# CS4220
# Computer Networks

## Lecture 5 Network Layer

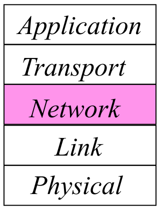
Dr. Xiaobo Charles Zhou
Department of Computer Science

## Network Layer Overview
Chapter 5

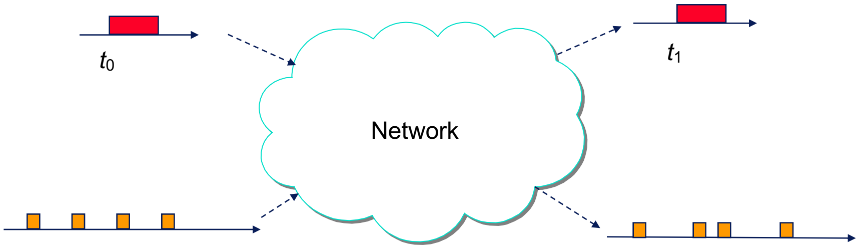
- Design Issues
- Routing Algorithms
- Congestion Control
- Quality of Service
- Internetworking
- Network Layer of the Internet

## The Network Layer

- Network Layer: the most complex layer
  - Requires the coordinated actions of multiple, geographically distributed network elements (switches & routers)
  - Must be able to deal with very large scales
    - Billions of users (people & communicating devices)
  - Biggest Challenges
    - Addressing: where should information be directed to?
    - Routing: what path should be used to get information there?



## Packet Switching



- Transfer of information as payload in data packets
- Packets undergo random delays & possible loss
- Different applications impose differing requirements on the transfer of information

## Network Layer Functions

**What are essential?**

- Routing**: mechanisms for determining the set of best paths for routing packets**
- Forwarding**: transfer of packets from inputs to outputs**
- Priority & Scheduling**: determining order of packet transmission**

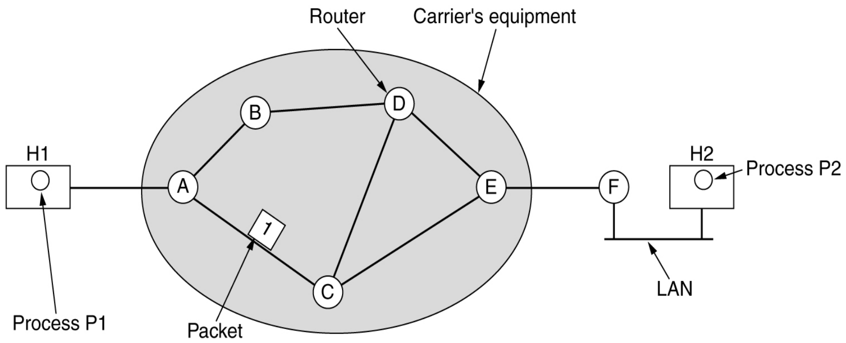
**Optional: congestion control, segmentation & reassembly, security**

## Network Layer Design Issues

- **Store-and-Forward Packet Switching**
- **Services Provided to the Transport Layer**
- **Implementation of Connectionless Service**
- **Implementation of Connection-Oriented Service**
- **Comparison of Virtual-Circuit and Datagram Subnets**

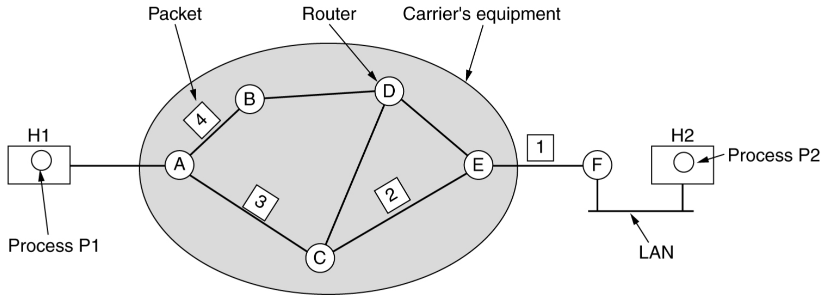
# Store-and-Forward Packet Switching

° **The environment of the network layer protocols for *end-to-end transmission*.**



Why a packet must be stored until it has fully arrived then forwarded?

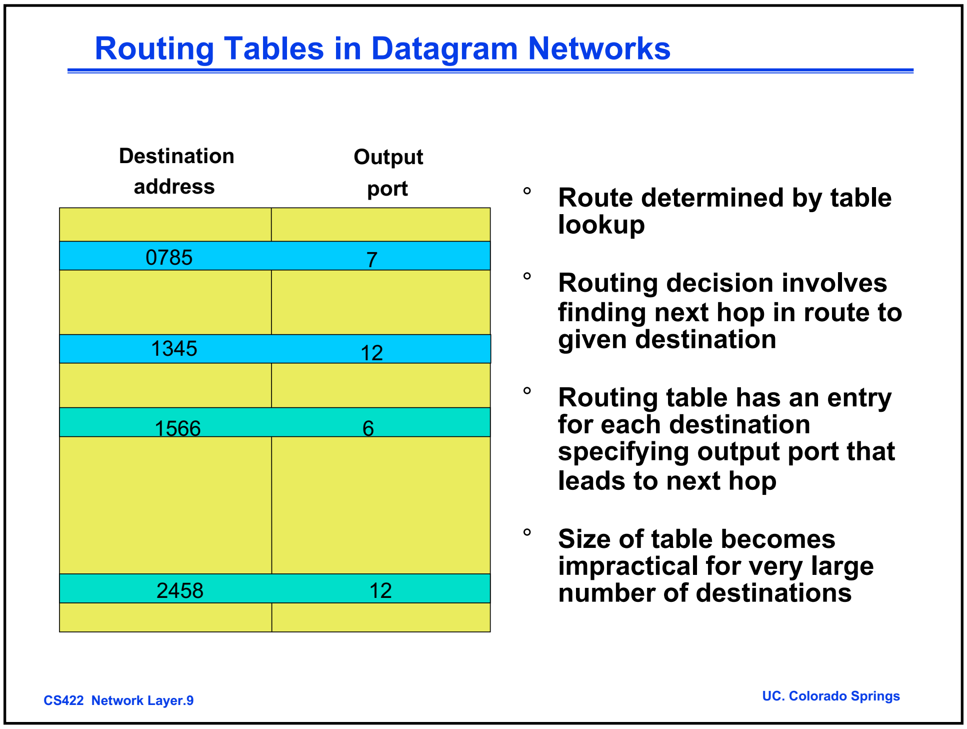
# Implementation of Connectionless Service



| A's table | | | | | | | |
|---|---|---|---|---|---|---|---|
| initially | | later | | C's table | | E's table | |
| A | – | A | – | A | A | A | C |
| B | B | B | B | B | A | B | D |
| C | C | C | C | C | – | C | C |
| D | B | D | B | D | D | D | D |
| E | C | E | B | E | E | E | – |
| F | C | F | B | F | E | F | F |

Dest. Line

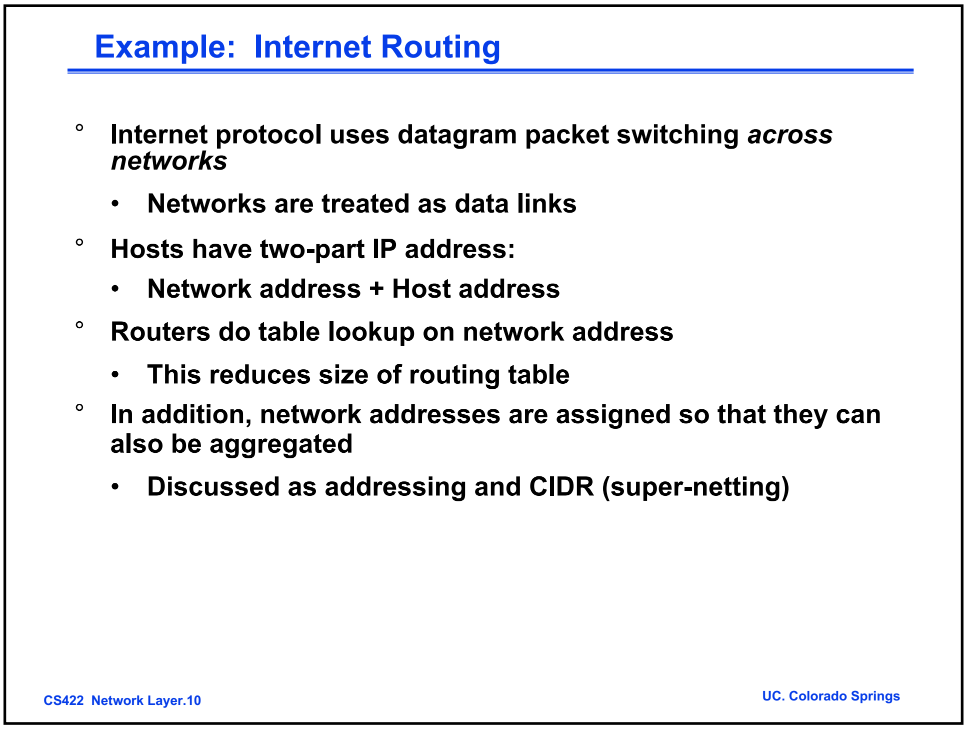
Routing in a datagram subnet.



## Routing Tables in Datagram Networks

| Destination address | Output port |
|---|---|
| | |
| 0785 | 7 |
| | |
| 1345 | 12 |
| | |
| 1566 | 6 |
| | |
| 2458 | 12 |
| | |

- Route determined by table lookup
- Routing decision involves finding next hop in route to given destination
- Routing table has an entry for each destination specifying output port that leads to next hop
- Size of table becomes impractical for very large number of destinations

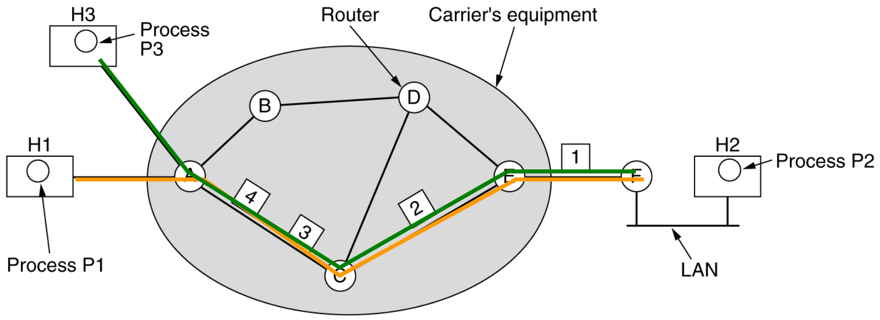


## Example: Internet Routing

- Internet protocol uses datagram packet switching *across networks*
  - Networks are treated as data links
- Hosts have two-part IP address:
  - Network address + Host address
- Routers do table lookup on network address
  - This reduces size of routing table
- In addition, network addresses are assigned so that they can also be aggregated
  - Discussed as addressing and CIDR (super-netting)

## Implementation of Connection-Oriented Service

**Routing within a virtual-circuit subnet.**



A's table

| In | | Out | |
|---|---|---|---|
| H1 | 1 | C | 1 |
| H3 | 1 | C | 2 |

C's table

| | | | |
|---|---|---|---|
| A | 1 | E | 1 |
| A | 2 | E | 2 |

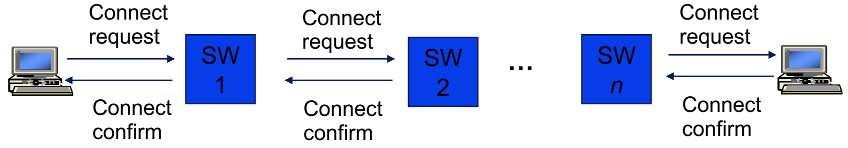
E's table

| | | | |
|---|---|---|---|
| C | 1 | F | 1 |
| C | 2 | F | 2 |

Label switching

Does VC subnets need the capability to route isolated packets from an arbitrary source to an arbitrary destination?

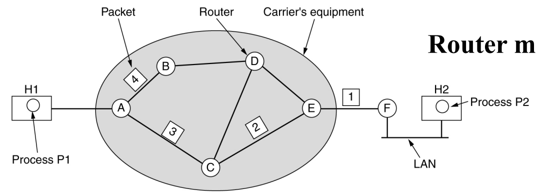
## Connection Setup



Resources do not need to be dedicated to VCs.

