

Computer Networks

Lecture 07

Network Layer

By
Dr. Khaled ELMenshawy

Lecture

07

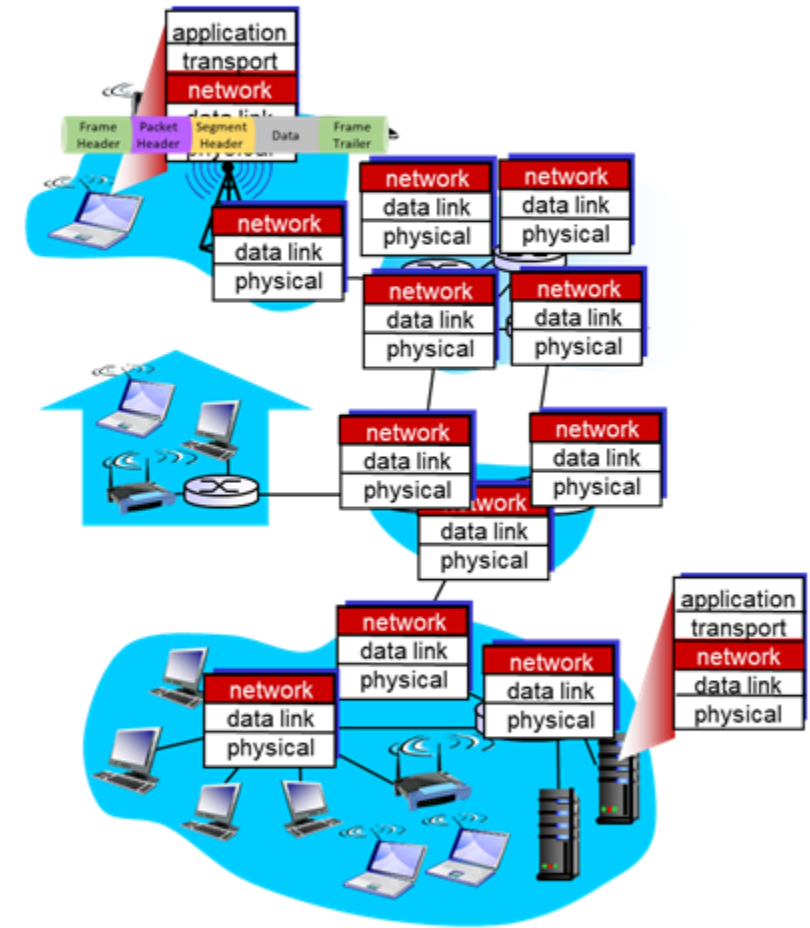
LEARNING OUTCOMES

- Understand network layer functions.
- Differentiate between the control plane and the data plane.
- Understand router architecture.
- Understand forwarding table.
- Recognize the types of switching fabrics.
- Understand the buffer management and Scheduling mechanism.
- Recognize IPv4 and IPv6 packet headers and Understand NAT and tunneling.
- Define routing algorithms.
- Differentiate between link-state and distance vector algorithms and how they operate.
- Understand Internet routing protocols (RIP, OSPF, and BGP).

Overview of Network layer

What is network layer?

- Network layer (or Internet layer) is responsible for transfer packets between different devices or networks and choosing the best data routing paths for network communication.
- Router examines header fields in all IP datagrams passing through it since network layer protocols in *every* host and router.
- On *sending side* encapsulates segments into datagrams and on *receiving side*, delivers segments to transport layer.

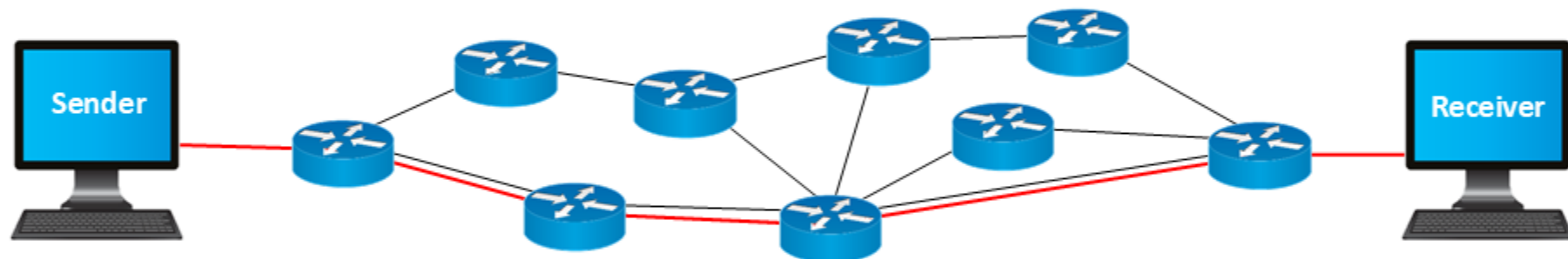


Network layer

Routing

Routing is process of planning trip from source to destination.

For example, determine route taken by packets from source to destination using routing algorithms.



Forwarding

Forwarding is the process of getting through single interchange.

For example, a router receives packets from hosts on one attached network and forwards them to hosts on another attached network or to another router for further forwarding.

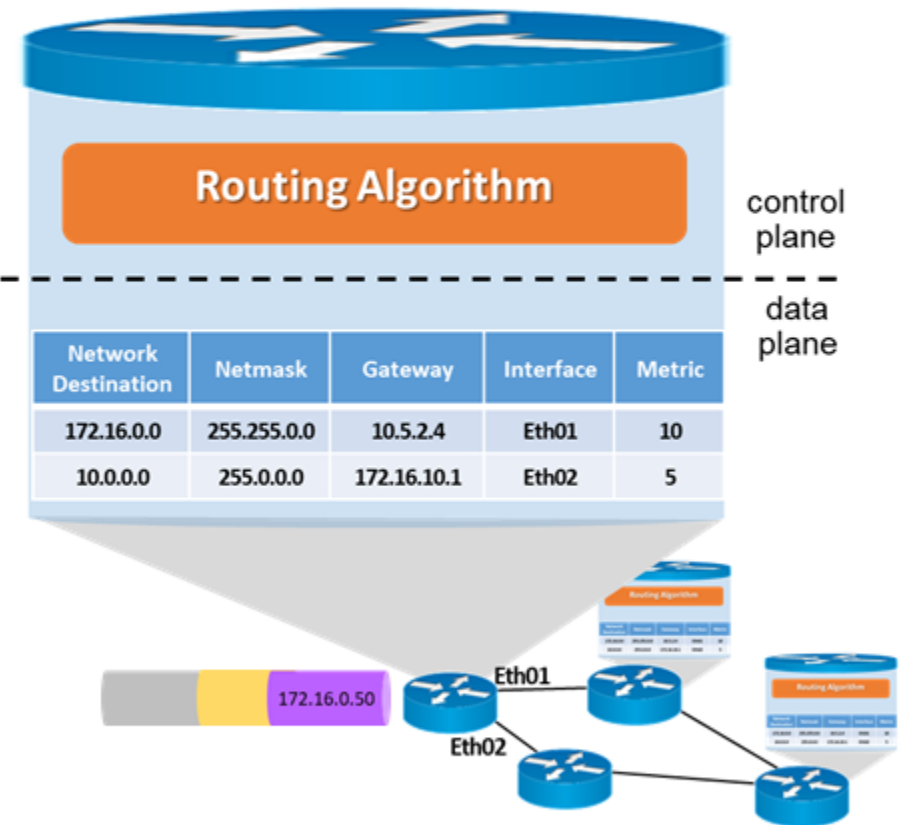
Network layer

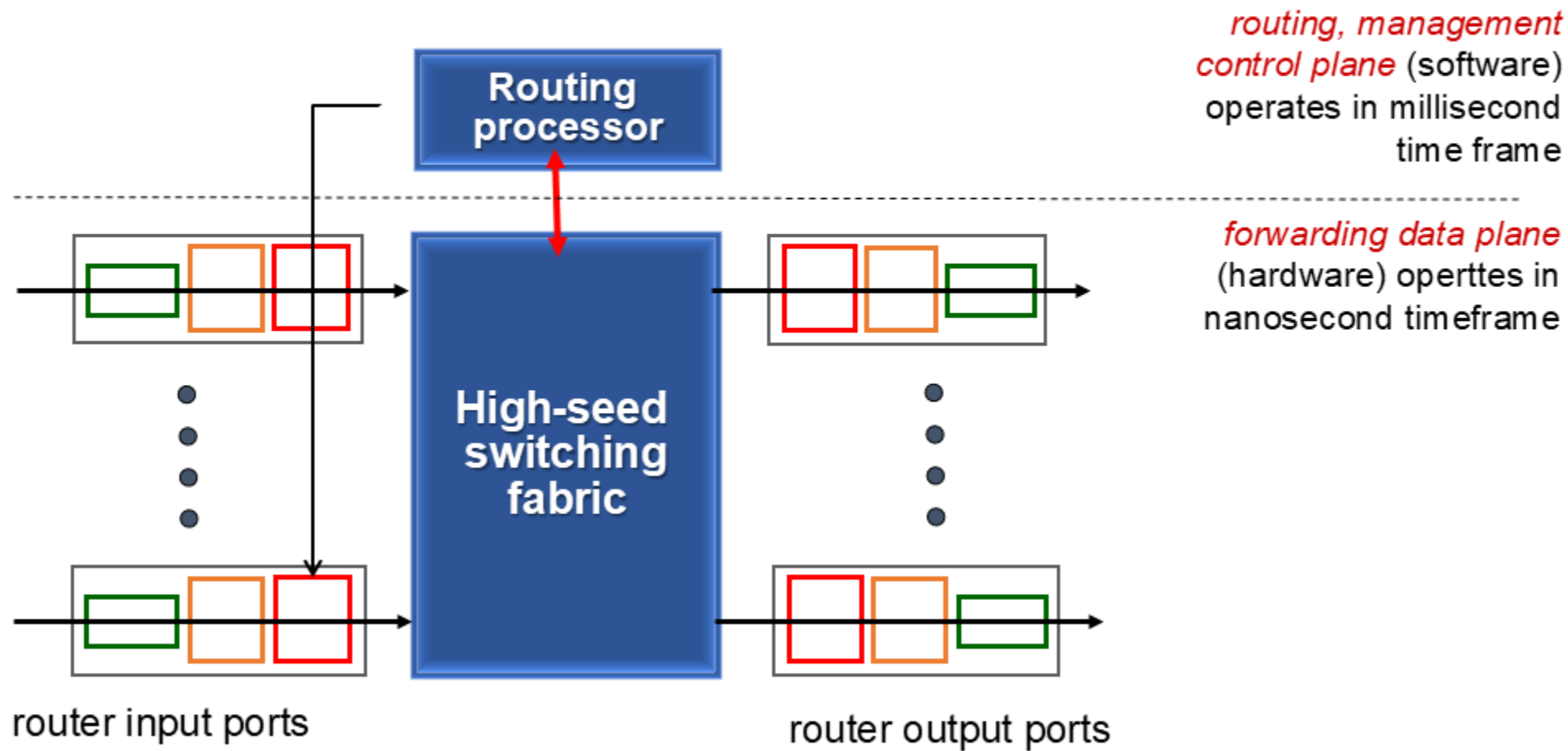
Control plane

All functions and processes (*routing algorithm*) that determine which path to use to send the packet or frame. Control plane is responsible for populating the routing table and drawing network topology. it is responsible for How packets should be forwarded.

