Network Layer

Dr. A Krishna Chaitanya,
Indian Institute of Information Technology Sri City

- Functionalities
  - forwarding
  - routing
  - connection setup
- Services
  - guaranteed delivery
  - guaranteed delivery with bounded delay
  - in-order packet delivery
  - guaranteed maximum jitter

Virtual-Circuit and Datagram Networks

- Virtual-Circuit: provides a connection-oriented service
  - a path
  - VC numbers
  - entries in the forwarding table corresponding to each VC
- Datagram Networks: connectionless service
  - routers forwards packets based on destination address range or following prefix matching rule

Inside a Router

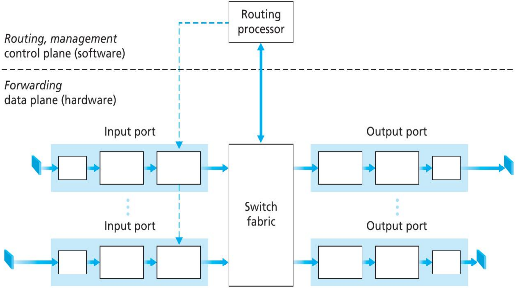


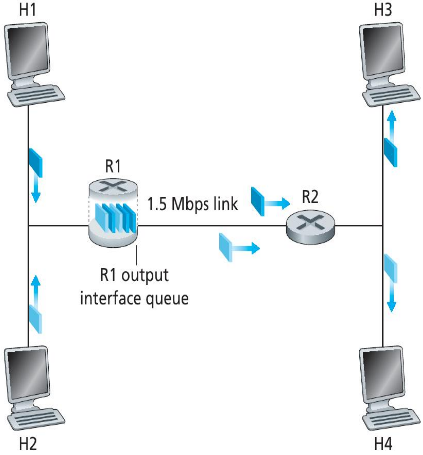
Figure 4.6 ♦ Router architecture

- Switching via memory
- Switching via bus
- Switching via interconnection of network

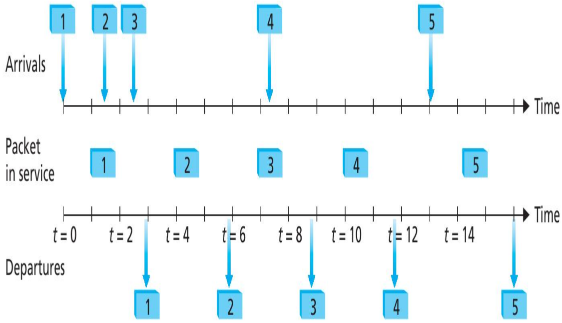
- Scheduling
  - FIFO
  - Priority Queue
  - Weighted Fair Queuing (WFQ)
- Policing
  - Leaky bucket

FIFO or FCFS

.

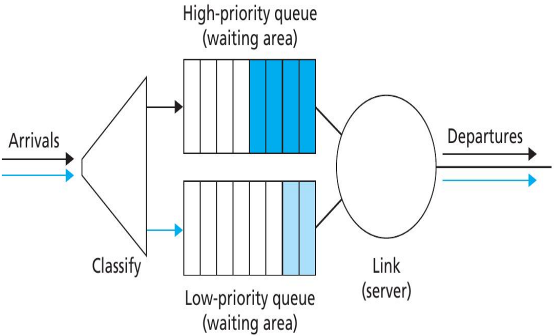


FIFO

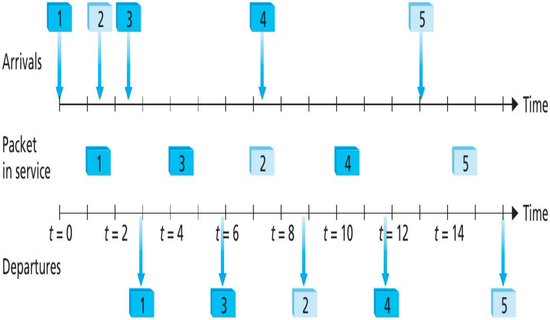


Arrivals
1
2
3
4
5
Time
Packet
in service
1
2
3
4
5
Time
t = 0
t = 2
t = 4
t = 6
t = 8
t = 10
t = 12
t = 14
Departures
1
2
3
4
5