- **Signaling messages propagate as route is selected**
- **Signaling messages identify connection and setup tables in switches**
- **Typically a connection is identified by a local tag, Virtual Circuit Identifier (VCI)**
- **Each switch only needs to know how to relate an incoming tag in one input to an outgoing tag in the corresponding output**
- **Once tables are setup, packets can flow along path**

## Two Tradeoffs of Virtual Circuits and Datagrams



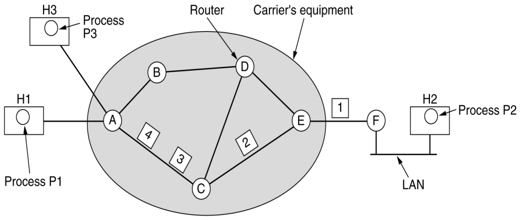
**Router memory space vs. bandwidth**

*Address length vs. # of entries*

A's table

| initially | | later | | C's table | | E's table | |
|---|---|---|---|---|---|---|---|
| A | – | A | – | A | A | A | C |
| B | B | B | B | B | A | B | D |
| C | C | C | C | C | – | C | C |
| D | B | D | B | D | D | D | D |
| E | C | E | B | E | E | E | – |
| F | C | F | B | F | E | F | F |

Dest. Line



**setup time vs. address parsing time**

**QoS vs. congestion avoidance**

| A's table | | | | C's table | | | | E's table | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| H1 | 1 | C | 1 | A | 1 | E | 1 | C | 1 | F | 1 |
| H3 | 1 | C | 2 | A | 2 | E | 2 | C | 2 | F | 2 |
| In | | Out | | | | | | | | | |

## Comparison of Virtual-Circuit and Datagram Subnets

| 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 Algorithms

- Routing algorithms: part of the network layer software responsible for deciding which output lines an incoming packet should be transmitted on

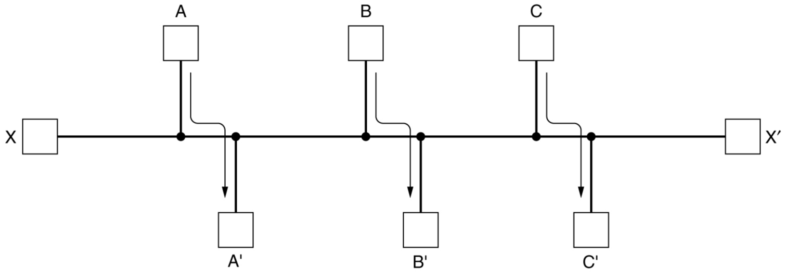
- Static vs. adaptive routing
  - The Optimality Principle
  - Shortest Path Routing
  - Flooding
  - Distance Vector Routing (RIP)
  - Link State Routing (OSPF)
  - Hierarchical Routing
  - Broadcast Routing
  - Multicast Routing
  - Routing for Mobile Hosts
  - Routing in Ad Hoc Networks

## Routing Algorithms (2)

- Desirable routing properties, but often contradictory
  - Correctness
  - Simplicity
  - Robustness
  - Stability
  - Fairness
  - optimality

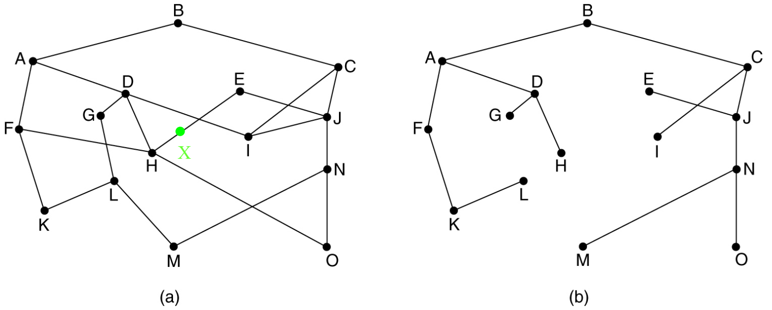
Q1: why there are contradictory goals?
Optimality vs. fairness

Should X-X' traffic be shut off to maximize the total flow on the horizontal link?



## The Optimality Principle

- Optimality principle: if router *J* is on the optimal path from router *I* to router *K*, then the optimal path from *J* to *K* also falls along the same route.



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

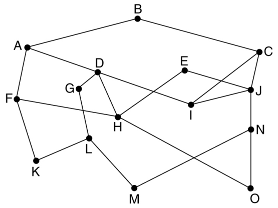
Q1: must a sink tree be unique? An example?

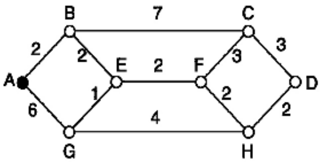
Q2: each packet will be delivered within a finite # of hops?

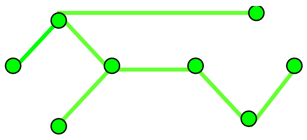
## Shortest Path Routing

- Shortest path: to choose a route between a given pair of routers, finds the shortest path between them on the graph.

Wait! What is a path length?

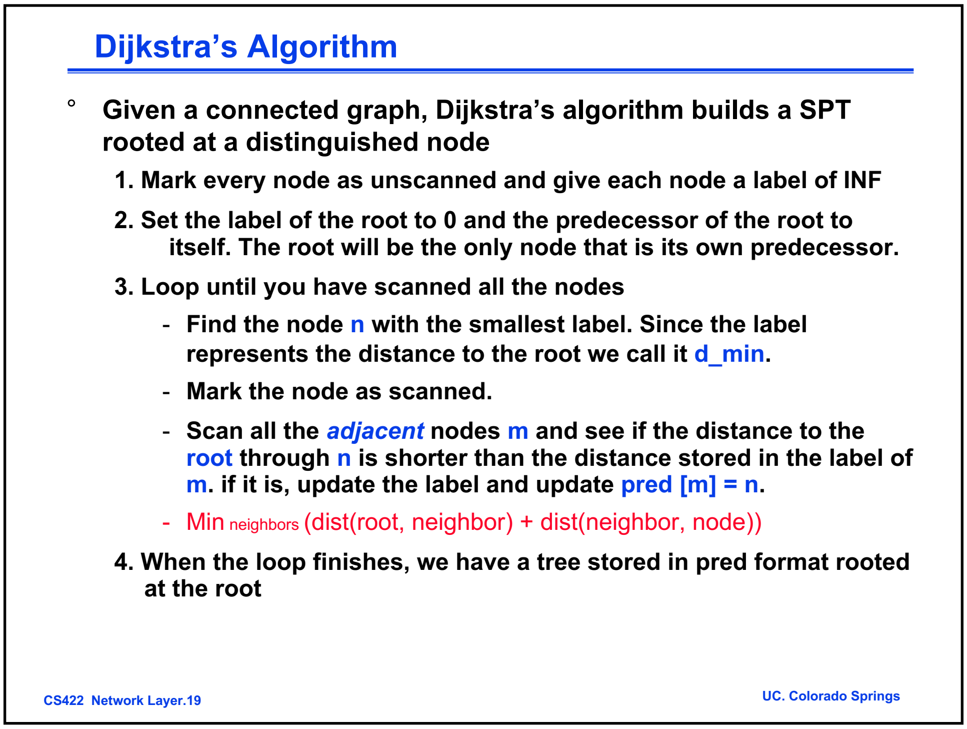






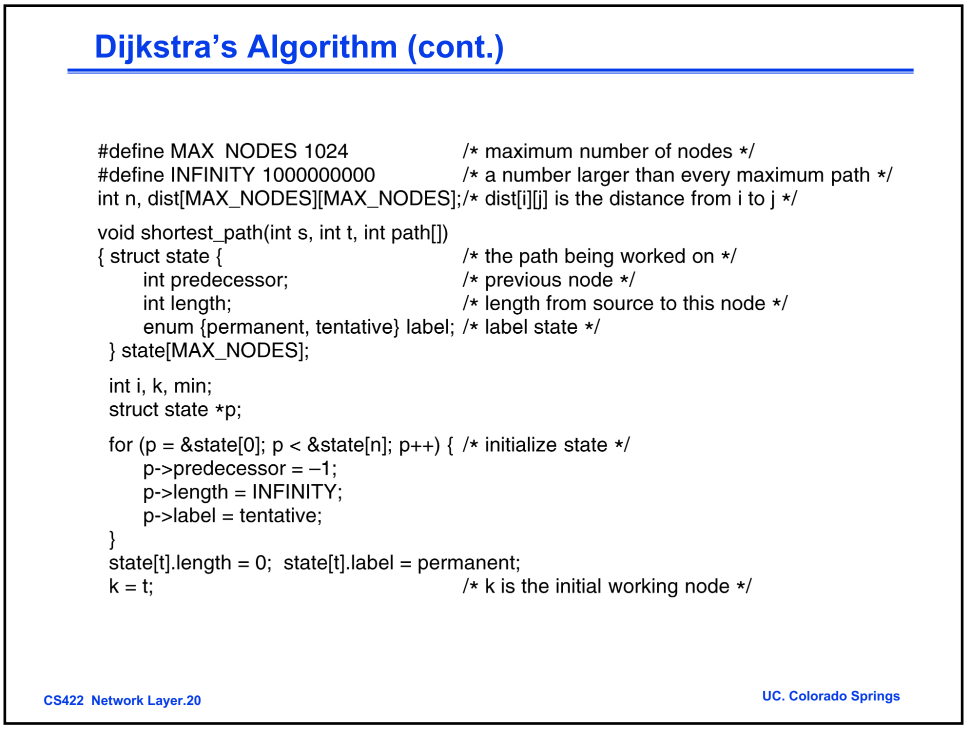
What is the difference between a sink tree and SPT?

Is SPT routing static or adaptive?



## Dijkstra’s Algorithm

- Given a connected graph, Dijkstra’s algorithm builds a SPT rooted at a distinguished node
  1. Mark every node as unscanned and give each node a label of INF
  2. Set the label of the root to 0 and the predecessor of the root to itself. The root will be the only node that is its own predecessor.
  3. Loop until you have scanned all the nodes
     - Find the node n with the smallest label. Since the label represents the distance to the root we call it d_min.
     - Mark the node as scanned.
     - Scan all the *adjacent* nodes m and see if the distance to the root through n is shorter than the distance stored in the label of m. if it is, update the label and update pred [m] = n.
     - Min$_{\text{neighbors}}$ (dist(root, neighbor) + dist(neighbor, node))
  4. When the loop finishes, we have a tree stored in pred format rooted at the root



## Dijkstra’s Algorithm (cont.)

```
#define MAX  NODES 1024                   /* maximum number of nodes */
#define INFINITY 1000000000               /* a number larger than every maximum path */
int n, dist[MAX_NODES][MAX_NODES];/* dist[i][j] is the distance from i to j */

void shortest_path(int s, int t, int path[])
{ struct state {                          /* the path being worked on */
      int predecessor;                    /* previous node */
      int length;                         /* length from source to this node */
      enum {permanent, tentative} label;  /* label state */
  } state[MAX_NODES];

  int i, k, min;
  struct state *p;

  for (p = &state[0]; p < &state[n]; p++) { /* initialize state */
      p->predecessor = –1;
      p->length = INFINITY;
      p->label = tentative;
  }
  state[t].length = 0;  state[t].label = permanent;
  k = t;                                  /* k is the initial working node */
```

## Dijkstra’s Algorithm (cont.)

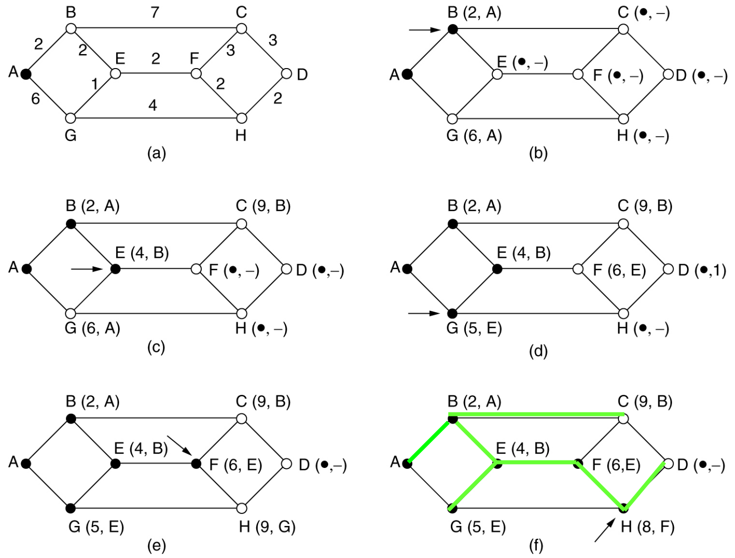
```
do {                                        /* Is there a better path from k? */
     for (i = 0; i < n; i++)                /* this graph has n nodes */
          if (dist[k][i] != 0 && state[i].label == tentative) {
               if (state[k].length + dist[k][i] < state[i].length) {
                    state[i].predecessor = k;
                    state[i].length = state[k].length + dist[k][i];
               }
          }

     /* Find the tentatively labeled node with the smallest label. */
     k = 0; min = INFINITY;
     for (i = 0; i < n; i++)
          if (state[i].label == tentative && state[i].length < min) {
               min = state[i].length;
               k = i;
          }
     state[k].label = permanent;
} while (k != s);

/* Copy the path into the output array. */
i = 0;  k = s;
do {path[i++] = k; k = state[k].predecessor; } while (k >= 0);
}
```

**Dijkstra's algorithm to compute the shortest path through a graph.**

## An Example of Dijkstra’s Algorithm



## Flooding

- Flooding: every incoming packet is sent out over every outgoing line except the one it arrived on.

Is flooding static or adaptive?

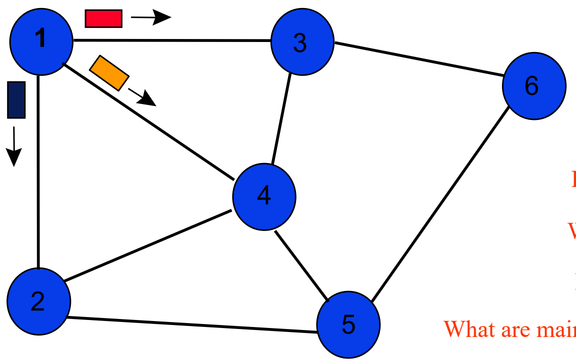
What is the major problem with flooding? Give an example.

How to handle the problem?

What are main nice properties of flooding?

How flooding can be terminated?

## A Flooding Example



Is flooding static or adaptive?

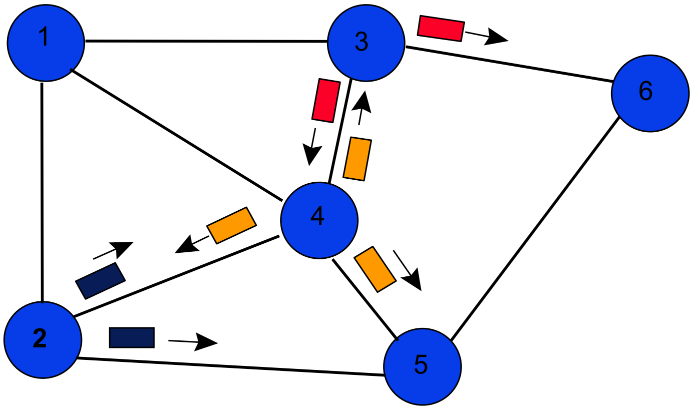
What is the major problem?

How to handle the problem?

What are main nice properties of flooding?

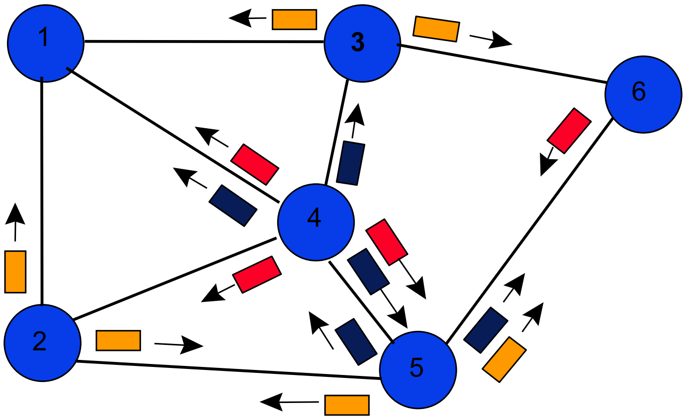
How flooding can be terminated?