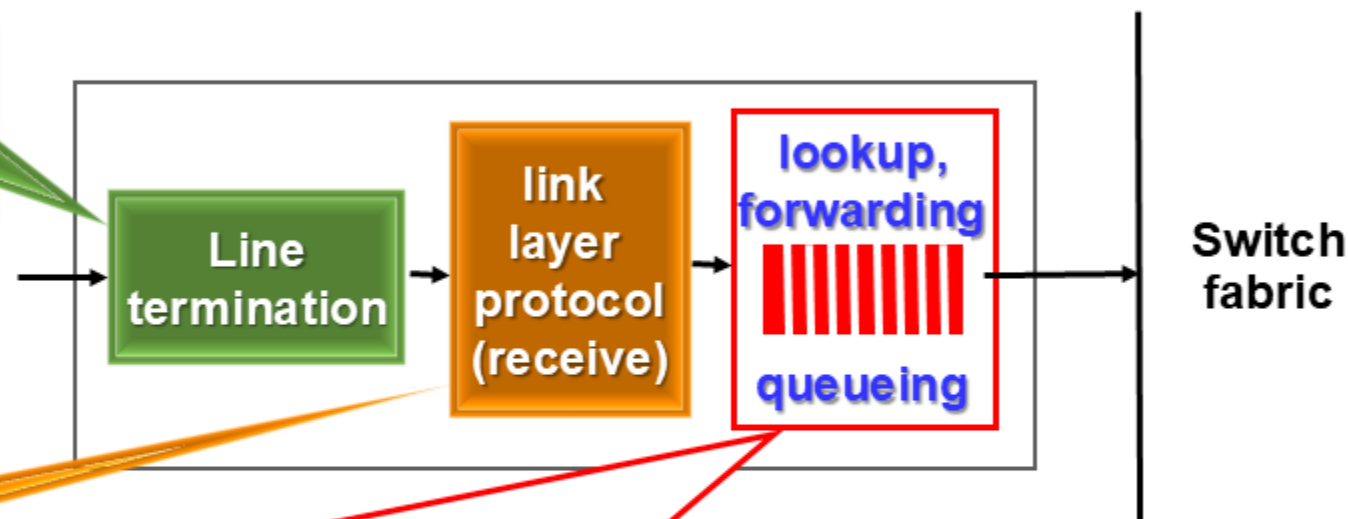
Data plane

All the functions and processes that forward packets/frames from one interface to another based on *control plane logic*.





Physical layer:
Cable (TP, Fiber,...)



Link layer:
Decapsulation

Decentralized switching:

- Using header field values, lookup output port using forwarding table in input port memory ("match plus action").
 - **Destination-based forwarding:** forward based only on destination IP address (traditional).
 - **Generalized forwarding:** forward based on any set of header field values.
- **Queueing:** if datagrams arrive faster than forwarding rate into switch fabric.

Forwarding tableDestination-based forwarding

Destination Address Range				Link interface
11001000	00010111	00010000	00000000	0
through				
11001000	00010111	00010111	11111111	1
through				
11001000	00010111	00011000	00000000	2
through				
11001000	00010111	00011001	00000000	3
through				
11001000	00010111	00011111	11111111	3
Otherwise				

Destination-based forwardingForwarding table

Destination Address Range	Link interface
11001000 00010111 00010000 00000000 through 11001000 00010111 00010111 11111111	0
11001000 00010111 00010000 00000100 through 11001000 00010111 00010000 00000111	3
11001000 00010111 00011000 00000000 through 11001000 00010111 00011000 11111111	1
11001000 00010111 00011001 00000000 through 11001000 00010111 00011111 11111111	2
Otherwise	3

Longest prefix matchingForwarding table

When looking for forwarding table entry for given destination address, use **longest address prefix** that matches destination address. Often performed using ternary content addressable memories (TCAMs)

Destination Address Range	Link interface
11001000 00010111 00010*** *****	0
11001000 00010111 00011000 *****	1
11001000 00010111 00011*** *****	2
Otherwise	3

Which interface?

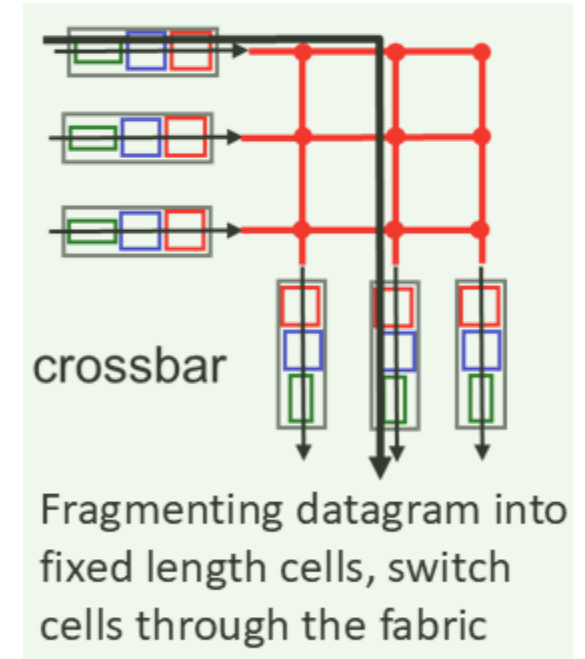
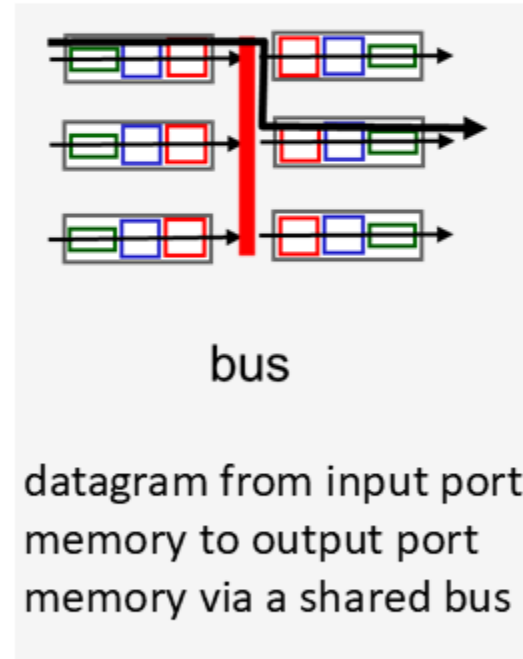
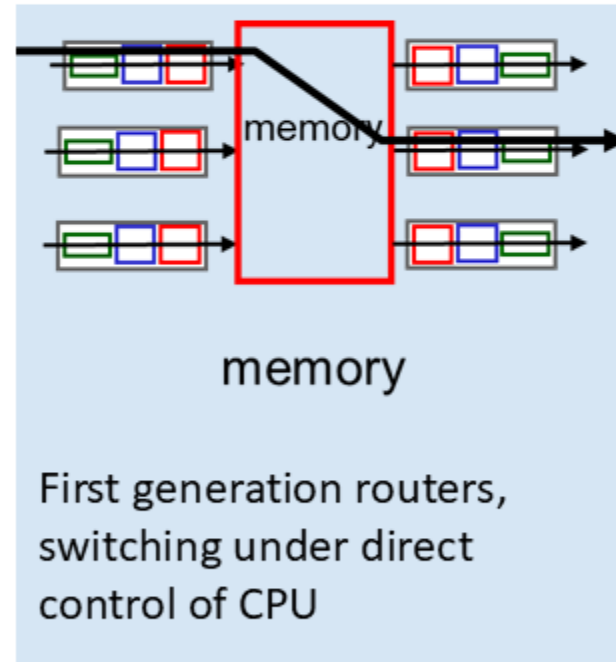
11001000 00010111 00010110 10100001 → **Interface 0**

11001000 00010111 00011000 10101010 → **Interface 1**

Switching fabrics

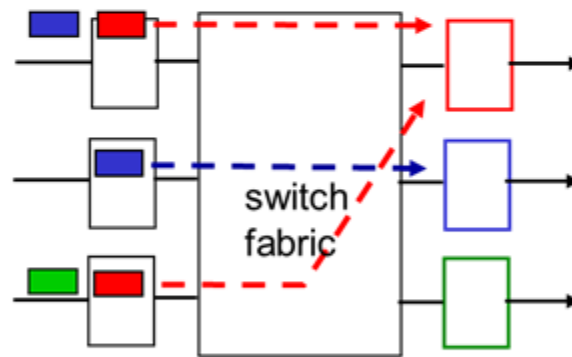
- Transfer packet from input buffer to appropriate output buffer.
- **Switching rate**: rate at which packets can be transfer from inputs to outputs.

Three types of switching fabrics

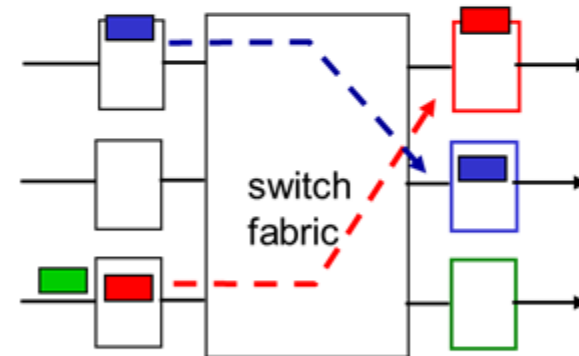


Input port queuing

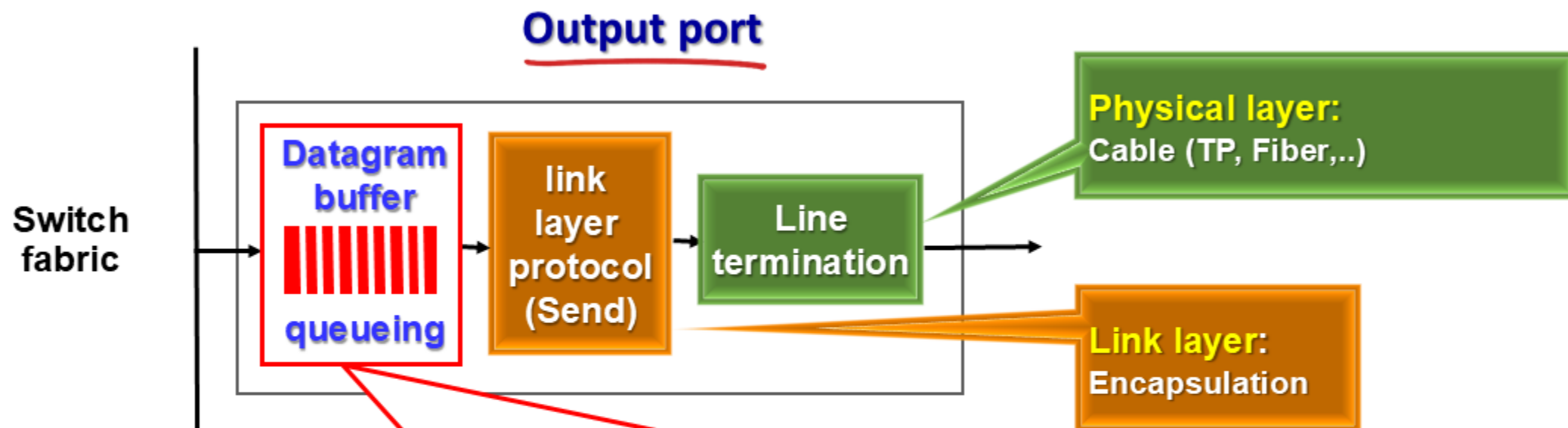
- Queueing may occur at input queues if the switch fabric is slower than input ports causing delay and loss due to input buffer overflow.
- Queued datagram at front of queue prevents others in queue from moving forward causing **Head-of-the-Line (HOL) blocking**.



only one red datagram can be transferred. lower red packet is blocked.