Priority Queuing

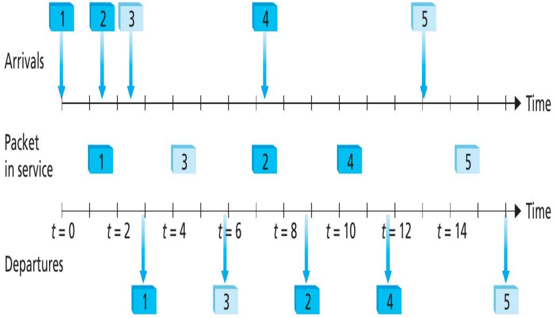


Priority Queuing



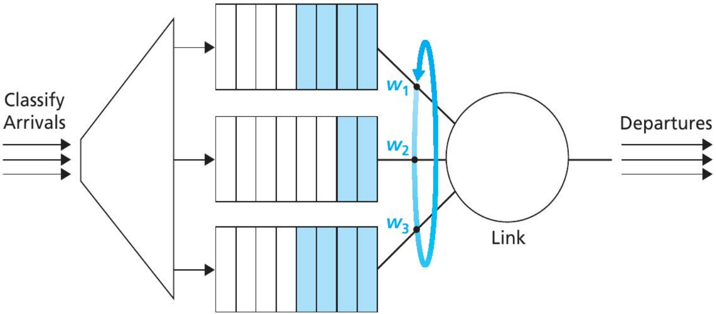
Round Robin

- Round robin queuing discipline
- No strict priority, schedule different queues in a round robin manner.
- Work-conserving round robin discipline



Robin.jpeg

Weighted Fair Queuing



- Bandwidth - R packets per second
- Class i will get a fraction of BW eqaul to $\frac{w_i}{\sum_j w_j}$

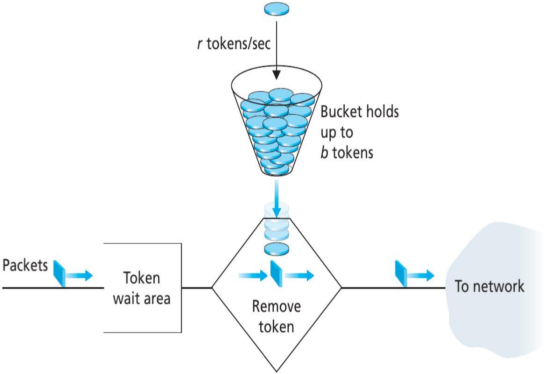
Policing: Leaky Bucket

- Restrictons:
  - average rate
  - peak rate
  - burst size

Policing: Leaky Bucket

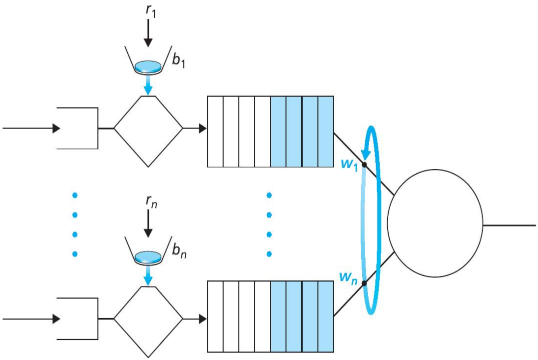
- Restrictons:
  - average rate
  - peak rate
  - burst size
- Leaky bucket:
  - a leaky bucket contains a maximum of b tokens
  - tokens are added to the bucket at rate r tokens per second
  - To transmit a packet, first remove token from the bucket and then transmit.

Leaky Bucket



- Maximum number of packets in an interval of t seconds: $rt + b$.

Leaky Bucket with WFQ



Maximum delay

- Consider flow 1: Its BW is $R\frac{w_1}{\sum w_j}$

Maximum delay

- Consider flow 1: Its BW is $R\frac{w_1}{\sum w_j}$
- A burst of b_1 packets have arrived.

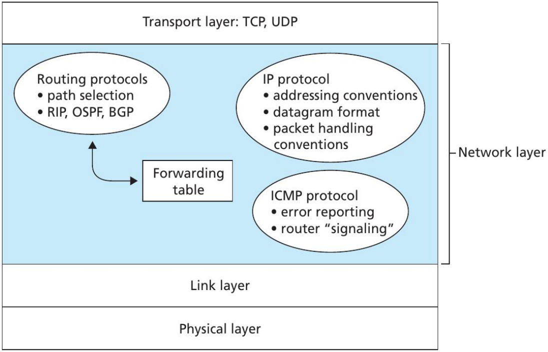
Maximum delay

- Consider flow 1: Its BW is $R\frac{w_1}{\sum w_j}$
- A burst of b_1 packets have arrived.
- Last packet will be served with in d_{max}

Maximum delay

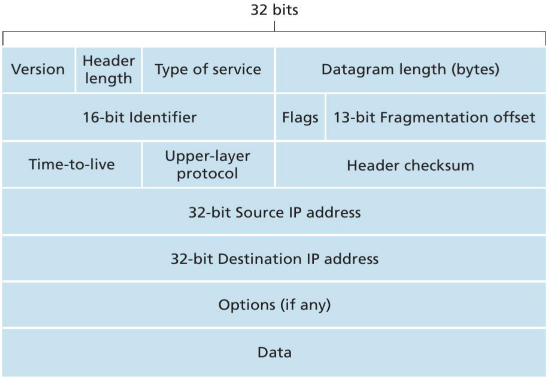
- Consider flow 1: Its BW is $R\frac{w_1}{\sum w_j}$
- A burst of b_1 packets have arrived.
- Last packet will be served with in d_{max}
- $d_{max} = \frac{b_1}{R\frac{w_1}{\sum w_j}}$

Internet Protocol



- IPv4 and IPv6

IPv4 Datagram



IPv4 Datagram

- Header checksum
  - needs to be computed at every router
  - TCP already has checksum, why do datagrams need checksum?