## A Flooding Example (cont.)



Flooding is initiated from Node 1:  Hop 2 transmissions

## A Flooding Example (cont.)



Flooding is initiated from Node 1:  Hop 3 transmissions

## Limited Flooding

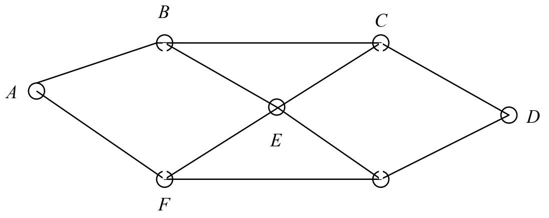
- Time-to-Live field in each packet limits number of hops to certain diameter
- Each switch adds its ID before flooding; discards repeats
- Source puts sequence number in each packet; switches records source address and sequence number and discards repeats

What are main nice properties of flooding?

Robustness; always follow shortest path

## Limited Flooding Example

- Suppose the following network uses flooding as the routing algorithm. If a packet sent by A to D has a maximum hop of 3, list all the routes it will take. Also tell how many hops worth of bandwidth it consumes. Assume the bandwidth weight of the lines is the same.

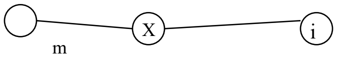


## Distance Vector Routing (RIP)

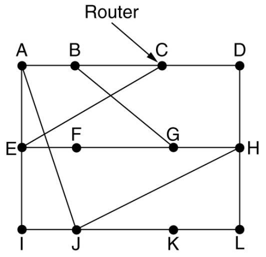
- **RIP Algorithm operates by having each router maintain a vector table giving the best known distance to each destination and which line to use to get there. The tables are updated by exchanging information with the neighbors.**

- **Vector table: one entry for each router in the subnet; each entry contains two parts: preferred outgoing line to use for that destination and an estimate of the time or distance to the destination.**

- **The router is assumed to know the distance to each neighbor and update the vector table *periodically* by changing it with neighbors.**
  - **# hops**
  - **Delay (ECHO)**



## An Example of RIP Updating Process



(a)

What is the major problem?

| To | A | I | H | K | New estimated delay from J | 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 | New routing table for J |  |

Vectors received from J's four neighbors

(b)

**(a) A subnet. (b) Input from A, I, H, K, and the new routing table for J.**

# The Count-to-Infinity Problem

- It converges to the correct answer quickly to good news but slowly to bad news.

| 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)

B knows A is 1 hop away while all other Routers still think A is down, why?

What is the spreading rate of good news?

How many exchanges needed in a N-hop subnet?

| 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)

Does B know that C’s path runs through B?

Why spreading rate of bad news so slow?

What is the core problem?

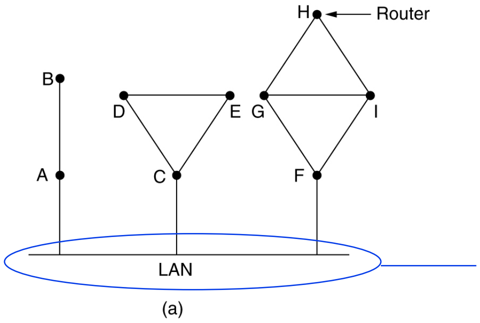
# Link State Routing

- RIP was replaced by link state routing in 1979, due to
  - Not taking bandwidth into account (use queue length)
  - Count-to-infinity problem
- In Link state routing, each router must
  - Discover its neighbors, learn their network address.
  - Measure the delay or cost to each of its neighbors.
  - Construct a packet telling all it has just learned.
  - Send this packet to all other routers.
  - Compute the shortest path to every other router.

**Does distance vector routing (RIP) knows the topology of the subnet?**

## Learning about the Neighbors

- To learn who its neighbors are, send a special HELLO packet on each point-to-point line.

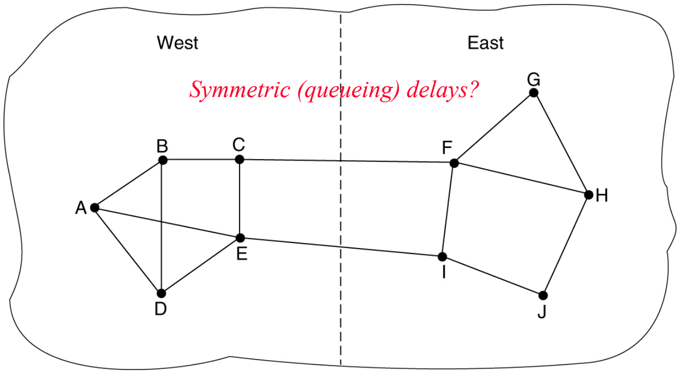


(a) Nine routers and a LAN.

## Measuring Line Cost

- To estimate the delay to its neighbors, send a special ECHO packet on each point-to-point line to get RTT.

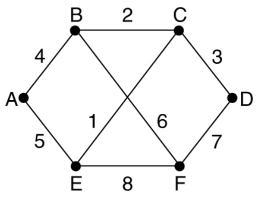
How to take the load into account when measuring the delay? Or should we?
Bottleneck oscillation between CF and EI links?



A subnet in which the East and West parts are connected by two lines.

## Building Link State Packets

- A state packet starts with the ID of the sender, a seq#, age, and a list of neighbors with delay information.



(a)

Link State Packets

| A | B | C | D | E | F |
|---|---|---|---|---|---|
| Seq. | Seq. | Seq. | Seq. | Seq. | Seq. |
| Age | Age | Age | Age | Age | Age |
| B 4 | A 4 | B 2 | C 3 | A 5 | B 6 |
| E 5 | C 2 | D 3 | F 7 | C 1 | D 7 |
|  | F 6 | E 1 |  | F 8 | E 8 |

(b)

**(a) A subnet.** **(b) The link state packets for this subnet.**

When to build the link state packets?

## Distributing the Link State Packets

- Flooding is used to distribute the link state packets.

What is the major problem with flooding?

How to handle the problem?

(source router, sequence number)

How to make the sequence number unique?

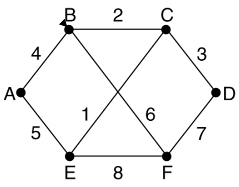
32-bit sequence number

What happens if a router crashes, losing its track, and starts again?

What happens if sequence number is corrupted, say 65,540, not 4.

*Age* field

## A Packet Buffer



(a)

Link State Packets

| A | B | C | D | E | F |
|---|---|---|---|---|---|
| Seq. | Seq. | Seq. | Seq. | Seq. | Seq. |
| Age | Age | Age | Age | Age | Age |
| B 4 | A 4 | B 2 | C 3 | A 5 | B 6 |
| E 5 | C 2 | D 3 | F 7 | C 1 | D 7 |
|  | F 6 | E 1 |  | F 8 | E 8 |

(b)

*The packet buffer for router B*

|  |  |  | Send flags |  |  | ACK flags |  |  |  |
|---|---|---|---|---|---|---|---|---|---|
| Source | Seq. | Age | A | C | F | A | C | F | Data |
| A | 21 | 60 | 0 | 1 | 1 | 1 | 0 | 0 |  |
| F | 21 | 60 | 1 | 1 | 0 | 0 | 0 | 1 |  |
| E | 21 | 59 | 0 | 1 | 0 | 1 | 0 | 1 |  |
| C | 20 | 60 | 1 | 0 | 1 | 0 | 1 | 0 |  |
| D | 21 | 59 | 1 | 0 | 0 | 0 | 1 | 1 |  |

## Computing the New Routes

- Once a router has accumulated a full set of link state packets, it can construct the entire subnet graph because every link is represented (every link is represented twice indeed, once for each direction)

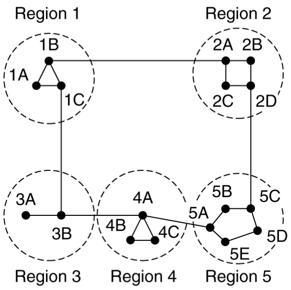
- Dijkstra’s algorithm can be run on the router to construct the shortest path to all possible destinations.

What is the memory required to store the input data for a subnet with *n* routers – each of them has *k* neighbors?

***OSPF is used in the Internet!***

# Hierarchical Routing

- The router routing table grows proportionally to the network size, consuming lots memory and CPU resources!



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 |

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 |

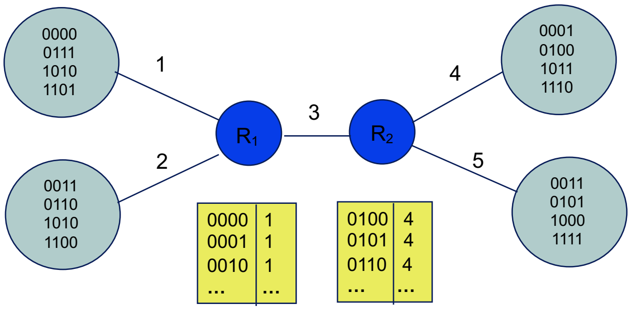
How table space gains?

What is the penalty?

How many levels desirable?

How addressing schemes in the Internet supports hierarchical routing?

# Non-Hierarchical Addresses and Routing

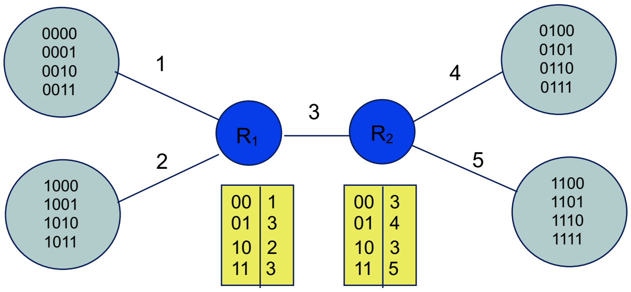


| | |
|---|---|
| 0000 | 1 |
| 0001 | 1 |
| 0010 | 1 |
| … | … |

| | |
|---|---|
| 0100 | 4 |
| 0101 | 4 |
| 0110 | 4 |
| … | … |

- No relationship between addresses & routing proximity
- Routing tables require 16 entries each, but what if there is address proximity support?

## Hierarchical Addresses and Routing



| | |
|---|---|
| 00 | 1 |
| 01 | 3 |
| 10 | 2 |
| 11 | 3 |

| | |
|---|---|
| 00 | 3 |
| 01 | 4 |
| 10 | 3 |
| 11 | 5 |

- Prefix indicates network where host is attached
- Routing tables require 4 entries each

## Hierarchical Routing (cont.)

- Consider a subnet with 720 routers.

  (1) how many table entries needed in each router if no hierarchy?

  (2) how many table entries needed in each router if a two-level hierarchy, 24 regions of 30 routers each? 53

  (3) how many table entries needed in each router if a three-level hierarchy, 8 clusters, each containing 9 regions of 10 routers? 25

  Kamoun & Kleinrock: Optimal number of levels for an $N$ route subnet is $\ln N$, requiring $e \cdot \ln N$ entries per router.

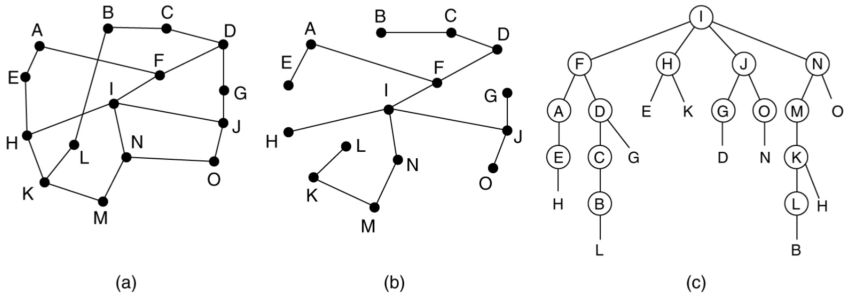
## Broadcast Routing

- Broadcast: send a message to all destination simultaneously!
  - how about the source sends a distinct message to each destination as Point-to-Point?
  - how about flooding?
  - Multi-destination routing: each message contains a list of destinations (bitmap)
  - Sink tree, or *spanning tree*, for directing routing
    - Excellent bandwidth utilization: minimal # of packets
    - Requiring knowledge of tree at each router

Does a router has the knowledge with distance vector routing?
How about with link state routing?

## Broadcast Routing – Reverse Path Forwarding

- The packet coming from the best route is often the first copy to arrive at the router!



Reverse path forwarding. (a) A subnet. (b) a Sink tree (though wrong). (c) The tree built by reverse path forwarding.

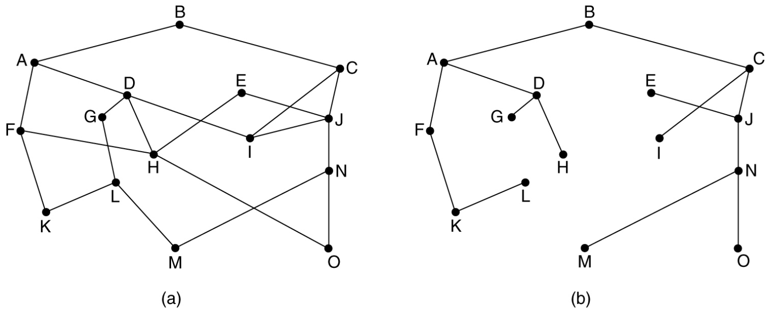
What is the key issue here?

How many packets generated by (b) and (c) respectively?

What is the principal advantage of reverse path forwarding?