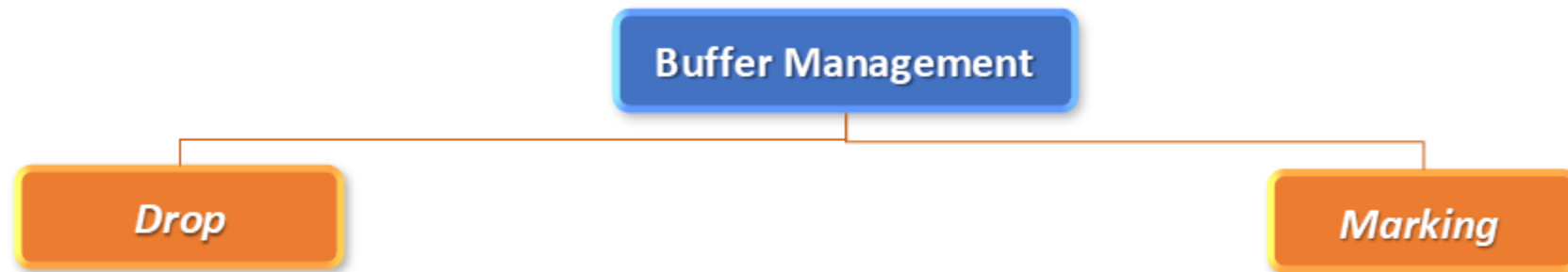
Green packet experiences HOL blocking.



- **Drop policy;** Buffering required when datagrams arrive from fabric faster than the transmission rate, datagram (packets) can be lost due to congestion, lack of buffers.
- **Scheduling mechanism;** Scheduling discipline chooses among queued datagrams for transmission, priority scheduling – who gets best performance, network neutrality.

How much buffering?

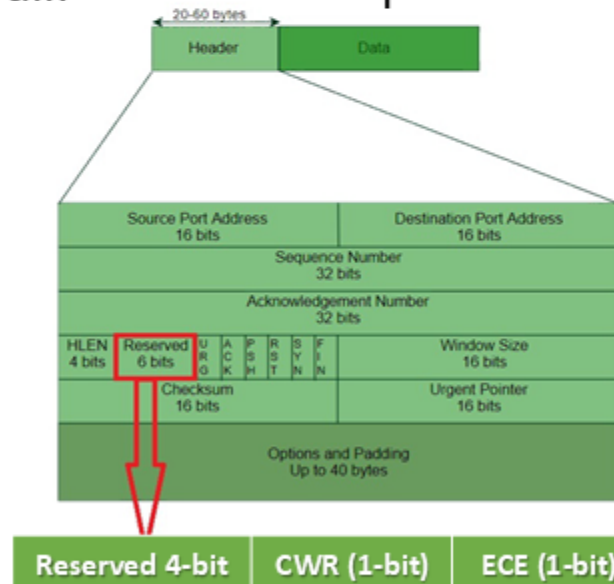
- *RFC 3439* rule of thumb: average buffering equal to “typical” RTT (e.g., 250 msec) times link capacity C
e.g., C = 10 Gbps link: buffer = $(250 \times 10^{-3}) \text{ Sec.} \times (10 \times 10^9) \text{ bps} = 2.5 \text{ Gbit buffer.}$
- More recent recommendation: with N flows (inputs), buffering equal to $\frac{RTT \cdot C}{\sqrt{N}}$
- Too much buffering can increase delays.



which packet to add, drop when buffers are full:

- **Tail drop**: drop arriving packet
- **Priority**: drop/remove on priority basis.

Which packets to mark to signal congestion (ECN)



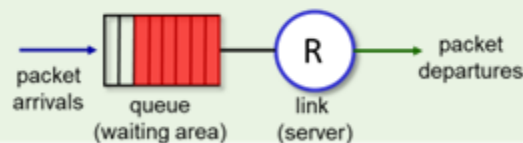
CWR	ECE	code	From	To
0	0	Non-ECN setup	any	any
0	1	ECN Echo	Rcv	Snd
1	0	cwnd reduced	Snd	Rcv
1	1	ECN Setup	Snd	Rcv

Scheduling mechanism

FCFS

First Come, First Served

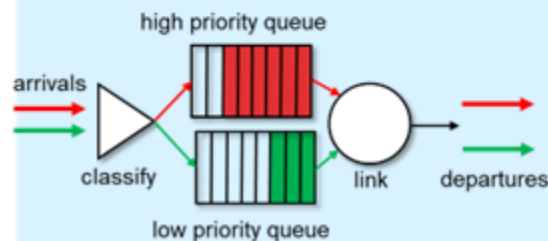
Where packets transmitted in order of arrival to output port. Also known as: First-in-first-out (FIFO)



Priority

Arriving traffic classified, according to priority, send packet from highest priority queue that has buffered packets.

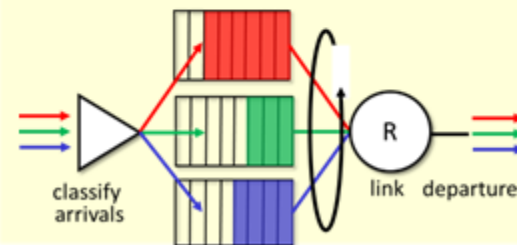
FCFS within priority class.



Round Robin

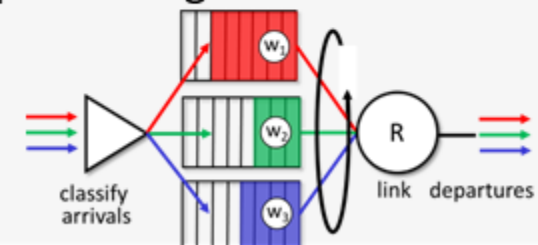
Arriving traffic classified, queued by class (in header) any header fields can be used for classification.

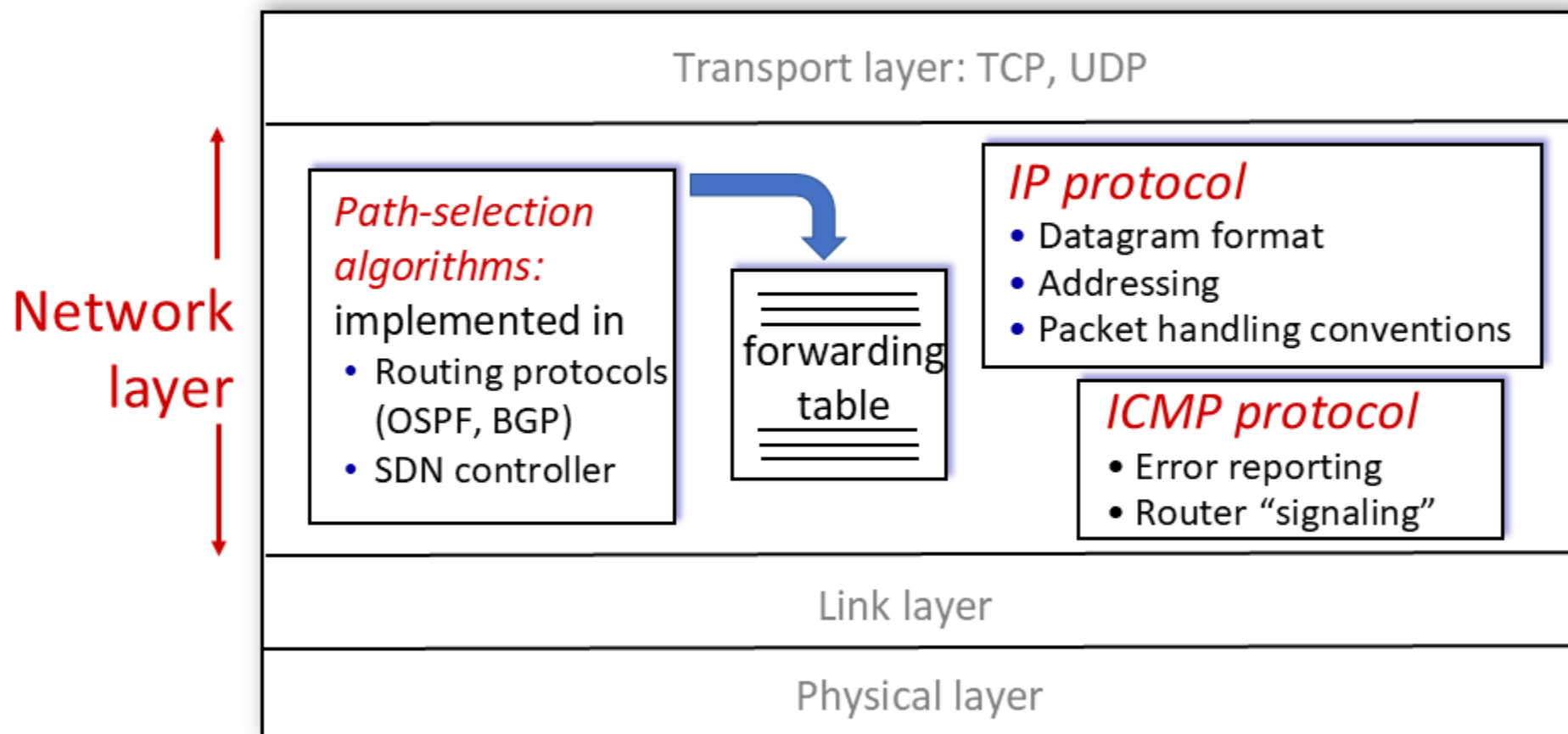
Cyclically scans class queues, sending one complete packet from each class.



