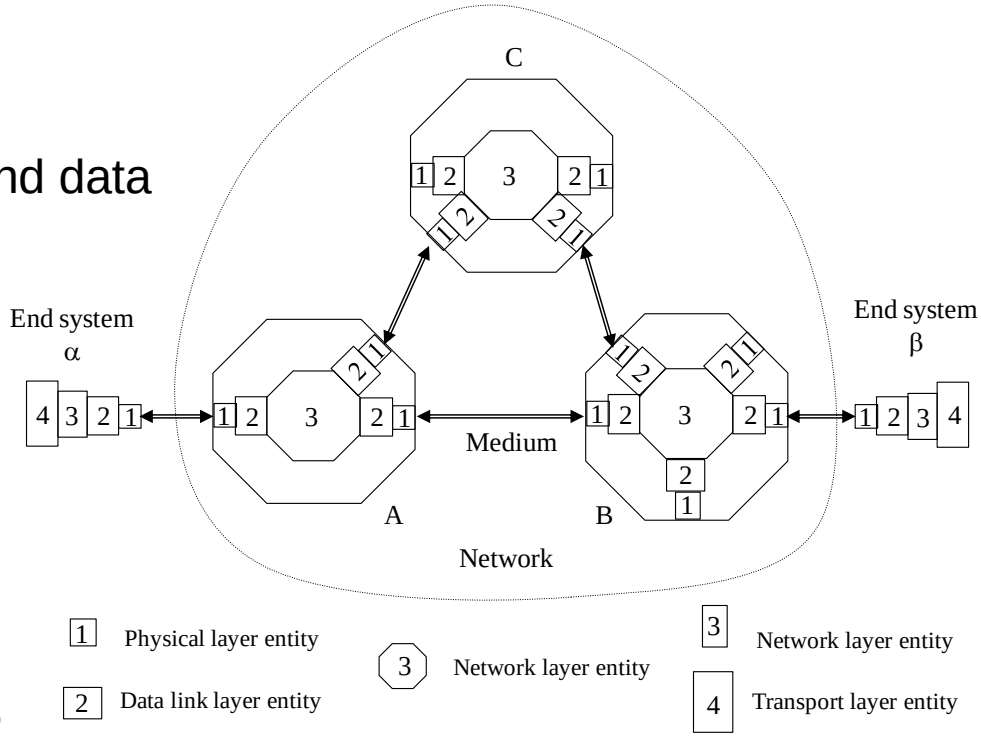
# Data Communication Networks
# Network Layer

M. R. Pakravan
Department of Electrical Engineering
Sharif University of Technology

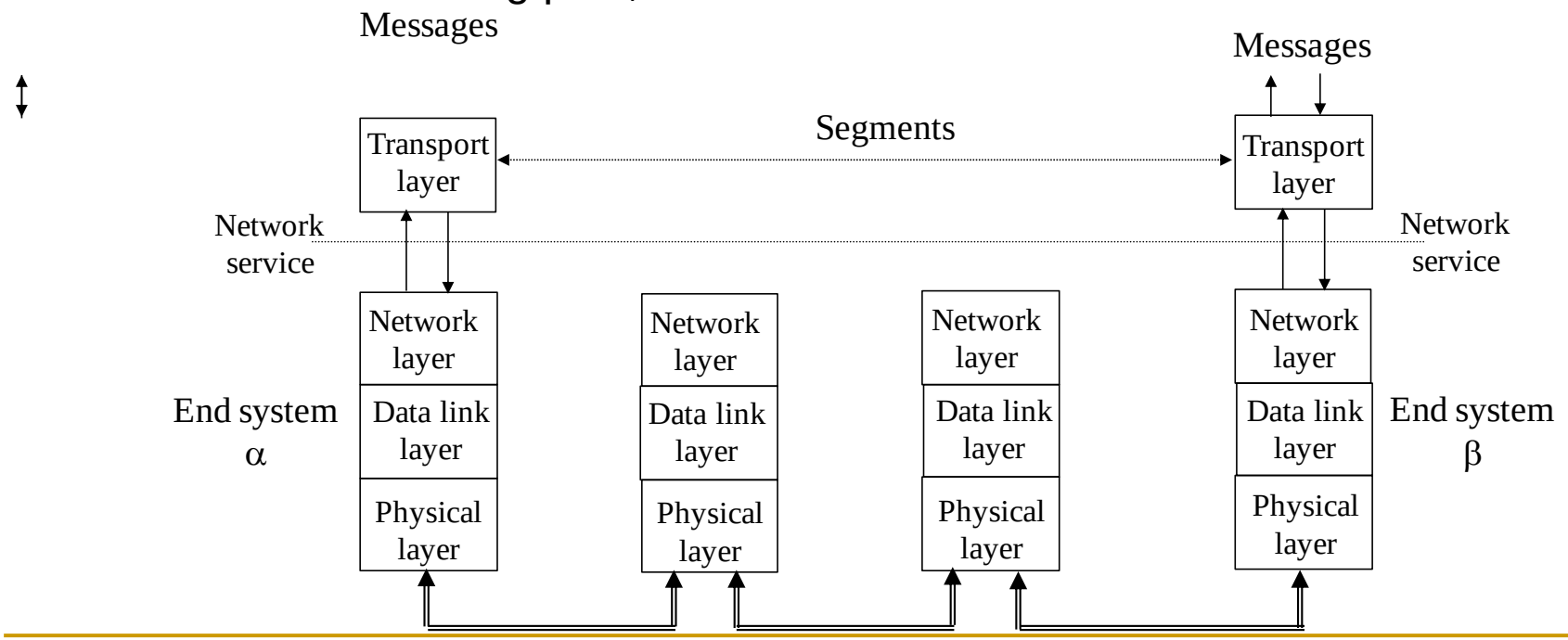
# The Network Layer

- Role of network layer: ***routing packets***
  - Deals with end-to-end communication
  - Aware of network topology
  - Choose appropriate route
  - Communicates with transport and data link layers
- Network layer topics
  - Design issues
  - Routing algorithms
  - Congestion control
  - Quality of Service
  - Internetworking
  - Network layer in the Internet: IP



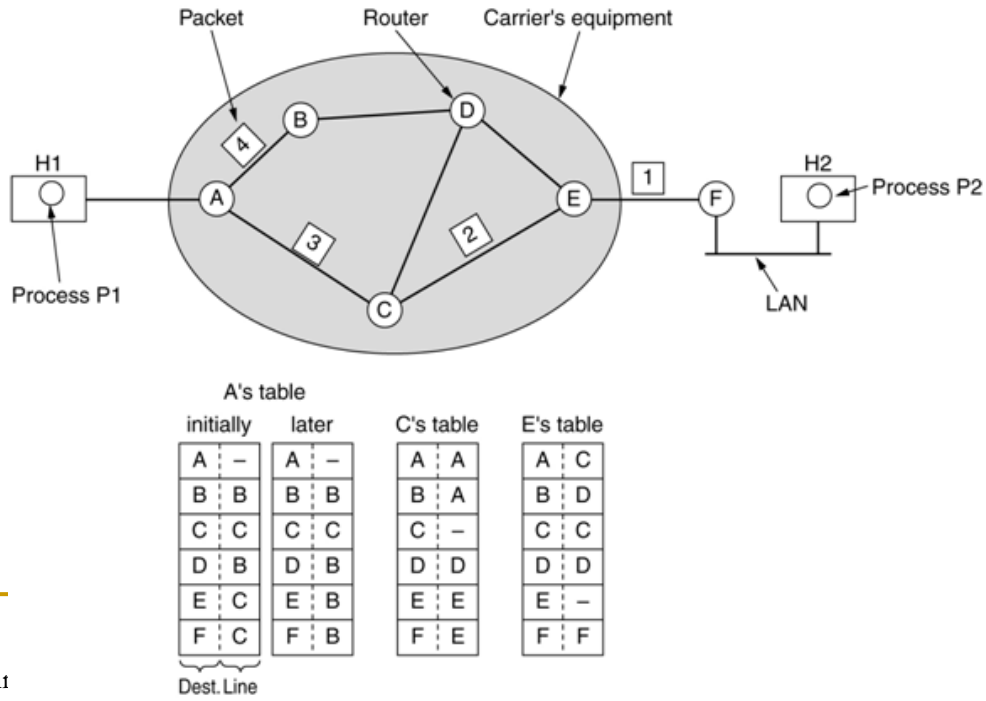
# Network Layer Design Issues

- Services provided to the transport layer
  - Independent of chosen subnet technology
  - Shield transport layer from topology, type and number of subnets
  - Uniform numbering plan, across LANs and WANs



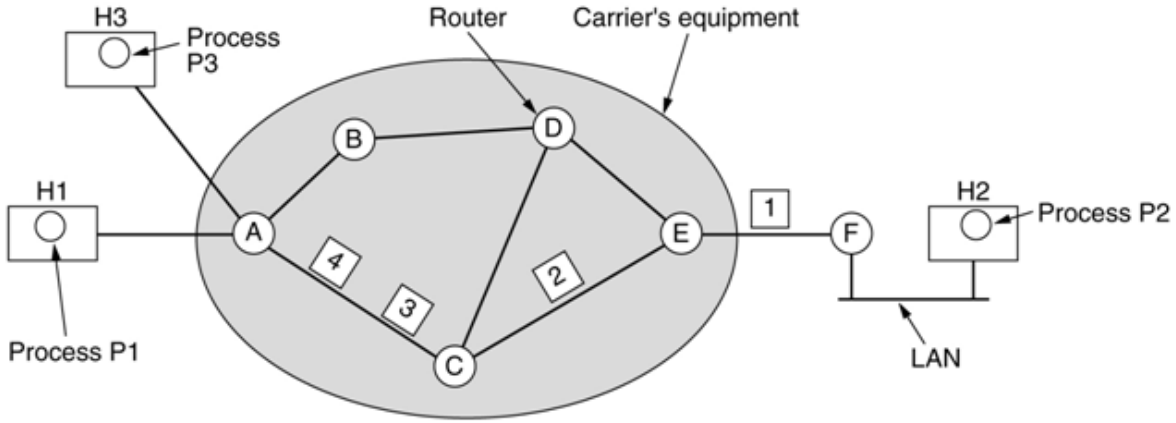
# Network layer design issues

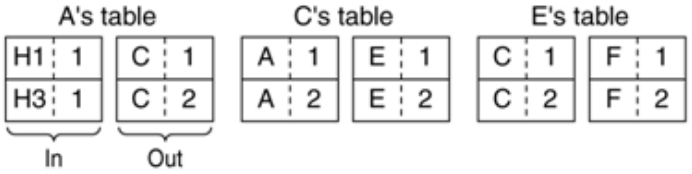
- **Connection-oriented or connectionless service?**
  - Where to put the complexity?
- Internet community: connectionless
  - Network will be unreliable anyway => handle this issue in higher layer (s)
  - Only SEND packet and RECEIVE packet needed
  - Hosts are getting powerful; so why not put the complexity there (in the transport layer) => easy to adapt
  - Speed more important than accuracy



# Network Layer Design Issues

- Telephone companies: NL should be connection-oriented
  - Set up connection first
  - Each connection has unique ID
  - Negotiate parameters (quality, cost)
  - Communication in both directions
  - Use of Acknowledge gives automatic flow control



# Virtual Circuits & Datagrams

- **Virtual circuits** :
  - Avoid routing every packet or cell independently
  - All packets/cells follow same route (= session routing)
  - Store # VC in routing tables (consuming some resources)
- **Datagrams**
  - Each packet contains destination and is independently routed
  - Also usable for connection-oriented service

| Issue | Datagram subnet | Virtual-circuit subnet |
|---|---|---|
| Circuit setup | Not needed | Required |
| Addressing | Each packet contains the full source and destination address | Each packet contains a short VC number |
| State information | Routers do not hold state information about connections | Each VC requires router table space per connection |
| Routing | Each packet is routed independently | Route chosen when VC is set up; all packets follow it |
| Effect of router failures | None, except for packets lost during the crash | All VCs that passed through the failed router are terminated |
| Quality of service | Difficult | Easy if enough resources can be allocated in advance for each VC |
| Congestion control | Difficult | Easy if enough resources can be allocated in advance for each VC |

# Routing

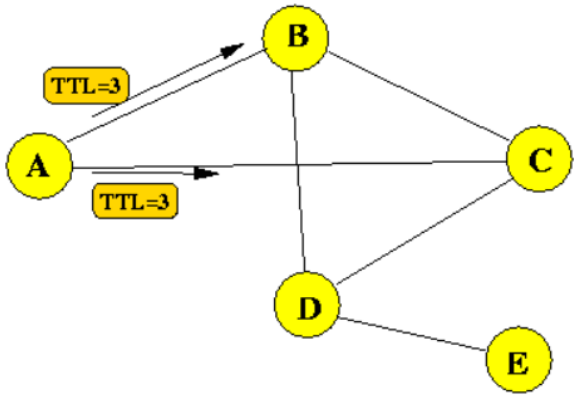
- Message / packet / cell has to pass multiple hops
- Routing algorithm : decide which outgoing link to use
- Routing algorithm requirements
  - Correctness
  - Simplicity
  - Robustness: cope with changing topology
  - Stability: converge quickly
  - Fairness: guarantee progression

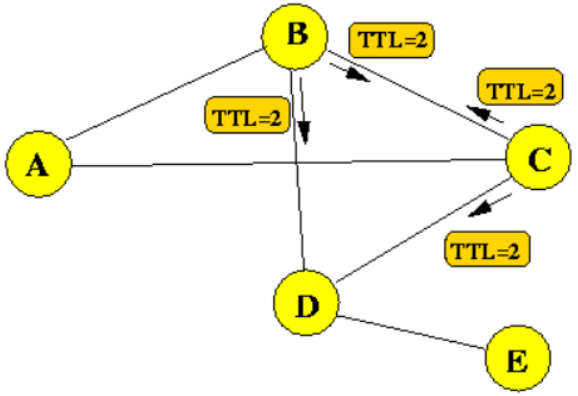
# Routing algorithms

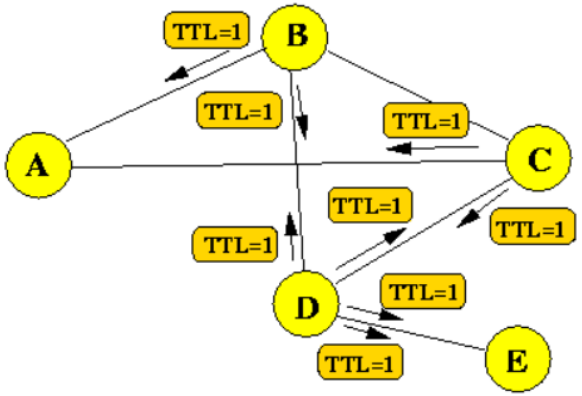
- Non-adaptive (static) routing:
  - Do not use measurements or estimates of current traffic and topology
  - Static routing: calculate route off-line or use deterministic algorithm
- Adaptive (dynamic) routing:
  - Routing decision depends on network status
  - Update state information regularly
  - What type of network information do we base the decision on?
    - Hop count
    - Distance, estimated transit time
    - Traffic rate
    - Congestion metrics

# Routing Algorithms: Flooding

- Static algorithm
- Idea: forward incoming packet on all other outgoing lines



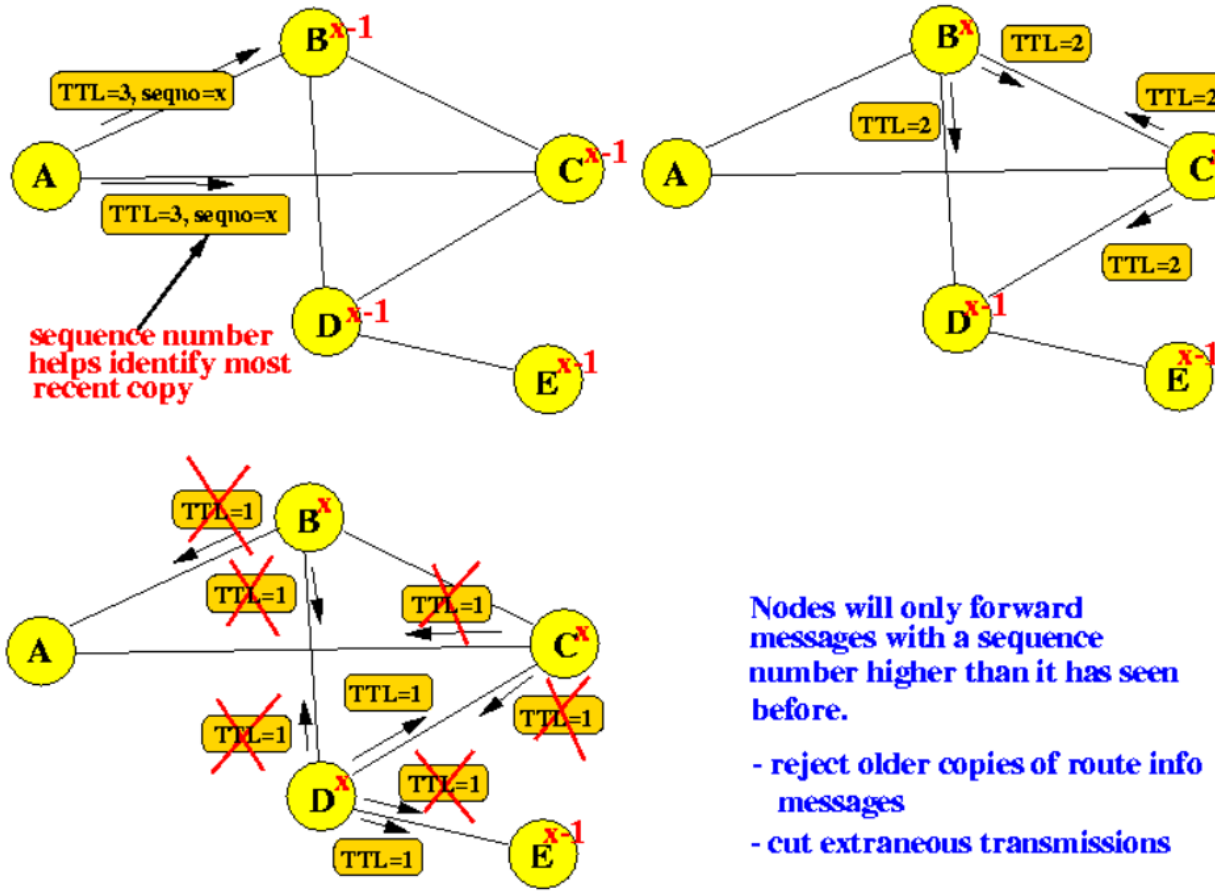

*Conclusion:*

*flooding sends a LARGE number of extraneous packets !*

*But: flooding does not use any routing information and will forward the message to ALL nodes in network*

# Routing Algorithms: Flooding

- Reducing the number of forwarded packets using sequence numbers



# Routing Algorithms: Flooding

- Flooding Properties:
  - Not practical in most applications
  - Generates too much traffic
  - Useful for:
    - Extreme robustness (military purpose)
    - Broadcast
    - Reference algorithm
  - It always finds the shortest delay path !!

# Routing

- Optimality principle:
  - If F is on optimal path of L $\Rightarrow$ B then the whole path F $\Rightarrow$ A $\Rightarrow$ B is the best path between F and B
- Consequence: optimal routes from all sources to a given root ( = sink node ) form a *sink tree* $\Rightarrow$ no loops.

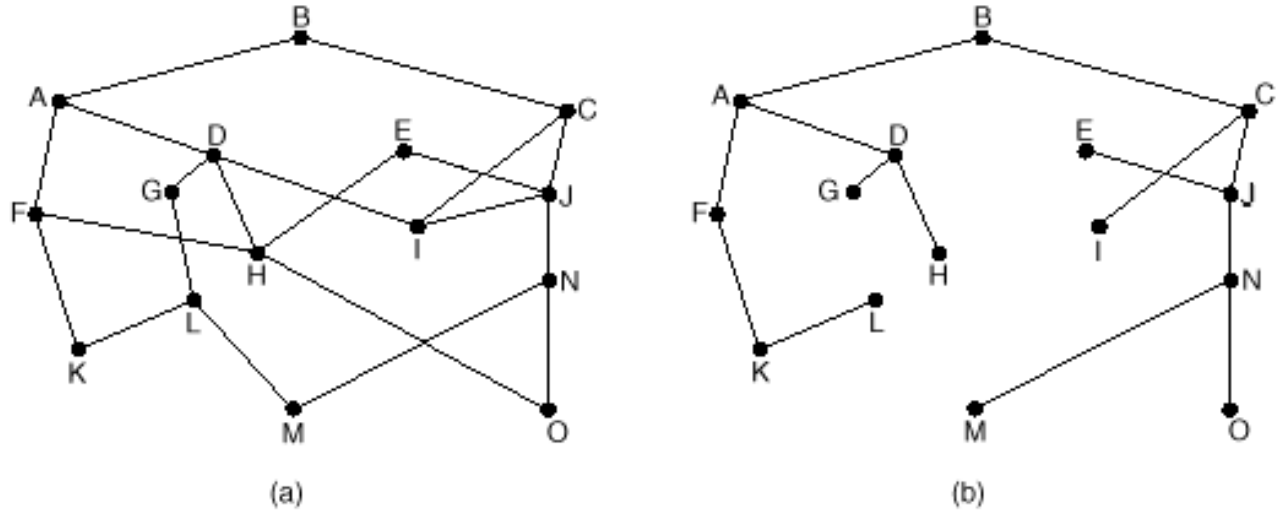



**Fig. 5-5.** (a) A subnet. (b) A sink tree for router *B*.

# Shortest Path Algorithm

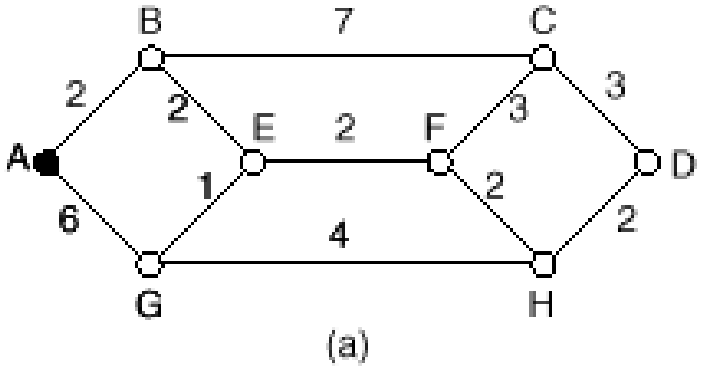
- Shortest Path Algorithm is used as part of many routing algorithms
- Build router graph G(N, L)
  - N is the set of nodes
  - L is the set of links (or edges)
- Choose path metric (weight of lines L)
  - # hops
  - Estimated delay ( using echo packets )
  - Physical distance
  - Queue length ( of outgoing lines )

# Shortest Path Algorithm

- Dijkstra's shortest path algorithm:
- Steps to find all shortest paths from A
- Initially: make A permanent: P = {A} : label all neighbors k of A with $D_k = d_{Ak}$
- Repeat until P contains all nodes
  - find next closest node and add it to P:
  - P= P + {i} , with $D_i$ = min $D_k$ (k not in P)
  - Updating labels of neighbours of P
  - $D_k$ = min [ $D_k$ , $D_k$ + $d_{ik}$ ] (k not in P)
- Key fact: the path to the next closest node i has to go through nodes from P only.