IPv4 Datagram

- Header checksum
  - needs to be computed at every router
  - TCP already has checksum, why do datagrams need checksum?
- Options
  - used for testing, debugging and security.
  - not used in general
  - Packet header is variable based on options used

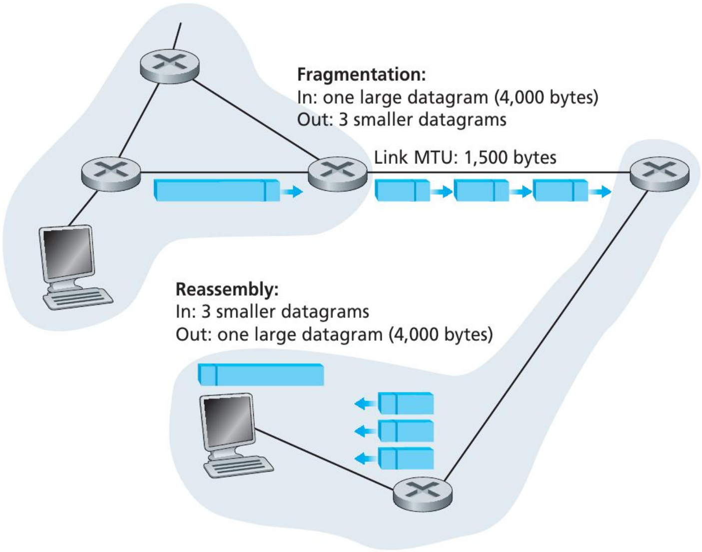
IPv4 Datagram

- Header checksum
    - needs to be computed at every router
    - TCP already has checksum, why do datagrams need checksum?
- Options
    - used for testing, debugging and security.
    - not used in general
    - Packet header is variable based on options used
- Data: Typically TCP/UDP segment.

Fragmentation

- Datagrams are often larger than MTU
- Fragmentation: Splitting a larger datagram into smaller frames suitable for transmission
- Link-layer in the end system reassembles the fragments and forwards to network layer.
- Datagram fields:
  - identification: all fragments of a datagram have same identification number.
  - flag: indicates whether it is last fragment or not
  - fragmentation offset: specifies the location of the fragment in the datagram

Fragmentation



Fragmentation

Fragment	Bytes	ID	Offset	Flag
1st fragment	1,480 bytes in the data field of the IP datagram	identification = 777	offset = 0 (meaning the data should be inserted beginning at byte 0)	flag = 1 (meaning there is more)
2nd fragment	1,480 bytes of data	identification = 777	offset = 185 (meaning the data should be inserted beginning at byte 1,480. Note that $185 \cdot 8 = 1{,}480$)	flag = 1 (meaning there is more)
3rd fragment	1,020 bytes (= 3,980–1,480–1,480) of data	identification = 777	offset = 370 (meaning the data should be inserted beginning at byte 2,960. Note that $370 \cdot 8 = 2{,}960$)	flag = 0 (meaning this is the last fragment)

- Who should be IP addressed?

IPv4 Addressing

- Who should be IP addressed?
  - Hosts?

IPv4 Addressing

- Who should be IP addressed?
  - Hosts?
  - Routers?

IPv4 Addressing

- Who should be IP addressed?
  - Hosts?
  - Routers?
- Hosts are connected to internet through a link.
- Interface: The boundary between host and physical link.

IPv4 Addressing

- Who should be IP addressed?
  - Hosts?
  - Routers?
- Hosts are connected to internet through a link.
- Interface: The boundary between host and physical link.
- It is the interface that will be IP addressed.

IP Address

- IP address is 32-bit long
- It is represented in dotted-decimal notation. Example, the address

  11000001 00100000 11011000 00001001

  will be represented by 193.32.216.9
- About 4 billion addresses available
- Who assigns IP addresses?