## Reverse Path Forwarding and Sink Tree

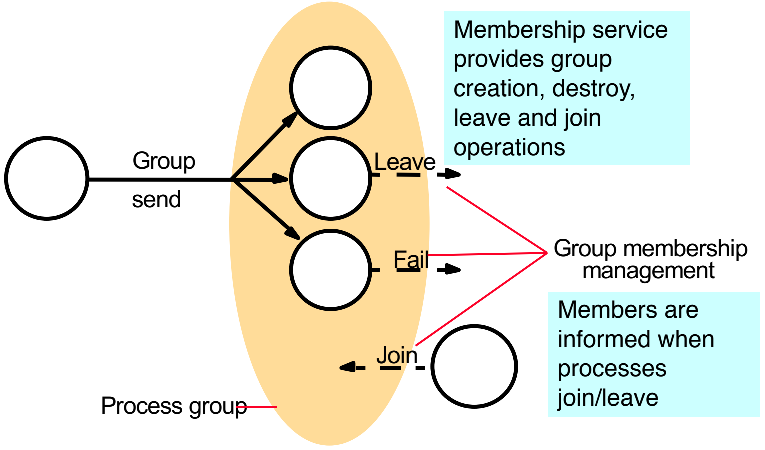
- How many packets are generated by a broadcast from B, using
  - reverse path forwarding
  - the sink tree.



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

## Group Communications (Multicast)

- Send a message to a group of destinations
  - Why not using point-to-point or broadcasting?

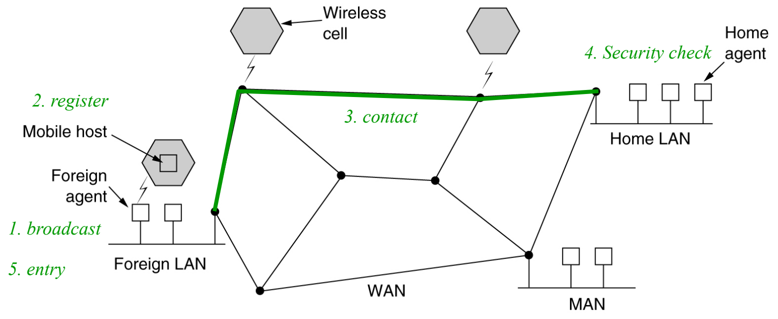


Membership service provides group creation, destroy, leave and join operations

Members are informed when processes join/leave

## Routing for Mobile Hosts (and Mobile IP)

- Mobile hosts: migratory hosts and roaming hosts which are away from home and still want to be connected.
  - Hosts are mobile with a permanent home; all routers are fixed
- Registration procedure makes routing feasible



A WAN to which LANs, MANs, and wireless cells are attached.

## MANET: Mobile Ad Hoc Networks

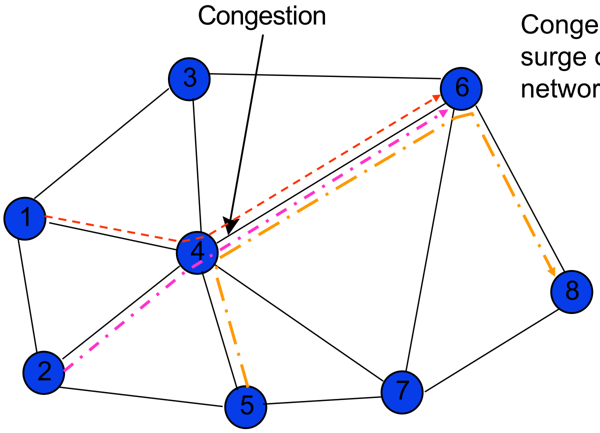
Possibilities when the routers are mobile:

- Military vehicles on battlefield.
  - No infrastructure.
- A fleet of ships at sea.
  - All moving all the time
- Emergency works at earthquake.
  - The infrastructure destroyed.
- A gathering of people with notebook computers.
  - In an area lacking 802.11.

## Traffic Management: Congestion Control

- **General Principles of Congestion Control**
- **Congestion Prevention Policies**
- **Congestion Control in Virtual-Circuit Subnets**
- **Congestion Control in Datagram Subnets**
- **Load Shedding**
- **Jitter Control**

## Why Congestion?



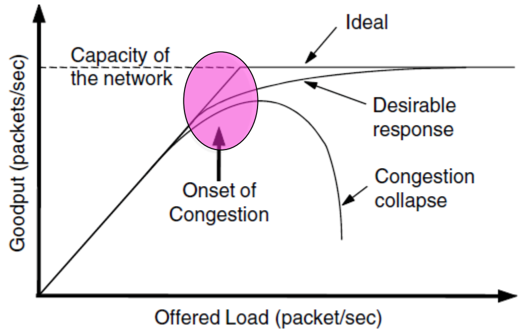
Congestion occurs when a surge of traffic overloads network resources

Approaches to Congestion Control:

- Preventive Approaches (open –loop): Scheduling & Reservations
- Reactive Approaches (closed-loop): Detect & Throttle/Discard

## Congestion

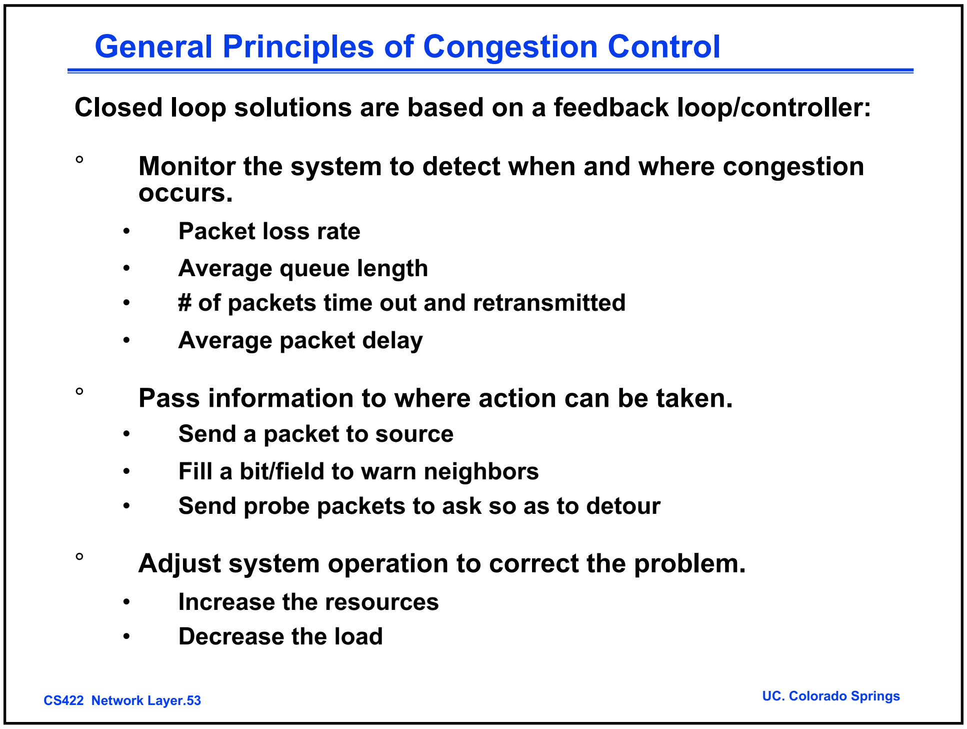
- Congestion results when too much traffic is offered; performance degrades due to loss/retransmissions
  - Goodput (=useful packets) trails offered load



When congestion occurs, what happens if insufficient memory?
What happens if an infinite amount of memory?
What about a slow CPU?

## Congestion Control vs. Flow Control

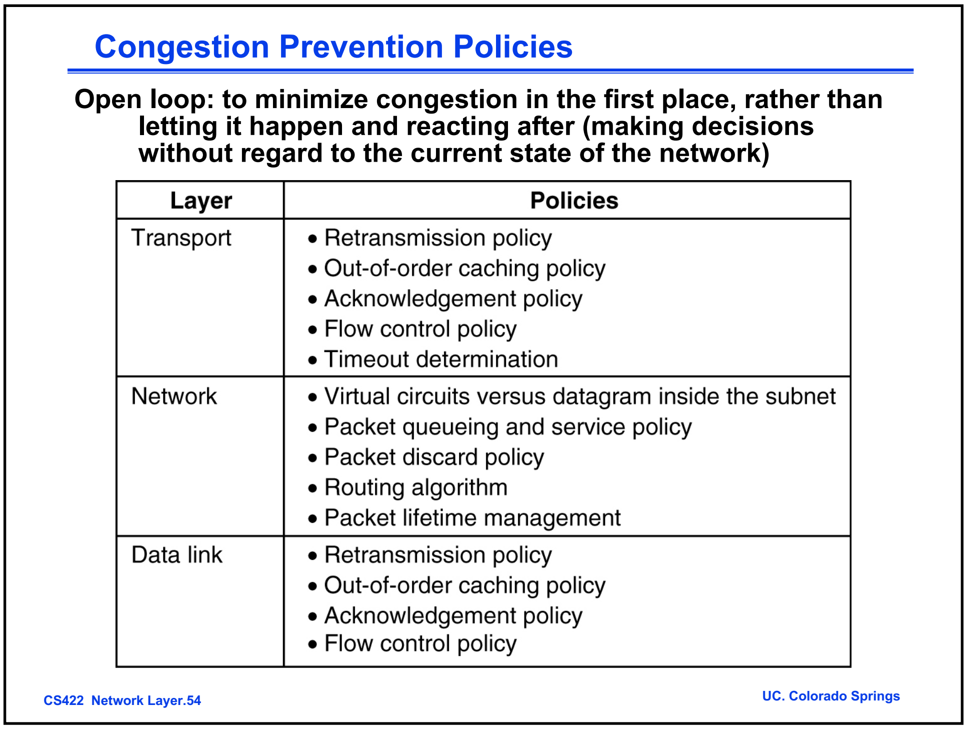
- Global issue in the subnet vs. point-to-point between a pair of sender and receiver
- Consider a network with 1000 Gbps, and a supercomputer tries to use it transfer a file to a PC at 1 Gbps. Is congestion control needed? Is flow control needed?
- Consider another case: a store-and-forward network with 1 Mbps lines and 1000 PCs, half of PCs want to transfer files at 100 kbps to the other half. Is flow control needed? Is congestion control needed?
- Key difference: is the network cannot handle the traffic or the receiver cannot handle the traffic!



## General Principles of Congestion Control

**Closed loop solutions are based on a feedback loop/controller:**

- **Monitor the system to detect when and where congestion occurs.**
  - **Packet loss rate**
  - **Average queue length**
  - **# of packets time out and retransmitted**
  - **Average packet delay**
- **Pass information to where action can be taken.**
  - **Send a packet to source**
  - **Fill a bit/field to warn neighbors**
  - **Send probe packets to ask so as to detour**
- **Adjust system operation to correct the problem.**
  - **Increase the resources**
  - **Decrease the load**



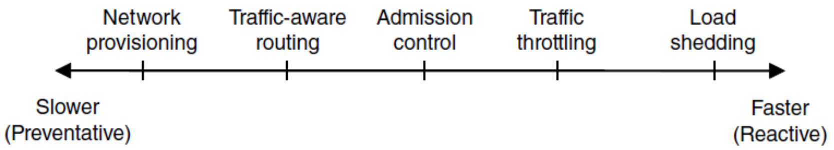
## Congestion Prevention Policies

**Open loop: to minimize congestion in the first place, rather than letting it happen and reacting after (making decisions without regard to the current state of the network)**

| 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 |

## Congestion Control – Approaches

- Network must do its best with the offered load
  - Different approaches at different timescales
  - Nodes should also reduce offered load (Transport)

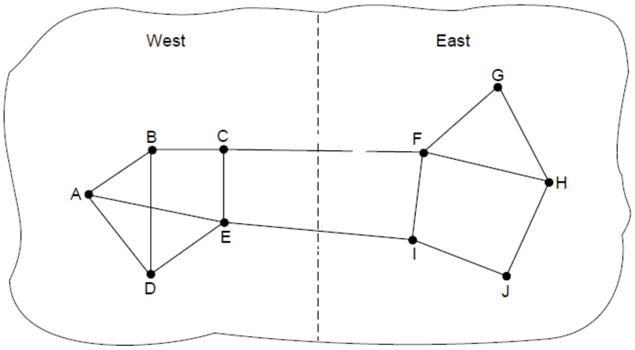


Network provisioning — Traffic-aware routing — Admission control — Traffic throttling — Load shedding

Slower (Preventative) — Faster (Reactive)

## Traffic-Aware Routing

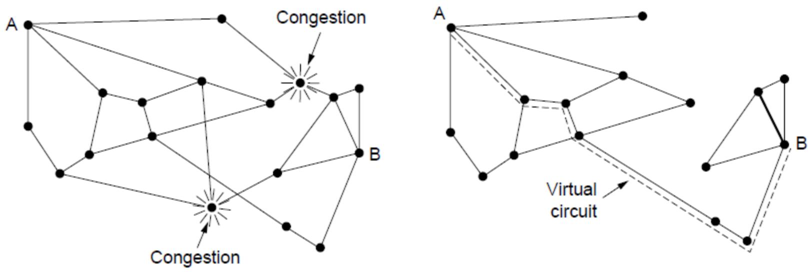
- Choose routes depending on traffic, not just topology
  - E.g., use *EI* for West-to-East traffic if *CF* is loaded
  - But take care to avoid oscillations



## Admission Control

° **Admission control allows a new traffic load only if the network has sufficient capacity, e.g., with virtual circuits**

- **Can combine with looking for an uncongested route**



*Network with some congested nodes*

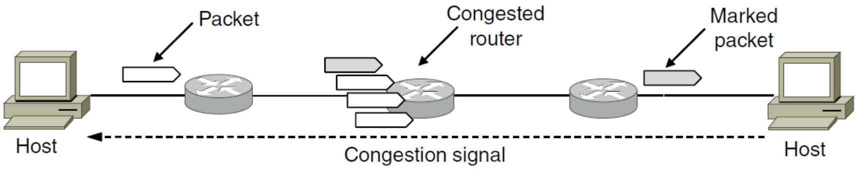
*Uncongested portion and route AB around congestion*

CN5E by Tanenbaum & Wetherall, © Pearson Education-Prentice Hall and D. Wetherall, 2011

## Traffic Throttling

○ **Congested routers signal hosts to slow down traffic**

- **ECN (Explicit Congestion Notification) marks packets and receiver returns signal to sender**

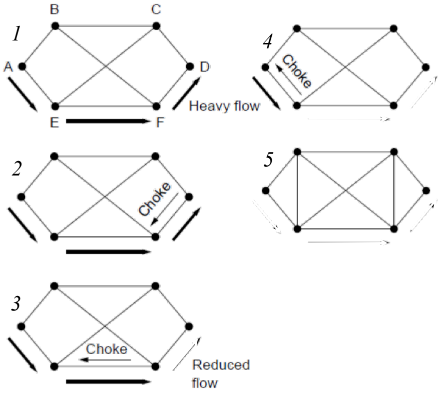


## Load Shedding (1)

When all else fails, network will drop packets (shed load)