# Shortest Path Algorithm

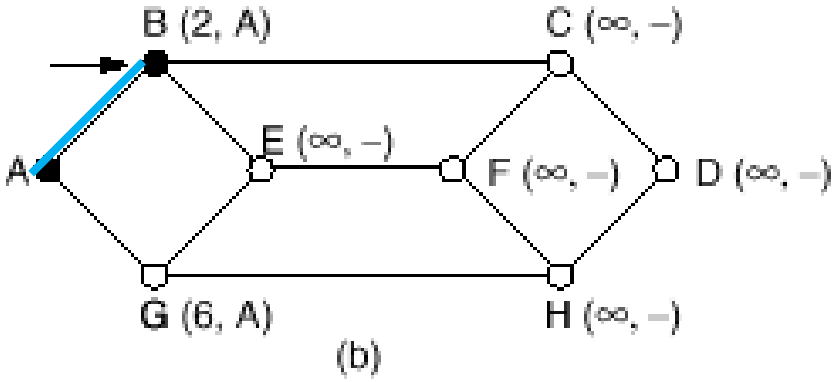
| | Base Set | B | C | D | E | F | G | H |
|---|---|---|---|---|---|---|---|---|
| 0 | A | 2,A | ∞ | ∞ | ∞ | ∞ | 6,A | ∞ |
| 1 | A,B | | 9,B | ∞ | 4,B | ∞ | 6,A | ∞ |
| 2 | A,B,E | | 9,B | ∞ | | 6,E | 5,E | ∞ |
| 3 | A,B,E,G | | 9,B | ∞ | | 6,E | | 9,G |
| 4 | A,B,E,G,F | | 9,B | ∞ | | | | 8,F |
| 5 | A,B,E,G,F,H | | 9,B | 10,H | | | | |
| 6 | A,B,E,G,F,H,C | | | 10,H | | | | |



(a)

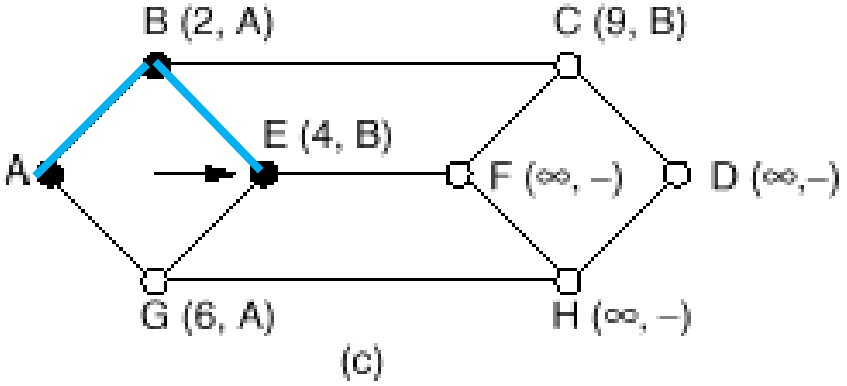
# Shortest Path Algorithm

| | Base Set | B | C | D | E | F | G | H |
|---|---|---|---|---|---|---|---|---|
| 0 | A | 2,A | ∞ | ∞ | ∞ | ∞ | 6,A | ∞ |
| 1 | A,B | | 9,B | ∞ | 4,B | ∞ | 6,A | ∞ |
| 2 | A,B,E | | 9,B | ∞ | | 6,E | 5,E | ∞ |
| 3 | A,B,E,G | | 9,B | ∞ | | 6,E | | 9,G |
| 4 | A,B,E,G,F | | 9,B | ∞ | | | | 8,F |
| 5 | A,B,E,G,F,H | | 9,B | 10,H | | | | |
| 6 | A,B,E,G,F,H,C | | | 10,H | | | | |



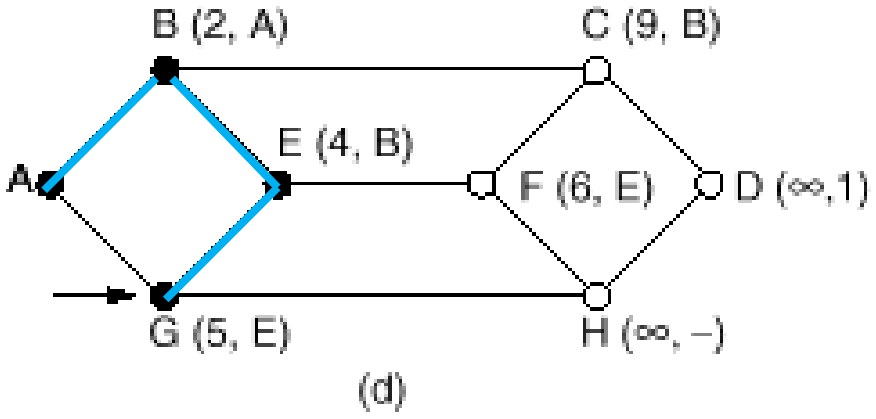
# Shortest Path Algorithm

| | Base Set | B | C | D | E | F | G | H |
|---|---|---|---|---|---|---|---|---|
| 0 | A | 2,A | ∞ | ∞ | ∞ | ∞ | 6,A | ∞ |
| 1 | A,B | | 9,B | ∞ | 4,B | ∞ | 6,A | ∞ |
| 2 | A,B,E | | 9,B | ∞ | | 6,E | 5,E | ∞ |
| 3 | A,B,E,G | | 9,B | ∞ | | 6,E | | 9,G |
| 4 | A,B,E,G,F | | 9,B | ∞ | | | | 8,F |
| 5 | A,B,E,G,F,H | | 9,B | 10,H | | | | |
| 6 | A,B,E,G,F,H,C | | | 10,H | | | | |



# Shortest Path Algorithm

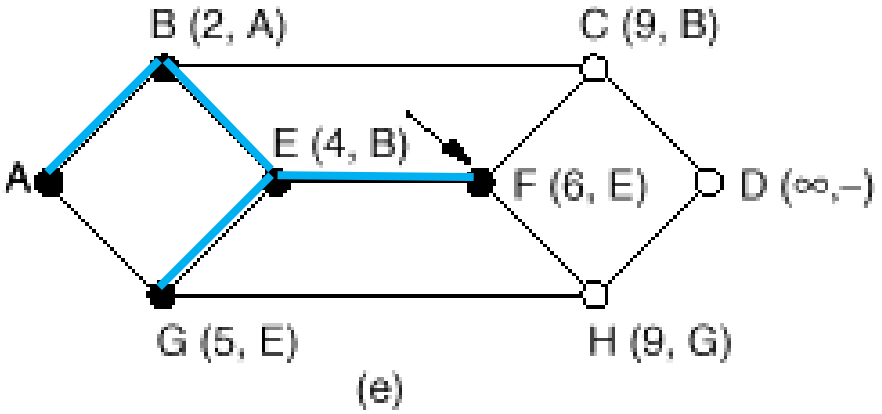
| | Base Set | B | C | D | E | F | G | H |
|---|---|---|---|---|---|---|---|---|
| 0 | A | 2,A | ∞ | ∞ | ∞ | ∞ | 6,A | ∞ |
| 1 | A,B | | 9,B | ∞ | 4,B | ∞ | 6,A | ∞ |
| 2 | A,B,E | | 9,B | ∞ | | 6,E | 5,E | ∞ |
| 3 | A,B,E,G | | 9,B | ∞ | | 6,E | | 9,G |
| 4 | A,B,E,G,F | | 9,B | ∞ | | | | 8,F |
| 5 | A,B,E,G,F,H | | 9,B | 10,H | | | | |
| 6 | A,B,E,G,F,H,C | | | 10,H | | | | |



(d)

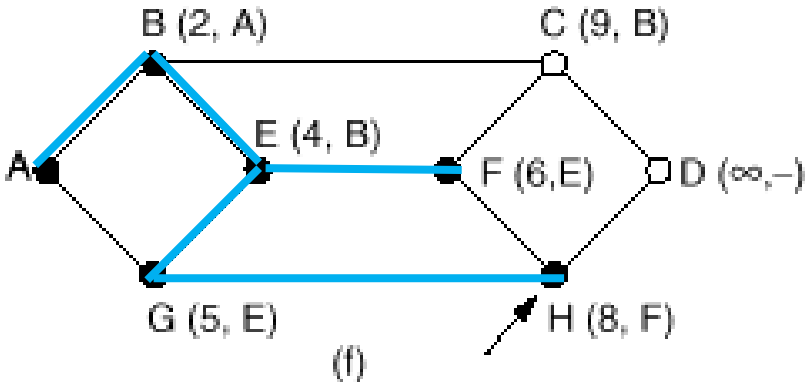
# Shortest Path Algorithm

| | Base Set | B | C | D | E | F | G | H |
|---|---|---|---|---|---|---|---|---|
| 0 | A | 2,A | ∞ | ∞ | ∞ | ∞ | 6,A | ∞ |
| 1 | A,B | | 9,B | ∞ | 4,B | ∞ | 6,A | ∞ |
| 2 | A,B,E | | 9,B | ∞ | | 6,E | 5,E | ∞ |
| 3 | A,B,E,G | | 9,B | ∞ | | 6,E | | 9,G |
| 4 | A,B,E,G,F | | 9,B | ∞ | | | | 8,F |
| 5 | A,B,E,G,F,H | | 9,B | 10,H | | | | |
| 6 | A,B,E,G,F,H,C | | | 10,H | | | | |



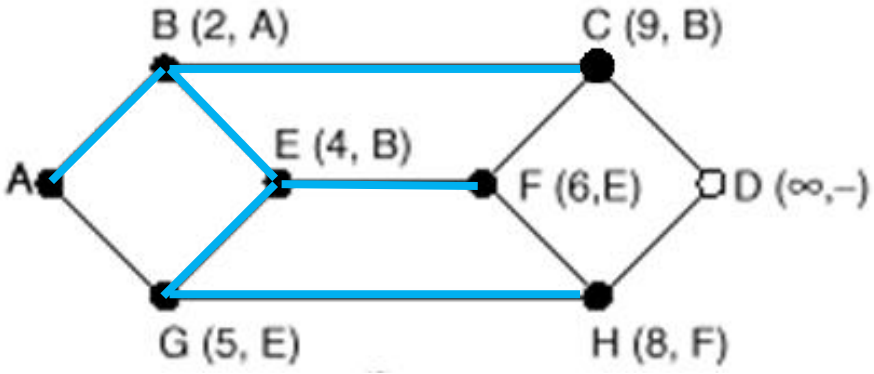
# Shortest Path Algorithm

| | Base Set | B | C | D | E | F | G | H |
|---|---|---|---|---|---|---|---|---|
| 0 | A | 2,A | ∞ | ∞ | ∞ | ∞ | 6,A | ∞ |
| 1 | A,B | | 9,B | ∞ | 4,B | ∞ | 6,A | ∞ |
| 2 | A,B,E | | 9,B | ∞ | | 6,E | 5,E | ∞ |
| 3 | A,B,E,G | | 9,B | ∞ | | 6,E | | 9,G |
| 4 | A,B,E,G,F | | 9,B | ∞ | | | | 8,F |
| 5 | A,B,E,G,F,H | | 9,B | 10,H | | | | |
| 6 | A,B,E,G,F,H,C | | | 10,H | | | | |



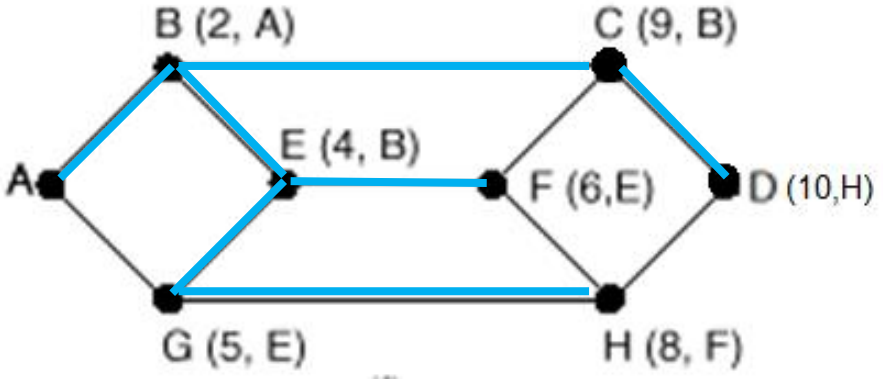
# Shortest Path Algorithm

| | Base Set | B | C | D | E | F | G | H |
|---|---|---|---|---|---|---|---|---|
| 0 | A | 2,A | ∞ | ∞ | ∞ | ∞ | 6,A | ∞ |
| 1 | A,B | | 9,B | ∞ | 4,B | ∞ | 6,A | ∞ |
| 2 | A,B,E | | 9,B | ∞ | | 6,E | 5,E | ∞ |
| 3 | A,B,E,G | | 9,B | ∞ | | 6,E | | 9,G |
| 4 | A,B,E,G,F | | 9,B | ∞ | | | | 8,F |
| 5 | A,B,E,G,F,H | | 9,B | 10,H | | | | |
| 6 | A,B,E,G,F,H,C | | | 10,H | | | | |



# Shortest Path Algorithm

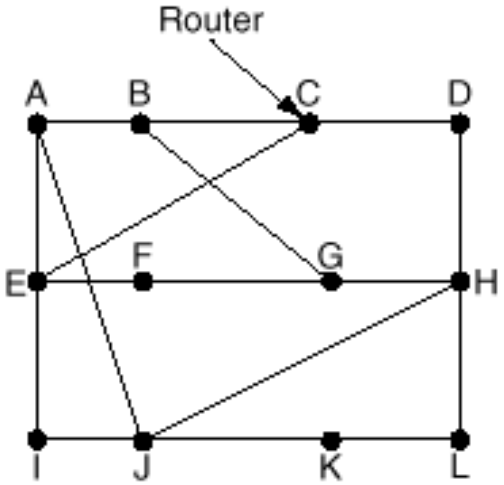
| | Base Set | B | C | D | E | F | G | H |
|---|---|---|---|---|---|---|---|---|
| 0 | A | 2,A | ∞ | ∞ | ∞ | ∞ | 6,A | ∞ |
| 1 | A,B | | 9,B | ∞ | 4,B | ∞ | 6,A | ∞ |
| 2 | A,B,E | | 9,B | ∞ | | 6,E | 5,E | ∞ |
| 3 | A,B,E,G | | 9,B | ∞ | | 6,E | | 9,G |
| 4 | A,B,E,G,F | | 9,B | ∞ | | | | 8,F |
| 5 | A,B,E,G,F,H | | 9,B | 10,H | | | | |
| 6 | A,B,E,G,F,H,C | | | 10,H | | | | |
| 7 | A,B,E,G,F,H,C,D | | | | | | | |



# Dynamic routing

- Distance vector routing
  - Used in early ARPANET
  - Each router maintains distance table indexed by destination number
  - Entry contains : outgoing link + estimated distance
- How to know distance?
  - Send echo packet to all neighbors; this packet gets back with timestamp therefore, distance to neighbor is determined
- Receive regularly updated tables from neighbors and update own table
  - D (a,b) = min (D (a,x) + D (x,b) )
    {for all neighbors x}
- Other metrics such as hop count
  or cost could also be used

# Dynamic routing




(a)

New estimated delay from J

| To | A | I | H | K | Line |
|---|---|---|---|---|---|
| A | 0 | 24 | 20 | 21 | 8 A |
| B | 12 | 36 | 31 | 28 | 20 A |
| C | 25 | 18 | 19 | 36 | 28 I |
| D | 40 | 27 | 8 | 24 | 20 H |
| E | 14 | 7 | 30 | 22 | 17 I |
| F | 23 | 20 | 19 | 40 | 30 I |
| G | 18 | 31 | 6 | 31 | 18 H |
| H | 17 | 20 | 0 | 19 | 12 H |
| I | 21 | 0 | 14 | 22 | 10 I |
| J | 9 | 11 | 7 | 10 | 0 – |
| K | 24 | 22 | 22 | 0 | 6 K |
| L | 29 | 33 | 9 | 9 | 15 K |

JA delay is 8 | JI delay is 10 | JH delay is 12 | JK delay is 6

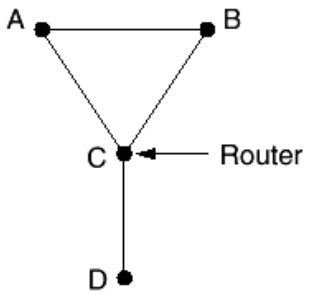
Vectors received from J's four neighbors

New routing table for J

(b)

# Distance vector routing

- Count-to-infinity problem:
  - Good news propagates much faster than bad news
- Split Horizon hack: (one of many efforts to reduce the impacts of this problem)
  - C does not report B the distance of node X for which packets have to flow through B
  - C tells D the truth about the distance to A, but tells B that CA= ∞
  - When AB fails, the bad news propagates as quickly as the good news
- Failure of Split Horizon
  - Init: CD=1, AD=2, BD=2, AB=1
  - CD fails, A and B tell C they can't reach D, so C sets CD=∞
  - Unfortunately, A hears from B that BD=2, so it sets AD=3 => wrong!




| A | B | C | D | E | |
|---|---|---|---|---|---|
| ∞ | ∞ | ∞ | ∞ | Initially |
| 1 | ∞ | ∞ | ∞ | After 1 exchange |
| 1 | 2 | ∞ | ∞ | After 2 exchanges |
| 1 | 2 | 3 | ∞ | After 3 exchanges |
| 1 | 2 | 3 | 4 | After 4 exchanges |

(a)

| A | B | C | D | E | |
|---|---|---|---|---|---|
| 1 | 2 | 3 | 4 | Initially |
| 3 | 2 | 3 | 4 | After 1 exchange |
| 3 | 4 | 3 | 4 | After 2 exchanges |
| 5 | 4 | 5 | 4 | After 3 exchanges |
| 5 | 6 | 5 | 6 | After 4 exchanges |
| 7 | 6 | 7 | 6 | After 5 exchanges |
| 7 | 8 | 7 | 8 | After 6 exchanges |
| ⋮ | | | | |
| ∞ | ∞ | ∞ | ∞ | |

(b)

# Link State Routing

- Solves slow convergence problem of the distance vector approach
- Idea:
  - Send all nearest neighbor information to all other routers
  - Each router uses this information to know the topology and determines shortest path to all nodes
- Link State Routing: 4 steps
  - Discover neighbors (and their IDs)
  - Measure delay (or cost) to reach neighbours
  - Send link state packets to all other routers
  - Compute topology and shortest path to every router
- Perform last three steps at
  - Regular intervals
  - When major event happens (link / node going down or coming up )

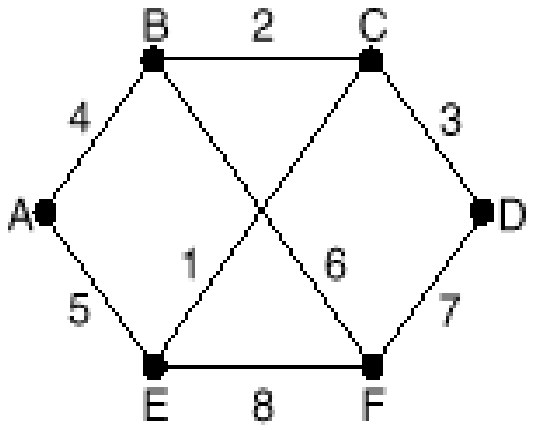
# Link State Routing

- Discover neighbor:
  - After booting, send HELLO packet to all neighbors
  - Neighbor should respond telling its identity
- Measure delay:
  - Send ECHO packet; neighbors respond immediately
  - Calculate trip time = delay/2
  - Average of multiple measurements may be used

# Link State Routing

- Sending link state packets:
  - Make link state packet
  - Use flooding to distribute packets to all routers
  - Use sequence numbers to reduce traffic
  - Routers keep (source, seq#) pairs and discard lower sequence numbers
  - Use age fields to handle corrupted sequence numbers
  - Age is decremented in regular intervals
  - Information is discarded when age is zero

# Link State Routing




| Source | Seq. | Age |
|---|---|---|
| A | 21 | 60 |
| F | 21 | 60 |
| E | 21 | 59 |
| C | 20 | 60 |
| D | 21 | 59 |

Link State Packets

| A | |
|---|---|
| Seq. | |
| Age | |
| B | 4 |
| E | 5 |

| B | |
|---|---|
| Seq. | |
| Age | |
| A | 4 |
| C | 2 |
| F | 6 |

| C | |
|---|---|
| Seq. | |
| Age | |
| B | 2 |
| D | 3 |
| E | 1 |

| D | |
|---|---|
| Seq. | |
| Age | |
| C | 3 |
| F | 7 |

| E | |
|---|---|
| Seq. | |
| Age | |
| A | 5 |
| C | 1 |
| F | 8 |

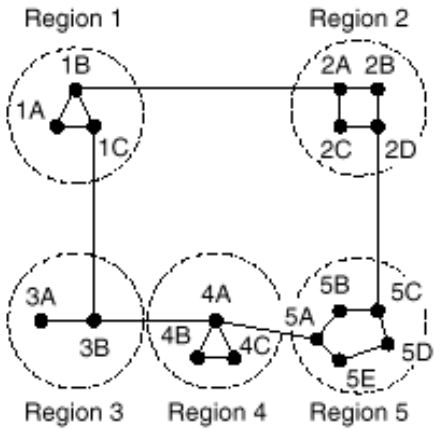
| F | |
|---|---|
| Seq. | |
| Age | |
| B | 6 |
| D | 7 |
| E | 8 |

# Link State Routing

- Route Calculation
  - Compute new routes (when packets from all sources have been received):
  - use shortest path algorithm on each node (e.g., Dijkstra's algorithm)
- Link state routing used widely.
  - Example: OSPF (open shortest path first) routing which is used in the Internet
- Problem with Link State Routing:
  - large memory requirements ~ O (NK), where: N=number of routers K=network node degree

# Hierarchical Routing

- Purpose: Reduce routing table size and routing computation time
- Solution: use multi-level topology




(a)

Full table for 1A

| Dest. | Line | Hops |
|---|---|---|
| 1A | – | – |
| 1B | 1B | 1 |
| 1C | 1C | 1 |
| 2A | 1B | 2 |
| 2B | 1B | 3 |
| 2C | 1B | 3 |
| 2D | 1B | 4 |
| 3A | 1C | 3 |
| 3B | 1C | 2 |
| 4A | 1C | 3 |
| 4B | 1C | 4 |
| 4C | 1C | 4 |
| 5A | 1C | 4 |
| 5B | 1C | 5 |
| 5C | 1B | 5 |
| 5D | 1C | 6 |
| 5E | 1C | 5 |

(b)

Hierarchical table for 1A

| Dest. | Line | Hops |
|---|---|---|
| 1A | – | – |
| 1B | 1B | 1 |
| 1C | 1C | 1 |
| 2 | 1B | 2 |
| 3 | 1C | 2 |
| 4 | 1C | 3 |
| 5 | 1C | 4 |

(c)

# Broadcast Routing Methods

- Send lots of individual packets
  - Lots of traffic
  - Source needs to know all destination addresses
  - Generally, not used
- Flooding
  - Generates too many packets
- Multi-destination routing
  - Packet contains list of destinations
  - Router splits packet among output lines and updates destination list
  - Again, all destination addresses must be known!

# Broadcast Routing Methods

- Sink tree (= *spanning tree*)
  - Each packet is sent along the lines of the sink tree of the sender
  - Most efficient method for broadcast to all
  - Sink tree is not always known (e.g., not when distance vector routing is used; link state routing does have this knowledge!)

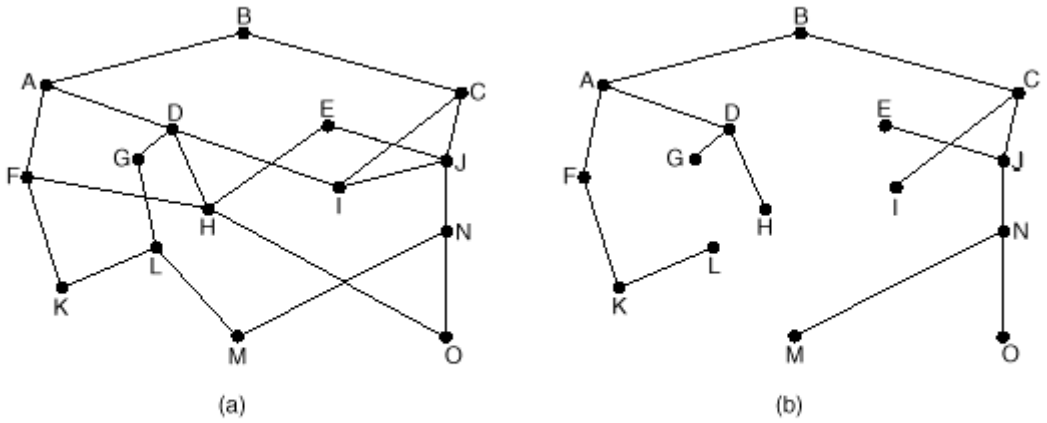



**Fig. 5-5.** (a) A subnet. (b) A sink tree for router *B*.

# Broadcast Routing: Reverse Path Forwarding

- Reverse path forwarding
  - Does not require network knowledge
  - Easy to implement
  - Transmission automatically finishes
  - Reasonably efficient
  - No list of destinations needed
  - No special mechanism needed to stop the forwarding
- Each router knows the normal line for sending packets to all other routers
- When a packet is received from a router, it is checked to see if it is coming from the “normal” line.
- Send packet to all neighbours, but discard incoming packets from ‘wrong’ lines
- Packets from A arriving at B on line L is ‘wrong’ if B does not send to A at line L
- The idea is that if a packet arrives at the wrong line, it probably did not follow the shortest route
- The overall total generated traffic is slightly more than required, but efficiency is reasonable

# Broadcast Routing: Reverse Path Forwarding

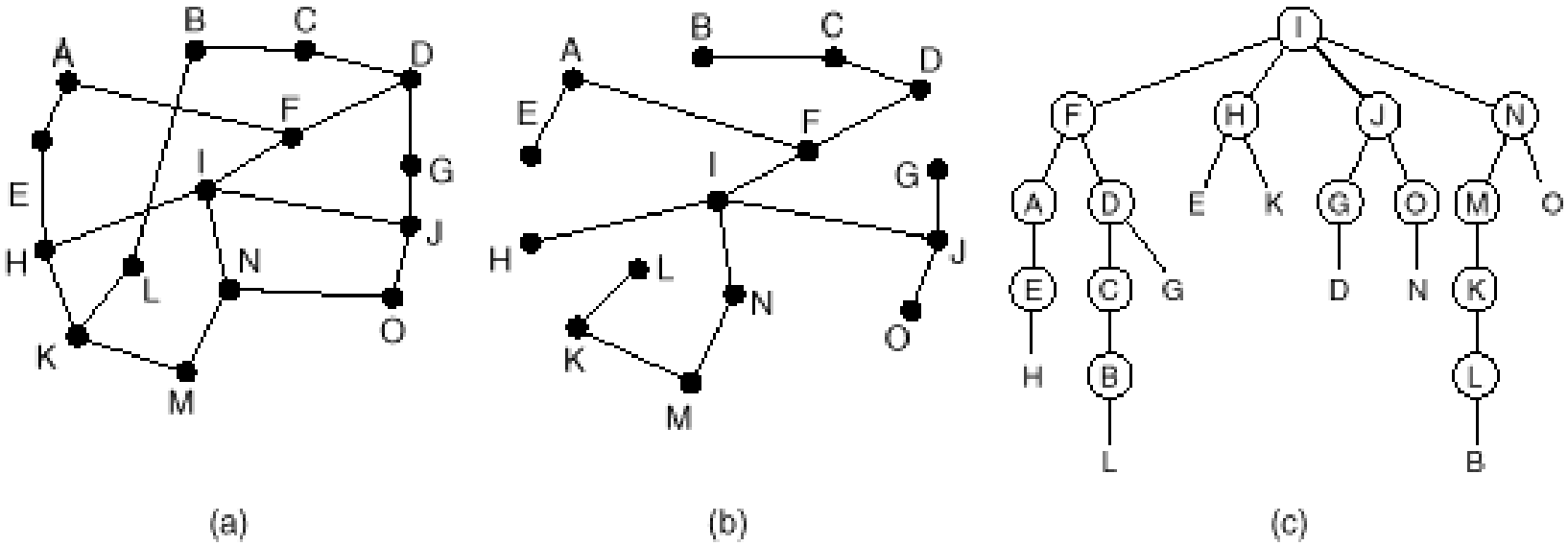
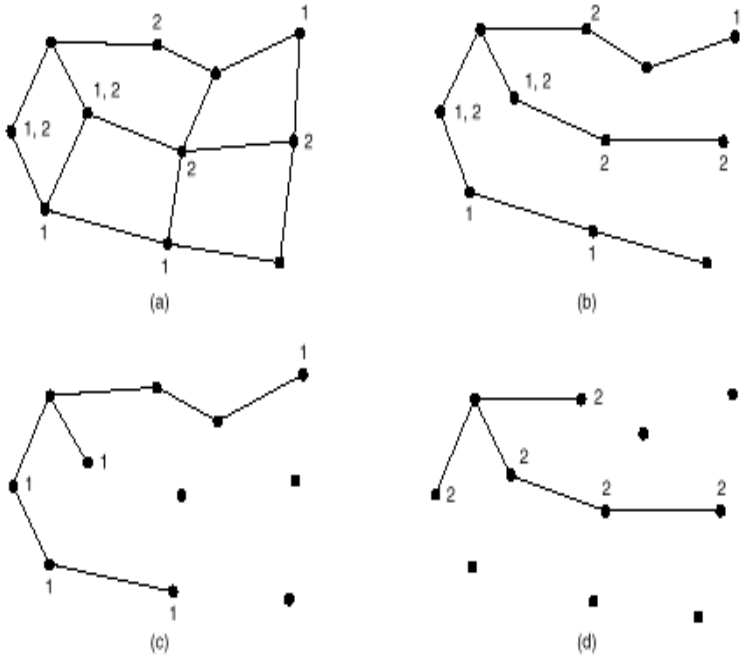



**Fig. 5-20.** Reverse path forwarding. (a) A subnet. (b) A spanning tree. (c) The tree built by reverse path forwarding.