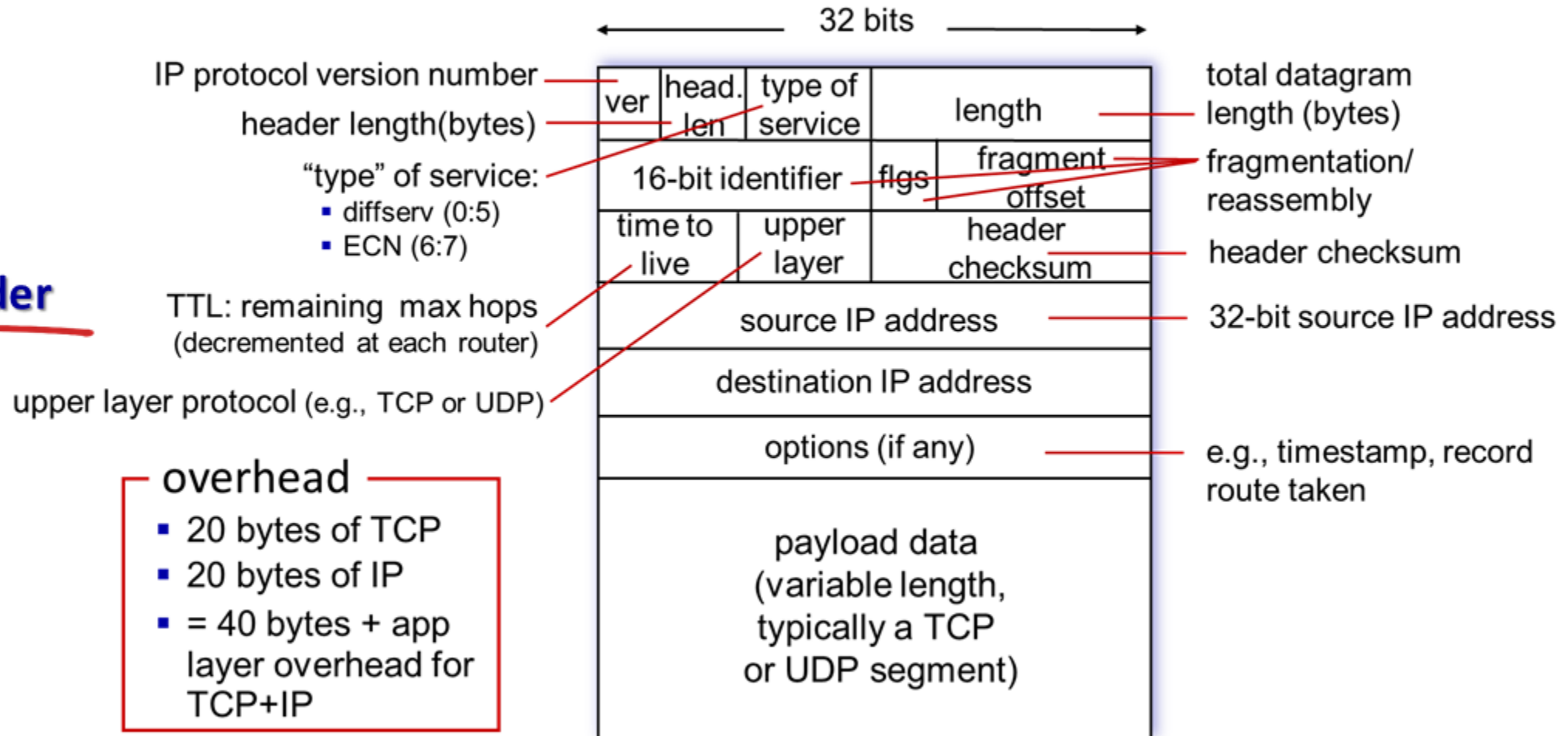
Weighted Fair Queuing WFQ

Based on Round Robin where each class (i) has weight, W_i , and gets weighted amount of service in each cycle: $W_i / \sum_i W_i$. No. of packets selected from each queue according to queue weight.



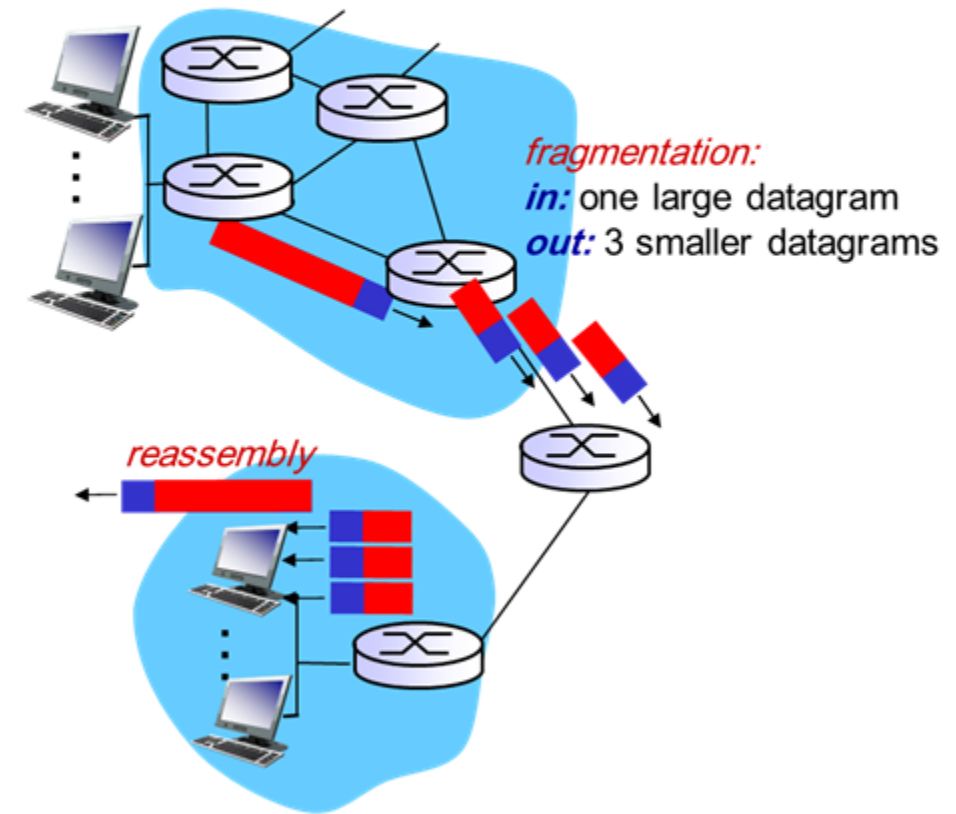


IPv4 Packet header



IP fragmentation, reassembly

- Network links have **MTU** (Max. Transfer Unit) - largest possible link-level frame.
 - Different link types, different MTUs
- Large IP datagram divided (“fragmented”) within net.
 - One datagram becomes several datagrams
 - “Reassembled” only at final destination
 - IP header bits used to identify, order related fragments.



IP fragmentation, reassemblyExample:

- 4000-byte datagram (3980 Byte for data + 20 Byte for header)
- MTU = 1500 bytes (1480 Byte for data + 20 Byte for header).

	length	ID	fragflag	offset	
	=4000	=x	=0	=0	

*one large datagram becomes
several smaller datagrams*

1480 bytes in data field

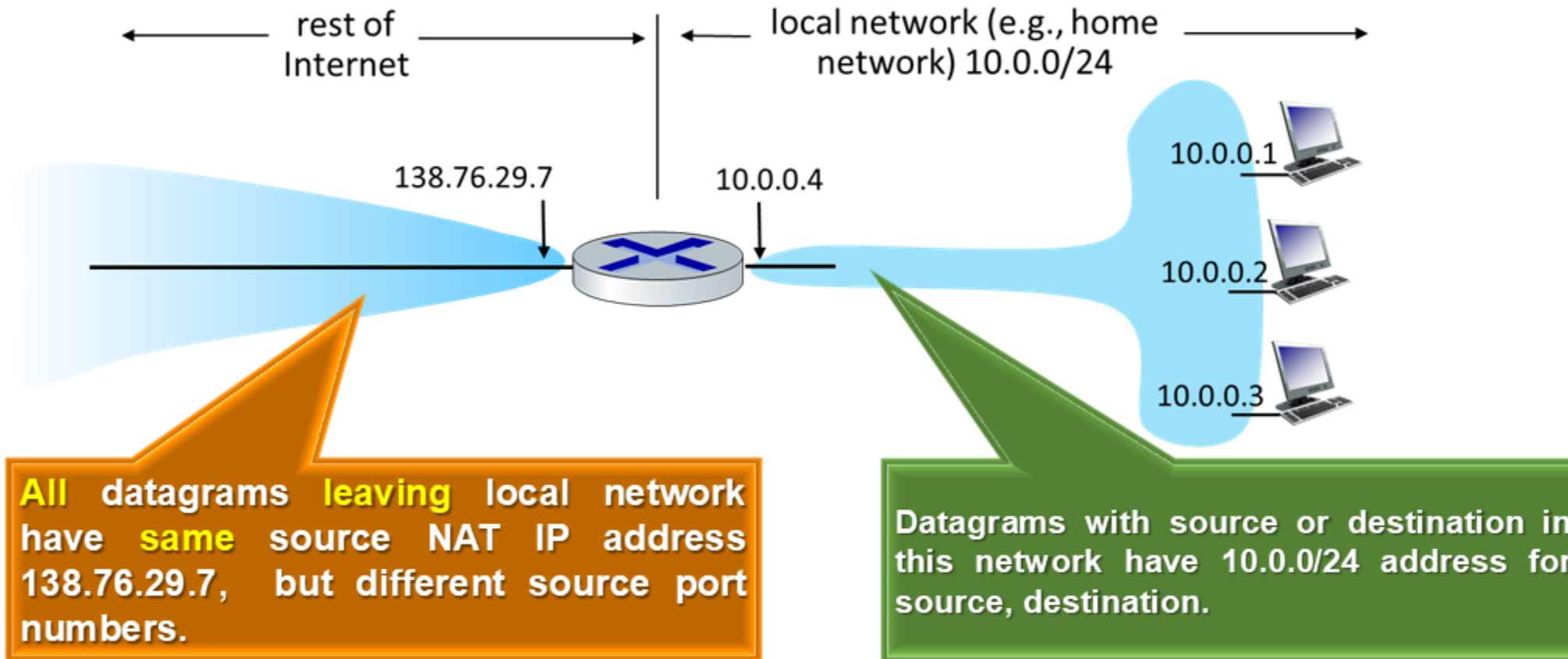
offset = 1480/8

offset = previous offset + 1480/8

	length	ID	fragflag	offset	
	=1500	=x	=1	=0	

	length	ID	fragflag	offset	
	=1500	=x	=1	=185	

	length	ID	fragflag	offset	
	=1040	=x	=0	=370	

NAT: network address translation

NAT: network address translation

- All devices in local network have 32-bit addresses in a “private” IP address space (*10/8, 172.16/12, 192.168/16* prefixes) that can only be used in local network.
- Advantages:
 - Just one IP address needed from provider ISP for all devices.
 - Can change addresses of host in local network without notifying outside world.
 - Can change ISP without changing addresses of devices in local network.
 - Security: devices inside local net not directly addressable, visible by outside world.