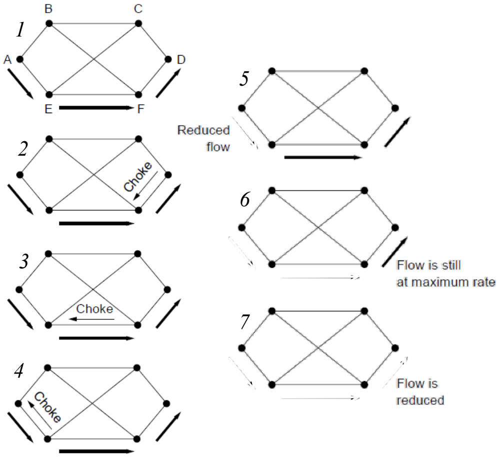
Can be done end-to-end or link-by-link by a **choke packet**

Link-by-link (right) produces rapid relief



## Load Shedding (2)

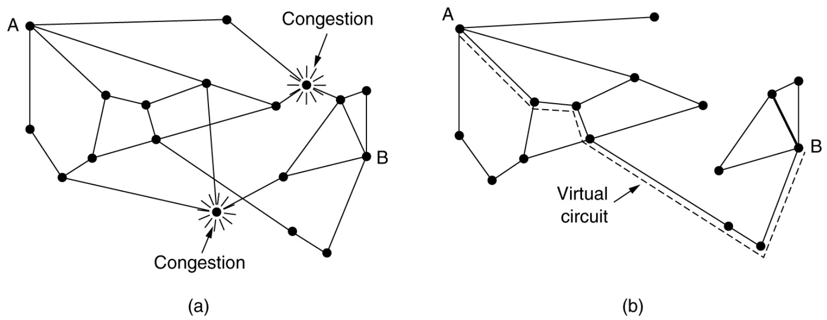
End-to-end (right) takes longer to have an effect, but can better target the cause of congestion



## Congestion Control in Virtual-Circuit Subnets

- Admission control: on/off model.
- Admitted but detouring
- Admitted with QoS agreement and resource reservation

What is the penalty (vs. over-provisioning)?



(a) A congested subnet. (b) A redrawn subnet, eliminates congestion and a virtual circuit from A to B.

## Congestion Control in Datagram Subnets

- Threshold-based utilization warning
  - Which factor used for threshold calculation?
  - How to measure the utilization? Instantaneously or smoothed?
  - How to set the threshold?
  - How many threshold levels?
- The Warning Bit in ACKs
- Choke packets to the source for slowing down

**Isn’t this approach too slow in reaction?**

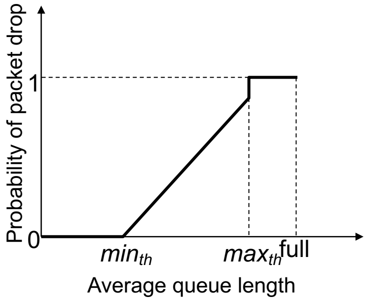
## Random Early Detection (RED)

- Isn't it more effective to deal with congestion after it is first detected than letting it gum up the works and then trying to deal with it?
  - Have time for action before it is too late
- A router takes action when its *average* queue length on some line exceeds a threshold
  - What actions the router can take? Tell the source to slow down or do something by itself? What TCP does?
  - How you compare Warning bit method and RED method?
    - Explicitly vs. Implicitly
    - When dropping is done

*RED may not good/feasible in wireless networks, why?*
*-- think about the assumption that a packet loss is due to congestion.*

## RED Algoithm

- Packets produced by TCP will reduce input rate in response to network congestion
- Early random drop (ERD): discard packets before buffers are full
- Random drop causes some sources to reduce rate before others, causing gradual reduction in aggregate input rate

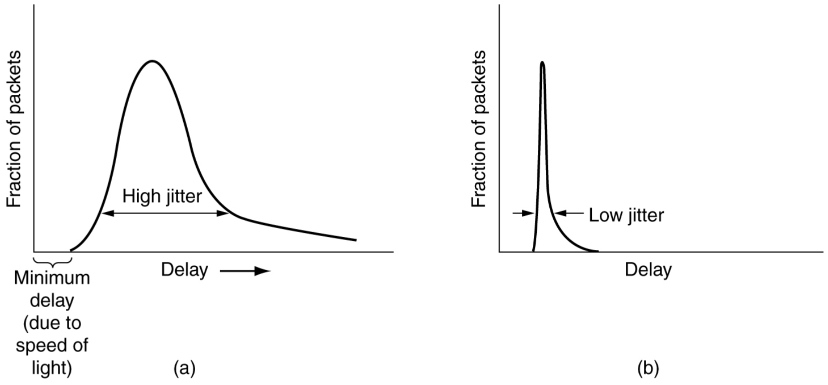


Algorithm:

- Maintain running average of queue length
- If $Q_{avg}$ < minthreshold, do nothing
- If $Q_{avg}$ > maxthreshold, drop packet
- If in between, drop packet according to probability
- Flows that send more packets are more likely to have packets dropped

## Jitter Control

° Jitter: the standard variation in the packet inter-arrival times

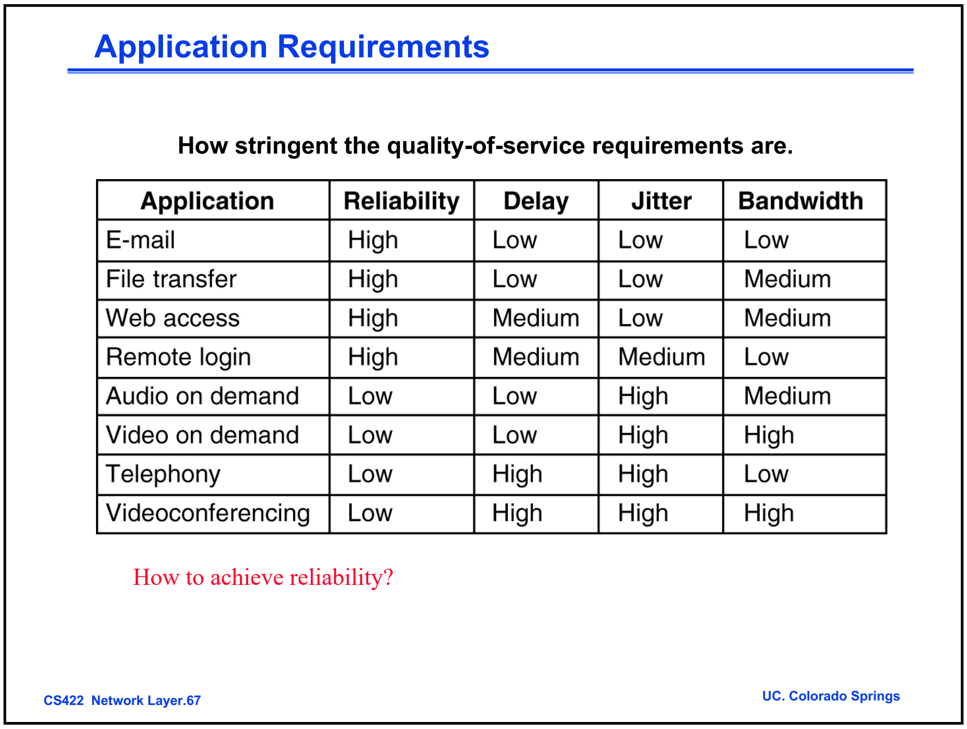


(a) High jitter. (b) Low jitter.

How to control jitter?

## Quality of Service (QoS)

- QoS Requirements
- Techniques for Achieving Good Quality of Service
- Integrated Services (IntServ)
- Differentiated Services (DiffServ)
- Label Switching and MPLS

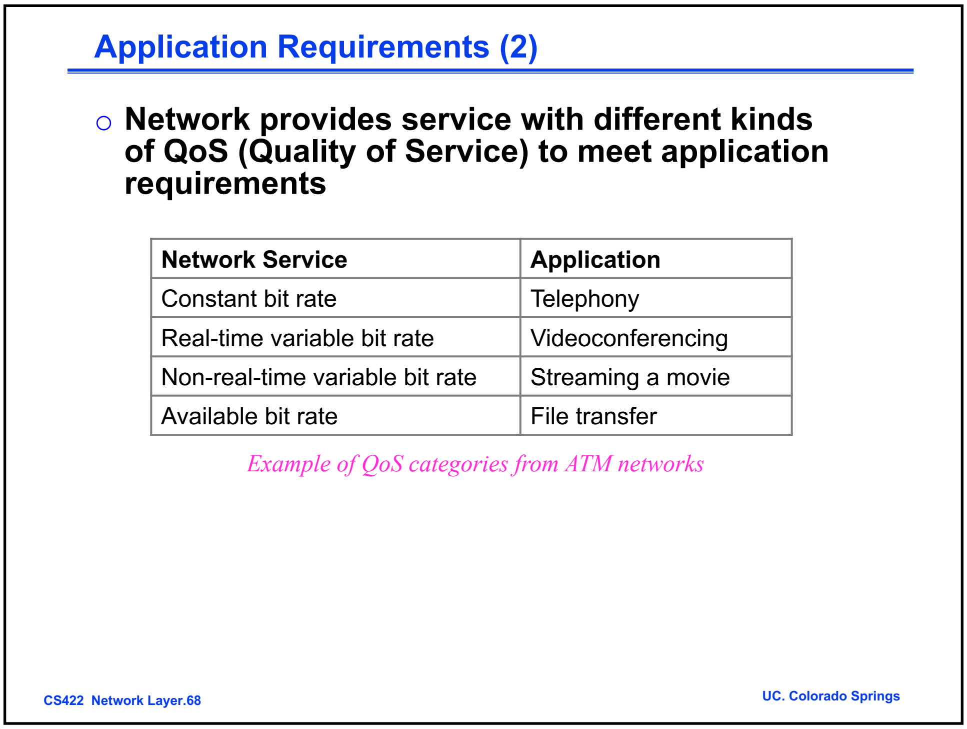


## Application Requirements

**How stringent the quality-of-service requirements are.**

| 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 |

How to achieve reliability?



## Application Requirements (2)

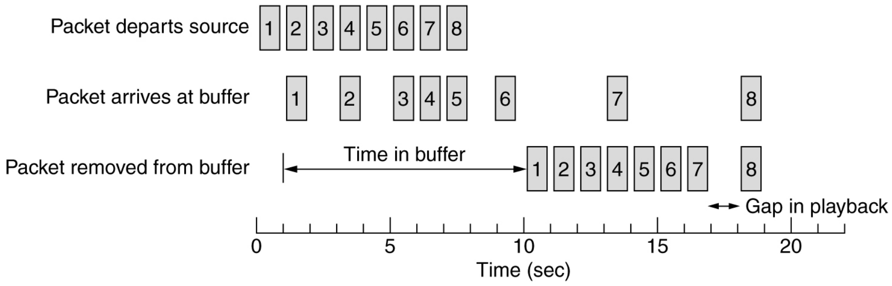
- **Network provides service with different kinds of QoS (Quality of Service) to meet application requirements**

| Network Service | Application |
|---|---|
| Constant bit rate | Telephony |
| Real-time variable bit rate | Videoconferencing |
| Non-real-time variable bit rate | Streaming a movie |
| Available bit rate | File transfer |

*Example of QoS categories from ATM networks*

## Achieving Good QoS - Buffering

**Smoothing the output stream by buffering packets.**



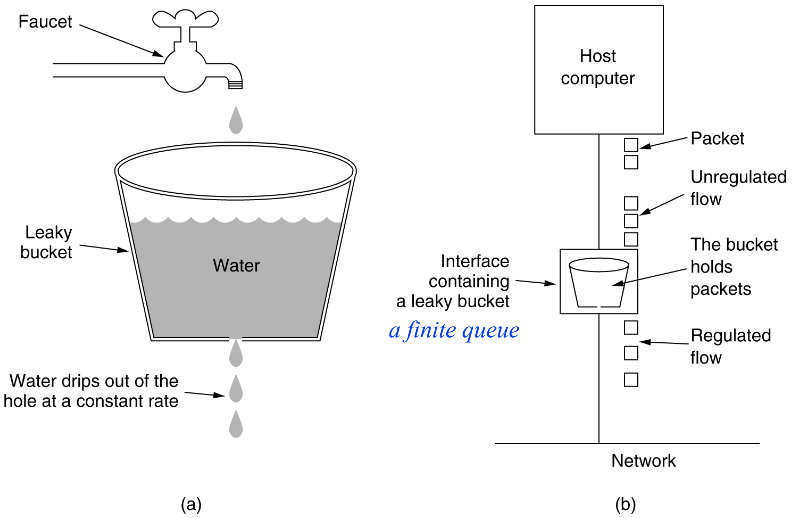
How buffering affect the QoS requirements?

What kind of applications benefit from buffering most?