# Multicast Routing

- Multicast: Sending packets to a group of nodes => Group management required
- Routers must know to which groups their hosts belong
- Each router computes spanning tree for each supported group
- Multicast packets are then forwarded to appropriate spanning trees only
- When multiple spanning trees share some paths, trees need to be pruned to limit the packet forwarding only to the relevant routers

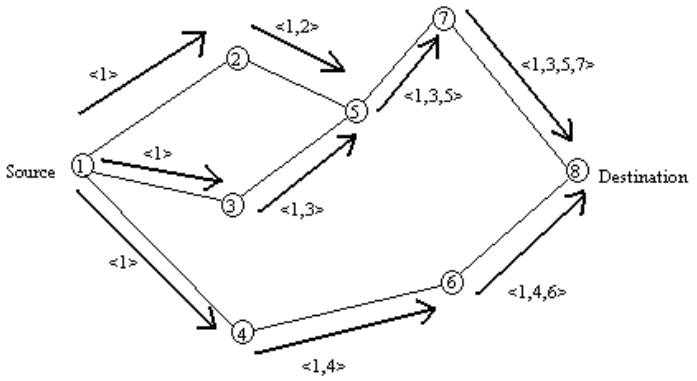


# Ad-Hoc Routing

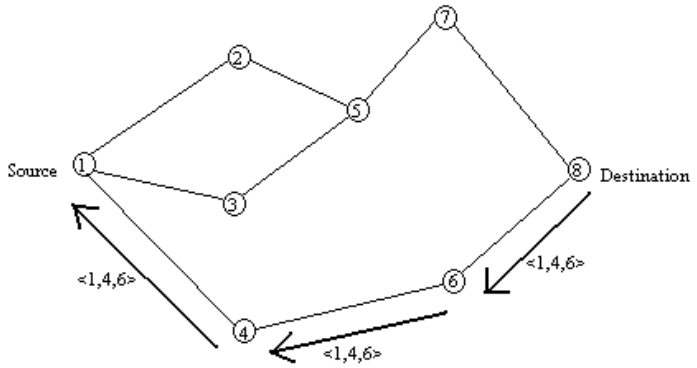
- Ad Hoc Networks: Network of nodes that just happens to be near each other. Nodes can move and start/stop working arbitrarily. Examples:
  - Military vehicles on battlefield. (No infrastructure.)
  - A fleet of ships at sea. (All moving all the time)
  - A gathering of people with notebook computers.
- Fixed-wire Routing techniques can't be used anymore
- Several methods have been proposed for ad hoc routing. Examples are:
  - Dynamic Source Routing (DSR)
  - Ad hoc On-demand Distance Vector (AODV) Routing
- Key issue for Ad Hoc routing methods is that routes are calculated on-demand to save bandwidth and battery life

# Dynamic Source Routing (DSR)

- On Demand routing protocol designed to restrict the routing overhead in ad hoc networks
- Route Request (RRQ) packets are generated by source and flooded in the network to inquire about a possible route to destination
- Each Route Request carries a sequence number generated by the source node and the path it has traversed.



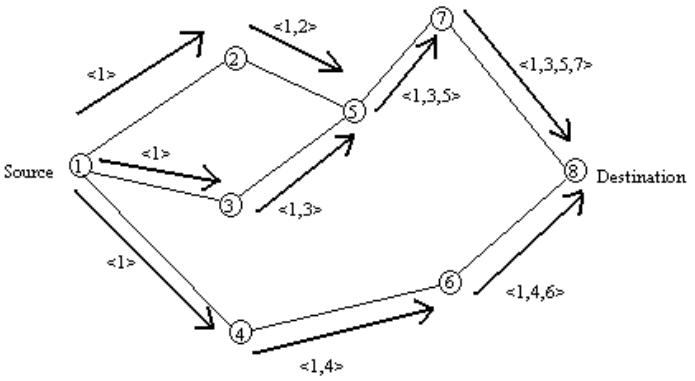

(a) Building Record Route during Route Discovery



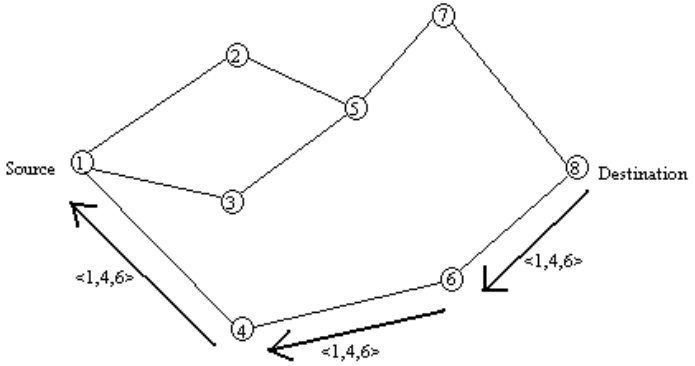

(b) Propogation of Route Reply with the Route Record

# Dynamic Source Routing (DSR)

- Each node, upon receiving a RRQ packet:
  - Checks the sequence number, records it and checks it to avoid forwarding duplicates
  - Checks time to live (TTL) of the packet to avoid forwarding old packets.
  - Adds its own ID to the packet
  - Rebroadcasts the packet to its neighbors
- Destination node replies to the source node using the trace ID of the first arrived RRQ packet.
- Source sends its data packets with a header containing the entire routing path. (That is why it is called source routing)




(a) Building Record Route during Route Discovery



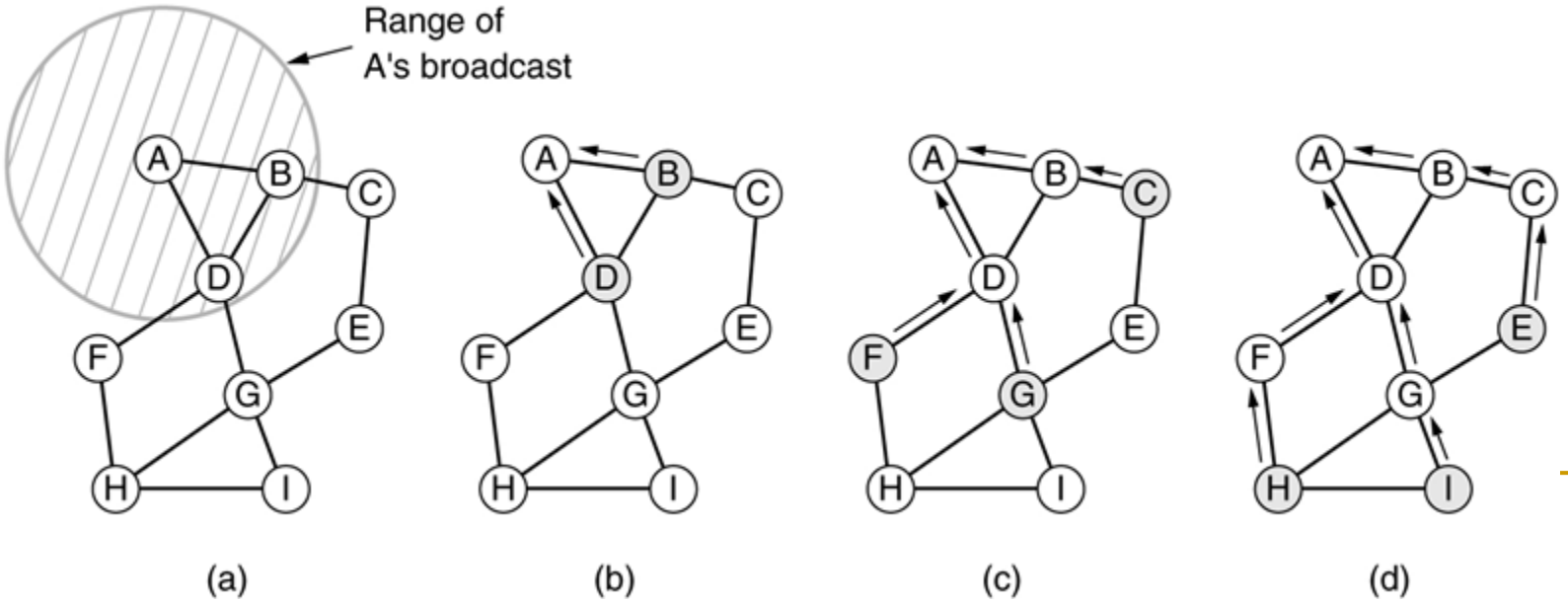

(b) Propogation of Route Reply with the Route Record

# Ad-hoc On-demand Distance Vector (AODV) Routing

- Node A constructs a ROUTE REQUEST (RRQ) packet and broadcasts it.
- RRQ Contains:
  - Request ID: a local counter maintained separately by each node, incremented for each RRQ packet it sends
  - {Source Address, Request ID} uniquely identify the RRQ packet
  - Destination Sequence Number (Dest_Seq#): Number identifying the last known value of Route Reply (RRP) from the desired destination
- Each node that hears the RRQ discards it if it is a duplicate
- Then it looks in the route table. If there is a fresh route to the destination, it sends a RRP to the source with the info (Saying: Send the packet to me and I'll route it!)
- Fresh route: Dest_Seq# of the RRQ is larger than the Dest_Seq# that exist in its routing table
- If it does not know the route, it increments the Hop count and re-broadcasts it.
- It also extracts the source data and uses it for reverse routing (Everybody knows where is A)

# AODV

- The message goes up to the destination (In this example, node I)
- I builds an RRP, where the Hop count, source and destination address are copied from RRQ,
- Dest_Seq# is copied from its current local counter value
- Life Time: Controls how long the route is valid
- Packet is sent back to the source address.
- In the reverse path, every node  knows how to get back to the source.

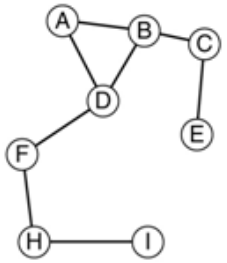


# AODV Route Maintenance

- Routing info should be updated regularly because of node's changing status
- Periodically, each node sends a Hello message to its neighbors
- If no response to Hello or no response to a data packet, assume the neighbor is dead
- Active neighbors of node D: The neighbors that depend on D for routing their packets.
- When a neighbor goes down, the affected active neighbors are informed of the event. Local routing table is modified to delete the route passing through the dead node.
- Each of those nodes in turn would notify their affected active neighbors of the event.
- The routing tables are modified all over the nodes and invalid routes are purged.

| Dest. | Next hop | Distance | Active neighbors | Other fields |
|---|---|---|---|---|
| A | A | 1 | F, G | |
| B | B | 1 | F, G | |
| C | B | 2 | F | |
| E | G | 2 | | |
| F | F | 1 | A, B | |
| G | G | 1 | A, B | |
| H | F | 2 | A, B | |
| I | G | 2 | A, B | |

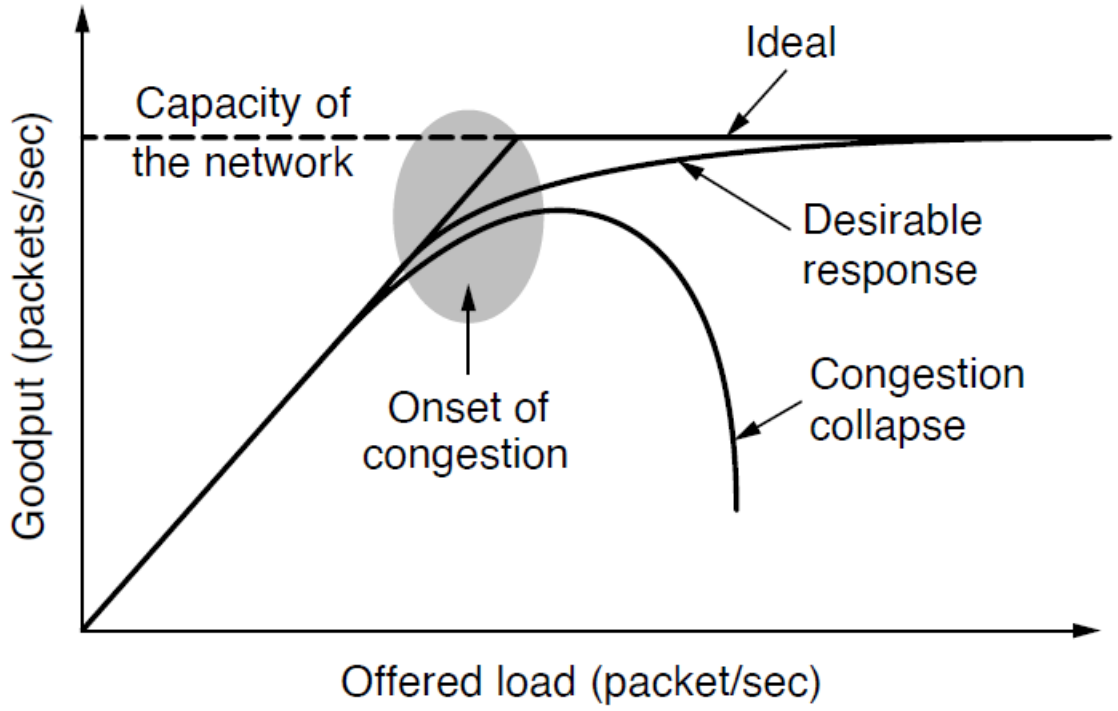
(a)




(b)

# Congestion control

- Problem: Network performance severely degrades at high load
- Examples of Reasons:
  - N to 1 communication: ***Hot-spot***
  - Sudden burst of traffic

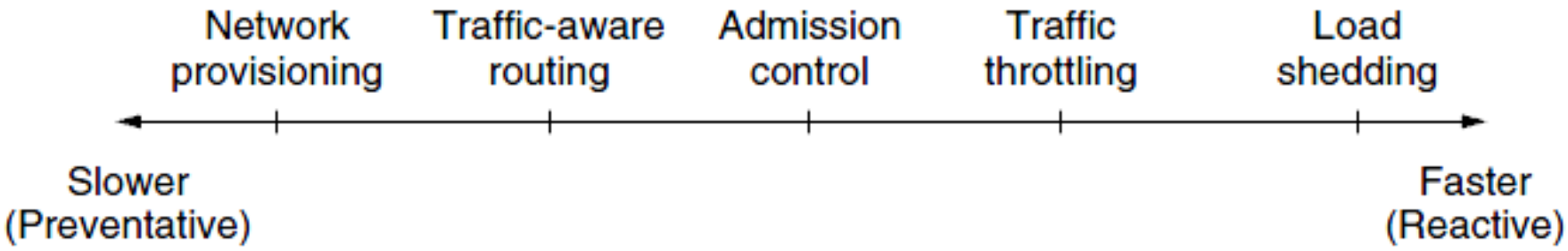


# Congestion Control

- What is the difference between congestion control and flow control
  - Congestion control
    - Global issue (network)
    - Indirect feedback
    - Sometimes slowdown messages are used
  - Flow control
    - Local issue (line, point-to-point connection )
    - Direct feedback from neighbor

# Congestion Control

- Time Scale of applying congestion control techniques



# Principles of Congestion Control

- Systems can be controlled by an open loop  or a closed loop mechanism
- Open loop: no feedback; solve problems at design time (to make sure they never occur; worst case design)
- Open loop issues:
  - No regard to current state of network
  - When to accept or discard new traffic? (Admission Control)
- Closed loop: Monitor the network on congestion
  - Metrics: % discarded packets, queue length, # timed out packets, average delay
- Pass this information to right place
  - Feedback packets: increase load
  - Reserved bit field in header of every packet
  - Probe packets can be sent periodically (ask about congestion)
- Adjust system operation
  - Avoid oscillation (proper response time needed)
  - Increase number of resources
  - Decrease load

# Congestion Prevention

- Open loop policies can be applied at different layers
- ***Data link layer policies***
  - Retransmission creates extra packets
    - Tune time out time
    - Use selective repeat instead of go back n
  - Acknowledgement policy
    - Use piggyback
  - Tight flow control (small window)
- ***Network layer policies***:
  - Any congestion control policies work only with virtual circuits, e.g.. reserving space
  - Routing
    - Spread traffic
    - Avoid hot spots
  - Discard policies
    - Remove long life-time packets
- ***Transport layer policies***:
  - Similar as previous policies
  - Choose proper time-out value for sender

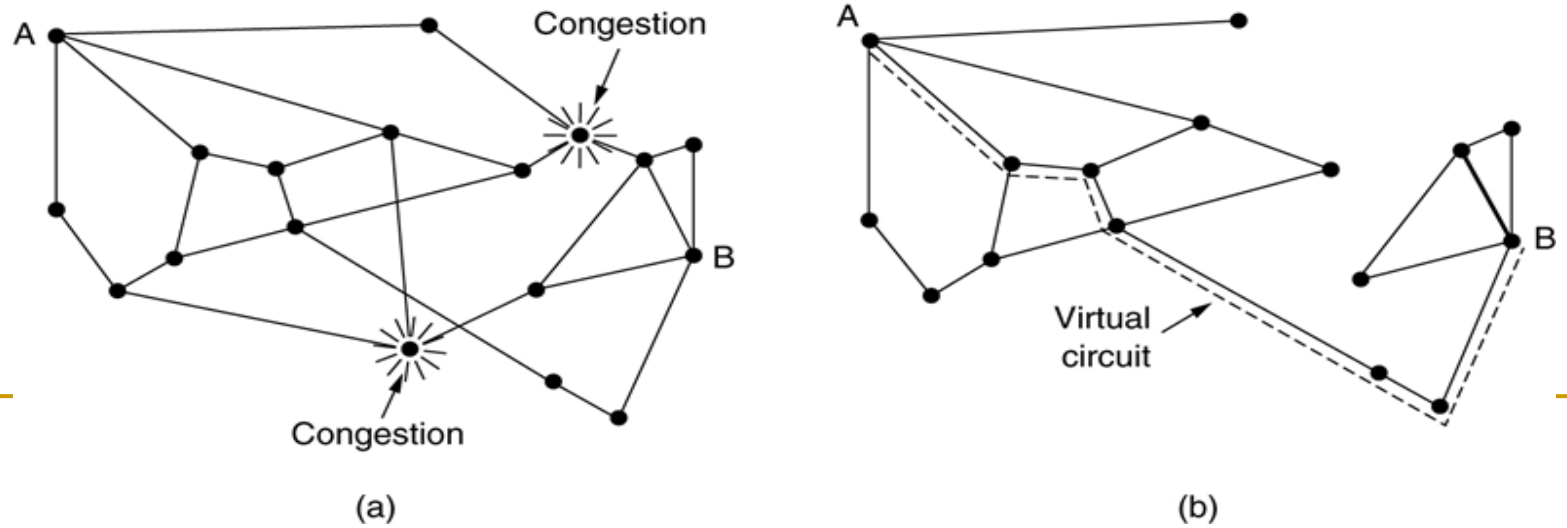
| Layer | Policies |
|---|---|
| Transport | • Retransmission policy<br>• Out-of-order caching policy<br>• Acknowledgement policy<br>• Flow control policy<br>• Timeout determination |
| Network | • Virtual circuits versus datagram inside the subnet<br>• Packet queueing and service policy<br>• Packet discard policy<br>• Routing algorithm<br>• Packet lifetime management |
| Data link | • Retransmission policy<br>• Out-of-order caching policy<br>• Acknowledgement policy<br>• Flow control policy |



**Fig. 5-23.** Policies that affect congestion.

# Congestion Control

- Congestion Control in Virtual Circuit subnets:
  - Use Admission Control to prevent congestion
    - If there is congestion, don't accept new traffic.
    - Reserve the necessary resource along the data path to prevent congestion
  - Negotiate a new VC that does not pass through the congested area



# Congestion Control in Datagram subnets

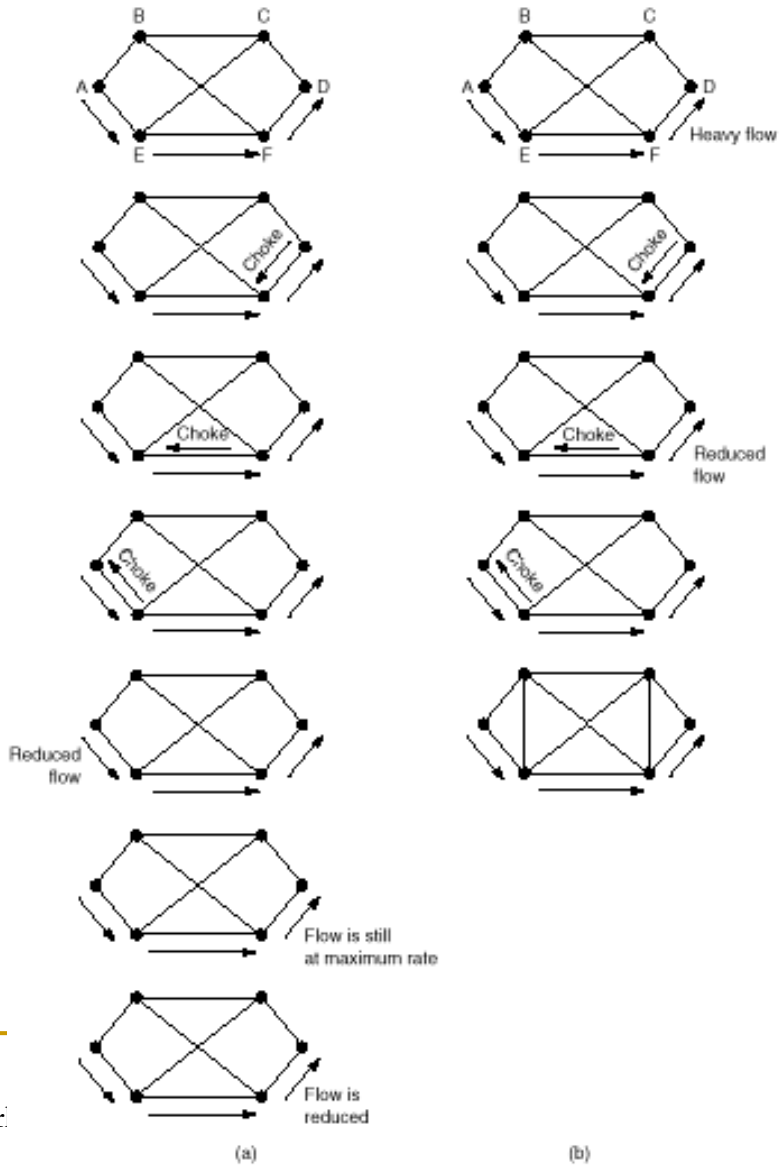
- Estimate congestion u by sampling certain activity *f* (e.g. Line utilization) on each outgoing line
- Calculate u using:

$$u_{new} = a u_{old} + (1-a) f$$

- Warning bit
  - Set a bit in the ip payload indicating that there is congestion along the data path. (Give warning when u > threshold)
  - The transport entity copies this bit in the acknowledge and send it back to the source
  - The source can estimate the congestion by measuring the fraction of ACKs it receive with a congestion bit indication

# Congestion Control in Datagram subnets

- Choke packets
  - Don’t wait for the destination to act. It may be too late!
  - Each router sends a choke packet to source when in warning state.
  - Source reduces the load by %X. waits for a period of time and if there was no more choke packets, goes back to normal operation
- Hop-by-hop choke packets
  - Choke packets may take a long time to cause effect
  - Solution: Choke packets should not effect the source host only, but also intermediate routers
  - Routers reduce traffic in certain direction
  - More buffer space required at routers !!



# Buffer Management

- Load Shedding
  - Allow routers to discard packets
  - What‘s the selection criterion ?
  - Use packet priority
    - Use priority field to decide which packet to drop
    - Users should pay more for higher priority packets

# Buffer Management

- It is best to drop packets as soon as there is congestion
- If there is congestion in a particular port, it is difficult to know whose fault it is (Or who is the source that has caused the congestion)
- Don’t send choke packets as it can add to the congestion
- TCP would slow down if packets are dropped (Reduced traffic rate from the source). This is implicit feedback from the network about its status.

