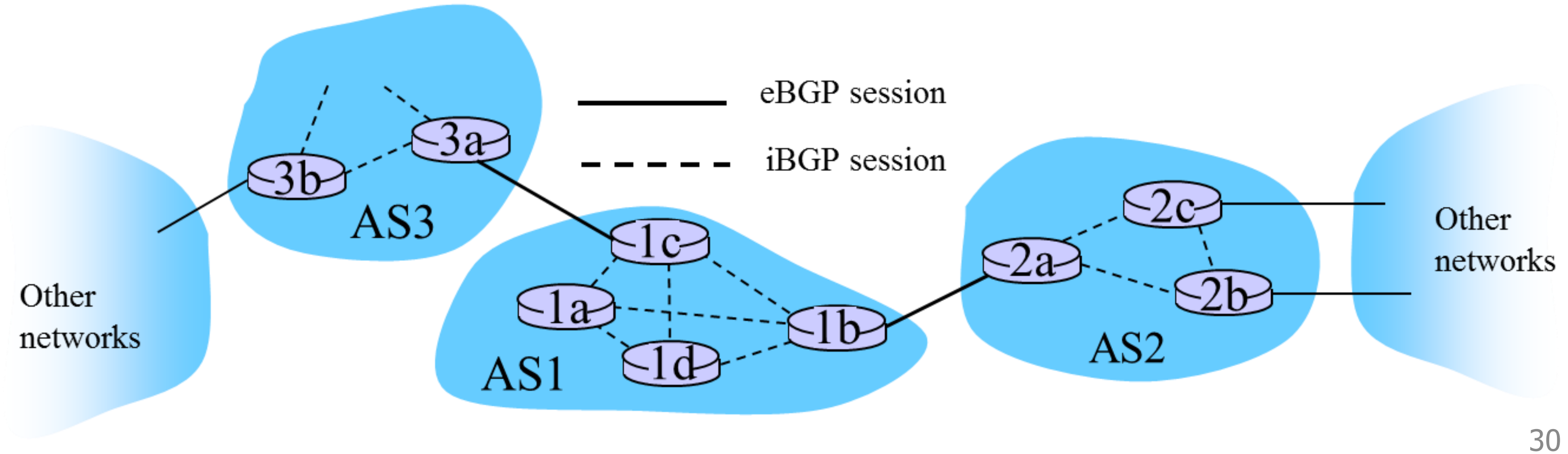


From router A to destination *subnets:*

subnet	hops
u	1
v	2
w	2
x	3
y	3
z	2

5.4 Border Gateway Protocol (BGP)

- **eBGP**
  - Send subnet reachability info to neighboring AS
  - Enable subnet to advertise its existence to the rest of the internet: “I am here”
- **iBGP**
  - Distribute subnet reachability information to all routers within an AS.
  - Determine least-cost path to other networks based on subnet reachability information and company policy.
    - Subnet reachability information: (see next slide)
    - Company policy:
      - E.g.: An AS charges high rate, so choose another path
- All routers throughout the network always learn new prefix and create new entry for the prefix in their respective forwarding table



5.4 Border Gateway Protocol (BGP)

- eBGP and iBGP service
  - **Step 1:** Router 3a use eBGP session between 3a and 1c to send subnet reachability info to AS1
  - **Step 2:** Router 1c use iBGP session to distribute prefix to all routers within AS1
  - **Step 3:** Router 1b use eBGP session between 1b and 2a to re-advertise subnet reachability info to AS2

- Subnet reachability info contain
  - **Prefix**
    - Example: AS3 can reach subnets
      - 128.16.64/24
      - 128.16.65/24
      - 128.16.66/24

      Then, AS3 send prefix 128.16.64/22 to AS1
  - **AS-PATH**
    - Contain AS through which prefix has passed
    - Example: Router 1b send AS-PATH = {AS3 AS1} to router 2a
  - **NEXT-HOP**
    - Contain IP address of a router's interface within an AS which prefix has passed
    - Example: Router 3a send NEXT-HOP = {IP address of router 3a's interface} to router 1c