## Achieving Good QoS – Traffic Shaping

- **Burstiness: flows are variable**
  - **video compression methods such as MPEG (1-4) are based on similarities between consecutive frames**
  - **can produce large variations in data rate, and affect delays**
- **SLA: how the carrier tell if the customer is following the agreement and what to do if it is not?**
- **Traffic shaping: regulate/smooth the average rate and burstiness of data transmission to reduce congestion and helps the carrier link up to its promise**
- **How to shape?**

## The Leaky Bucket Algorithm



**(a) A leaky bucket with water.** **(b) a leaky bucket with packets.**

## The Leaky Bucket Example

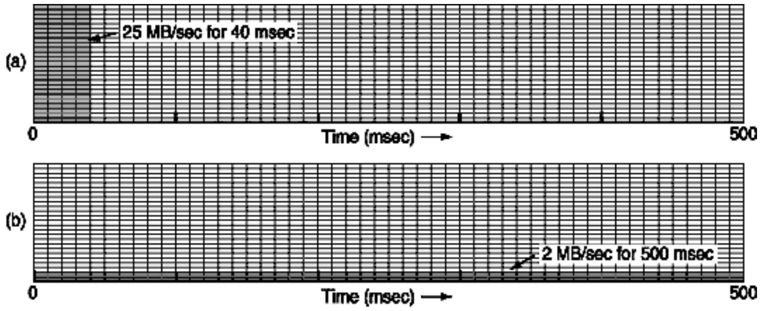
- **Data comes to a router in 1 MB bursts, that is, an input runs at 25 MB/s (burst rate) for 40 msec. The router is able to support 2 MB/s output (leaky) rate. The router uses a leaky bucket for traffic shaping.**

  **(1) How large the bucket should be so there is no data loss?**

  **(2) Now, if the leaky bucket size is 1MB, how long the *maximum burst interval* can be?**

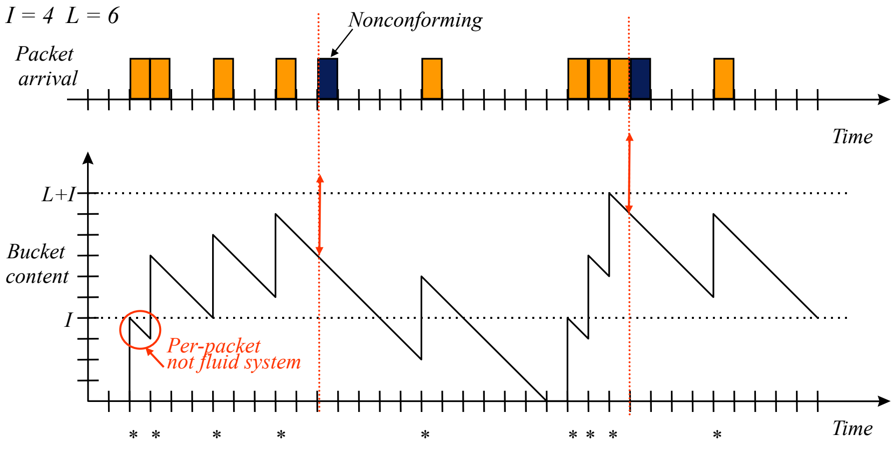
## The Leaky Bucket Example Diagram

◦ Example: data comes to a router in 1 MB bursts, that is, an input runs at 25 MB/s for 40 msec. The router is able to support 2 MB/s outgoing (leaky) rate. The leaky bucket size is 1MB.



(a) Input to a leaky bucket. (b) Output from a leaky bucket.

## Leaky Bucket Example

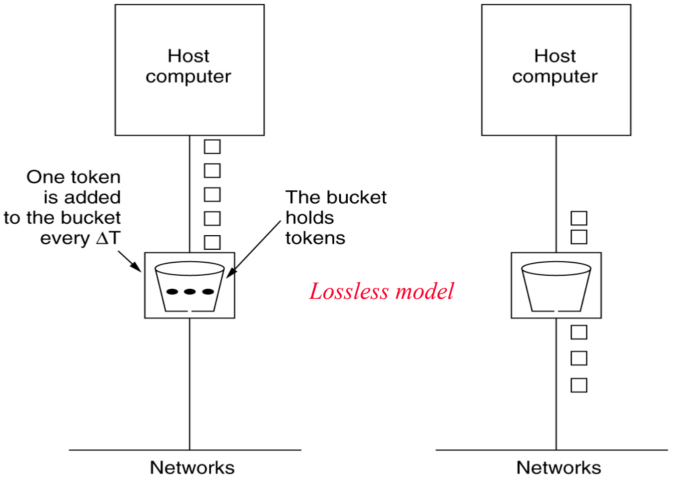


*Non-conforming packets not allowed into bucket & hence not included in calculations*

*maximum burst size (MBS = 3 packets)*

## The Token Bucket Algorithm

**Does the leaky bucket algorithm allow *saving*: what happens that no flow come in during some time, and a burstiness occurs?**



*Lossless model*

(a) **Before.** (b) **After.**

## The Token Bucket Example 1

° **A network uses a token bucket for traffic shaping. A new token is put into the bucket every 1 msec. Each token is good for one packet, which contains 100 bytes of data. What is the maximum sustainable (input) data rate?**

# The Token Bucket Example 2

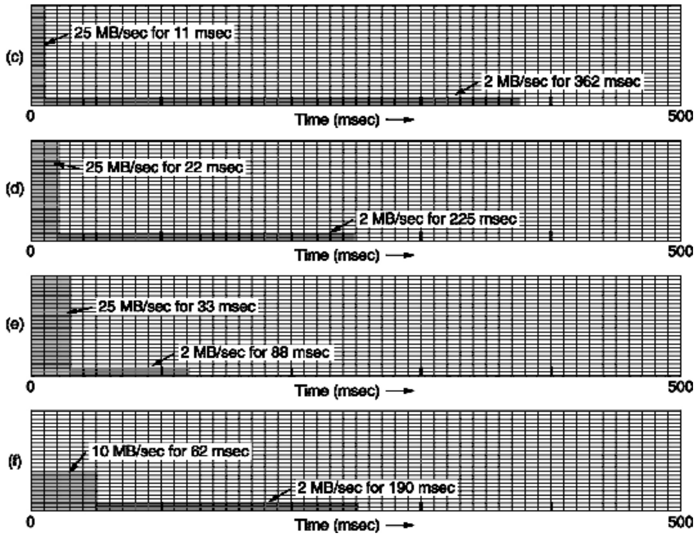
- **Given: the token bucket capacity *C*, the token arrival rate *p*, and the maximum output rate *M*, calculate the maximum burst interval *S***

  **$C + pS = MS$**

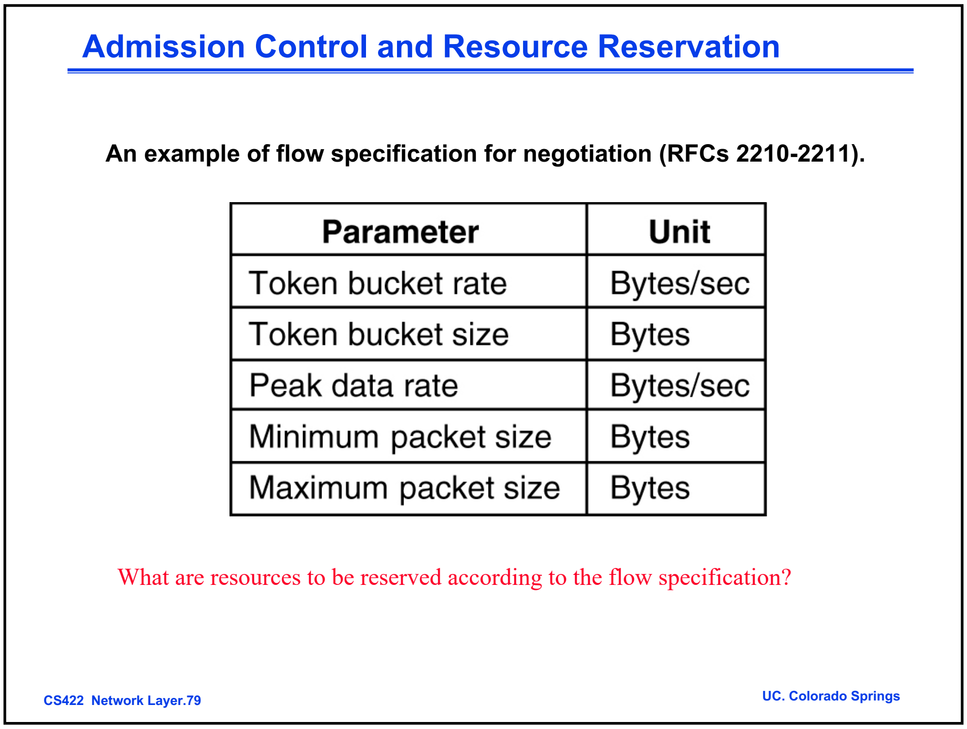
- **Example 2: data comes to a router in 1 MB bursts, that is, an input runs at 25 MB/s (burst rate) for 40 msec. The router uses a token bucket with capacity of 250KB for traffic shaping. Initially, the bucket is full of tokens. And, the tokens are generated and put into the bucket in a rate of 2 MB/s.**

  **What will be the output from the token bucket?**

# The Token Bucket Example 2 Diagram



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 of 1MB.**

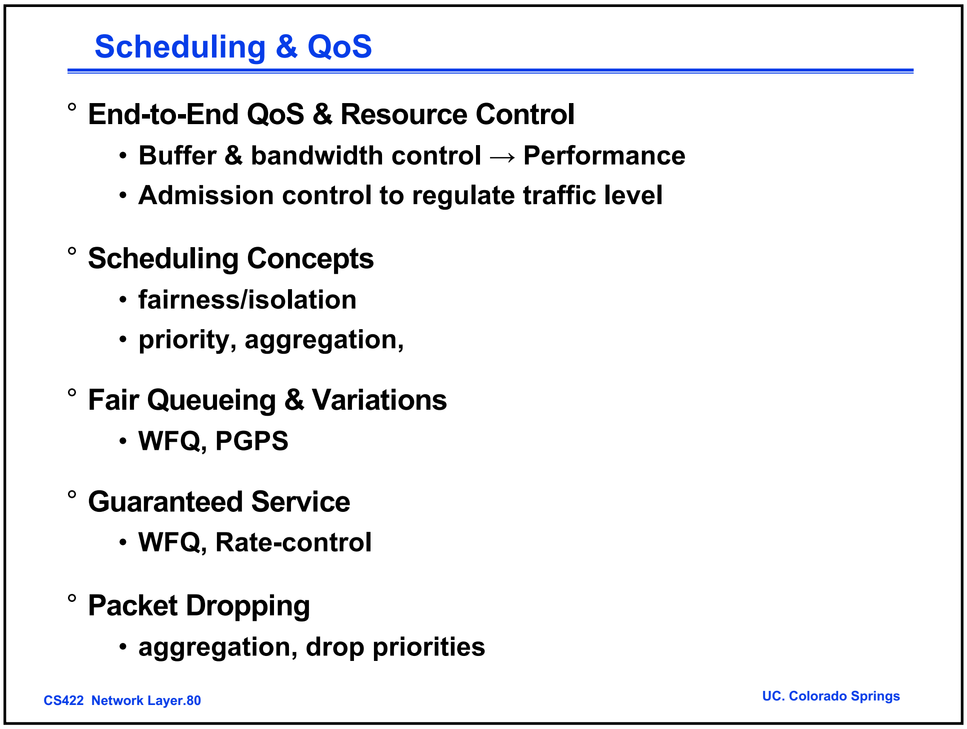


## Admission Control and Resource Reservation

**An example of flow specification for negotiation (RFCs 2210-2211).**

| Parameter | Unit |
|---|---|
| Token bucket rate | Bytes/sec |
| Token bucket size | Bytes |
| Peak data rate | Bytes/sec |
| Minimum packet size | Bytes |
| Maximum packet size | Bytes |

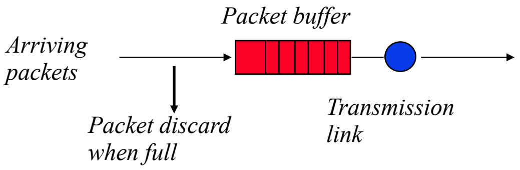
What are resources to be reserved according to the flow specification?



## Scheduling & QoS

- **End-to-End QoS & Resource Control**
  - **Buffer & bandwidth control → Performance**
  - **Admission control to regulate traffic level**
- **Scheduling Concepts**
  - **fairness/isolation**
  - **priority, aggregation,**
- **Fair Queueing & Variations**
  - **WFQ, PGPS**
- **Guaranteed Service**
  - **WFQ, Rate-control**
- **Packet Dropping**
  - **aggregation, drop priorities**

## FIFO Queueing



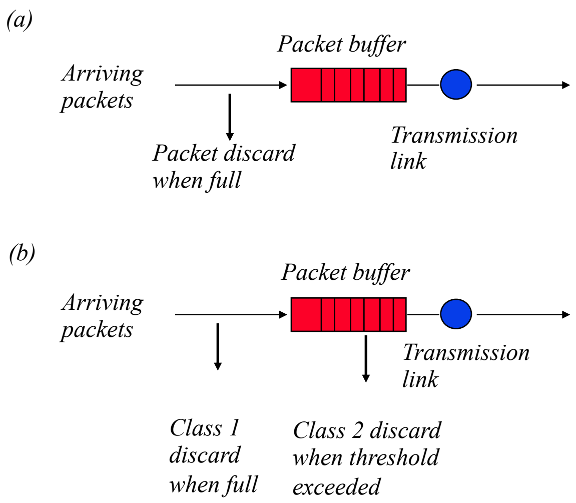
- All packet flows share the same buffer
- Transmission Discipline:  First-In, First-Out
- Buffering Discipline:  Discard arriving packets if buffer is full (Alternative:  random discard; pushout head-of-line, i.e. oldest, packet)

*How about aggressiveness vs. fairness?*

## FIFO Queueing

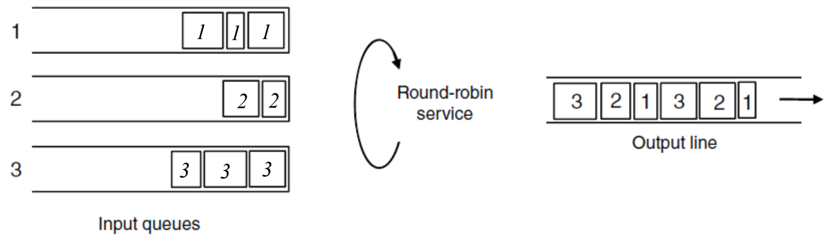
- Cannot provide differential QoS to different packet flows
  - Different packet flows interact strongly
- Statistical delay guarantees via load control
  - Restrict number of flows allowed (connection admission control)
  - Difficult to determine performance delivered
- Finite buffer determines a maximum possible delay
- Buffer size determines loss probability
  - But depends on arrival & packet length statistics
- Variation:  packet enqueueing based on queue thresholds
  - some packet flows encounter blocking before others
  - higher loss, lower delay