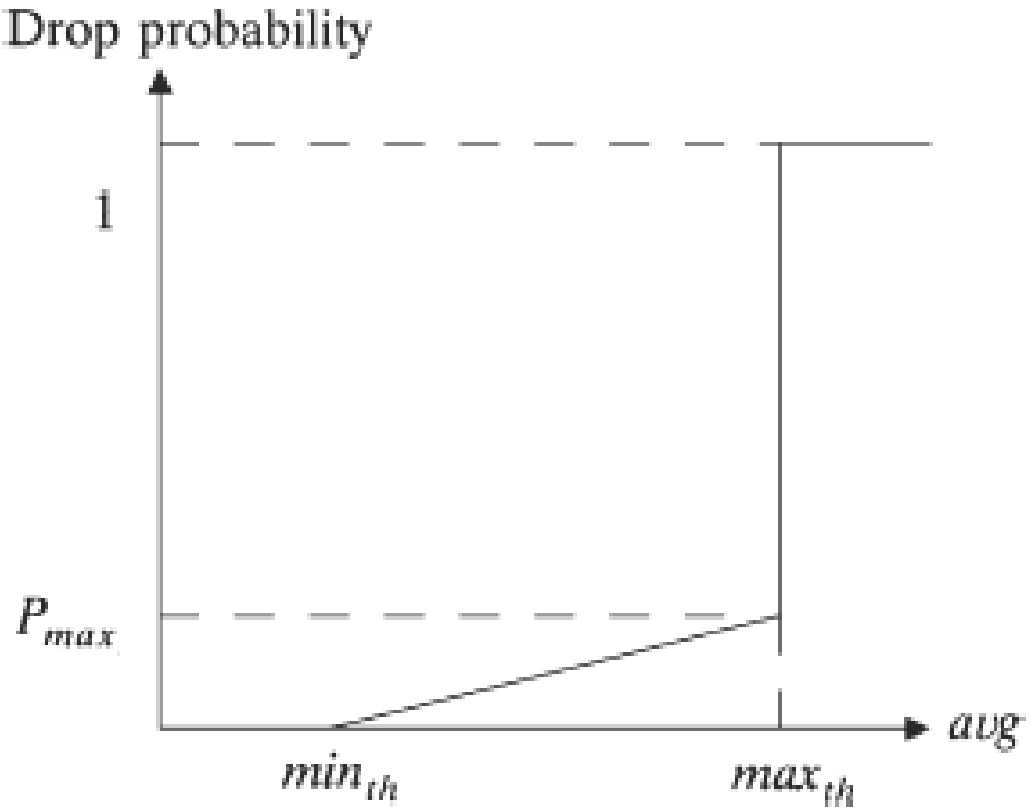
# Random Early Detection (RED)

- This is a proactive buffer management approach in which the router discards one or more packets before the buffer becomes completely full
- Average Queue Length (*avg*): Moving average of the Queue length in the past

$$avg(t+1) = avg(t) \times (1-\alpha) + L(t) \times \alpha$$

- Packets are marked with a probability P as a function of *avg*
- The probability that a packet is marked from a particular connection is roughly proportional to that connection's share of the bandwidth at the router.
- The minimum and maximum thresholds $min_{th}$ and $max_{th}$ are determined by the desired average queue size.
- The average queue size that makes the desired tradeoffs such as the tradeoff between maximizing throughput and minimizing delay depends on network characteristics

# Random Early Detection (RED)

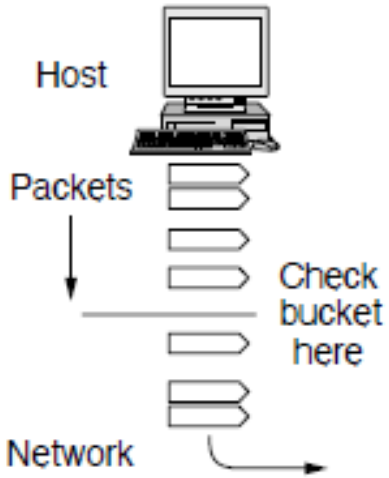


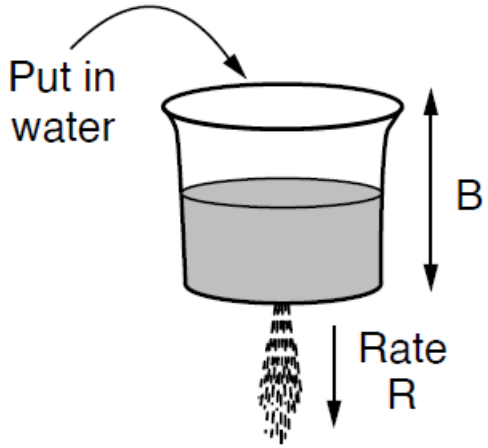
# Traffic Shaping

- Traffic shaping is used to avoid bursty traffic
  - Regulate the average rate on server side
  - Negotiate a contract between sender and network. (Service Level Agreement (SLA))
  - Specially important for real-time applications
- ***Traffic control***
  - Leaky bucket
  - Token bucket
  - Hybrid solution

# Traffic Shaping – Leaky Bucket

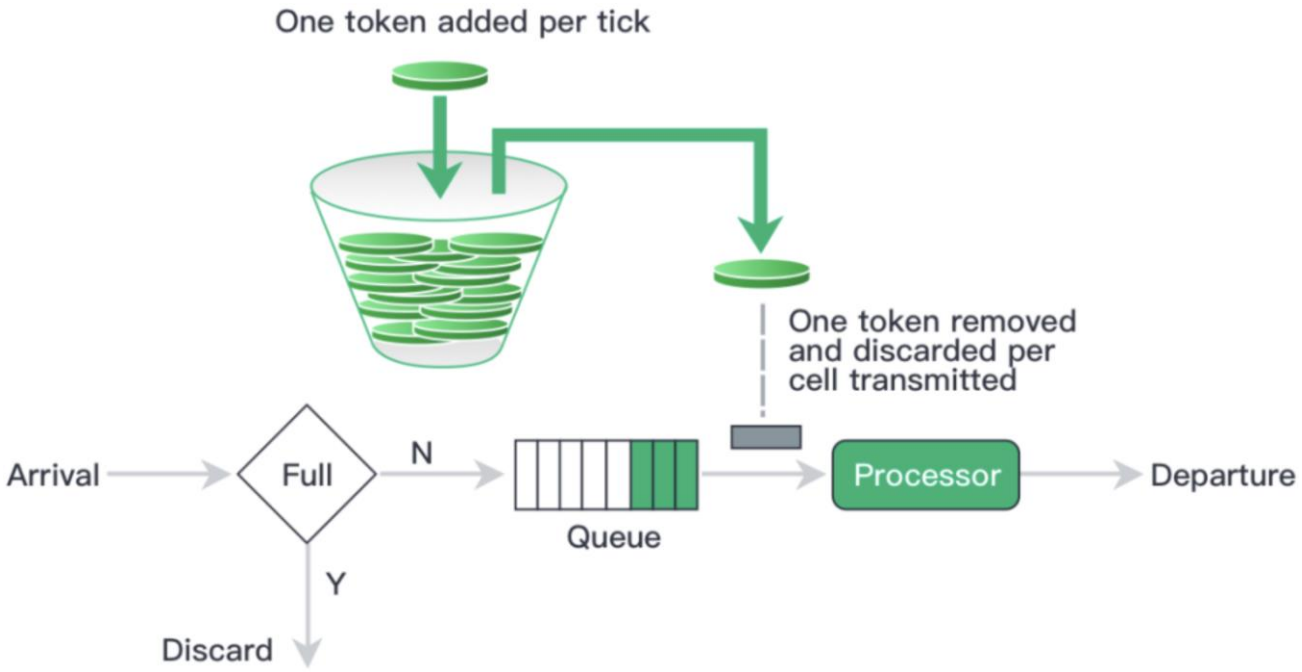
- Leaky bucket algorithm
  - Sender puts packets or bytes in a queue
  - When queue is full the input is discarded
  - Queue output data at constant packet or byte rate
  - Equal to a single-server queuing system with constant service time



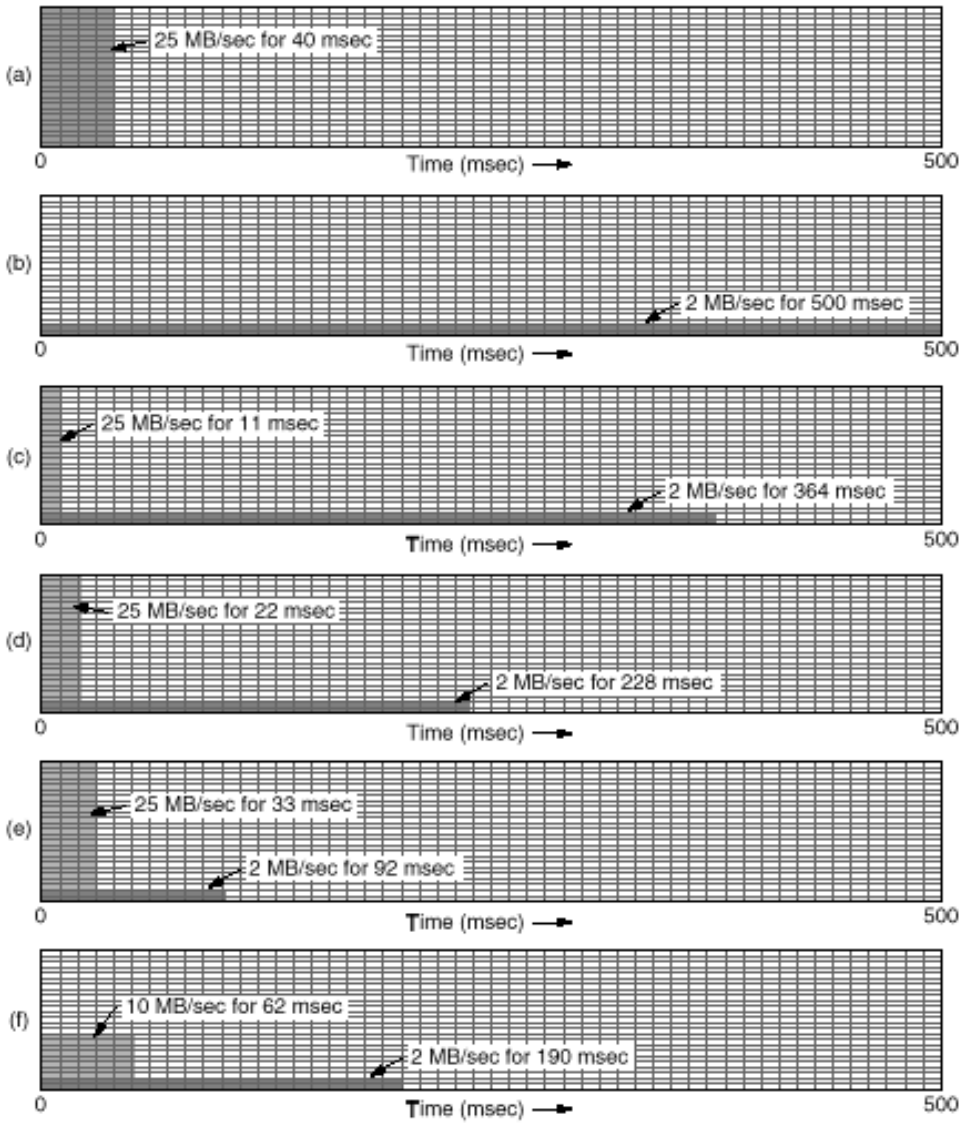
# Traffic Shaping -Token Bucket

- Token bucket algorithm
  - Bucket holds tokens, generated 1 per T sec. up till certain max. value
  - Idle host time builds up ‘transmission rights’ (= tokens)



# Traffic Control Example

- (a) Input to a leaky bucket.
- (b) Output from a leaky bucket.
- Output from a token bucket with capacities of
  - (c) 250 KB,
  - (d) 500 KB,
  - (e) 750 KB,
  - (f) Output from a 500KB token bucket feeding a 10-MB/sec leaky bucket.



# Resource Reservation

- Resource Reservation
  - To provide QoS, some sort of fixed path for the flow seems to be necessary
  - The idea: Reserve critical resources along the path of a flow
  - Critical Resources:
    - Bandwidth
    - Buffer space
    - CPU processing cycle (This can be tricky)

- Example for resource relation:
  - Mean processing capacity of the processor, $\mu=10^6$ packets/sec
  - Mean arrival rate of packets to the processor (Poisson distribution), $\lambda=0.95\times 10^6$ packets/sec
  - Mean delay experienced by a packet is:

$$\rho = \frac{\lambda}{\mu} \Rightarrow T = \frac{1}{\mu} \times \frac{1}{1-\rho}$$

$$T = 1\mu\text{sec} \times \frac{1}{1-0.95} = 20\mu\text{sec}$$

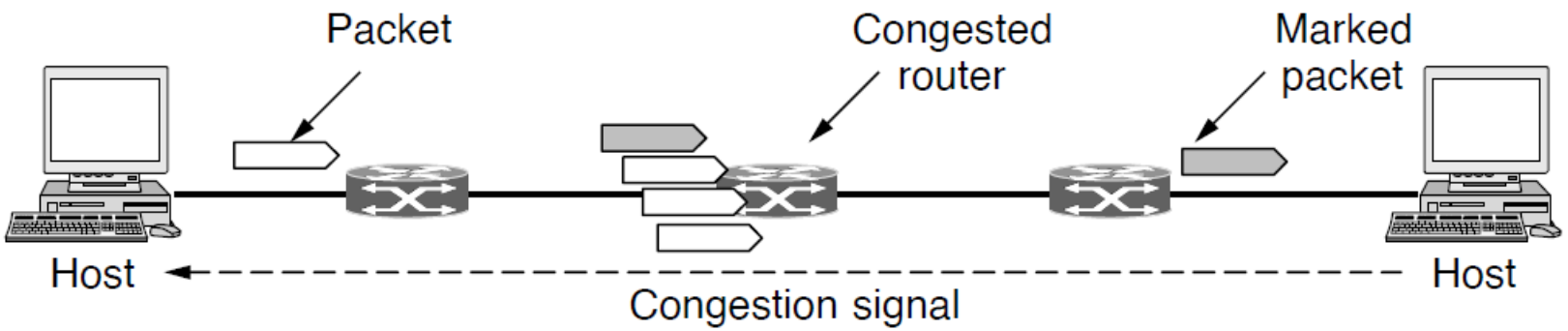
  - So, the port bandwidth, buffer space requirement and processing capacity all are related and impact QoS parameters.

# Admission Control

- The flow can be specified in multiple ways.
- Sender generates the flow spec and sends it to the destination
- Routers along a flow-path will study the flow parameters, reserve enough resources and reduce the requested level of service if they can't meet it.
- At the other end, a decision will be made on the provided QoS to the flow (control the flow admission by reservation of resources)
- An example of flow specification (RFC2210):
  - Token Bucket Rate (Bytes/sec) (Maximum sustained transmitter rate)
  - Token bucket size (Bytes) (Maximum burst size)
  - Peak data rate (Bytes/sec)
  - Minimum packet size (Bytes)
    - CPU cycles are used for each processed packet regardless of its size. This will help the router in its CPU allocation decision
  - Maximum packet size (Bytes)
    - Used for buffer allocation decision

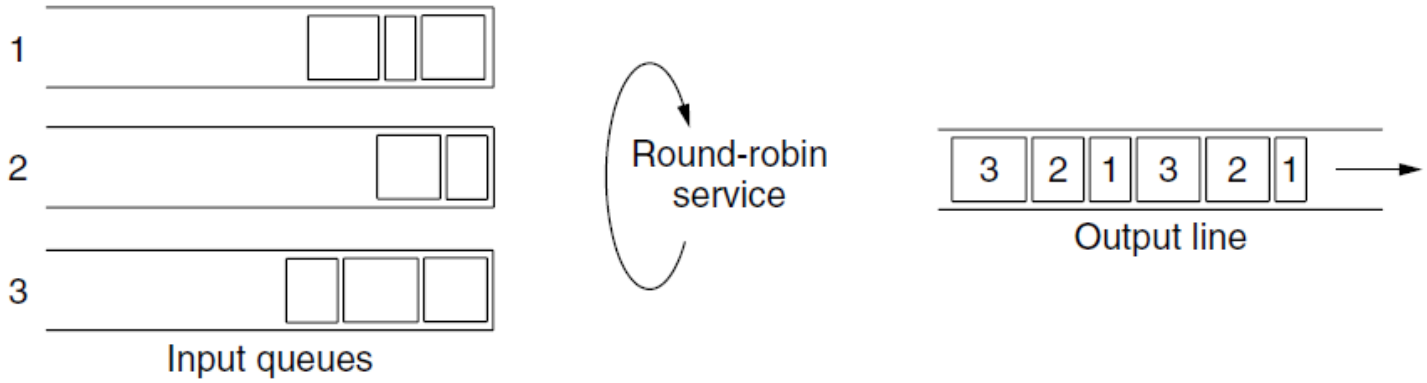
# Explicit Congestion Notification

- If there is a congested router, it marks the IP packets.
- Receiver sends a congestion notification on the reverse path to notify the sender so it can reduce its traffic rate temporarily.



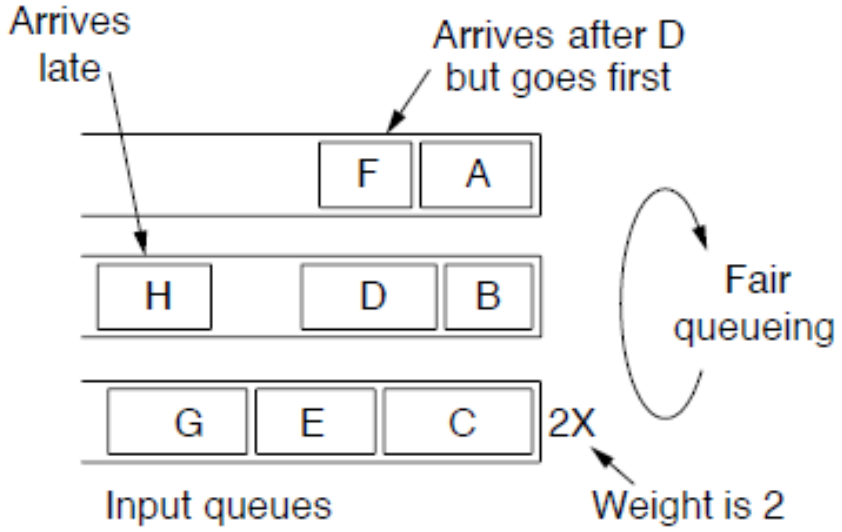
# Fair Queuing

- If multiple sources compete for the same destination port of a router, then the most aggressive sender can block all others
- To remedy that, Fair Queuing algorithm was proposed:
  - Go round robin around among senders, send one packet of each source on the destination port.
- What if packet sizes are different?
  - Go round robin, byte by byte on packets. When a packet got all its credits, send it out



# Weighted Fair Queuing

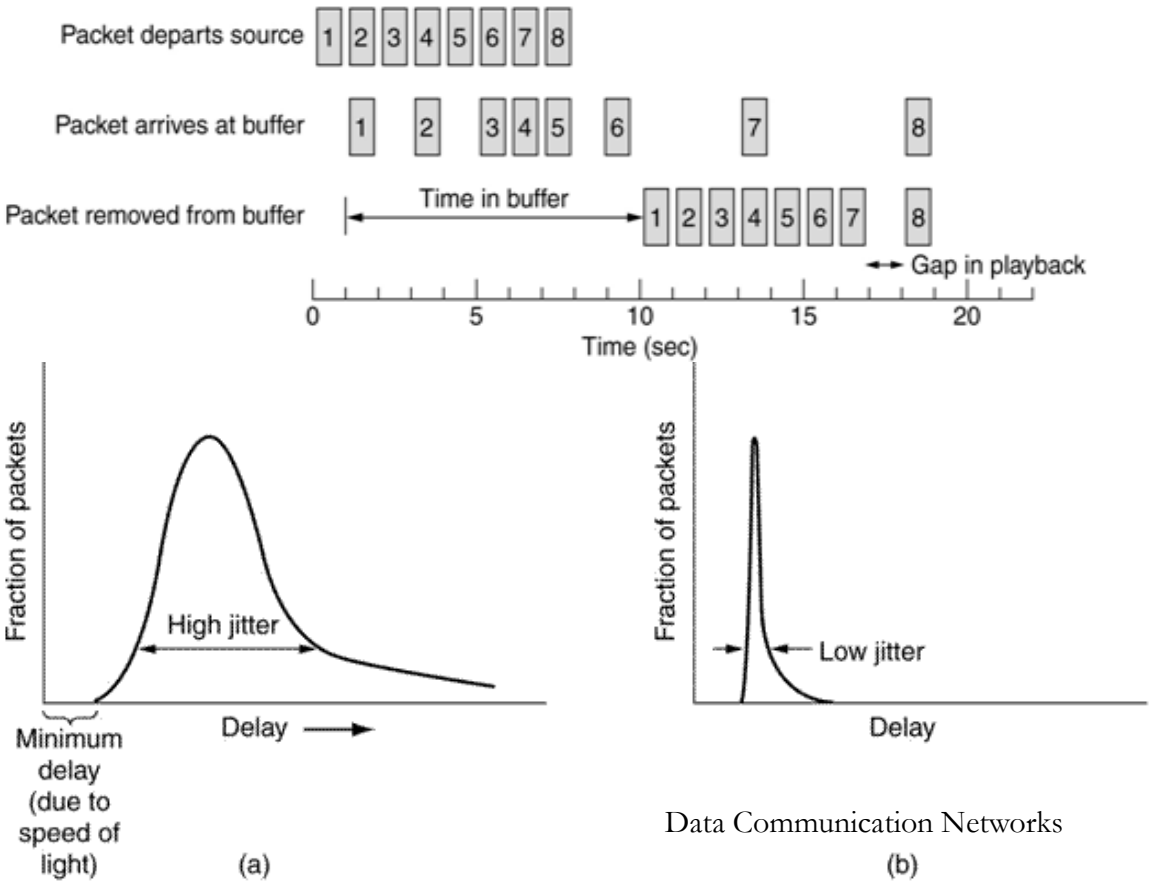
- We can assign weight to queues and consider the length of packets to allow fair service.
- Priority of service is determined by calculating the virtual finish time of each packet.




| Packet | Arrival time | Length | Finish time | Output order |
|---|---|---|---|---|
| A | 0 | 8 | 8 | 1 |
| B | 5 | 6 | 11 | 3 |
| C | 5 | 10 | 10 | 2 |
| D | 8 | 9 | 20 | 7 |
| E | 8 | 8 | 14 | 4 |
| F | 10 | 6 | 16 | 5 |
| G | 11 | 10 | 19 | 6 |
| H | 20 | 8 | 28 | 8 |

# Jitter Control

- For audio/video streaming applications, variance of delay (Jitter) should be controlled
- Buffering at the destination can be used to reduce the effects of network-induced Jitter on the packet arrival time.



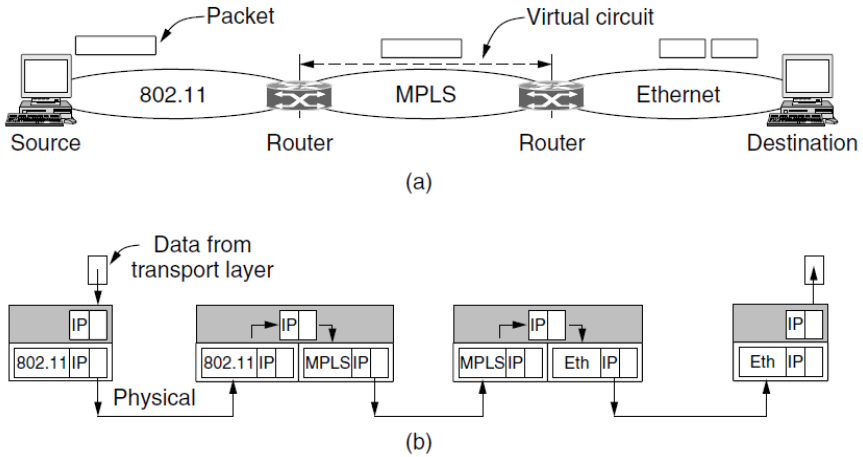
# Quality of Service (QoS)

- QoS parameters:
  - Reliability
  - Rate/Throughput
  - Delay/Jitter
- Applications require different types of QoS

| Application | Reliability | Delay | Jitter | Bandwidth |
|---|---|---|---|---|
| E-mail | High | Low | Low | Low |
| File transfer | High | Low | Low | Medium |
| Web access | High | Medium | Low | Medium |
| Remote login | High | Medium | Medium | Low |
| Audio on demand | Low | Low | High | Medium |
| Video on demand | Low | Low | High | High |
| Telephony | Low | High | High | Low |
| Videoconferencing | Low | High | High | High |

# Internetworking

- How to connect and route on networks using different protocol stacks
  - Different protocols will always exist (vendor policies)
  - Cheaper networks encourage variety (it becomes a departmental decision)
- Interconnection of different networks is called an **internet**



# Internetworking

- Various boxes connecting networks or segments are used:
- Physical layer: Repeaters
  - Copy individual bits between cable segments
- Data link layer: Bridges, L2 Switches
  - Store and forward data link frames between LANs
- Network layer: Multi-protocol routers
  - Forward packets between dissimilar networks (often called gateways)
- Transport layer: Gateways
  - Connect byte streams in the transport layer e.g. TCP socket connection
- Application gateways:
  - Proxy servers
  - Firewall

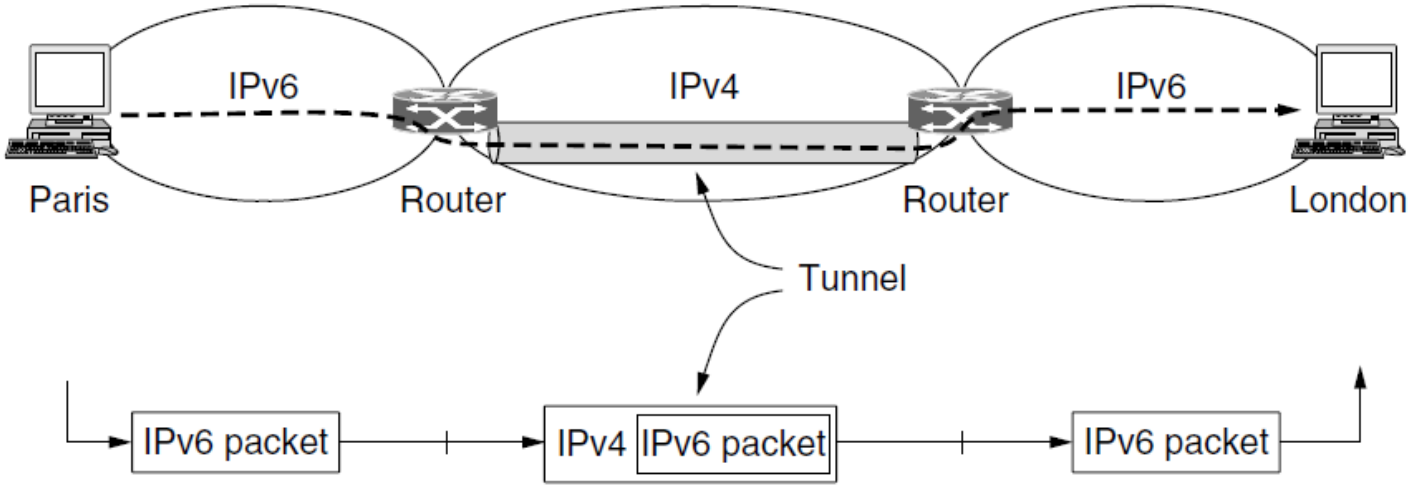
# Internetworking

- How networks differ?
  - Connectionless service may change the order of packets
  - Protocol conversion: e.g.. From IP to IPX
  - Address conversion (range, hierarchy)
  - Quality of service may or may not be provided
  - Different packet length
  - Different methods (if at all) for flow control, security, accounting, etc.