IP Address

- IP address is 32-bit long
- It is represented in dotted-decimal notation. Example, the address

  11000001 00100000 11011000 00001001

  will be represented by 193.32.216.9
- About 4 billion addresses available
- Who assigns IP addresses?
- International Corporation for Assigned Names and Numbers (ICANN)

IP Address

- IP address is 32-bit long
- It is represented in dotted-decimal notation. Example, the address

  11000001 00100000 11011000 00001001

  will be represented by 193.32.216.9
- About 4 billion addresses available
- Who assigns IP addresses?
- International Corporation for Assigned Names and Numbers (ICANN)
- How to assign IP addresses?

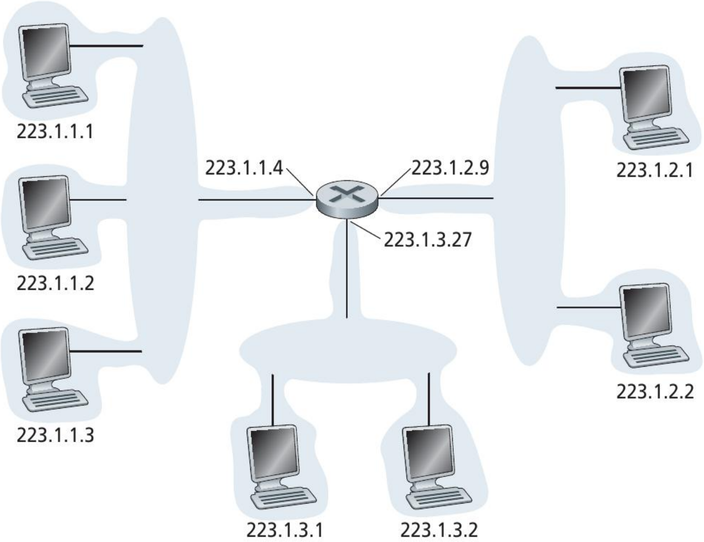
IP Address

- IP address is 32-bit long
- It is represented in dotted-decimal notation. Example, the address

  11000001 00100000 11011000 00001001

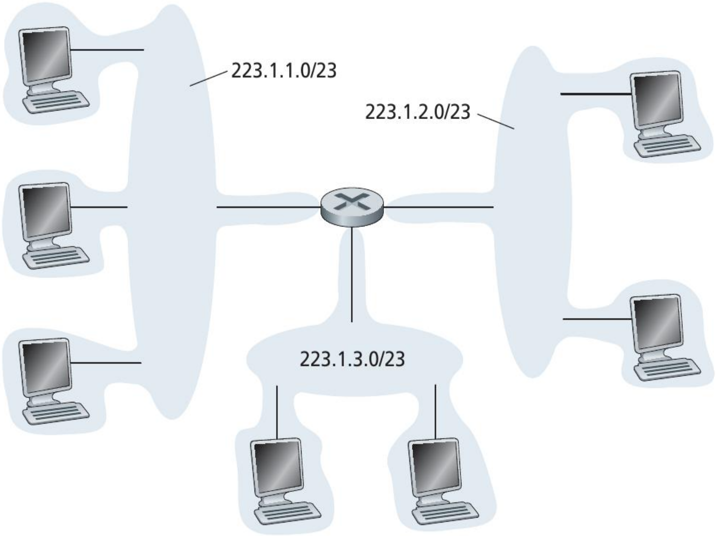
  will be represented by 193.32.216.9
- About 4 billion addresses available
- Who assigns IP addresses?
- International Corporation for Assigned Names and Numbers (ICANN)
- How to assign IP addresses?
- Subnet: Detach each interface from its host or router, creating islands of isolated networks. Each of these isolated networks is called a subnet.

Subnet



223.1.1.1
223.1.1.2
223.1.1.3
223.1.1.4
223.1.2.9
223.1.3.27
223.1.2.1
223.1.2.2
223.1.3.1
223.1.3.2

Subnet



IP Addressing

- The internet's addressing strategy is known as Classless Interdomain Routing (CIDR)
- IP broadcast address: 255.255.255.255
- Classful addressing:
  - Class A: a.b.c.d/8
  - Class B: a.b.c.d/16
  - Class C: a.b.c.d/24
- CIDR: a.b.c.d/x