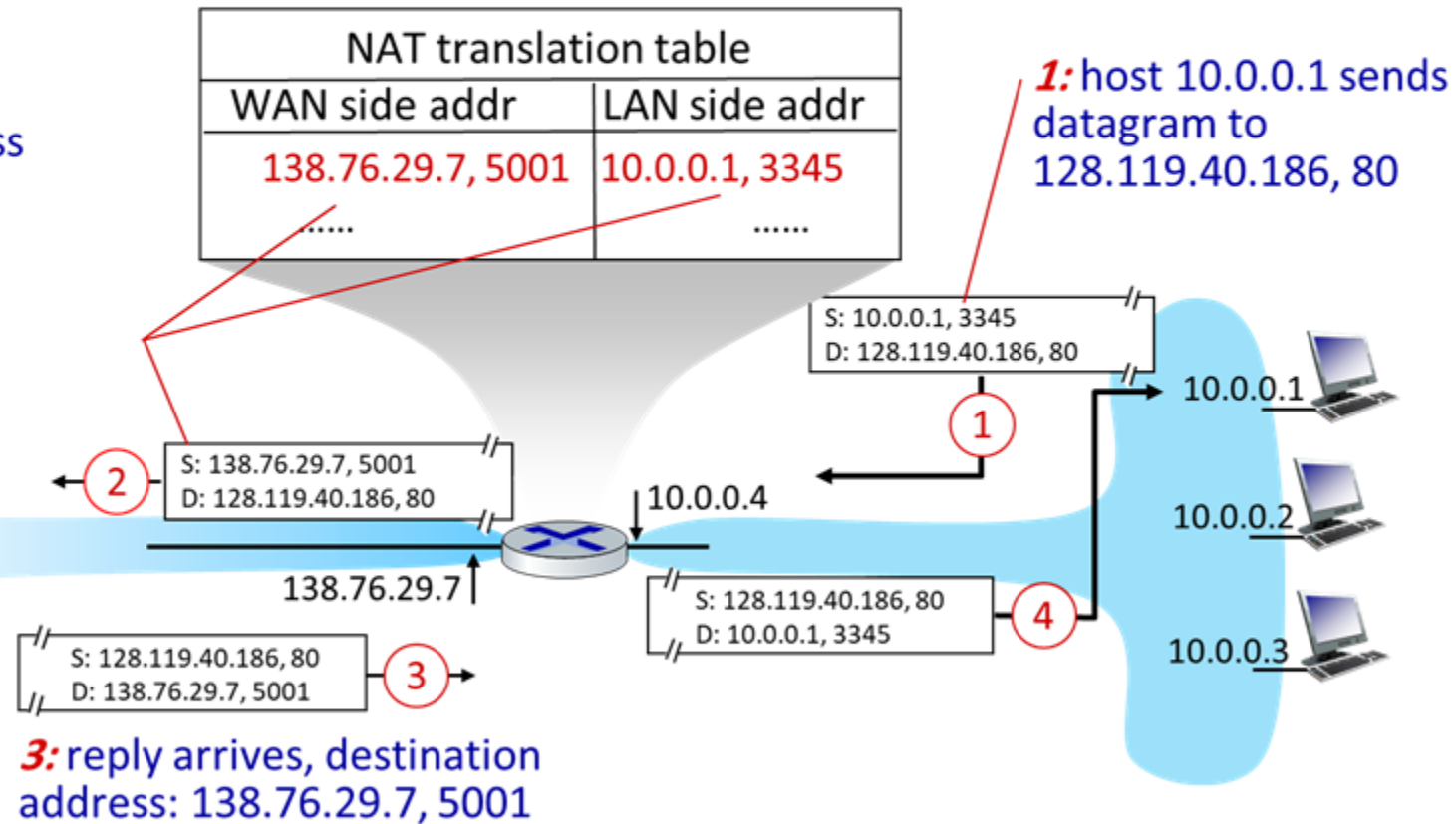
NAT: network address translation

Implementation: NAT router must (transparently):

- **Outgoing datagrams:** replace (source IP address, port #) of every outgoing datagram to (NAT IP address, new port #)
 - remote clients/servers will respond using (NAT IP address, new port #) as destination address.
- **Remember (in NAT translation table)** every (source IP address, port #) to (NAT IP address, new port #) translation pair.
- **Incoming datagrams:** replace (NAT IP address, new port #) in destination fields of every incoming datagram with corresponding (source IP address, port #) stored in NAT table.

NAT: network address translation

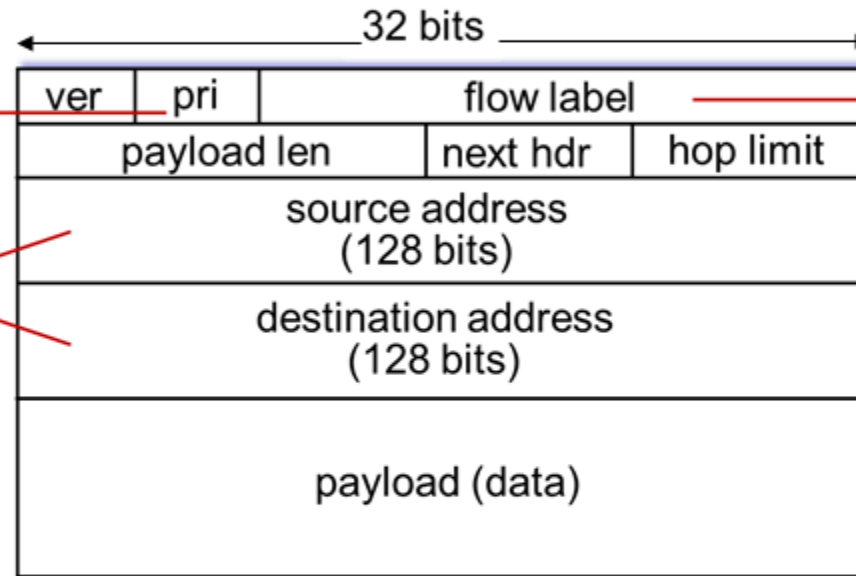
2: NAT router changes datagram source address from 10.0.0.1, 3345 to 138.76.29.7, 5001, updates table



Priority: identify
priority among
datagrams in flow

128-bit
IPv6 addresses

IPv6 Packet header



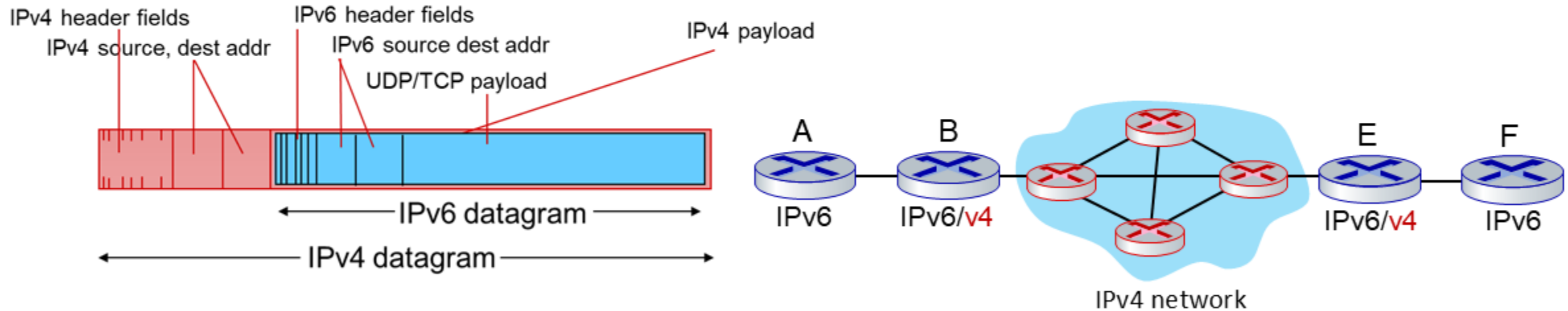
Flow label: identify
datagrams in same
"flow." (concept of
"flow" not well defined).

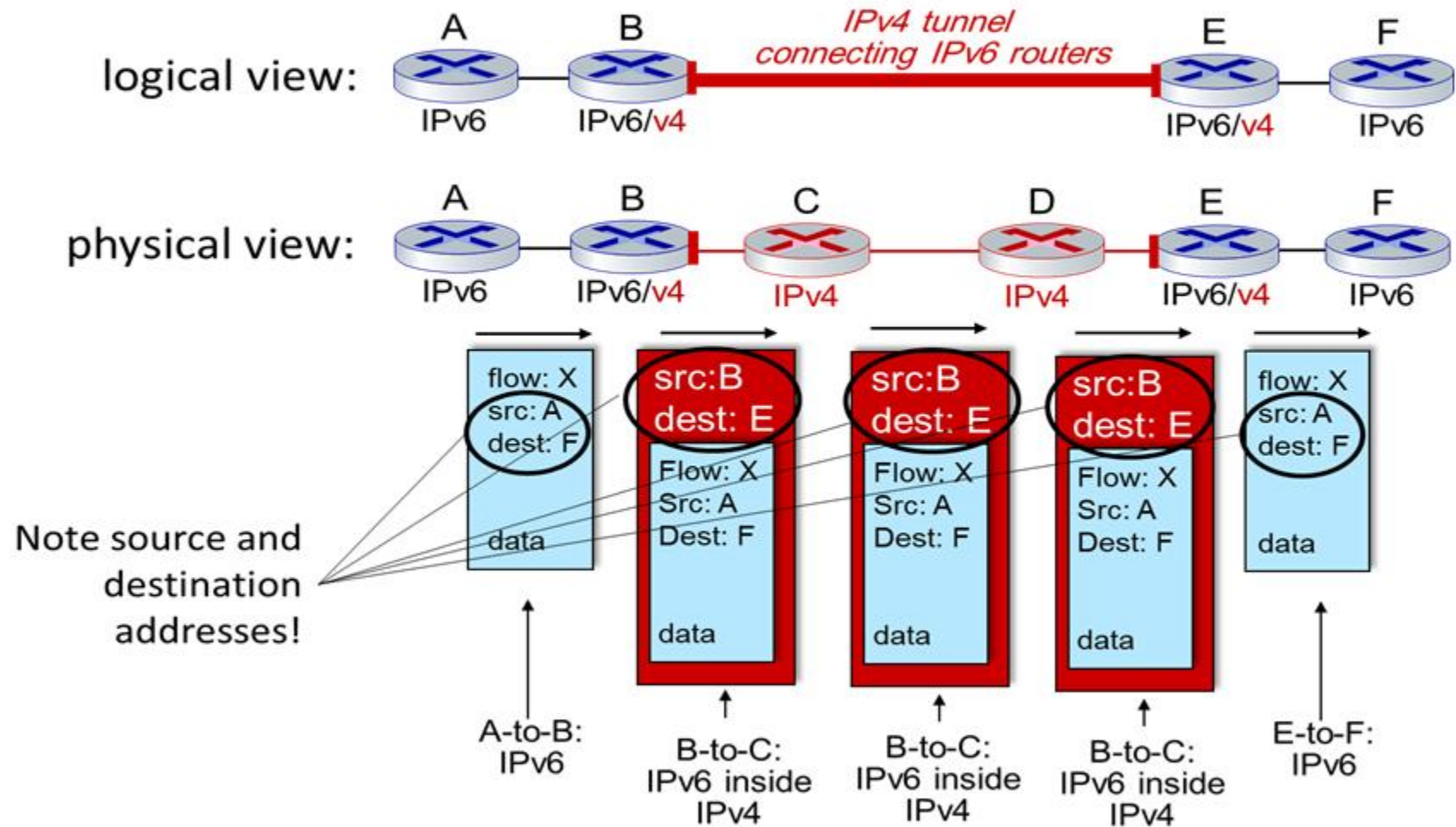
What's missing (compared with IPv4):

- No checksum (to speed processing at routers)
- No fragmentation/reassembly
- No options (available as upper-layer, next-header protocol at router)

Tunneling

Tunneling: IPv6 datagram carried as payload in IPv4 datagram among IPv4 routers (“packet within a packet”).