| Item | Some Possibilities |
|---|---|
| Service offered | Connection oriented versus connectionless |
| Protocols | IP, IPX, SNA, ATM, MPLS, AppleTalk, etc. |
| Addressing | Flat (802) versus hierarchical (IP) |
| Multicasting | Present or absent (also broadcasting) |
| Packet size | Every network has its own maximum |
| Quality of service | Present or absent; many different kinds |
| Error handling | Reliable, ordered, and unordered delivery |
| Flow control | Sliding window, rate control, other, or none |
| Congestion control | Leaky bucket, token bucket, RED, choke packets, etc. |
| Security | Privacy rules, encryption, etc. |
| Parameters | Different timeouts, flow specifications, etc. |
| Accounting | By connect time, by packet, by byte, or not at all |

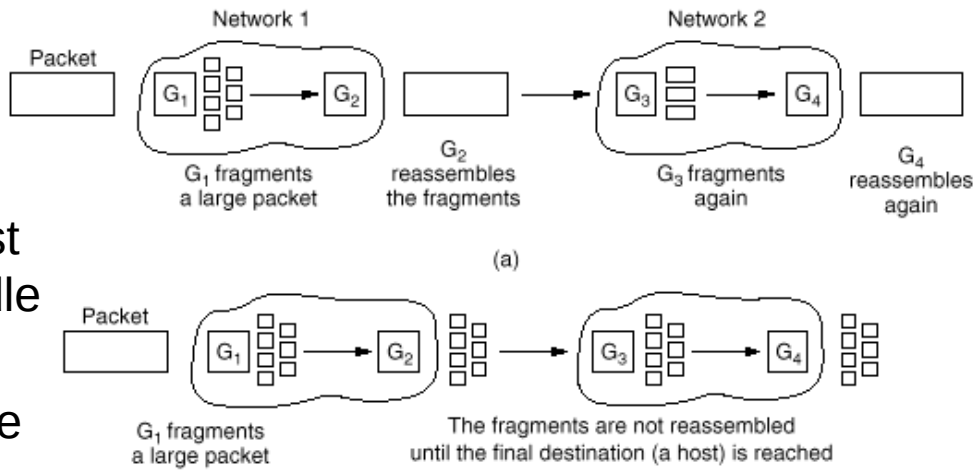
# Tunneling

- Tunnelling
  - If source and destination hosts are on same type of network but there is a different network in between, *tunnelling* can be used
  - Packet encapsulation performed by multi protocol router



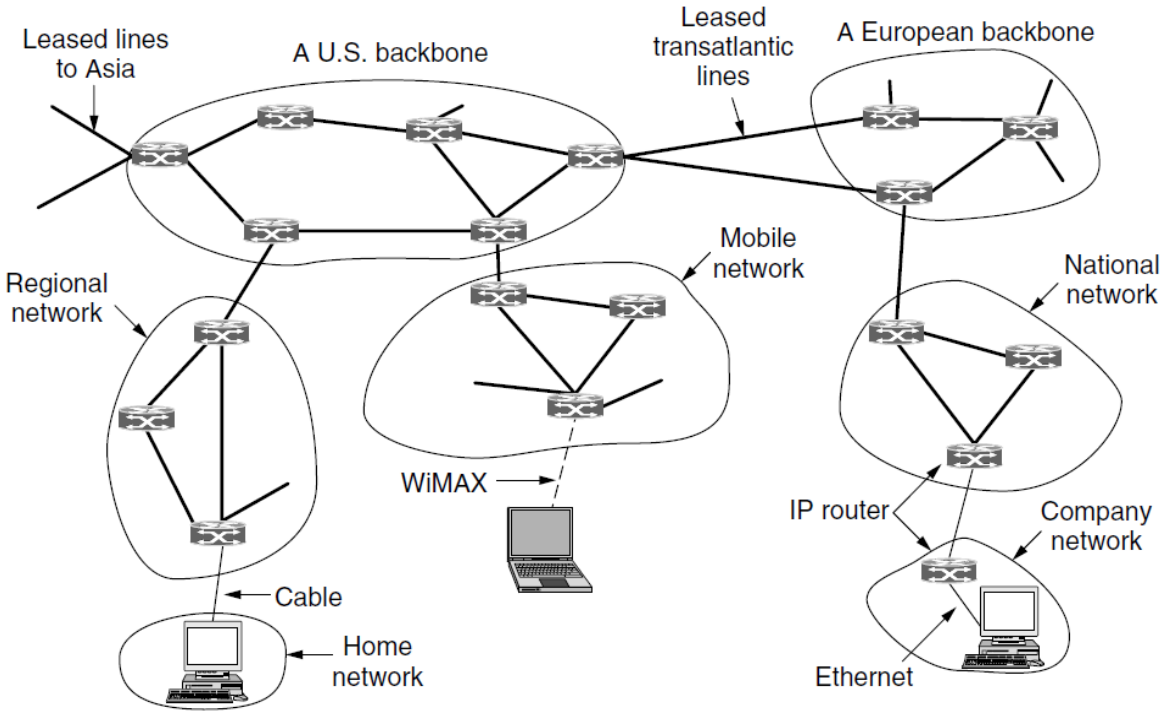
# Fragmentation

- All networks impose maximum packet payload size (Example: 65,515 bytes in IP)
- Multi protocol routers (gateways) may fragment packets. Two options:
  - Transparent :
    - Packet leaves network unfragmented and fragments have to use same route
  - Non-transparent
    - Fragments reassembled at destination host
    - All hosts must support this assemble feature
    - Added overhead for all packets
- Problem: if one fragment lost, end-to-end retransmission required
- Another approach: choose smallest unit which every network can handle
- Always chop large packets to this size and follow the same procedure



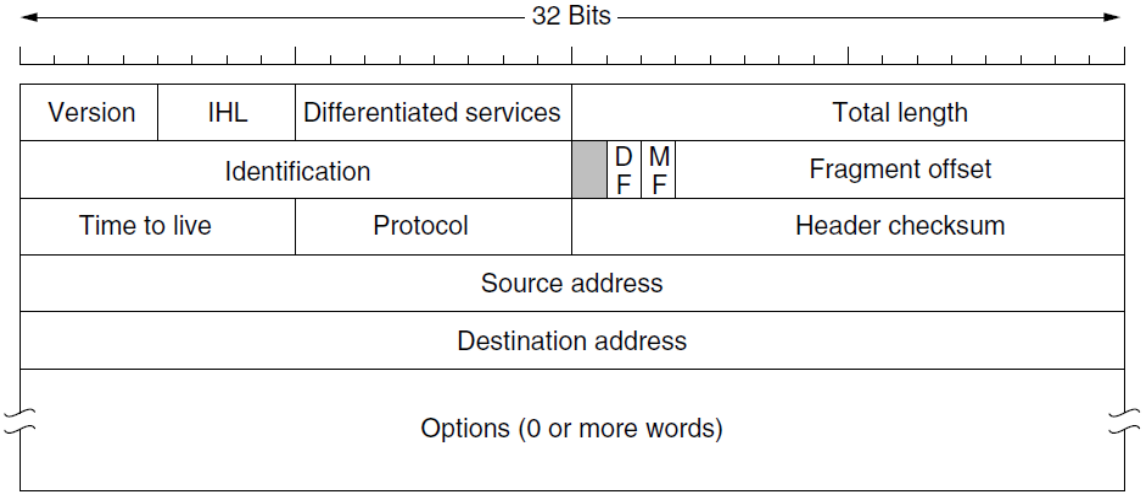
# The Internet Network Layer: IP

- Internet: The Internet is an interconnected collection of many networks
- IP = Internet Protocol (provides datagram service)
- Transport layer breaks messages up in datagrams of about 1500 bytes
- Datagrams are reassembled at destination host (by Transport layer)



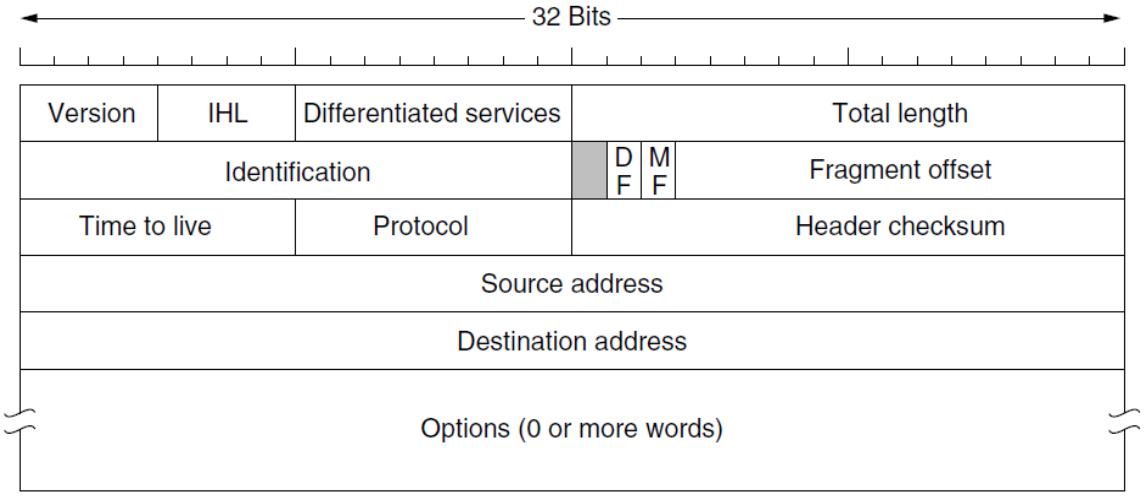
# The Internet Network Layer: IP

- IP datagram contains a header of minimal 20 bytes, containing fields for:
  - Version (= 4 for IPv4)
  - IHL: internet header length
  - Differentiated services
    - Distinguish between different classes of service.
  - Total length
  - Identification
    - Label fragments belonging to same datagram
  - DF (do not fragment) bit
    - Note: datagrams < 576 bytes should be supported



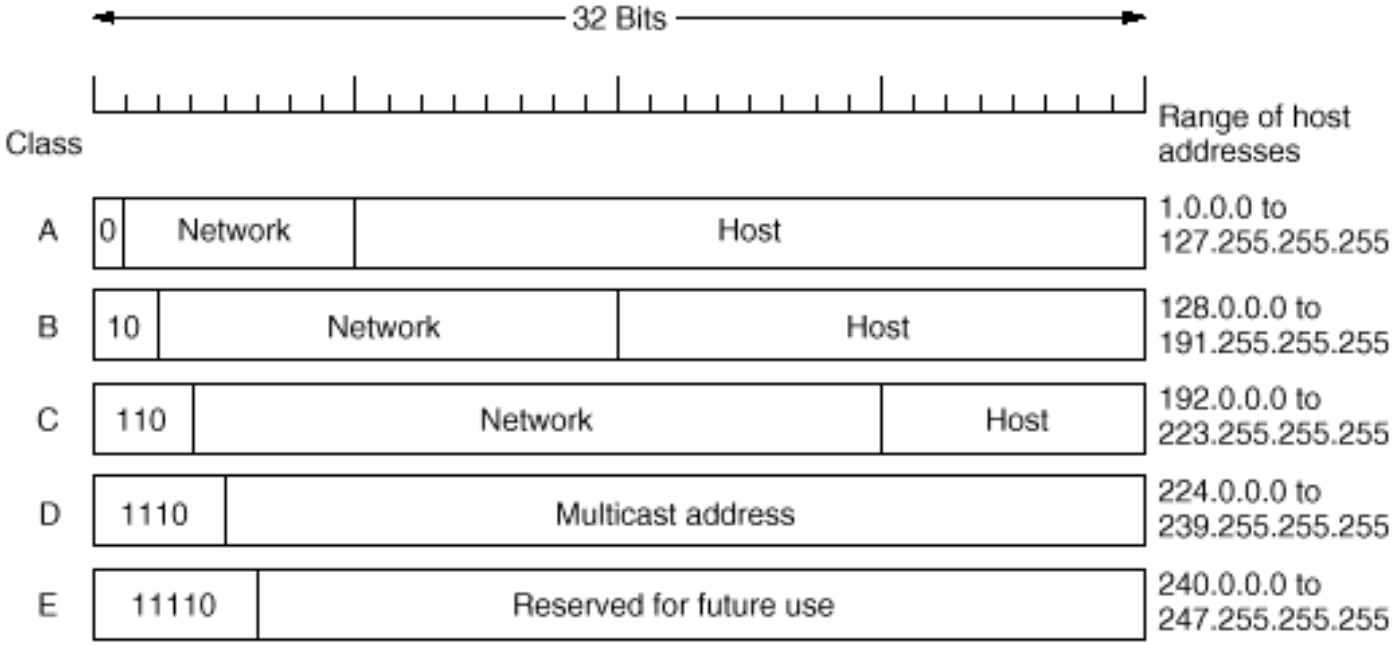
# The Internet Network Layer: IP

- MF (more fragments) bit: is this the final fragment?
- Fragment offset: Location of the fragment within the datagram
- TTL (time to live)
  - Counter decremented at each hop
  - Discard datagram when zero: send control message back to source
- Protocol field:
  - Indicate which transport layer protocol to use: e.g.. UDP or TCP
  - Allows direct delivery to right TL protocol handler
- Header checksum
- Source and Destination IP addresses



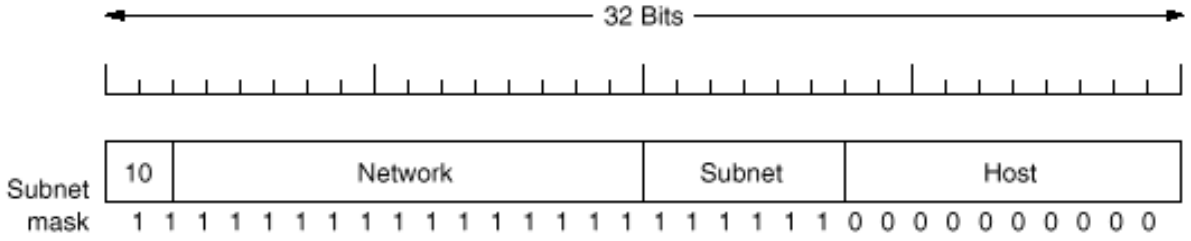
# The Internet Network Layer: IP

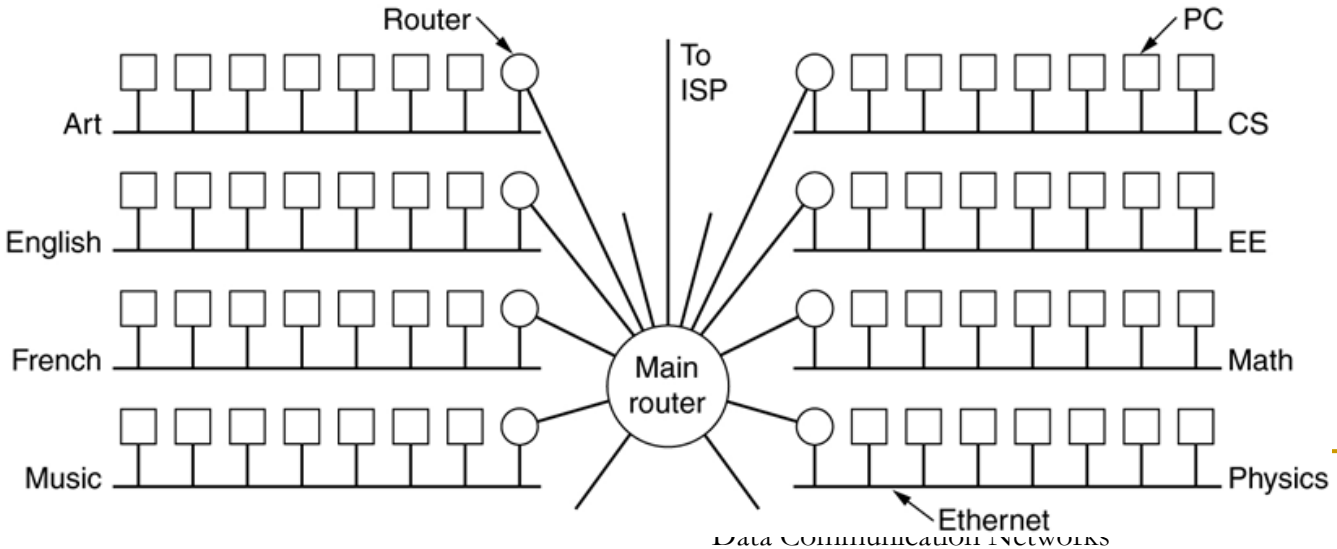
- Every IP-host (on the net) needs unique 32-bit address
- Assignment by ICANN (Internet Corporation for Assigned Names and Numbers)
- Five classes: A - E
- Dotted decimal notation: Example: 129.245.96.122



# Subnets

- A single class A, B or C address refers to one network.
- The network can be designed to contain subnets that can be used for internal routing.



# Subnet routing: How is routing performed:

- Normally router contains table with the following type of addresses
  - (network#, 0)
  - (this network#, host#)
  - This is a 2-level hierarchy
  - If incoming network address not listed send packet to default router (with more extensive table)
- With subnetting (for internal use only):
  - Router table contains additional entries:
    - (this network#, subnet#, 0)
    - (this network#, this subnet#, host#)
  - 3-level hierarchy
  - Use host masking to quickly determine subnet
- Decision on how many bits to use for subnet and how many bits for Host addresses is an internal design parameter for each company or university

# Classless Inter Domain Routing

- Idea: assign the addresses on a variable-sized blocks, without regard to classes
- Use Masking to quickly determine outgoing link
- Masking example:
  - Assume company needs 2k addresses, and gets: 194.24.0.0 - 194.24.7.255 (8 C blocks)
  - Mask = 255.255.248.0 (248 = 1111.1000b)
  - CIDR Notation: 194.24.0.0/21
- Routing tables contain (base address, mask) triples
- Incoming address is masked with each entry
- If: (address AND mask) = base address, send packet to appropriate output port

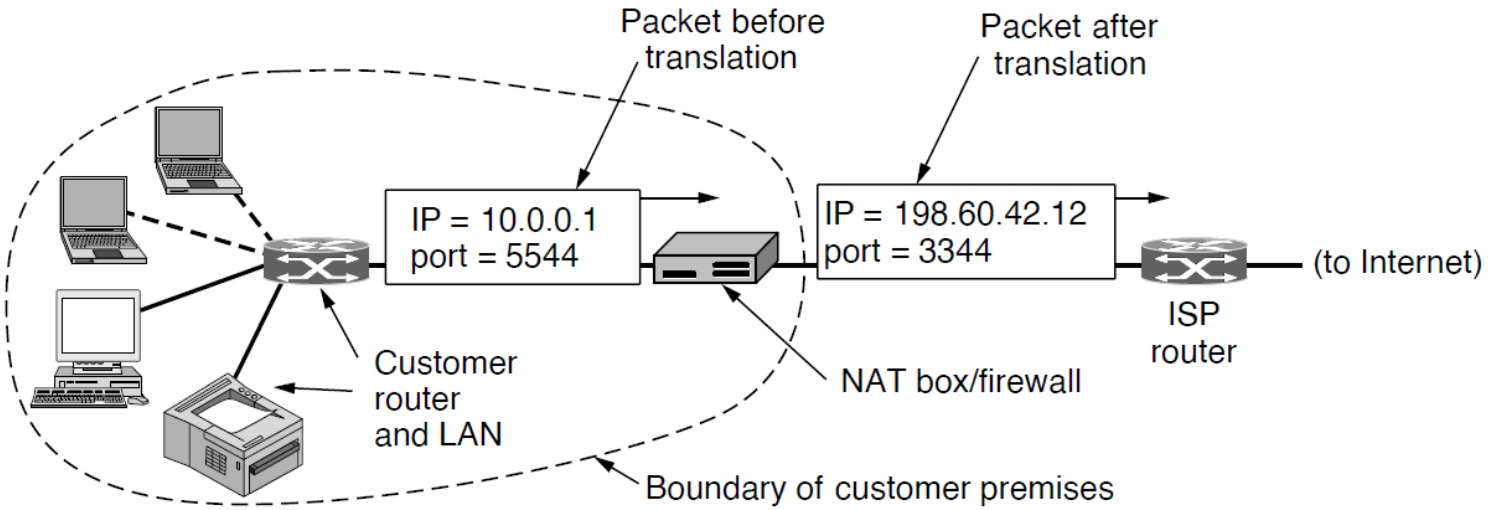
# Longest Prefix Matching

- The router compares the destination IP address, bit-by-bit, with the entries in the routing table.
- The entry that has the longest number of network bits that match the IP destination address is always the best match (or best path)
  - Example: Packet has a destination address 192.168.1.33
  - Which path to choose?

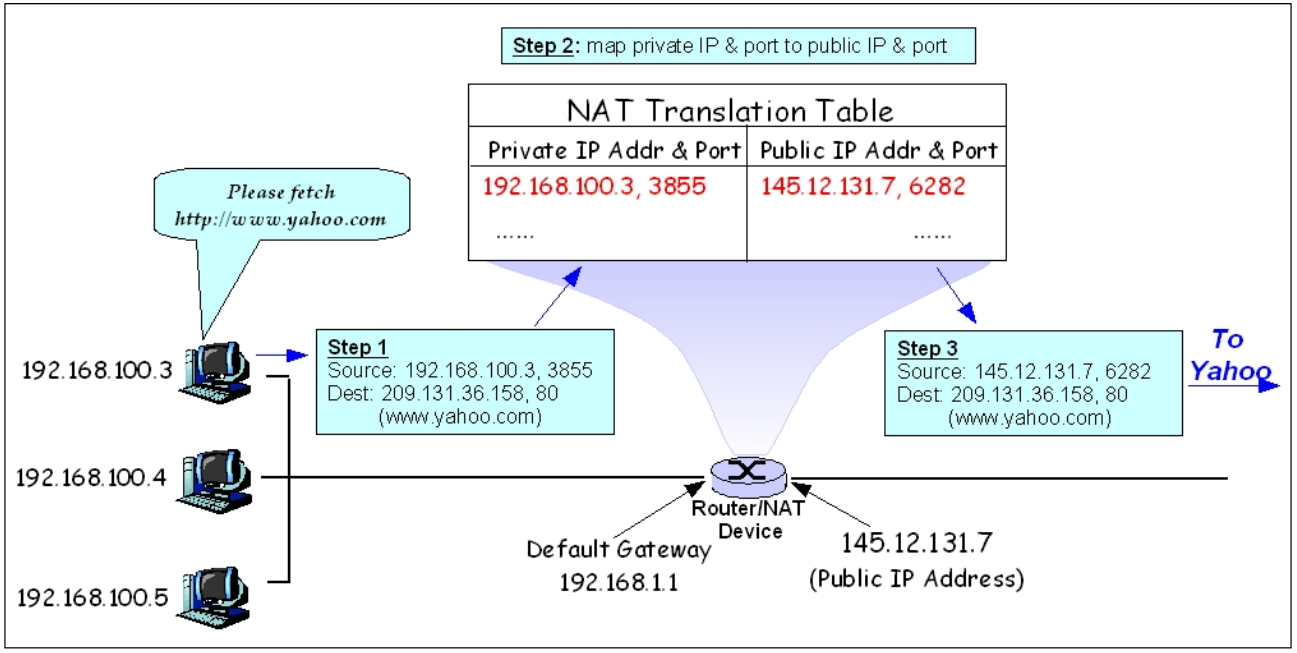
| Address | Converted Binary Address |
|---|---|
| 192.168.1.33 (destination IP address) | **11000000.10101000.00000001.001**00001 |
| 192.168.1.32/28 | **11000000.10101000.00000001.001**00000 (<–Best match) |
| 192.168.1.0/24 | **11000000.10101000.00000001.**00000000 |
| 192.168.0.0/16 | **11000000.10101000.**00000000.00000000 |

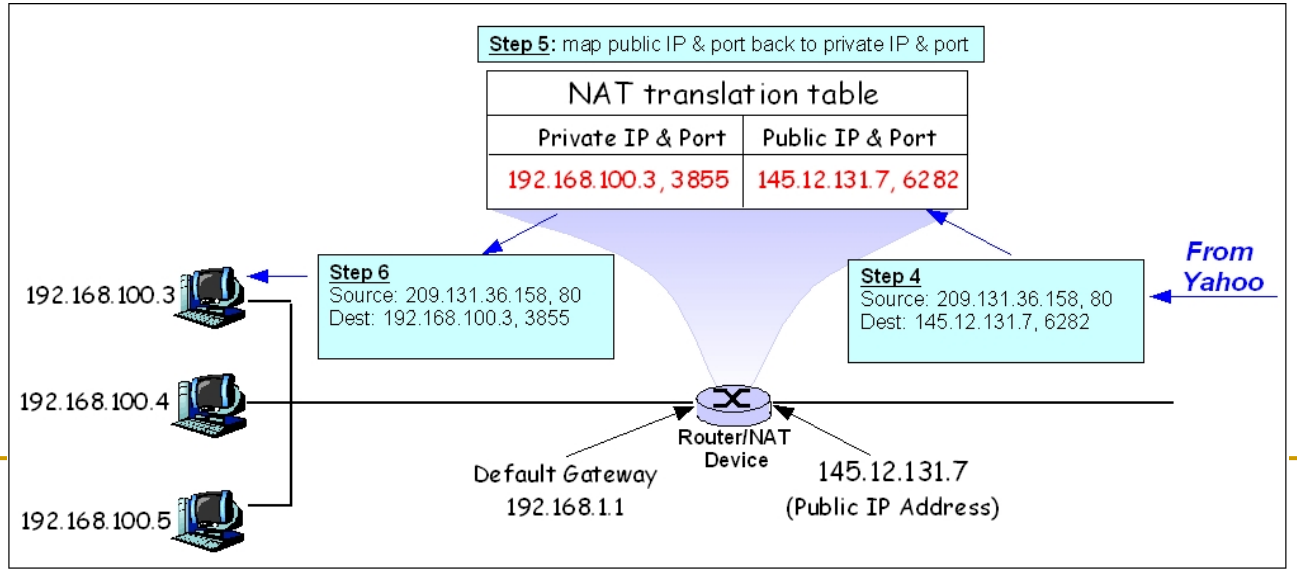
# Network Address Translation

- Allow each host in the internal network to maintain a permanent private IP address.
- Replace the {TCP Source, IP source} addresses with {virtual TCP source, Virtual IP source} of each outgoing IP packet
- On the reply, use the combination of TCP and IP addresses to lookup the local host
- Reserved NAT addresses
  - IP packets with these addresses should not appear on internet
  - 10.0.0.0 – 10.255.255.255/8 (16,772,216 hosts)
  - 172.16.0.0 – 172.31.255.255/12 (1,0478,576 hosts)
  - 192.168.0.0 – 192.168.255.255/16 (65,636 hosts)