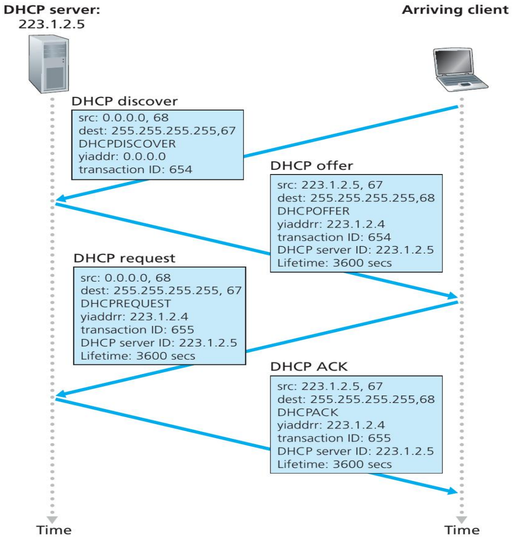
Obtaining a Block of Addresses

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

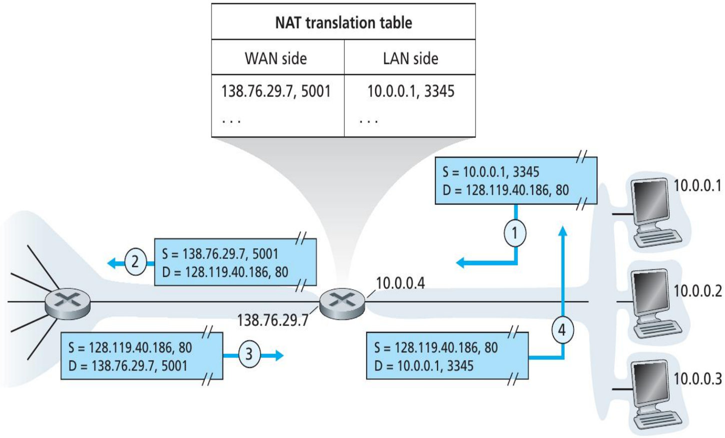
Dynamic Host Configuration Protocol

- Allows hosts to obtain IP address automatically
- Also known as plug and play protocol
- Client-server protocol
- Each server may have DHCP server. If a subnet does not have DHCP server, it will have DHCP relay agent that knows the address of DHCP server.

DHCP


DHCP server:
223.1.2.5
Arriving client
DHCP discover
src: 0.0.0.0, 68
dest: 255.255.255.255,67
DHCPDISCOVER
yiaddr: 0.0.0.0
transaction ID: 654
DHCP offer
src: 223.1.2.5, 67
dest: 255.255.255.255,68
DHCPOFFER
yiaddrr: 223.1.2.4
transaction ID: 654
DHCP server ID: 223.1.2.5
Lifetime: 3600 secs
DHCP request
src: 0.0.0.0, 68
dest: 255.255.255.255, 67
DHCPREQUEST
yiaddrr: 223.1.2.4
transaction ID: 655
DHCP server ID: 223.1.2.5
Lifetime: 3600 secs
DHCP ACK
src: 223.1.2.5, 67
dest: 255.255.255.255,68
DHCPACK
yiaddrr: 223.1.2.4
transaction ID: 655
DHCP server ID: 223.1.2.5
Lifetime: 3600 secs
Time
Time

Network Address Translation (NAT)



Internet Control Message Protocol (ICMP)

- Typically used for error reporting. Example: "Destination network unreachable"
- Can be used for congestion control
- ICMP messages have:
  - type and code field
  - header and the first 8 bytes of IP datagram that caused the ICMP message

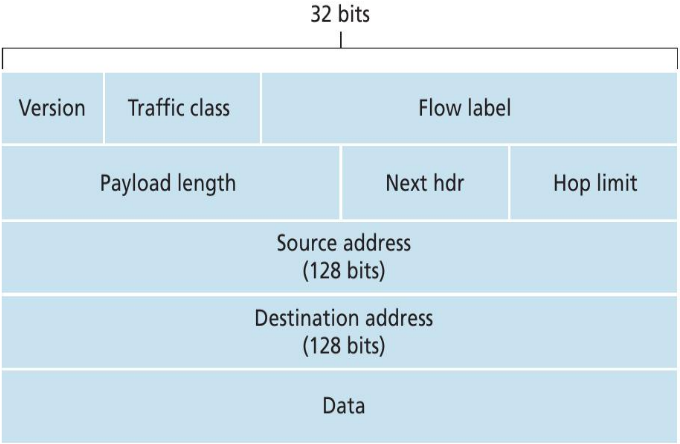
ICMP

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

IPv6

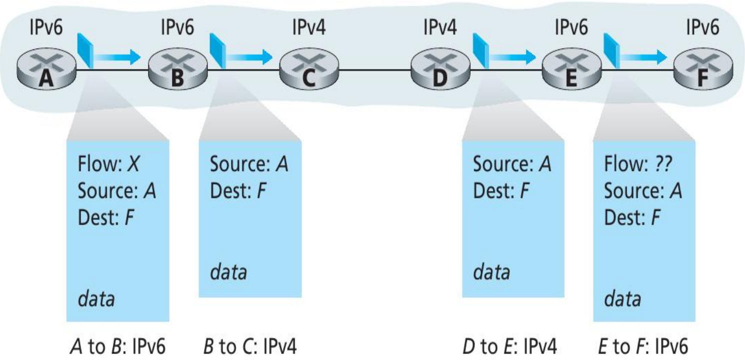
- Internet Engineering Task Force (IETF) developed IPv6
- Expanded addressing capabilities: 128 bits
- A streamlined 40-byte header: fixed length header
- Flow labeling and priority:
  - labeling of packets belonging to particular flows
  - ICMP packets can be given high priority than IP datagrams.
- No fragmentation and reassembly at router
- No checksum computation, and no options field.