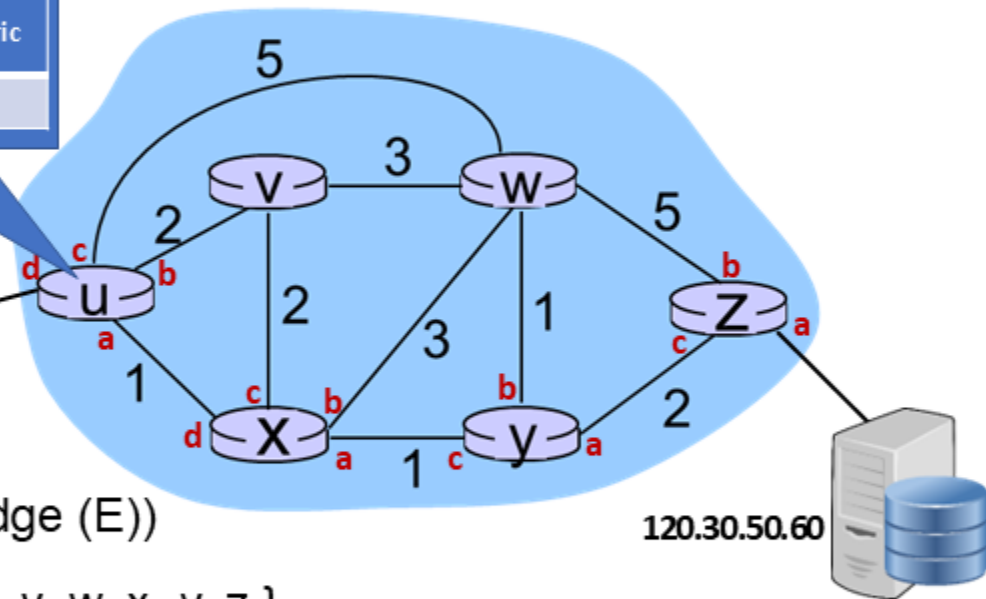
Tunneling

Graph abstraction

Link-Cost is calculated by routing algorithms using many different techniques and methods based on the routing algorithms in use. Some of the parameters used for calculating the link-cost are as follows:

- Hop count.
- Path reliability.
- Path speed.
- Bandwidth.
- Latency.
- Congestion.

Network Destination	Interface	Metric
120.0.0.0	a	1

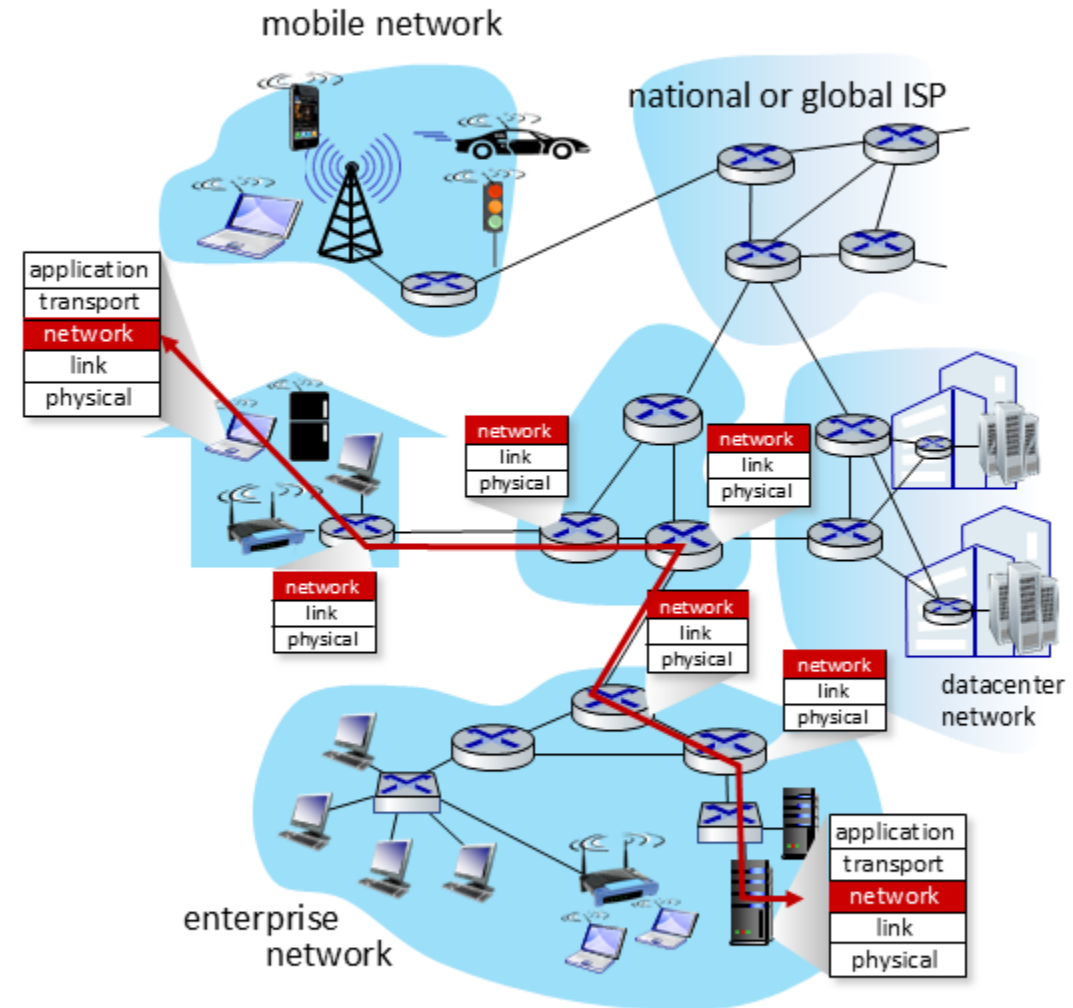


- Graph: $G = (\text{Node}(N), \text{Edge}(E))$
 - $N = \text{Set of routers} = \{u, v, w, x, y, z\}$
 - $E = \text{Set of links} = \{(u,v), (u,x), (v,x), (v,w), (x,w), (x,y), (w,y), (w,z), (y,z)\}$
- each link has a cost, *e.g.*, $C(u,v)=2$

Routing protocols

Routing protocol goal: determine “good” paths from sending hosts to receiving host, through network of routers.

- **Path:** sequence of routers packets traverse from given initial source host to final destination host.
- **“Good”:** least “cost”, “fastest”, “least congested”.



Routing algorithm classification

Static: routes change

slowly over time

Global: all routers have *complete* topology, link cost info.

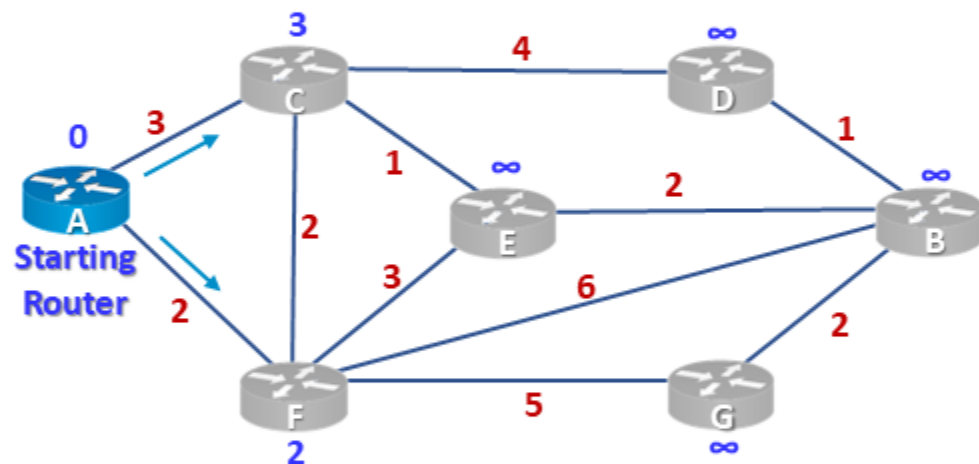
- “**Link-state**” algorithms.

Dynamic: routes change more quickly

- periodic updates or in response to link cost changes

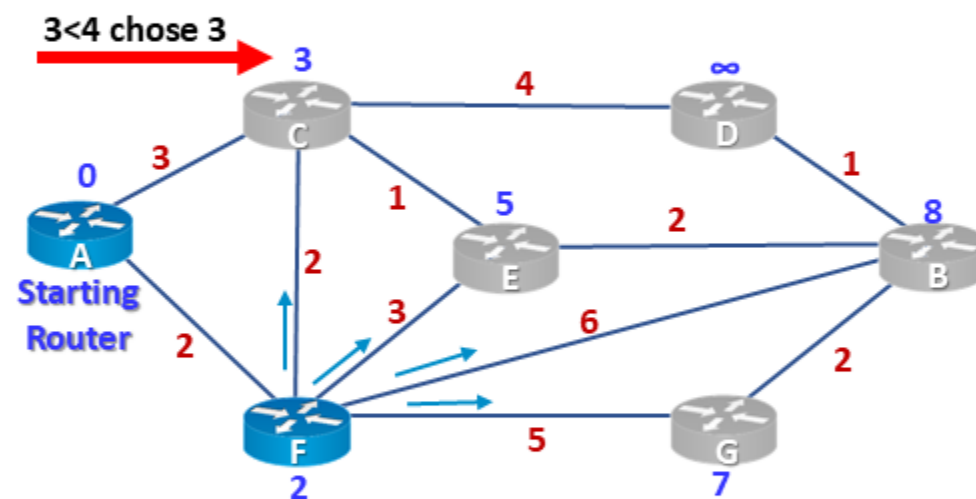
Decentralized: iterative process of computation, exchange of info with neighbors.