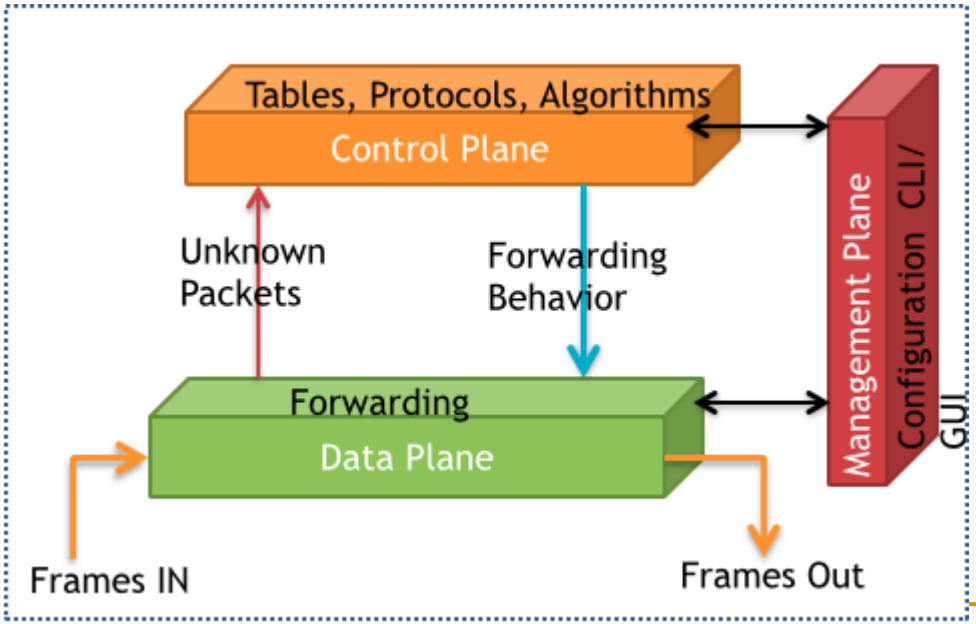
# Network Address Translation



# Network Planes

- Network systems contain:
  - Data Plane
  - Control Plane
  - Management Plane



# Internet Control Protocols

- Example of Internet Control protocols:
  - Internet Control Message Protocol (ICMP)
  - Address Resolution Protocol (ARP)
  - Dynamic Host Configuration Protocol (DHCP)

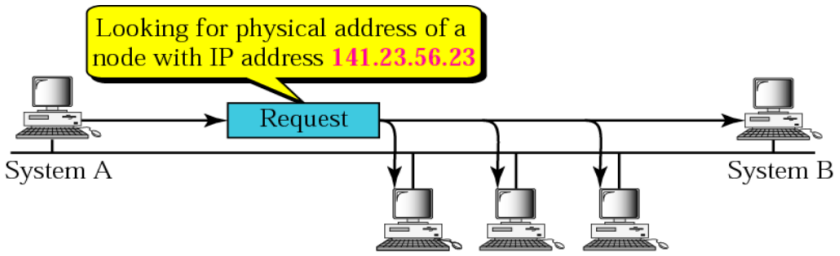
# ICMP

- A protocol to exchange messages between routers to allow proper operation of the network
- Messages are embedded in IP packet

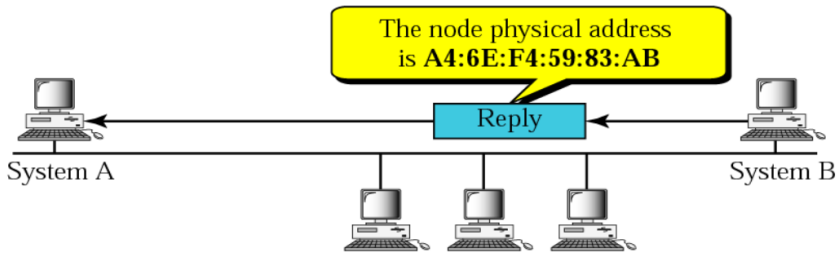
| Message type | Description |
|---|---|
| Destination unreachable | Packet could not be delivered |
| Time exceeded | Time to live field hit 0 |
| Parameter problem | Invalid header field |
| Source quench | Choke packet |
| Redirect | Teach a router about geography |
| Echo and echo reply | Check if a machine is alive |
| Timestamp request/reply | Same as Echo, but with timestamp |
| Router advertisement/solicitation | Find a nearby router |

# Address Resolution Protocol (ARP)

- The Address Resolution Protocol (ARP) is a communication protocol used for discovering the link layer address, such as a MAC address, associated with a given internet layer address, typically an IPv4 address
- Example:
  - System A broadcasts ARP Request message on LAN : give me the Ethernet address of 141.23.56.23
  - All LAN nodes receive the message, only the target unicasts the reply.
- Address mappings are cached temporarily



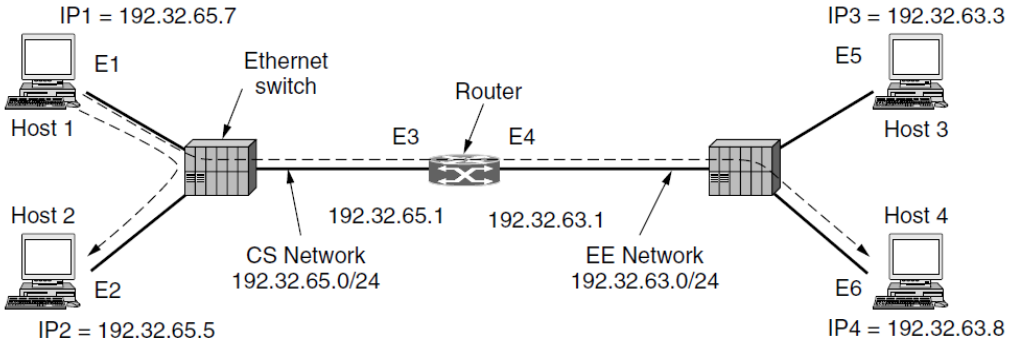

a. ARP request is broadcast




b. ARP reply is unicast

# Address Resolution Protocol (ARP)

- Remote Address Example: Host 1 wants to send information to Host 4
  - Host 1 uses ARP message; gateway responds
  - Host 1 Sends packet directly to gateway. (Host 1 must know the default (gateway) Ethernet address)
  - Routers forward the packet to the destination LAN
  - On destination LAN ARP is used by the router again to get the destination Ethernet address

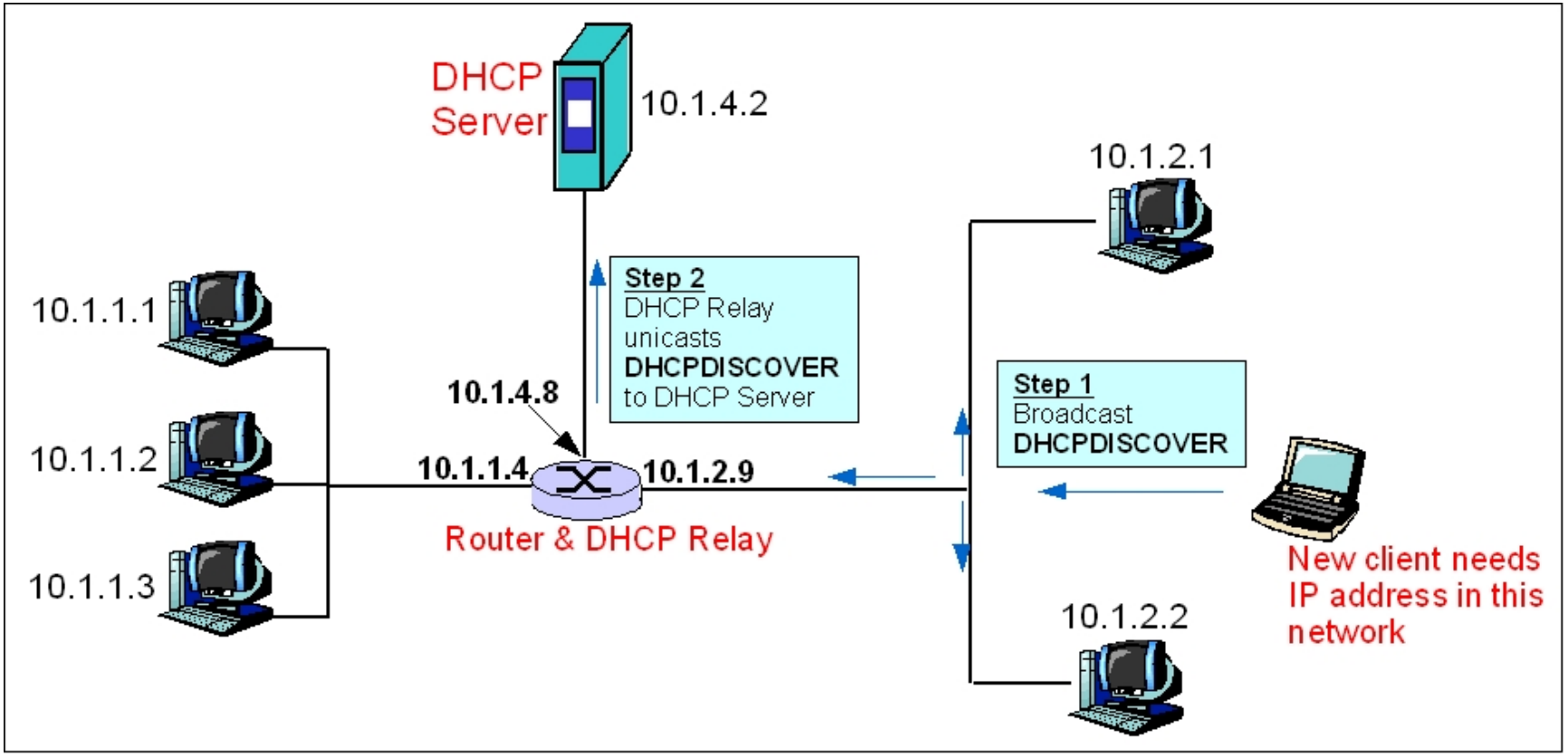



| Frame | Source IP | Source Eth. | Destination IP | Destination Eth. |
|---|---|---|---|---|
| Host 1 to 2, on CS net | IP1 | E1 | IP2 | E2 |
| Host 1 to 4, on CS net | IP1 | E1 | IP4 | E3 |
| Host 1 to 4, on EE net | IP1 | E4 | IP4 | E6 |

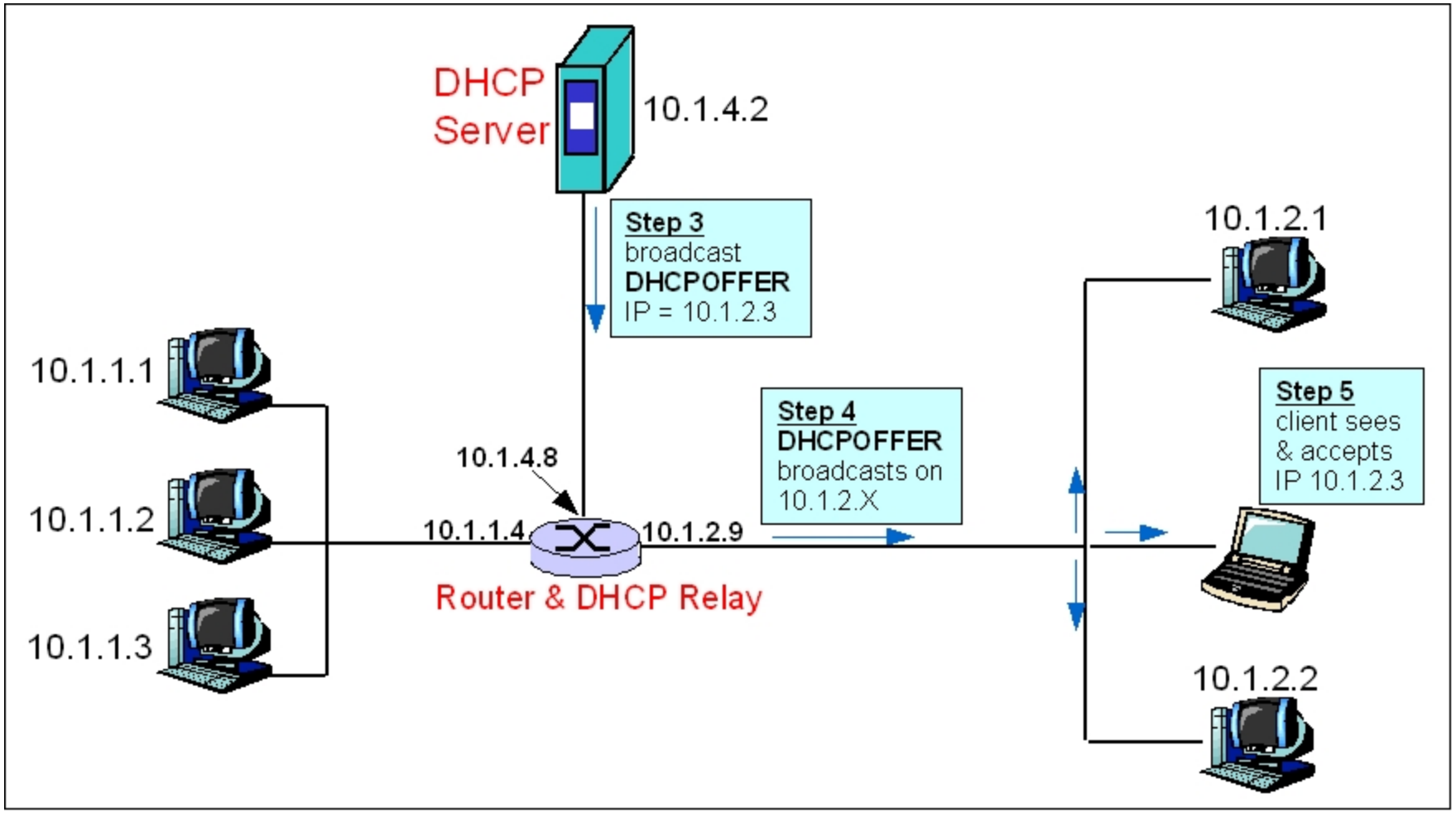
# Dynamic Host Configuration Protocol (DHCP)

- DHCP: Dynamic Host Configuration Protocol (RFC 2131, 2132)
  - A client-server architecture for configuration of host IP addresses
  - A newly booted workstation broadcasts a DHCP DISCOVER packet
  - The DHCP server replies by giving it an IP address
  - The IP address is leased for a period of time.
  - Host must ask for renewal of the lease before it expires.
  - If the lease expires, the IP is returned to the pool of available IP addresses.
  - If DHCP server is not on the same LAN, a DHCP proxy can relay the DHCP messages over network to the DHCP server.
  - DHCP proxy should know the IP address of the DHCP server.