IPv6 Datagram

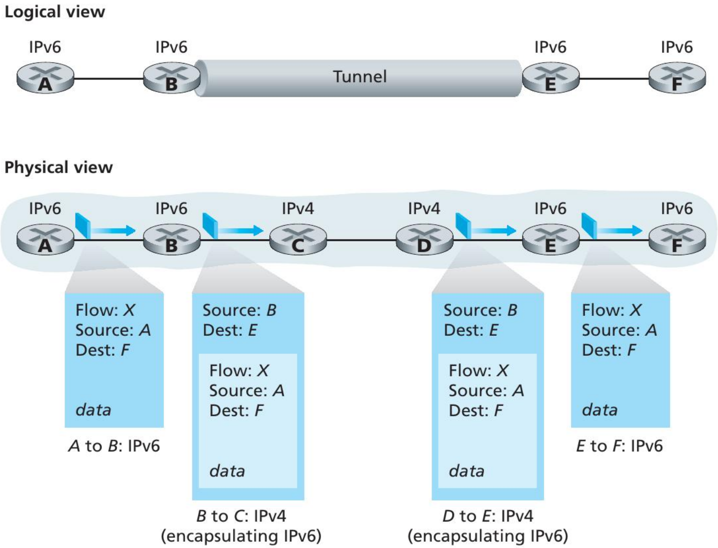


- IPv4 to IPv6
  - Dual-Stack approach
  - Tunneling

Dual-Stack Approach



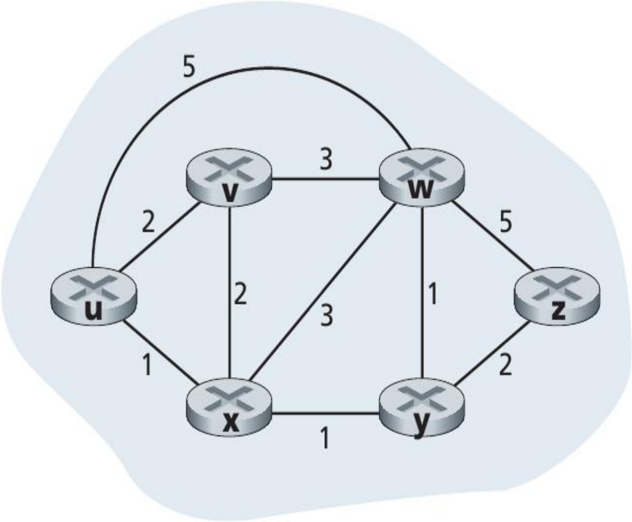
Tunneling



Logical view
IPv6
IPv6
IPv6
IPv6
A
B
Tunnel
E
F
Physical view
IPv6
IPv6
IPv4
IPv4
IPv6
IPv6
A
B
C
D
E
F
Flow: X
Source: A
Dest: F
data
A to B: IPv6
Source: B
Dest: E
Flow: X
Source: A
Dest: F
data
B to C: IPv4
(encapsulating IPv6)
Source: B
Dest: E
Flow: X
Source: A
Dest: F
data
D to E: IPv4
(encapsulating IPv6)
Flow: X
Source: A
Dest: F
data
E to F: IPv6

Routing

Notation



Notation

- We represent a network by an undirected graph $G = (N, E)$
- N is the set of Nodes (routers)
- E is the set of edges connecting nodes (links)
- $c(x, y)$ is the cost of the edge between x and y.
- Cost of a path $(x_1, \ldots, x_p)$ is sum of costs of edges along the path: $c(x_1, x_2) + \cdots + c(x_{p-1}, x_p)$
- We aim to find paths with least cost.

Classification

- Global vs Decentralized:
  - Global routing algorithm: requires global information about links and costs at every router. Also known as Link-State algorithm
  - Decentralized routing algorithm: no node has complete information
- Static vs Dynamic routing
- Load-sensitive vs Load-insensitive routing

Link-State Routing Algorithm

- We study Dijkstra's algorithm
- $D(v)$: cost of the least cost path from source to destination v as of this iteration
- $p(v)$: previous node along the current least cost path from the source to v
- N' : subset of N. If $v \in N$, then least cost path to v from source is definitely known.