## FIFO w/o and w/ Discard Priority



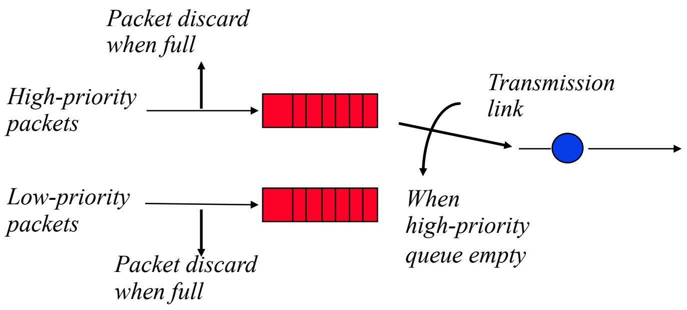
## Packet Scheduling

- **Packet scheduling divides router/link resources among traffic flows with alternatives to FIFO (First In First Out)**



*Example of round-robin queuing*

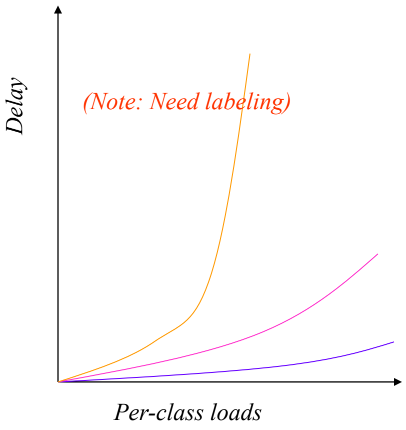
## HOL Priority Queueing



- High priority queue serviced until empty
- High priority queue has lower waiting time
- Buffers can be dimensioned for different loss probabilities
- Surge in high priority queue can cause low priority queue to saturate

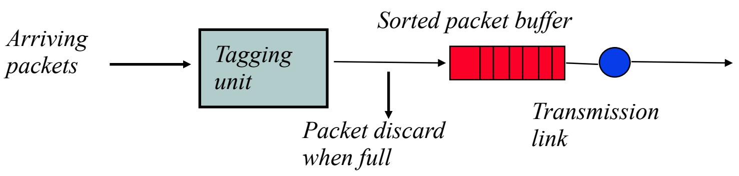
## HOL Priority Features

*Strict priority vs. WTP*



- Provides differential QoS
- Pre-emptive priority: lower classes invisible
- Non-preemptive priority: lower classes impact higher classes through residual service times
- High-priority classes can hog all of the bandwidth & starve lower priority classes
- Need to provide some isolation between classes

## Earliest Due Date Scheduling



Arriving packets

Tagging unit

Sorted packet buffer

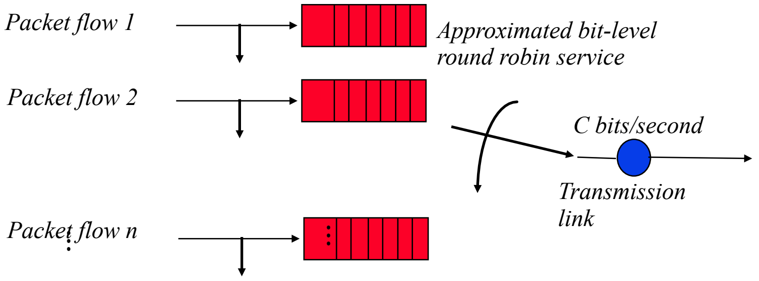
Packet discard when full

Transmission link

- Queue in order of “due date”
  - packets requiring low delay get earlier due date
  - packets without delay get indefinite or very long due dates

*What we will learn in real-time systems?*

## Fair Queuing / Generalized Processor Sharing (GPS)



Packet flow 1

Packet flow 2

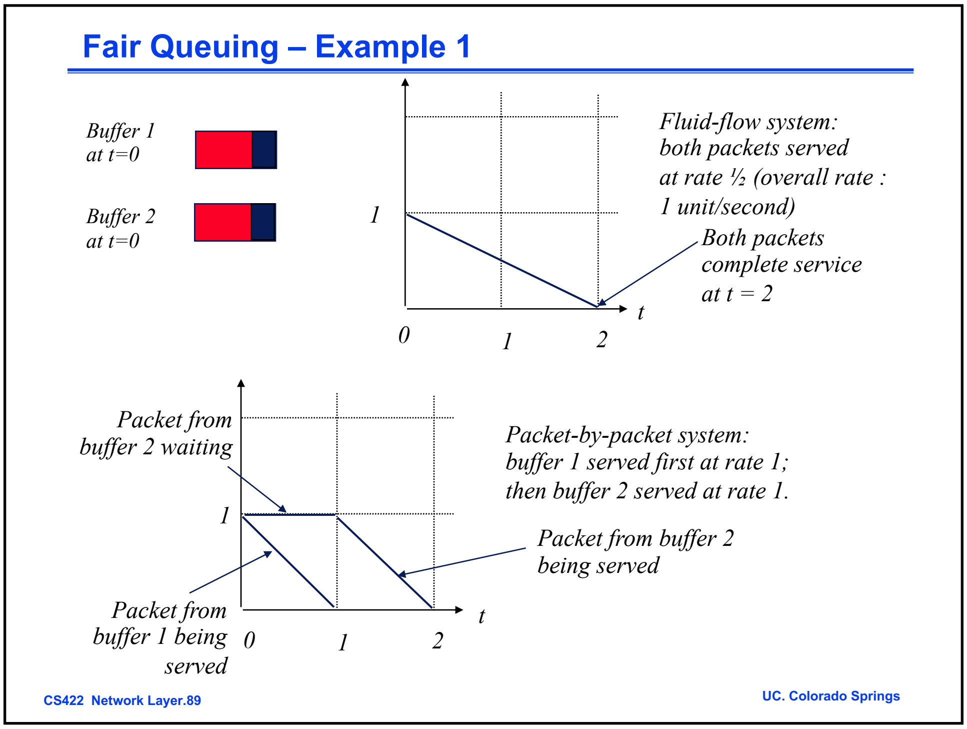
Packet flow n

Approximated bit-level round robin service

C bits/second

Transmission link

- Each flow has its own logical queue: prevents hogging; allows differential loss probabilities
- C bits/sec allocated equally among non-empty queues
  - transmission rate = C / n(t), where n(t)=# non-empty queues
- Idealized system assumes *fluid flow* from queues
- Implementation requires approximation: simulate fluid system; sort packets according to completion time in ideal system



## Fair Queuing – Example 1

Buffer 1 at t=0

Buffer 2 at t=0

Fluid-flow system: both packets served at rate ½ (overall rate : 1 unit/second)

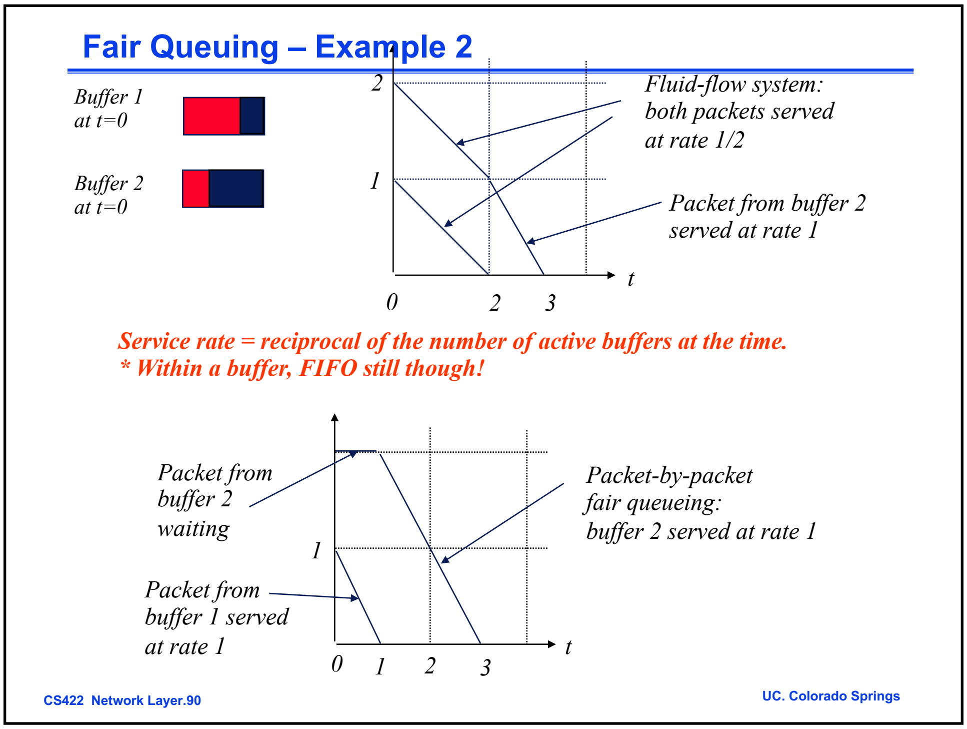
Both packets complete service at t = 2

Packet from buffer 2 waiting

Packet-by-packet system: buffer 1 served first at rate 1; then buffer 2 served at rate 1.

Packet from buffer 2 being served

Packet from buffer 1 being served



## Fair Queuing – Example 2

Buffer 1 at t=0

Buffer 2 at t=0

Fluid-flow system: both packets served at rate 1/2

Packet from buffer 2 served at rate 1

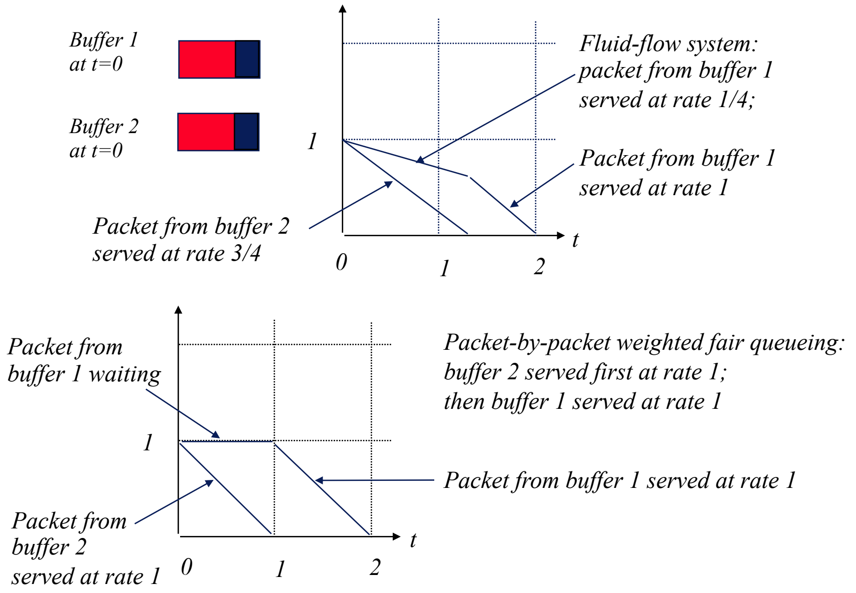
*Service rate = reciprocal of the number of active buffers at the time.*
** Within a buffer, FIFO still though!*

Packet from buffer 2 waiting

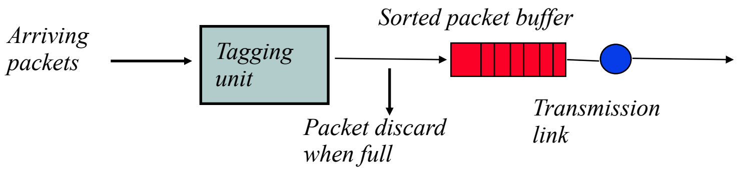
Packet-by-packet fair queueing: buffer 2 served at rate 1

Packet from buffer 1 served at rate 1

# WFQ



# Packetized GPS/WFQ



- Compute packet completion time in ideal system
  - add tag to packet
  - sort packet in queue according to tag
  - serve according to HOL
- WFQ and its many variations form the basis for providing QoS in packet networks

## Admission Control (1)

- Admission control takes a traffic flow specification and decides whether the network can carry it
  - Sets up packet scheduling to meet QoS

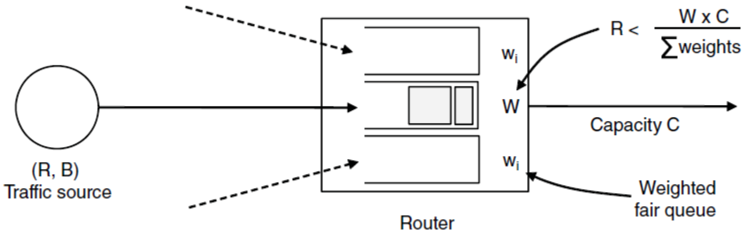
| Parameter | Unit |
|---|---|
| Token bucket rate | Bytes/sec |
| Token bucket size | Bytes |
| Peak data rate | Bytes/sec |
| Minimum packet size | Bytes |
| Maximum packet size | Bytes |

*Example flow specification*

## Admission Control (2)

Construction to guarantee bandwidth B and delay D:

- Shape traffic source to a (R, B) token bucket
- Run WFQ with weight W / all weights > R/capacity
- Holds for all traffic patterns, all topologies

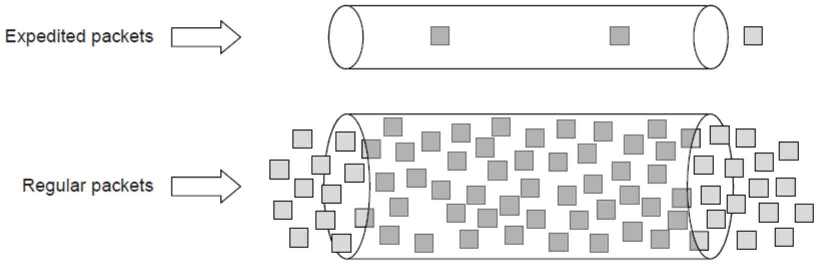


## Integrated Services

- Design with QoS for each flow; handles multicast traffic.
- Admission with RSVP (Resource reSerVation Protocol):
  - Receiver sends a request back to the sender
  - Each router along the way reserves resources
  - Routers merge multiple requests for same flow
  - Entire path is set up, or reservation not made

## Differentiated Services (1)

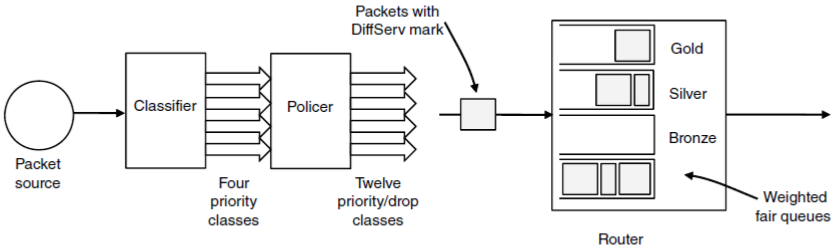
- Design with classes of QoS; customers buy what they want
  - Expedited class is sent in preference to regular class
  - Less expedited traffic but better quality for applications