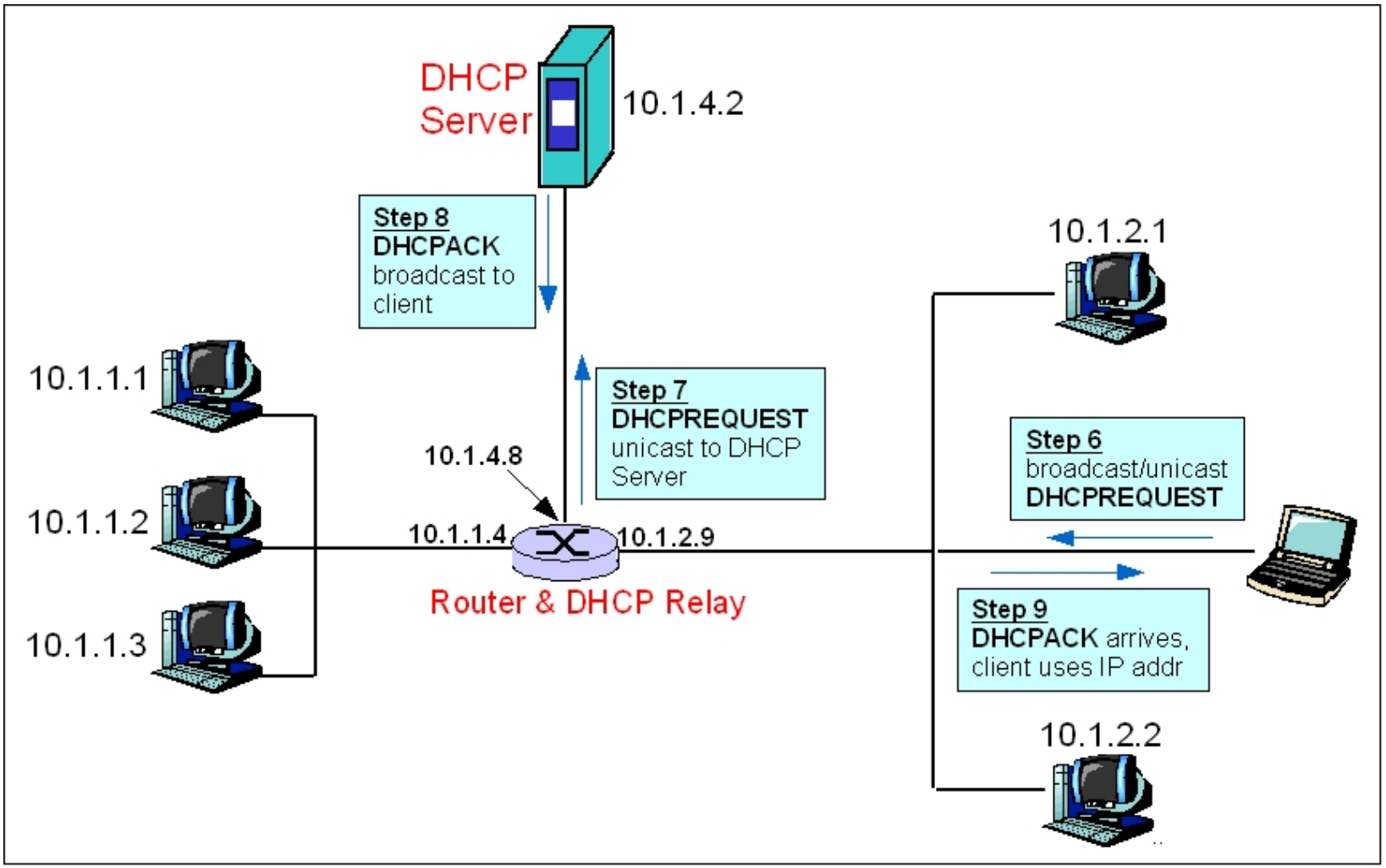
# DHCP Example



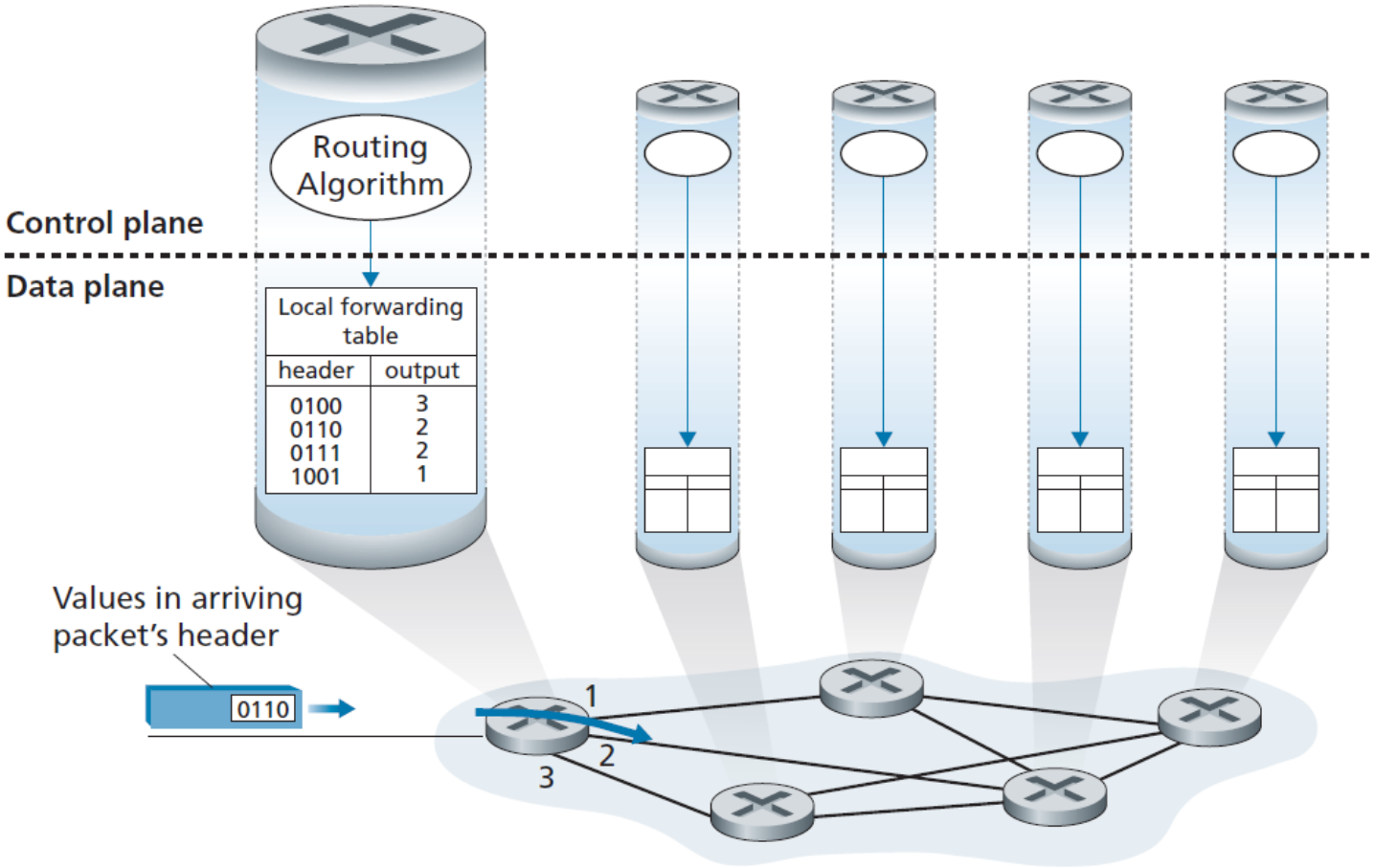
# DHCP Example



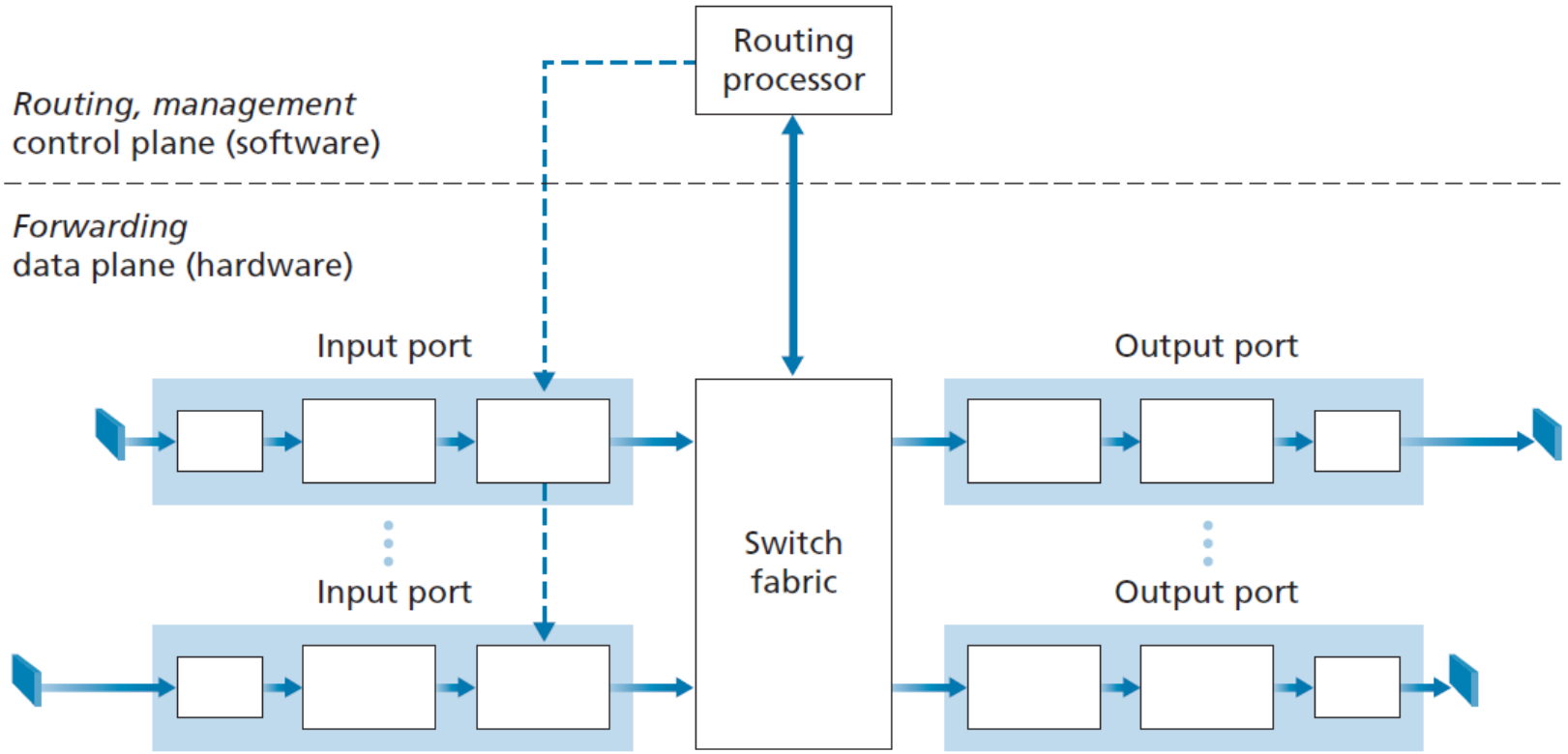
# DHCP Example



# Routing: Control Plane and Data Plane

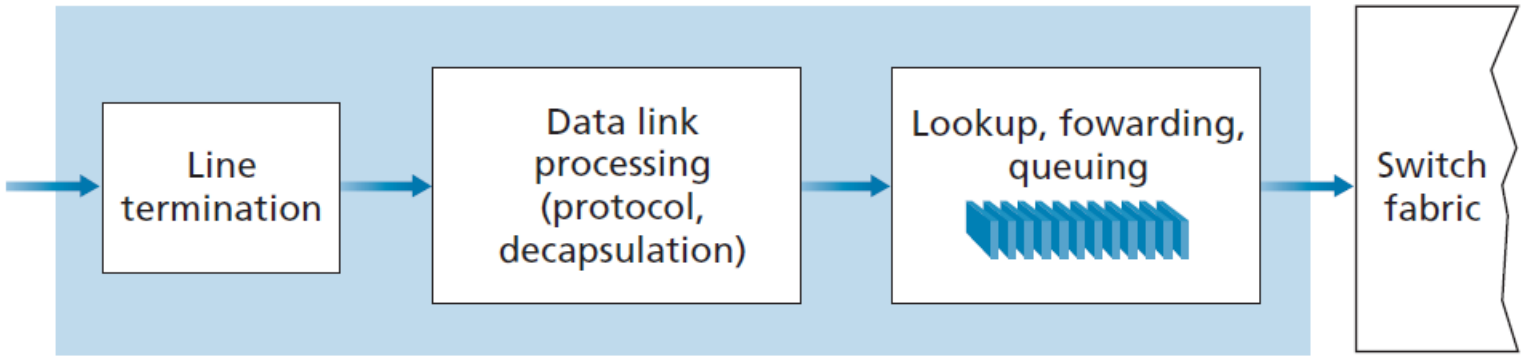


# Router Architecture

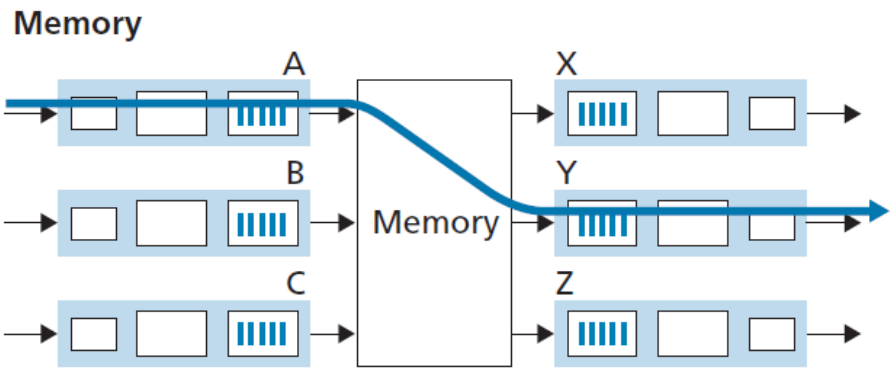


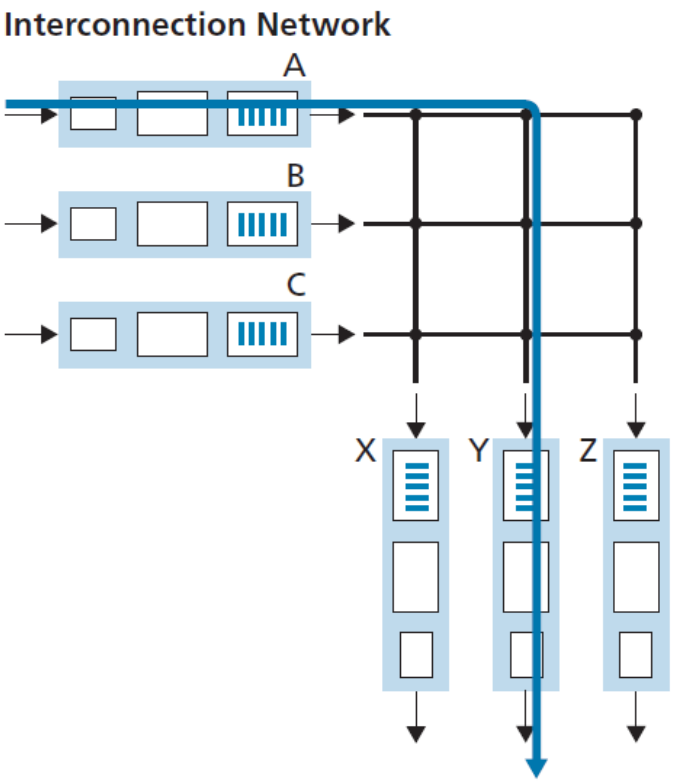
# Input Port Processing

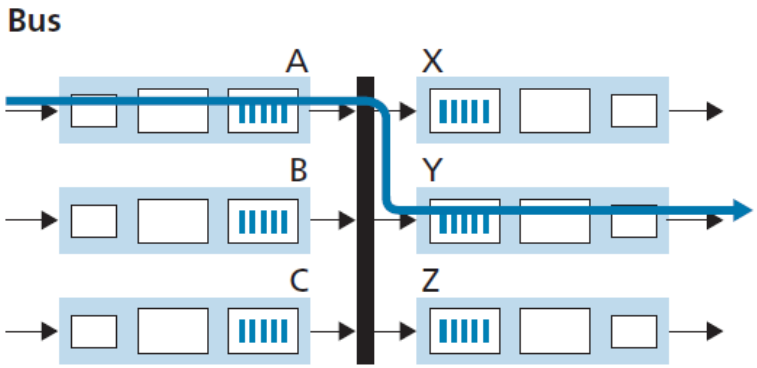
- Line Termination
- Data Link Processing
- Address Lookup
- Queuing



# Switching

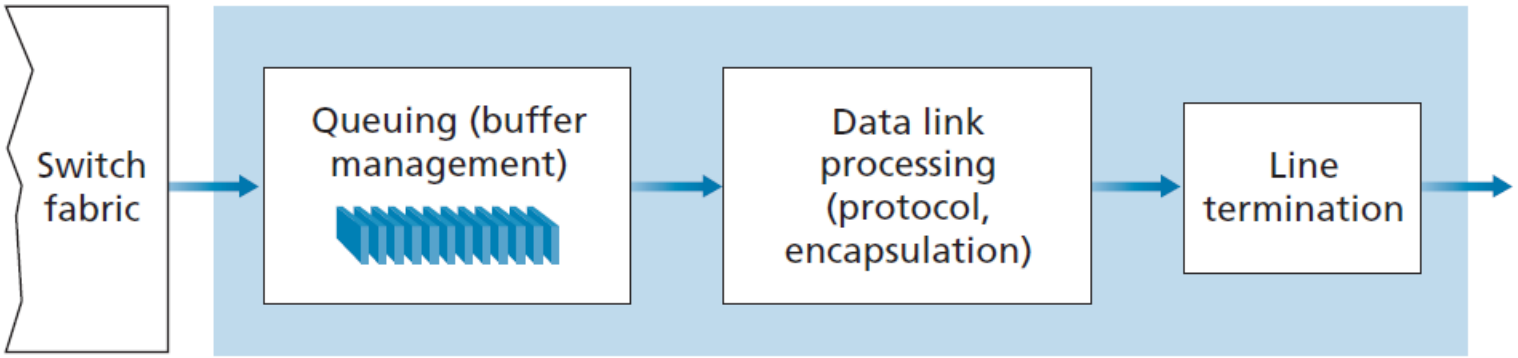

Memory
A
B
C
Memory
X
Y
Z


Interconnection Network
A
B
C
X
Y
Z


Bus
A
B
C
X
Y
Z

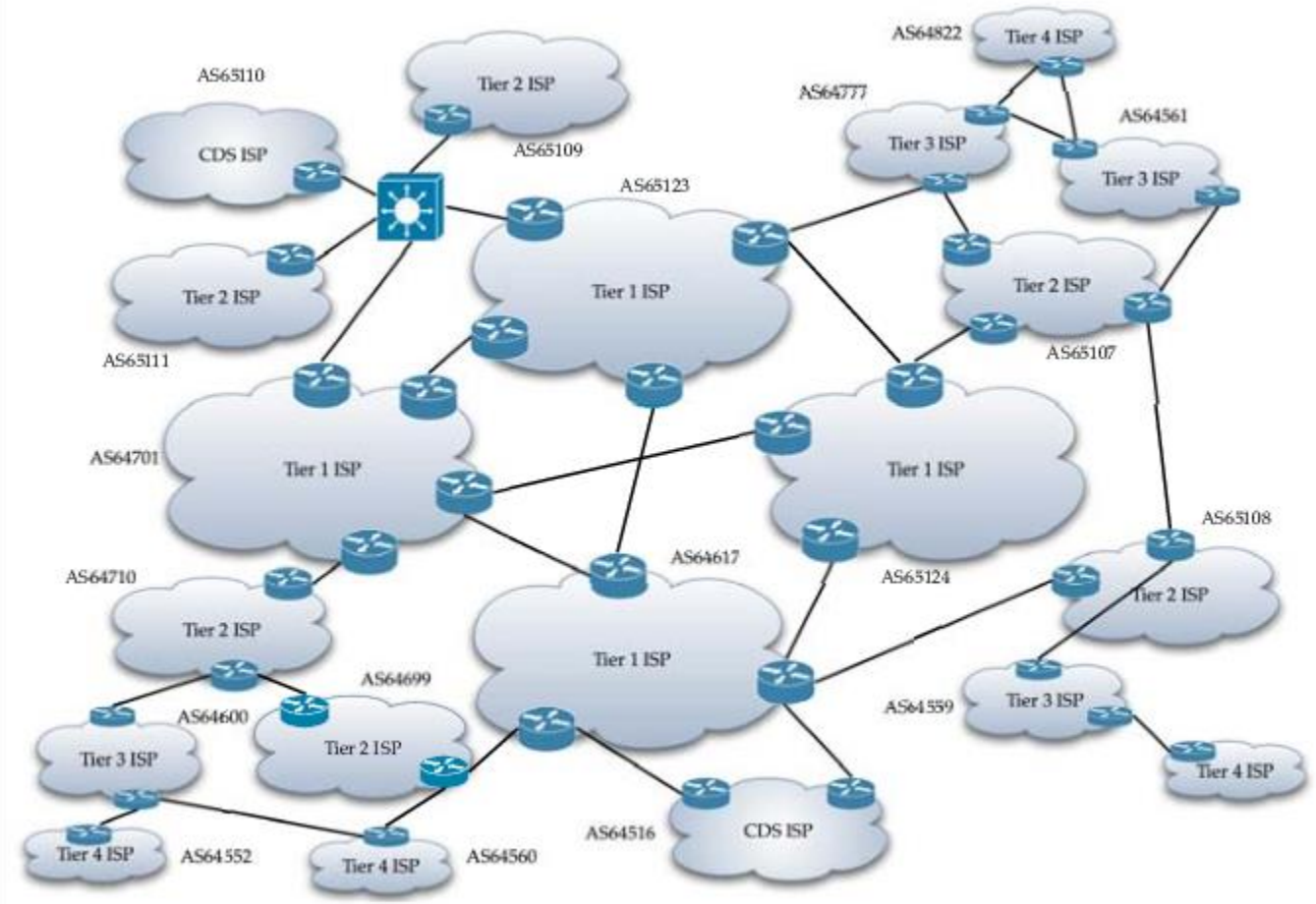
# Output Port Processing

- Queuing (Buffer management)
- Data Link Processing
- Line Termination



# Routing on the Internet

- Internet is made up of many independent networks or ASs (Autonomous Systems) that are connected.



# Internet Routing Protocols

## Intradomain Routing (Interior Gateway Protocol)

- Routing protocols used inside an AS.
- OSPF and IS-IS are popular protocols.

## Interdomain Routing (Exterior Gateway Protocol)

- Routing protocols between ASs.
- BGP protocol is widely used.

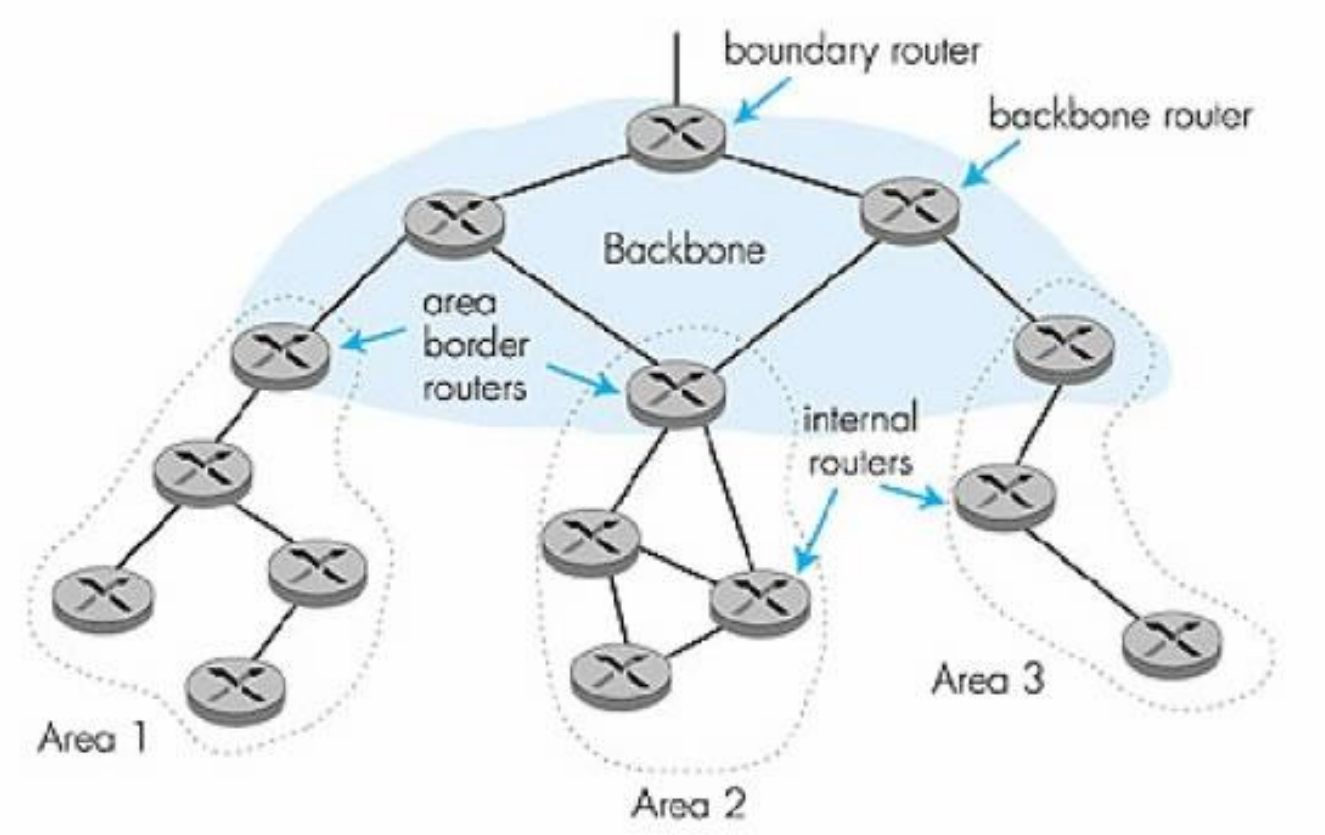
# Open Shortest Path First (OSPF)

- OSPF is a link-state protocol that uses flooding of link-state information and a Dijkstra's least-cost path algorithm.
- With OSPF, each router constructs a complete topological map (that is, a graph) of the entire autonomous system.
- Each router then locally runs Dijkstra's shortest-path algorithm to determine a shortest-path tree to all subnets, with itself as the root node.

# OSPF Features

- Security.
  - Exchanges between OSPF routers (for example, link-state updates) can be authenticated.
  - With authentication, only trusted routers can participate in the OSPF protocol within an AS, thus preventing malicious intruders from injecting incorrect information into router tables.
- Multiple same-cost paths.
  - When multiple paths to a destination have the same cost, OSPF allows multiple paths to be used

# Open Shortest Path First (OSPF)



# Hierarchy in OSPF

- An OSPF autonomous system can be configured hierarchically into areas.
- Each area runs its own OSPF link-state routing algorithm, with each router in an area broadcasting its link state to all other routers in that area.
- Within each area, one or more area border routers are responsible for routing packets outside the area. Lastly, exactly one OSPF area in the AS is configured to be the backbone area.
- The primary role of the backbone area is to route traffic between the other areas in the AS.
- The backbone always contains all area border routers in the AS and may contain non-border routers as well.
- Inter-area routing within the AS requires that the packet be first routed to an area border router (intra-area routing), then routed through the backbone to the area border router that is in the destination area, and then routed to the final destination.

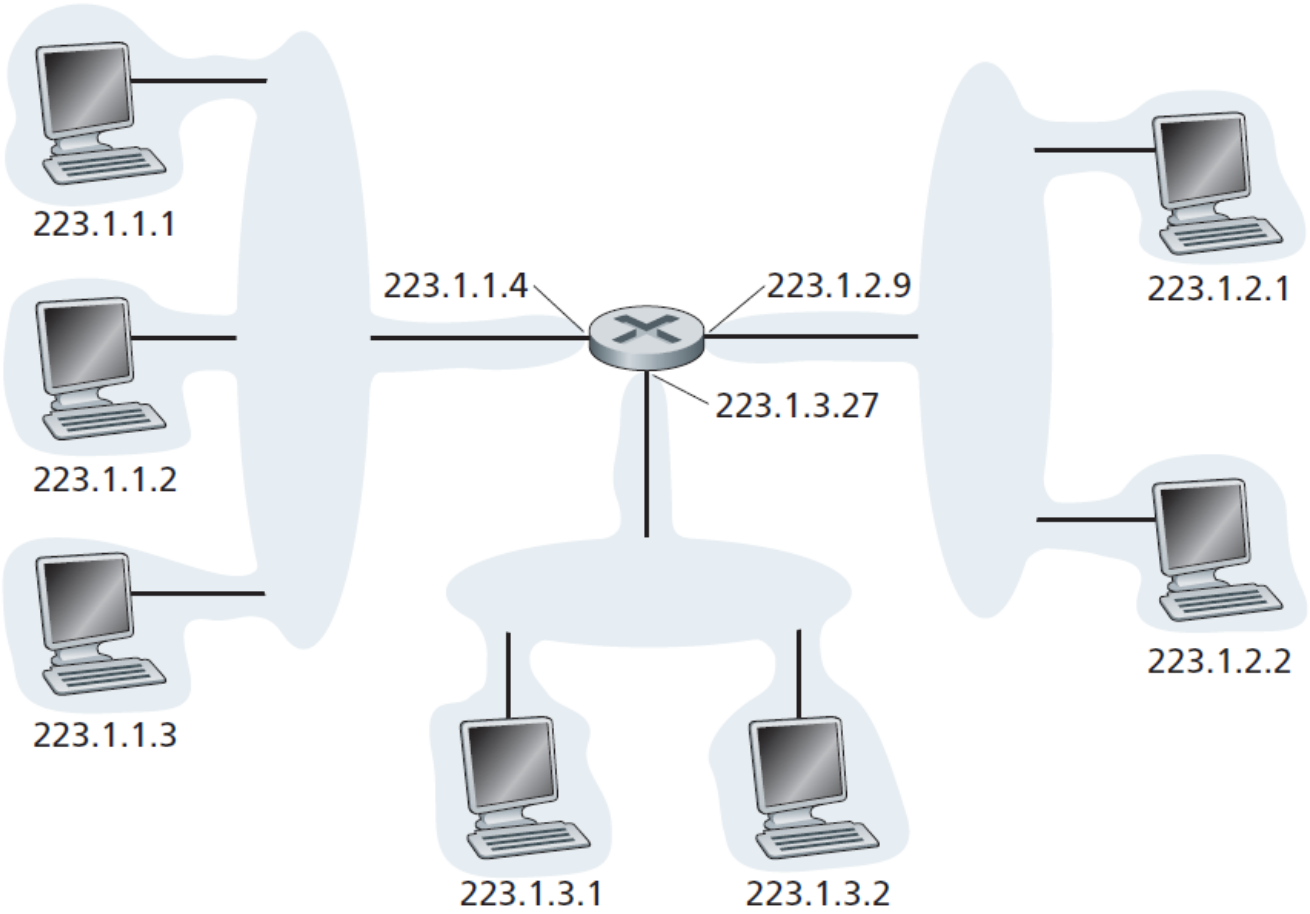
# Border Gateway Protocol (BGP)

- A decentralized protocol that is used to connect all ASs that are connected to form the internet.
- In BGP, packets are not routed to a specific destination address, but instead to CIDRized prefixes, with each prefix representing a subnet or a collection of subnets.
  - Router forwarding table entry example: (138.16.68/22,K) where K is the interface number of the router.

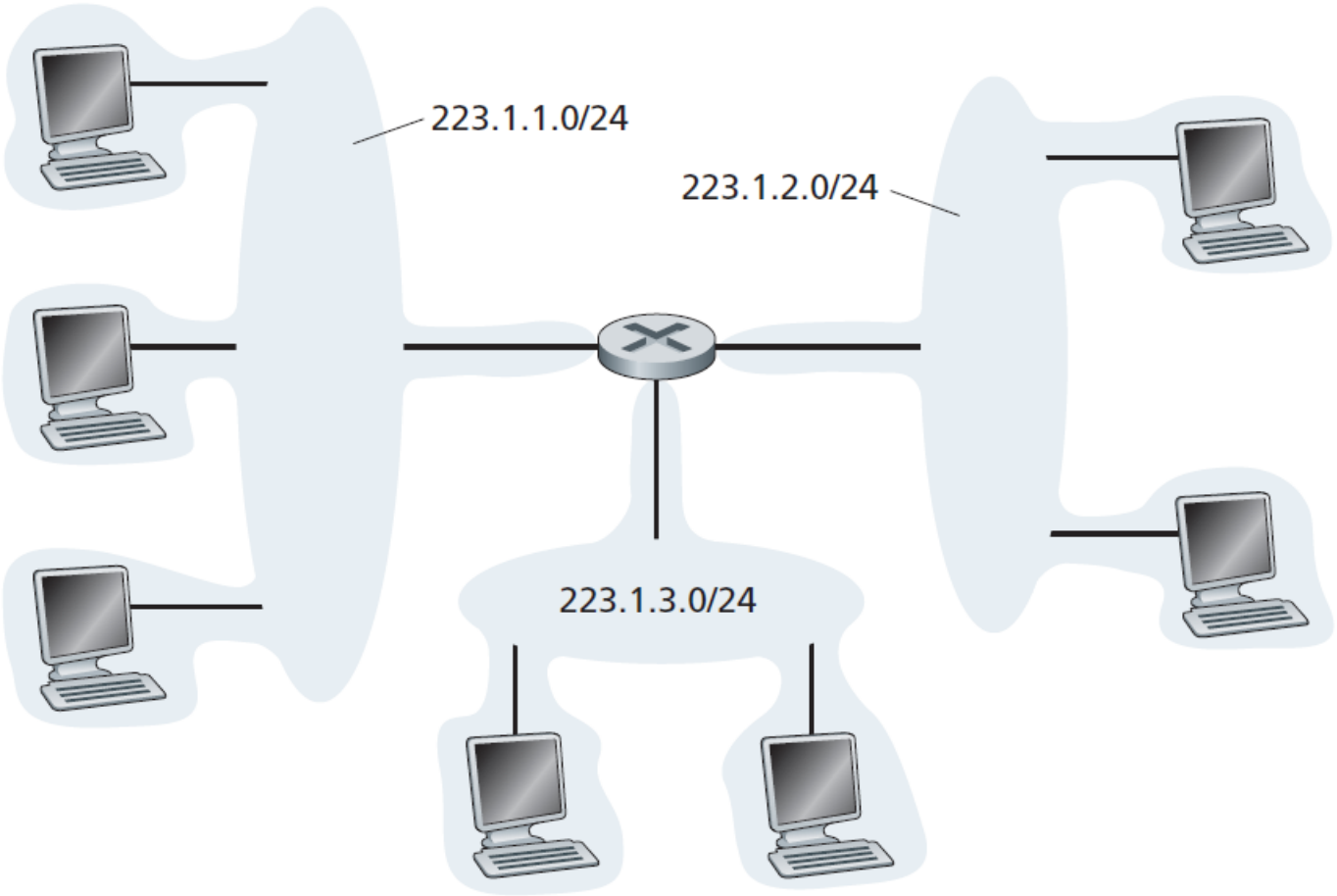
# Border Gateway Protocol (BGP)

- BGP main objectives:
  - Obtain prefix reachability information from neighboring ASs.
    - BGP allows each subnet to advertise its existence to the rest of the Internet.
  - Determine the “best” routes to the prefixes.
    - The best route will be determined based on policy as well as the reachability information.