LS Algorithm

```
Initialization:
  N' = {u}
  for all nodes v
    if v is a neighbor of u
      then D(v) = c(u,v)
    else D(v) = ∞

Loop
  find w not in N' such that D(w) is a minimum
  add w to N'
  update D(v) for each neighbor v of 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
   least path cost to w plus cost from w to v */
until N'= N
```

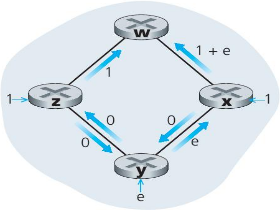
LS Routing Algorithm: Example

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					

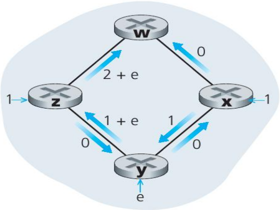
LS Routing: Pathology

- Congestion-sensitive routing
- Link-costs are equal to the load carried on the link.
- Link costs are not symmetric: $c(u, v) \neq c(v, u)$
- $c(u, v) = c(v, u)$, if load on the link in both directions is same

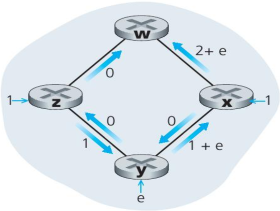
LS Routing: Pathology



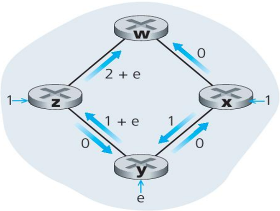
a. Initial routing



b. *x*, *y* detect better path to *w*, clockwise



c. *x*, *y*, *z* detect better path to *w*, counterclockwise



d. *x*, *y*, *z*, detect better path to *w*, clockwise

Distance-Vector (DV) Routing Algorithm

- Decentralized,asynchronous
- Iterative process
- $d_x(y)$ denotes cost of least cost path from x to y
- Bellman-Ford equation

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

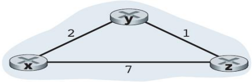
  v is a neighbor of x.
- Each node x maintains the following routing information:
  - For each neighbor v, the cost $c(x,v)$
  - Node x's distance vector, $\mathbf{D}_x = [D_x(y) : y \in N]$
  - Distance vectors of each of its neighbors $\mathbf{D}_v$

DV Algorithm

At each node, *x*:

```
Initialization:
   for all destinations y in N:
      D_x(y) = c(x,y)   /* if y is not a neighbor then c(x,y) = ∞ */
   for each neighbor w
      D_w(y) = ? for all destinations y in N
   for each neighbor w
      send distance vector D_x = [D_x(y): y in N] to w

loop
   wait (until I see a link cost change to some neighbor w or
         until I receive a distance vector from some neighbor w)

   for each y in N:
      D_x(y) = min_v{c(x,v) + D_v(y)}

   if D_x(y) changed for any destination y
      send distance vector D_x = [D_x(y): y in N] to all neighbors

forever
```

DV Example



Node x table

from	x	y	z
x	0	2	7
y	∞	∞	∞
z	∞	∞	∞

from	x	y	z
x	0	2	3
y	2	0	1
z	7	1	0

from	x	y	z
x	0	2	3
y	2	0	1
z	3	1	0

Node y table

from	x	y	z
x	∞	∞	∞
y	2	0	1
z	∞	∞	∞

from	x	y	z
x	0	2	7
y	2	0	1
z	7	1	0

from	x	y	z
x	0	2	3
y	2	0	1
z	3	1	0

Node z table

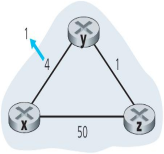
from	x	y	z
x	∞	∞	∞
y	∞	∞	∞
z	7	1	0

from	x	y	z
x	0	2	7
y	2	0	1
z	3	1	0

from	x	y	z
x	0	2	3
y	2	0	1
z	3	1	0

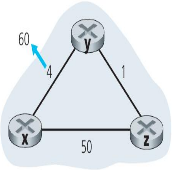
Time

DV Algorithm: Link Cost Changes



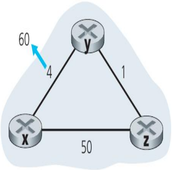
- Focus on distance tables entries of y and z to x
- At t_0, cost has changed to 1 from 4. y updates its table with $D_y(x) = 1$ and informs z
- At t_1, z receives update from y and updates its table $D_z(x) = 2$
- At t_2, y receives update from z and no changes in table.