- routers initially only know link costs to attached neighbors
- “**Distance vector**” algorithms

Dijkstra's Link-State Routing AlgorithmLeast-cost from A to B

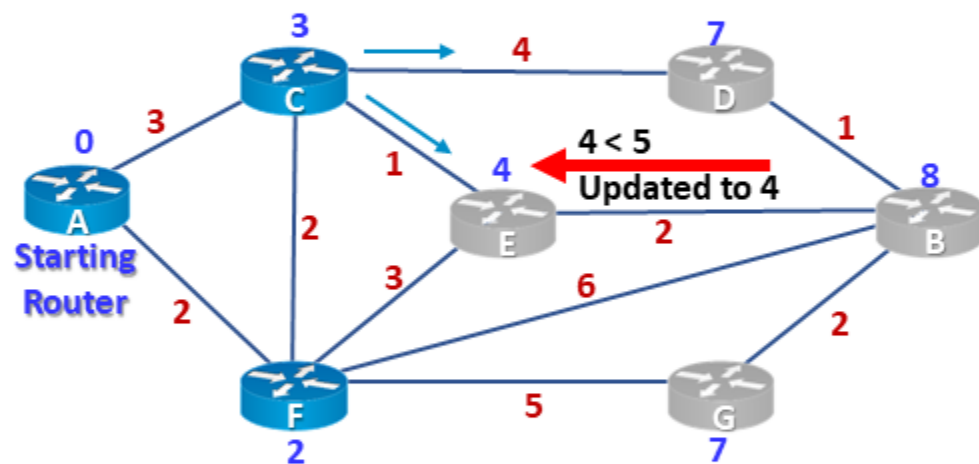
1

Calculate the cost of the adjacent routers to (A) and set the cost to all other routers to ∞



2

Move to the next router with the smallest cost (F) then update the cost of the connected routers to (F).

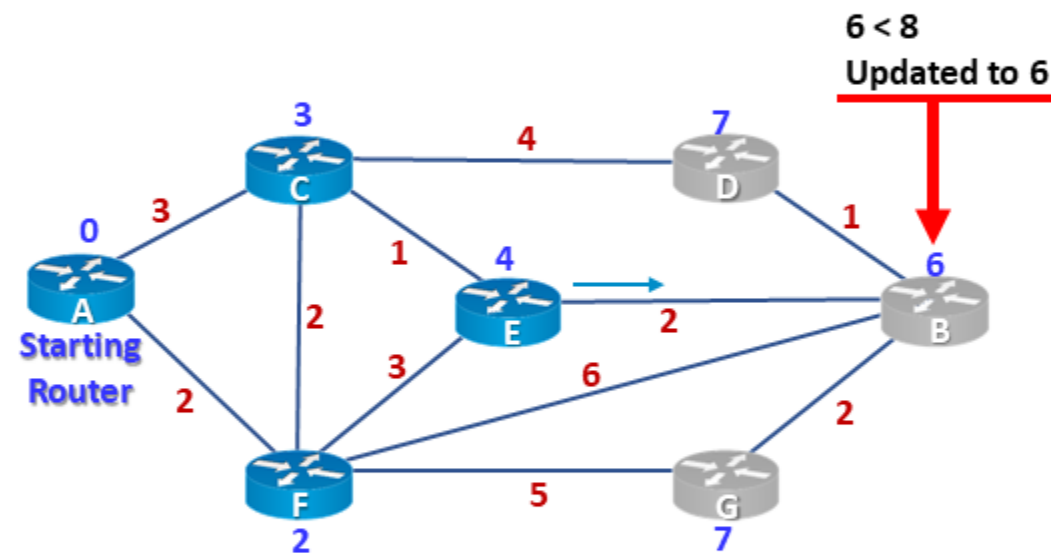
Dijkstra's Link-State Routing AlgorithmLeast-cost from A to B

3

Move to the next router with the smallest cost (C) then update the cost of the connected routers to (C).

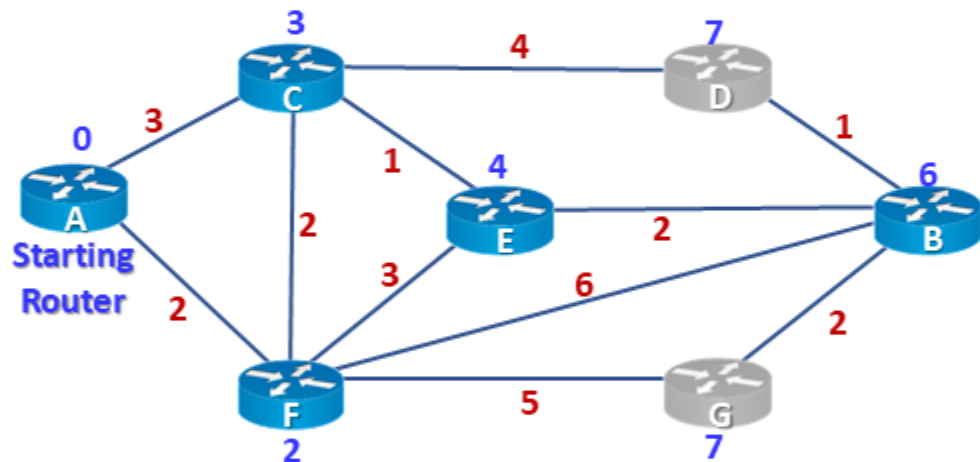
4

Move to the next router with the smallest cost (E) then update the cost of the connected routers to (E).



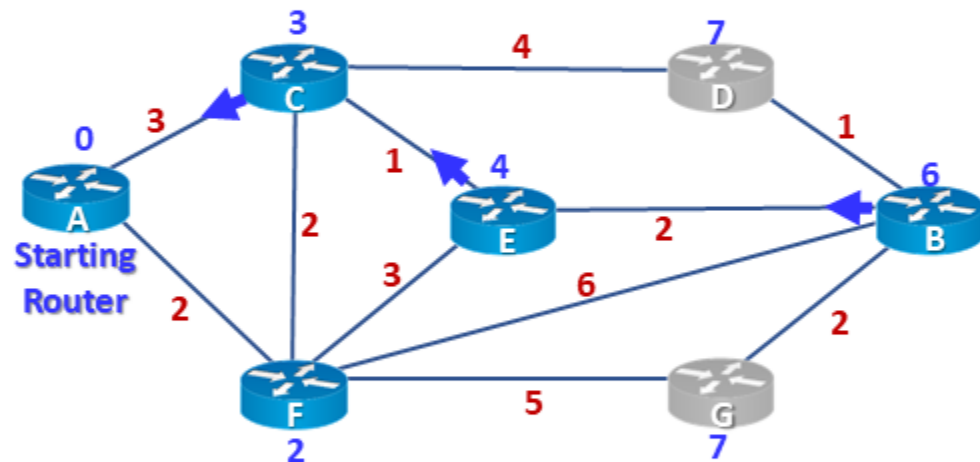
Dijkstra's Link-State Routing Algorithm

Least-cost from A to B



5

Move to the next router with the smallest cost (B), the least-cost path between (A) and (B) is found.



6

To reconstruct the least-cost path between (A) & (B), keep tracking the previous router to the beginning.