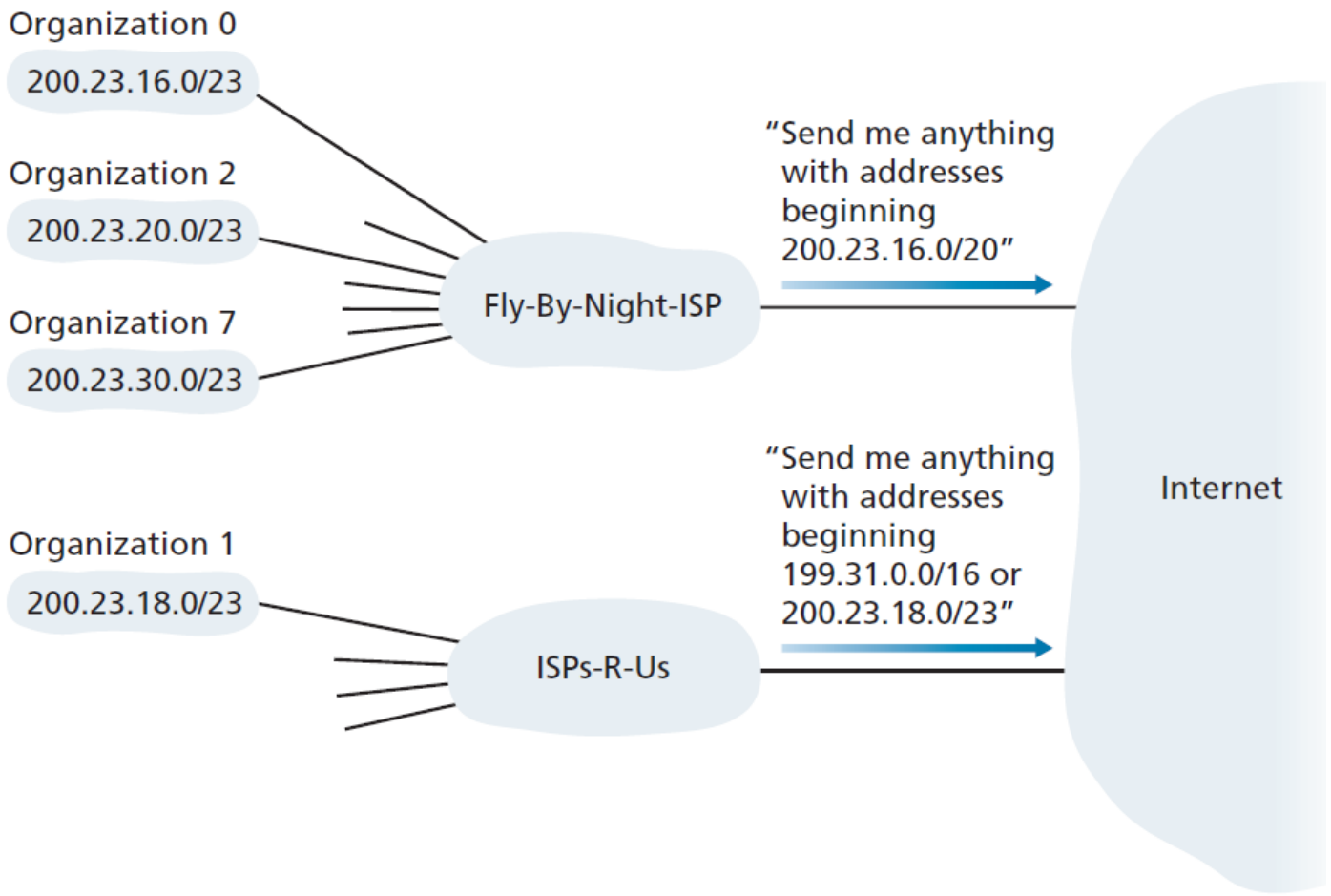
# Example: IP Routing in Subnet



# Example: IP Routing in Subnet



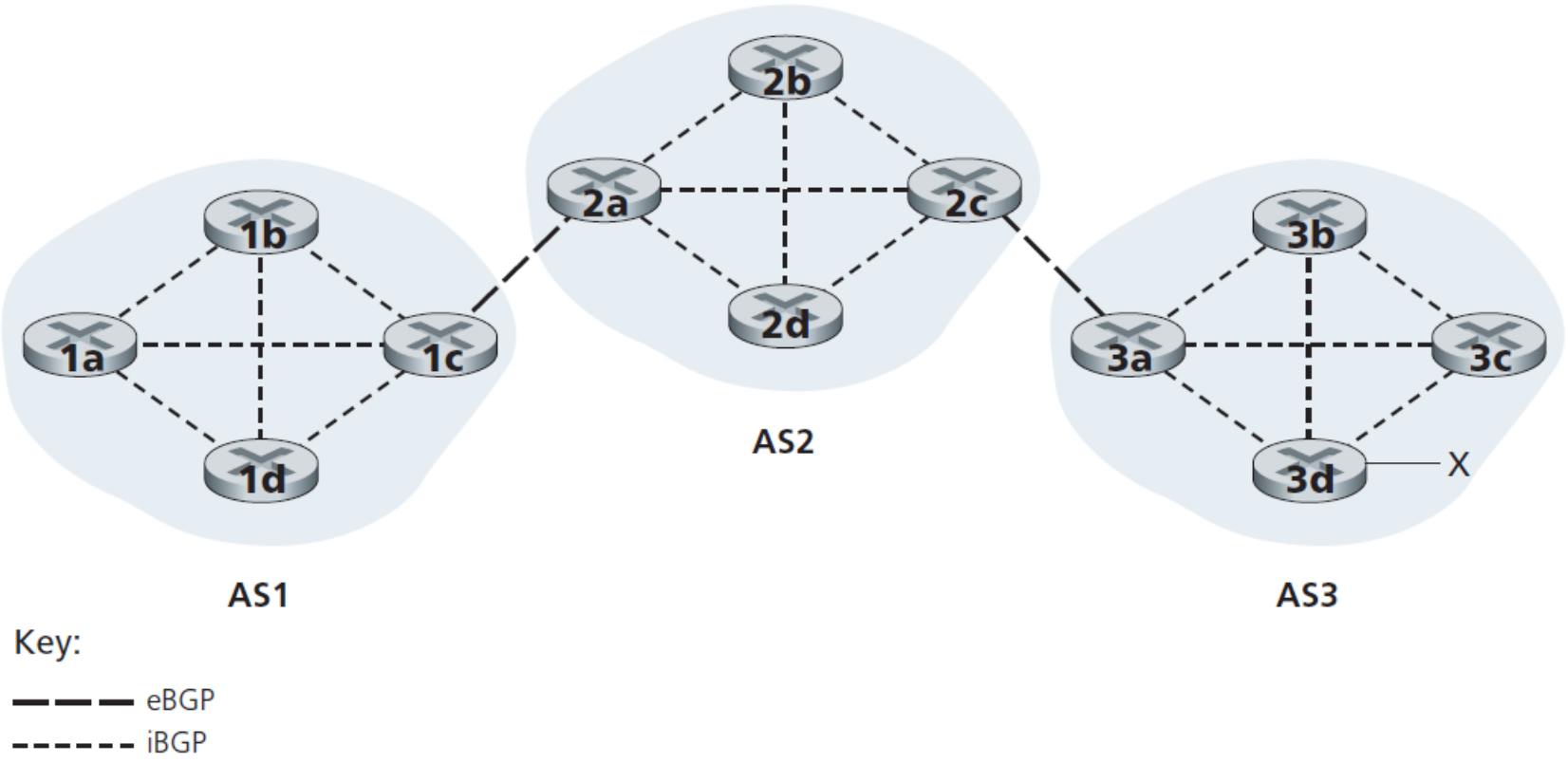
# Hierarchical Addressing and Route Aggregation



# Border Gateway Protocol (BGP)

- In BGP pairs of routers exchange routing information over semi-permanent TCP connections
- Each such TCP connection, along with all the BGP messages sent over the connection, is called a BGP connection.
- Furthermore, a BGP connection that spans two ASs is called an external BGP (eBGP) connection
- A BGP session between routers in the same AS is called an internal BGP (iBGP) connection.

# BGP Example 1

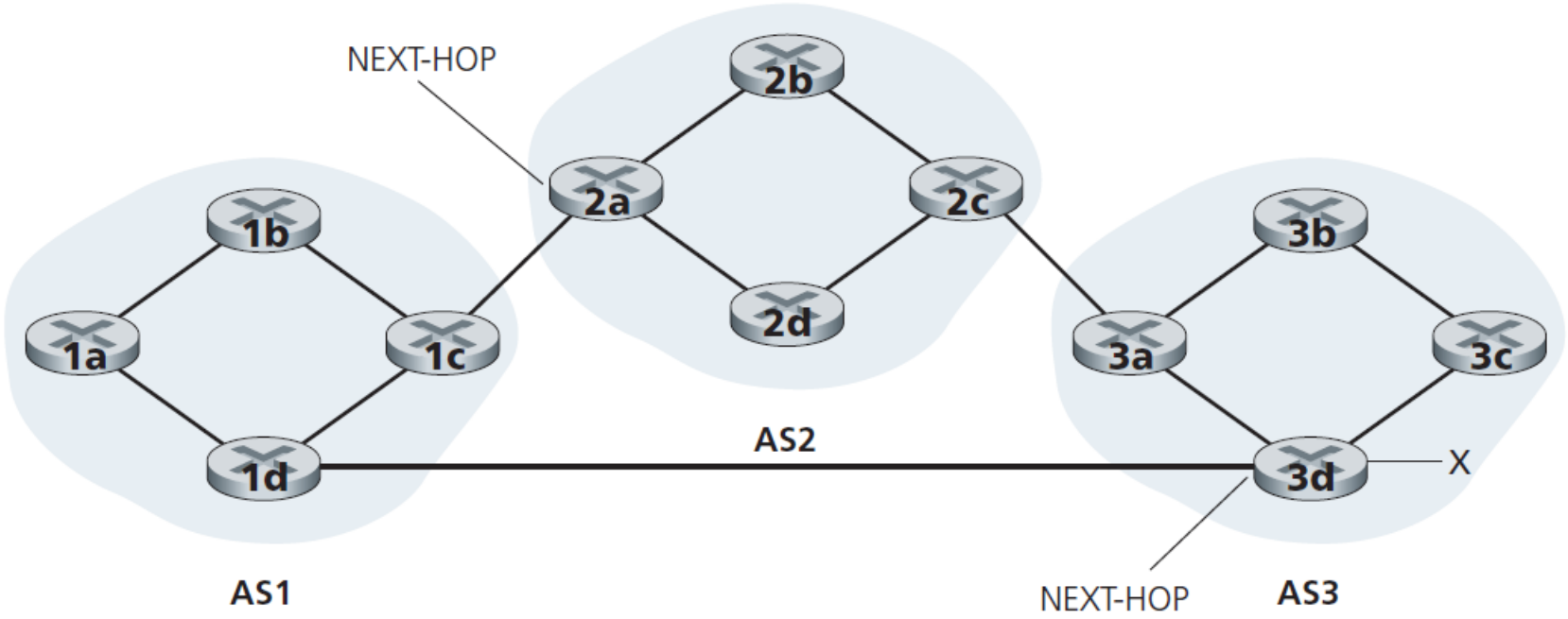


# BGP Example 1

1) Gateway router 3a sends an eBGP message “AS3 x” to gateway router 2c.
2) Gateway router 2c then sends the iBGP message “AS3 x” to all the other routers in AS2, including to gateway router 2a.
3) Gateway router 2a then sends the eBGP message “AS2 AS3 x” to gateway router 1c.
4) Gateway router 1c uses iBGP to send the message “AS2 AS3 x” to all the routers in AS1.

- Now, all routers in AS1 know that the subnet x is reachable via the path AS2-AS3-x

# BGP Example 2



# BGP Example 2

- When a router advertises a prefix across a BGP connection, it includes with the prefix several BGP attributes.
- A prefix along with its attributes is called a route.
- Two of the more important attributes are AS-PATH and NEXT-HOP.
- The AS-PATH attribute contains the list of ASs through which the eBGP advertisement has passed.
- The NEXT-HOP is the IP address of the router interface that begins the AS-PATH.
- In this example, each router in AS1 becomes aware of two BGP routes to prefix x:
  - IP address of leftmost interface for router 2a; AS2 AS3; x
  - IP address of leftmost interface of router 3d; AS3; x

# BGP Route Selection

- A router learns all routes to a given prefix and performs route selection based on:
  - Local preference of the route which can be assigned based on the policy of the network operator
  - Shortest AS path which is the lowest number of ASs in the route
  - Shortest path from each router to the next-hop

# BGP Policy

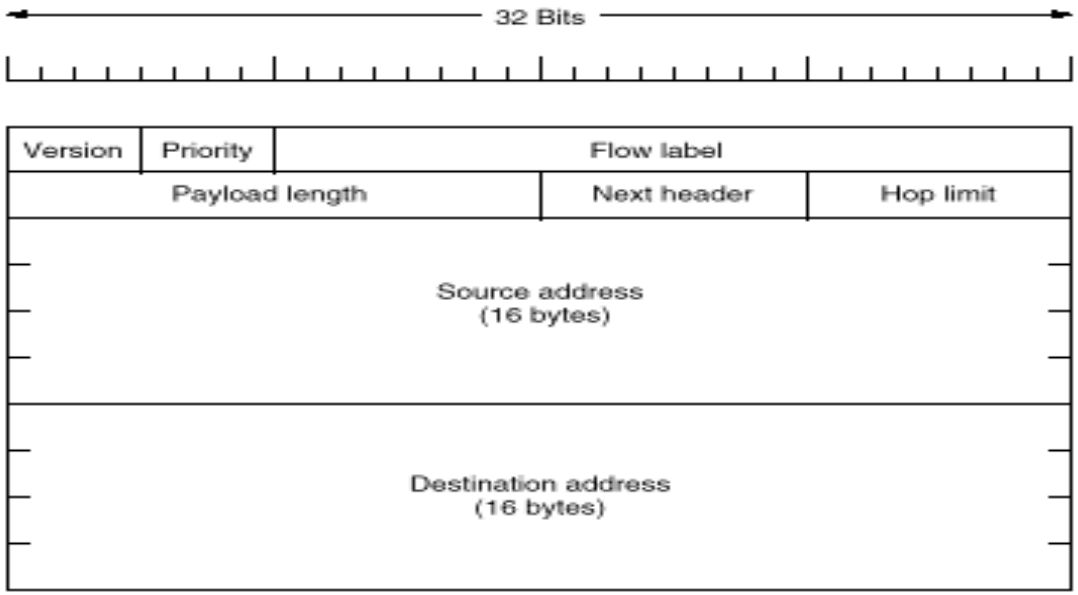
- One of the most important aspects of BGP is that it is designed so that network owners can apply their policy in routing.
- Examples of policy:
  - Any traffic flowing across an ISP's backbone network must have either a source or a destination (or both) in a network that is a customer of that ISP.
  - Use AS1 instead of AS2 because it is cheaper.
  - Don't use AT&T in Australia because performance is poor.
  - Traffic starting or ending at Apple should not transit Google

# IPv6 Protocol

- To expand the feature set and capabilities of IP, in 1990 work on new protocol started with these requirements:
  - Support billions of hosts
  - Reduce routing table size
  - Simplify protocol (reduces processing power requirement in routers)
  - Better security (authentication and privacy)
  - Multicasting
  - Mobile hosts
  - Allow protocol to evolve (room for expansion)
  - Coexist with IPv4
- Important Features:
  - Longer addresses than IPv4: 16 byte addresses
  - Header simplification (fixed size of 40 bytes)
  - Better support for options
  - Improved Security capabilities
  - Support for packet priority
  - Support for pseudo connections (using the flow label)
  - Support very long datagrams

# IPv6 Protocol Features

- Version: Always set to 6
- Priority: Handle packets with different classes of service
  - 0-7 non-real-time (flow controlled)
  - 8-15 real-time (no flow control)
- Flow: flow label which is used for flow-driven routing
- Payload Length: packets with size >64k are supported
- Next Header: Headers can optionally be extended over multiple packets.



# Multi Protocol Label Switching

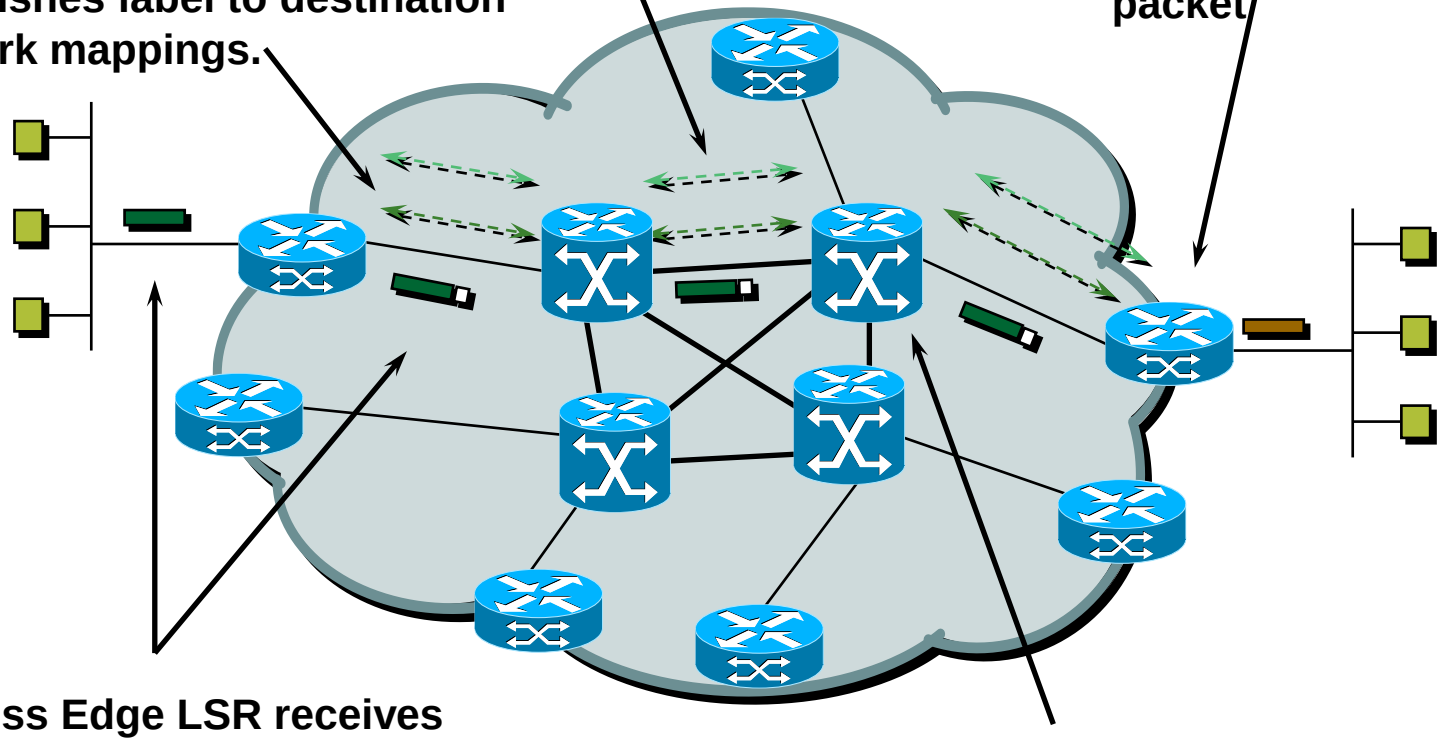
- Multiprotocol Label Switching (MPLS) is a mechanism in high-performance telecommunications networks which directs and carries data from one network node to the next.
- MPLS makes it easy to create "virtual links" between distant nodes. It can encapsulate packets of various network protocols.
- In an MPLS network, data packets are assigned labels. Packet-forwarding decisions are made solely on the contents of this label, without the need to examine the packet itself. This allows one to create end-to-end circuits across any type of transport medium, using any protocol
- MPLS requires a small overhead, while providing connection-oriented services for variable-length frames.

# MPLS Operation

**1a. Existing routing protocols (e.g. OSPF, IS-IS) establish reachability to destination networks**

**1b. Label Distribution Protocol (LDP) establishes label to destination network mappings.**

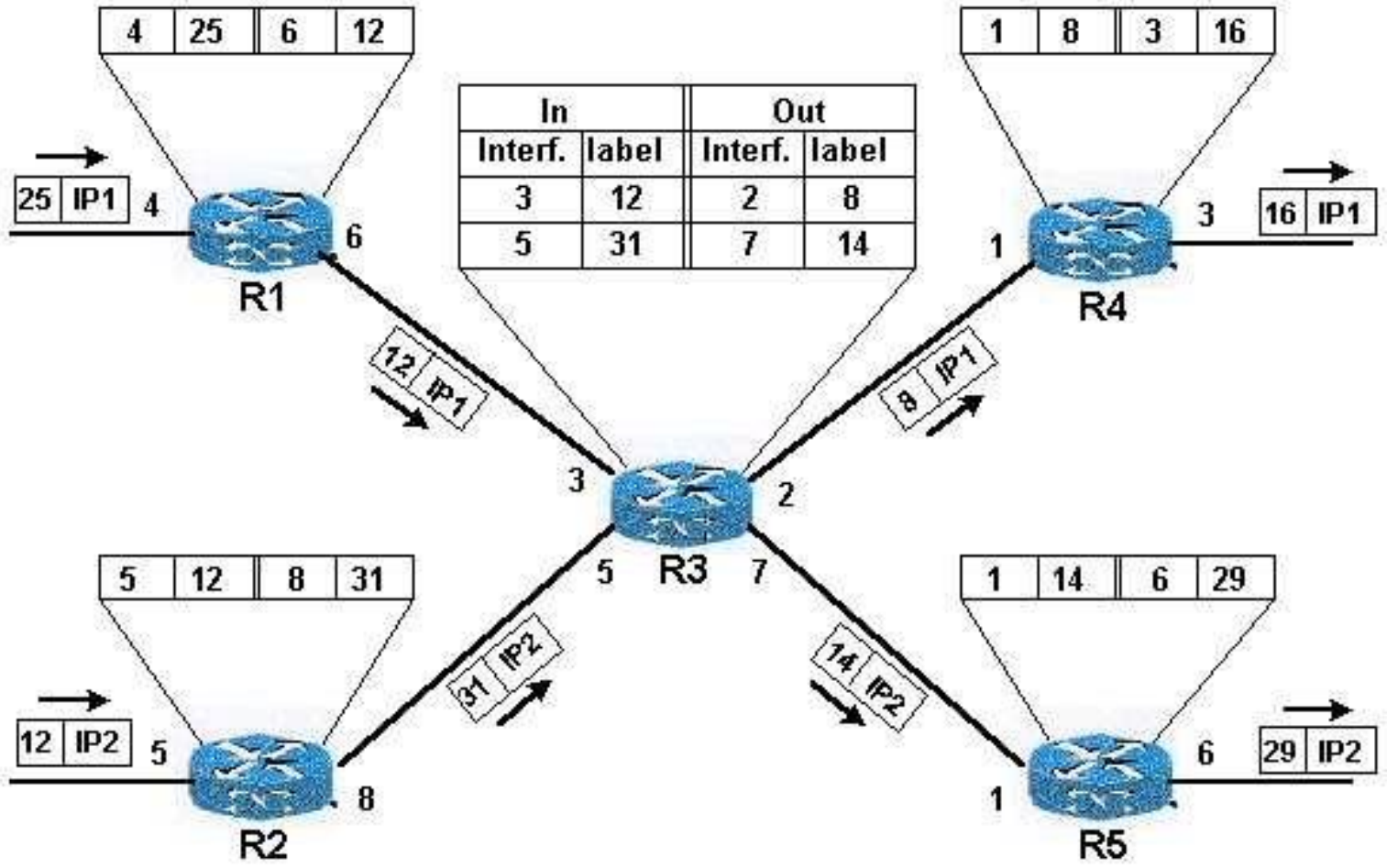
**4. Edge LSR at egress removes label and delivers packet**



**2. Ingress Edge LSR receives packet, performs Layer 3 value-added services, and "labels" packets**

**3. LSR switches packets using label swapping**

# Label Switching Example



# Multi Protocol Label Switching

- Forwarding Equivalency Class (FEC)
  - The FEC is used to describe an association of discrete packets with a destination address and a class of traffic. It allows the grouping of packets into classes.
  - The matching of the FEC with a packet is achieved by using a label to identify each FEC. For different classes of service, different FECs and their associated labels are used.
- Label Switching Router (LSR)
  - Forwards MPLS packets using label switching
  - Capable of forwarding native IP packets
  - Executes one or more IP routing protocols
  - Participates in MPLS control protocols
  - A group of LSRs with the same MPLS level forms an MPLS domain.
- Label Switched Path (LSP)
  - The path through one or more LSRs followed by packets in a particular FEC is called a label-switched path

# MPLS Advantages

- Simplified header processing:
  - It is not necessary for MPLS routers to inspect the packet's network layer headers. Rather, they are only required to do a table lookup and label replacement.
- A network with a state
  - Packets that enter the network from different routers are distinguishable, so that forwarding decisions that depend on the ingress router can be easily made.
  - This is not true of conventional forwarding, since packets are memoryless about their ingress routers.
- More flexible per packet operation
  - The label table of an MPLS switch may contain more information than just the next hop and the new label for a particular FEC.
  - For example, an entry may include the precedence or class of service predetermined for the corresponding FEC.
  - On the contrary, conventional forwarding can only consider information encoded in the packet header.
- Create a Network Path for Traffic Engineering
  - To improve the performance it is sometimes desirable, rather than letting the route be chosen by dynamic routing algorithms from hop to hop in the network, to force a packet to follow a route that is explicitly chosen before the packet enters the network. This capability is necessary to support traffic engineering.

# MPLS Label

- Labels are assigned by downstream routers
- Labels are local to each path and change at each hop
- When there is a hierarchy in the MPLS network, labels can be stacked. Only top level labels are used to process the packet
- Label value contains:
  - Info for next hop
  - Operations to do on the label stack:
    - Swap label
    - Pop an entry off the label stack
    - Replace top label and then push another label