DV Algorithm: Link Cost Changes



- Before link cost changes:
  $D_y(x) = 4, D_y(z) = 1, D_z(y) = 1, D_z(x) = 5$

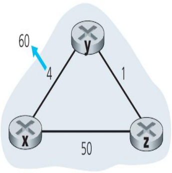
DV Algorithm: Link Cost Changes



- Before link cost changes:
  $D_y(x) = 4, D_y(z) = 1, D_z(y) = 1, D_z(x) = 5$
- At t_0, cost has changed to 60 from 4. y updates its table with

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

DV Algorithm: Link Cost Changes

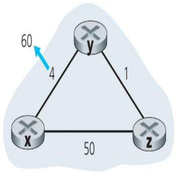


- Before link cost changes:
  $D_y(x) = 4, D_y(z) = 1, D_z(y) = 1, D_z(x) = 5$
- At t_0, cost has changed to 60 from 4. y updates its table with

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

- $D_y(x) = min\{60 + 0, 1 + 5\} = 6$

DV Algorithm: Link Cost Changes

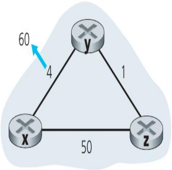


- Before link cost changes:
  $D_y(x) = 4, D_y(z) = 1, D_z(y) = 1, D_z(x) = 5$
- At t_0, cost has changed to 60 from 4. y updates its table with

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

- $D_y(x) = min\{60 + 0, 1 + 5\} = 6$
- At t_1, z receives update from y and updates its table
  $D_z(x) = min\{50 + 0, 1 + 6\} = 7$

DV Algorithm: Link Cost Changes

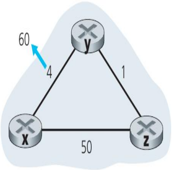


- Before link cost changes:
  $D_y(x) = 4, D_y(z) = 1, D_z(y) = 1, D_z(x) = 5$
- At t_0, cost has changed to 60 from 4. y updates its table with

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

- $D_y(x) = min\{60 + 0, 1 + 5\} = 6$
- At t_1, z receives update from y and updates its table
  $D_z(x) = min\{50 + 0, 1 + 6\} = 7$
- At t_2, y receives update from z and updates table as
  $D_y(x) = 8$. and this process repeats.

DV Algorithm: Link Cost Changes



- Before link cost changes:
  $D_y(x) = 4, D_y(z) = 1, D_z(y) = 1, D_z(x) = 5$
- At t_0, cost has changed to 60 from 4. y updates its table with

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

- $D_y(x) = min\{60 + 0, 1 + 5\} = 6$
- At t_1, z receives update from y and updates its table
  $D_z(x) = min\{50 + 0, 1 + 6\} = 7$
- At t_2, y receives update from z and updates table as
  $D_y(x) = 8$. and this process repeats.
- Count-to-infinity!

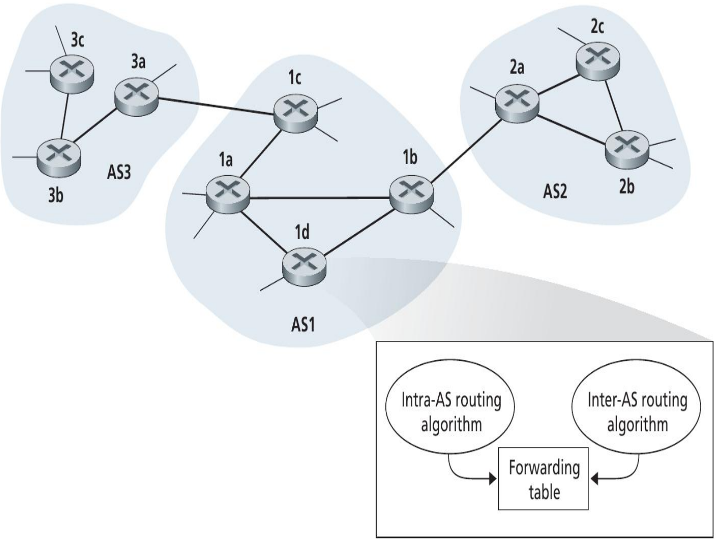
Poisoned Reverse

- If z routes through y, z will inform y that $D_z(x) = \infty$
- y cannot route to x via z as there is no path!
- When $c(x, y) = 60$, y updates its table with $D_y(x) = 60$!
- After receiving an update z routes to x via direct path and updates its table with $D_z(x) = 50$
- After receiving update from z, y recomputes route to x via z and informs z with $D_y(x) = \infty$ (infact it is 51!)

Hierarchical Routing

- Scale: number of routers in internet is very large. Which algorithm to use?
- Administrative autonomy: an organization should be able to run and administer its network as it wishes.
- These problems can be solved by organizing routers into autonomous systems (AS)
- Each AS will have a gateway router

Hierarchical routing



Routing in Internet

- Intra-AS routing
  - Routing information protocol (RIP): based on DV algorithm
  - Open shortest path first (OSPF): based on LS algorithm
- Inter-AS routing
  - Border Gateway Protocol (BGP)

Details of these algorithms (Section 4.6) are left for self study!
These are part of our CCN course