Dijkstra's Link-State Routing Algorithm

1

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) = ∞
```

2

2 Loop

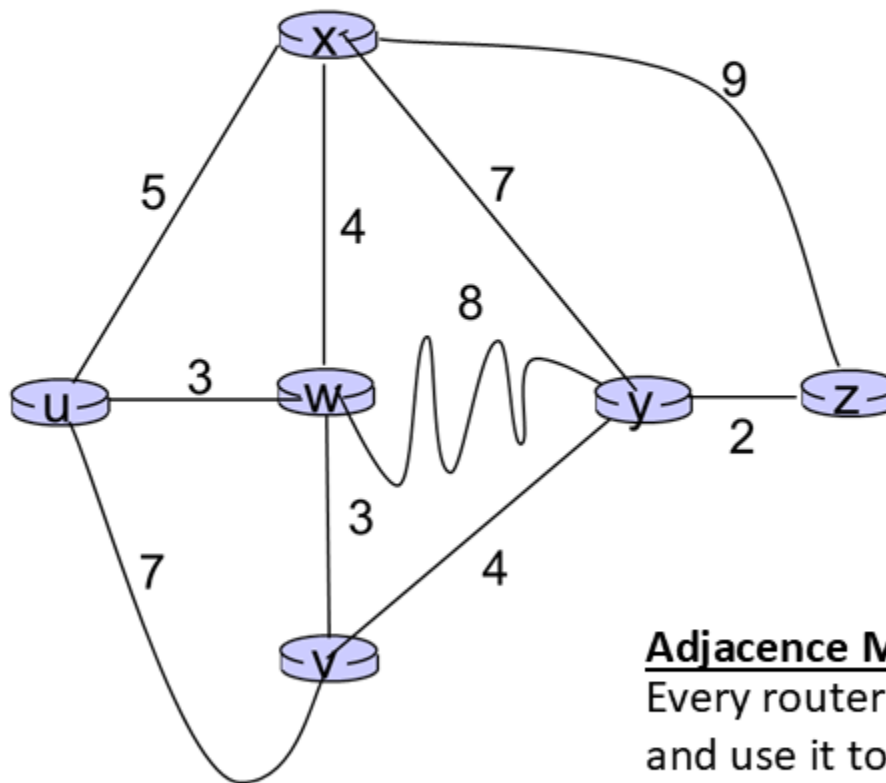
```
8 find w not in N' such that D(w) is a minimum
9 add w to N'
10 update D(v) for all v adjacent to w and not in N' :
11   D(v) = min( D(v), D(w) + c(w,v) )
12   /* new cost to v is either old cost to v or known
13   shortest path cost to w plus cost from w to v */
14
15 until all nodes in N'
```

notation

- $c_{x,y}$: direct link cost from node x to y ; $= \infty$ if not direct neighbors
- $D(v)$: *current* estimate of cost of least-cost-path from source to destination v
- $p(v)$: predecessor node along path from source to v
- N' : set of nodes whose least-cost-path *definitively* known

Dijkstra's Link-State Routing 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	7,U	3,U	5,U	∞	∞
1	UW	6,W		5,U	11,W	∞
2	UWX	6,W			11,W	14,X
3	UWXV				10,V	14,X
4	UWXVY					12,Y



	U	V	W	X	Y	Z
U	0	7	3	5	∞	∞
V	7	0	3	∞	4	∞
W	3	3	0	4	8	∞
X	5	∞	4	0	7	9
Y	∞	4	8	7	0	2
Z	∞	∞	∞	9	2	0

Adjacency Matrix:

Every router in the network has this matrix and use it to calculate least-cost paths.

Distance Vector Routing Algorithm

Based on *Bellman-Ford* (BF) equation (dynamic programming):

Bellman-Ford equation

Let $D_x(y)$: cost of least-cost path from x to y .

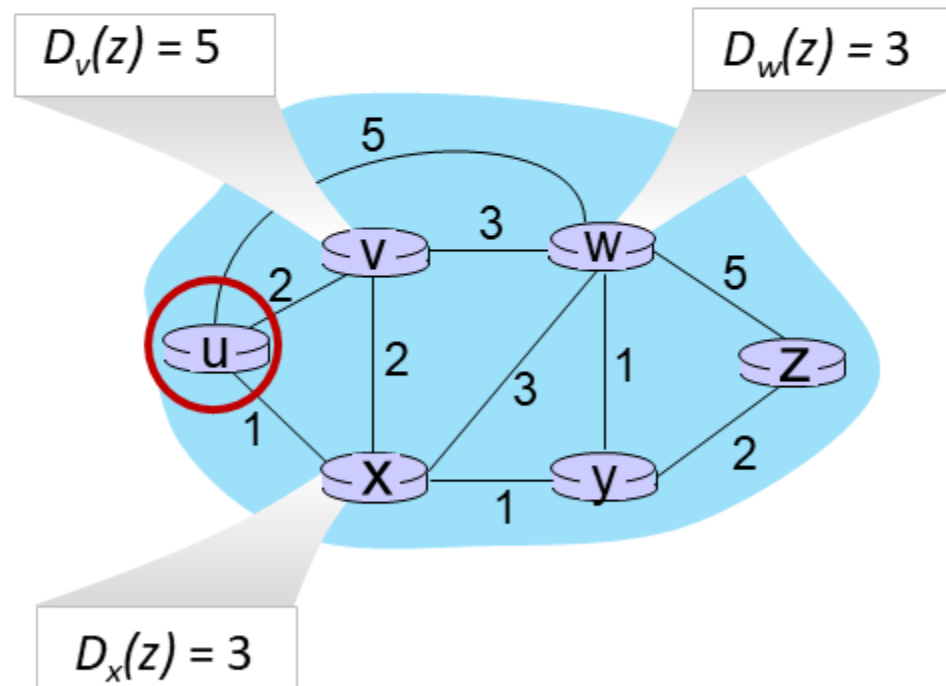
Then:

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

Where: $D_v(y)$ is estimated least-cost-path cost, and $c_{x,v}$ is direct cost of link from x to v

Distance Vector Routing Algorithm

Suppose that u 's neighboring nodes, x, v, w , know that for destination z :



Bellman-Ford equation says:

$$D_u(z) = \min \{ c_{u,v} + D_v(z), \\ c_{u,x} + D_x(z), \\ c_{u,w} + D_w(z) \}$$

$$= \min \{ 2 + 5, \\ 1 + 3, \\ 5 + 3 \} = \{ 7, 4, 8 \} = 4$$

node achieving minimum (x) is next hop on estimated least-cost path to destination (z)

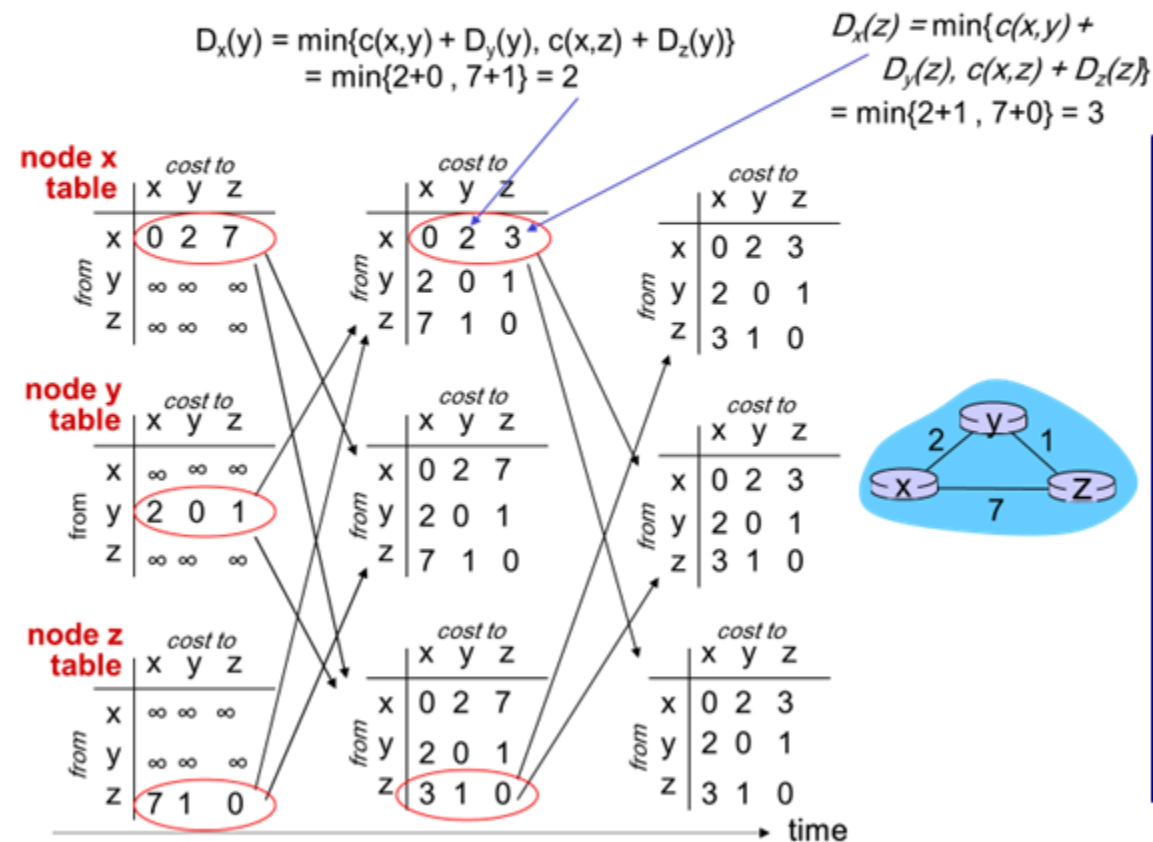
Distance Vector Routing Algorithm

➤ *Iterative, asynchronous:* each local iteration caused by:

- local link cost change
- DV update message from neighbor

➤ *Distributed:*

- Each node notifies neighbors only when its DV changes
 - Neighbors then notify their neighbors if necessary



each node:

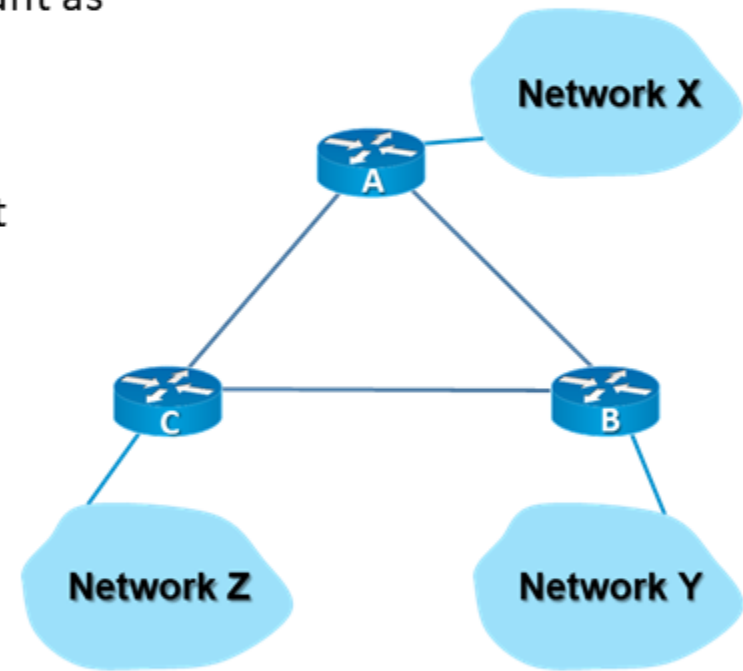
wait for (change in local link cost or msg from neighbor)

recompute estimates

if DV to any dest has changed, *notify* neighbors

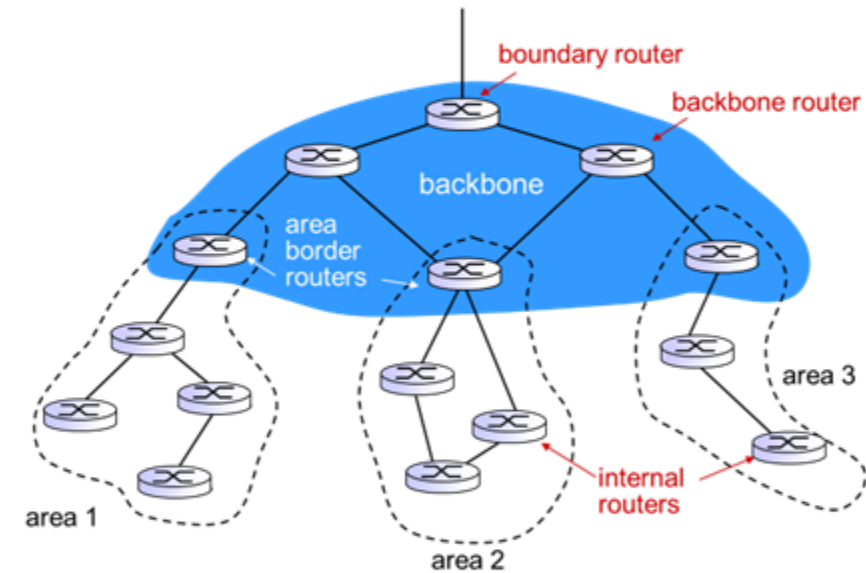
Routing Information Protocol (RIP)

- Routing Information Protocol (RIP) is a **distance vector protocol** that uses hop count as its primary metric.
- Support up to **15 hops max.**
- Exchanged with neighbors **every 30 sec** in response message, if no advertisement heard **after 180 sec.**, then neighbor/link declared dead:
 - Routes via neighbor invalidated
 - New advertisements sent to neighbors
 - Neighbors in turn send out new advertisements (if tables changed)
- RIP v1.0 does not support CIDR (Classless Inter-Domain Routing) and has no authentication.
- RIP v2.0 support CIDR and authentication.



Open Shortest Path First Protocol (OSPF)

- Open shortest path first (OSPF) is a **link-state routing protocol** that is used to find the best path between the source and the destination router using Dijkstra's algorithm.
- All OSPF messages authenticated .
- Two-level hierarchy: local area, backbone:
 - link-state advertisements only in area
 - each nodes has detailed area topology; only know direction (shortest path) to nets in other areas.
- Area border routers: "summarize" distances to nets in own area, advertise to other Area Border routers.
- Backbone routers: run OSPF routing limited to backbone.
- boundary routers: connect to other AS's.
- Integrated unicast- and multicast support.



Border Gateway Protocol (BGP)

- Border Gateway Protocol (BGP) is a hybrid **link-state and Distance Vector routing protocol** that is used to exchange of routing and reachability information among edge routers in order to directs packets between autonomous systems (AS).
- BGP provides each AS a means to:
 - **eBGP**: obtain subnet reachability information from neighboring ASs.
 - **iBGP**: propagate reachability information to all AS-internal routers.
- Two important attributes:
 - **AS-PATH**: contains ASs through which prefix advertisement has passed: e.g., AS 67, AS 17.
 - **NEXT-HOP**: indicates specific internal-AS router to next-hop AS. (Router selects route based on shortest AS-PATH)
 - **Example**: Prefix:138.16.64/22 ; AS-PATH: AS3 AS131 ; NEXT-HOP: 201.44.13.125