*CN5E by Tanenbaum & Wetherall, © Pearson Education-Prentice Hall and D. Wetherall, 2011*

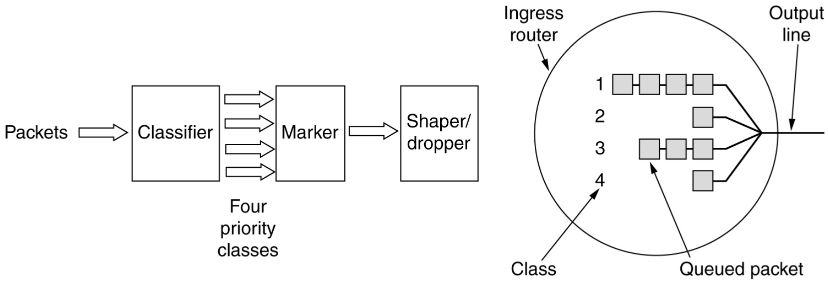
## Differentiated Services (2)

- Implementation of DiffServ:
  - Customers mark desired class on packet
  - ISP shapes traffic to ensure markings are paid for
  - Routers use WFQ to give different service levels



## Assured Forwarding (priority-based)

A possible implementation of the data flow for assured forwarding
(4 priority classes with three dropping probabilities)



8-bit TOS in the IP header for packet marking (IP compatible)

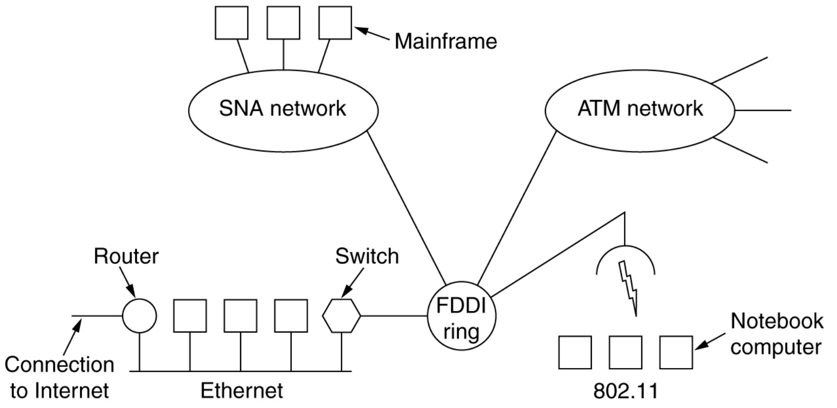
MPLS: MultiProtocol Label Switching

## Internetworking

- How Networks Differ
- How Networks Can Be Connected
- Concatenated Virtual Circuits
- Connectionless Internetworking
- Tunneling
- Internetwork Routing
- Fragmentation

## Connecting Networks

A collection of interconnected networks.



Many different networks exist and numerous protocols are in wide use.

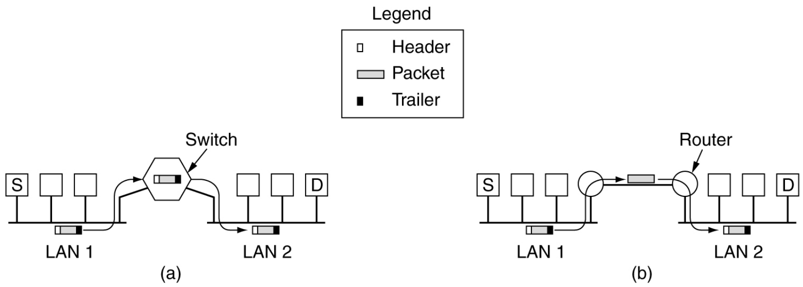
## How Networks Differ

| 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 |

## How Networks Can Be Connected

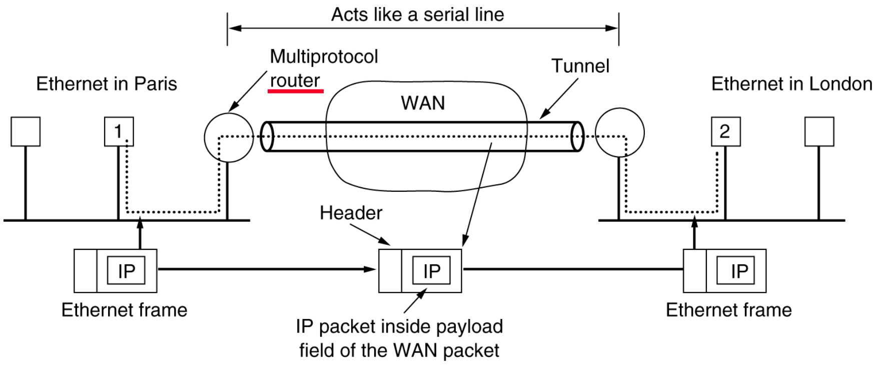
**(a) Two Ethernets connected  by a switch/bridge.**

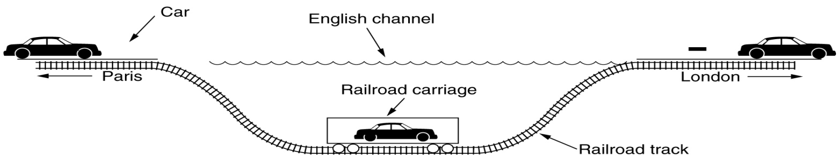
**(b) Two Ethernets connected by (multi-protocol) routers.**



*What is an essential difference between switched case and the routed case?*

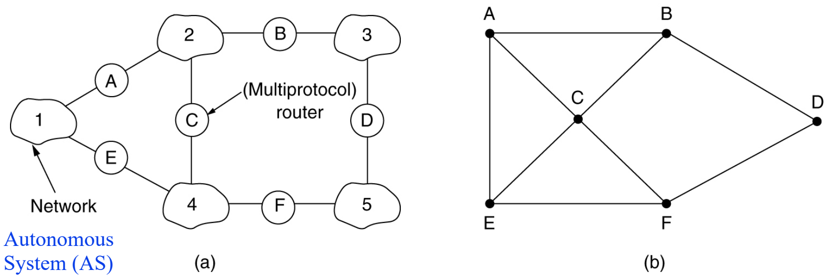
## Tunneling (with multi-protocol routers)





## Inter-network Routing (vs. Intra-network Routing)

**(a) An internetwork. (b) A graph of the internetwork.**



Autonomous System (AS)

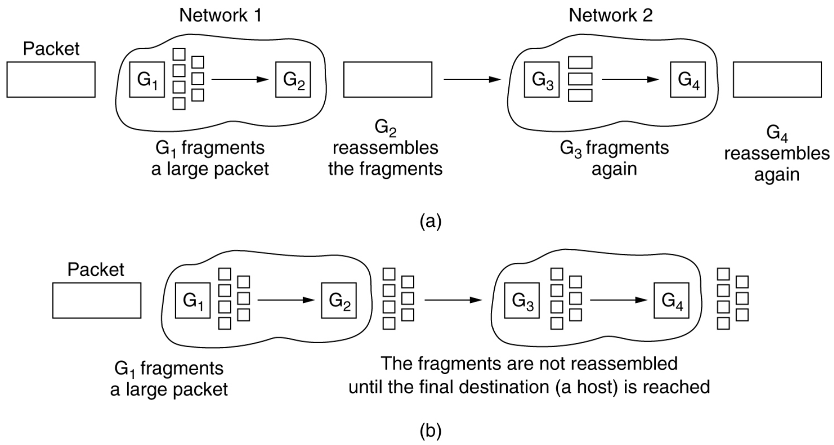
**Two-level routing:**

**Interior Gateway Routing Protocol (OSPF)**

**The Exterior Gateway Routing Protocol (BGP)**

## Fragmentation (Segmentation)

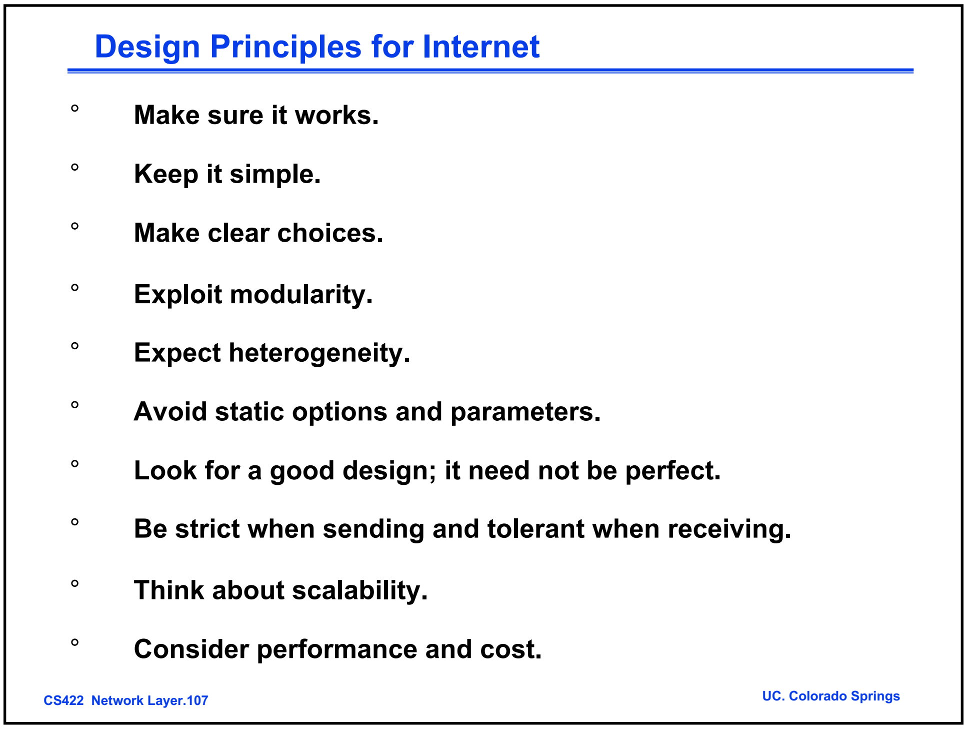
- Each network imposes some maximum size on its packets (ATM 46B, IP 64KB), making tunnelling/travelling a problem.


Network 1
Network 2
Packet
G1
G2
G3
G4
G1 fragments a large packet
G2 reassembles the fragments
G3 fragments again
G4 reassembles again
(a)
Packet
G1
G2
G3
G4
G1 fragments a large packet
The fragments are not reassembled until the final destination (a host) is reached
(b)

(a) Transparent fragmentation (ATM). (b) Nontransparent fragmentation (IP)

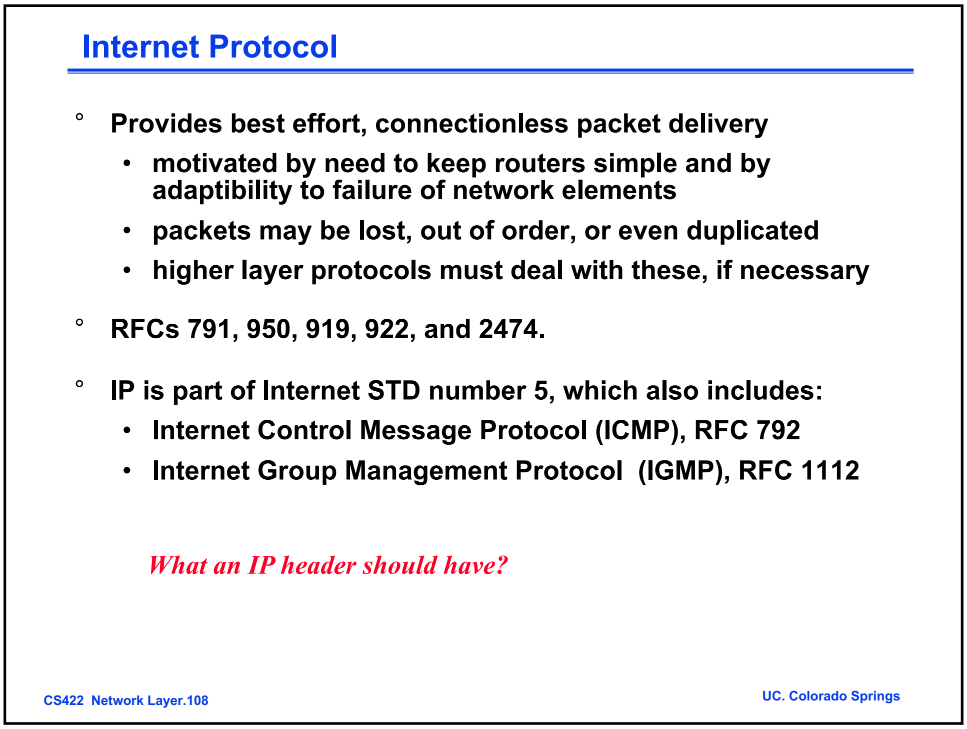
## The Network Layer (IP) in the Internet

- The IP Protocol
- IP Addresses
- Internet Control Protocols
- OSPF – The Interior Gateway Routing Protocol
- BGP – The Exterior Gateway Routing Protocol
- Internet Multicasting
- Mobile IP
- IPv6



## Design Principles for Internet

- Make sure it works.
- Keep it simple.
- Make clear choices.
- Exploit modularity.
- Expect heterogeneity.
- Avoid static options and parameters.
- Look for a good design; it need not be perfect.
- Be strict when sending and tolerant when receiving.
- Think about scalability.
- Consider performance and cost.



## Internet Protocol

- Provides best effort, connectionless packet delivery
  - motivated by need to keep routers simple and by adaptibility to failure of network elements
  - packets may be lost, out of order, or even duplicated
  - higher layer protocols must deal with these, if necessary
- RFCs 791, 950, 919, 922, and 2474.
- IP is part of Internet STD number 5, which also includes:
  - Internet Control Message Protocol (ICMP), RFC 792
  - Internet Group Management Protocol (IGMP), RFC 1112

*What an IP header should have?*

## IP Packet Header (v4)

0 4 8 16 19 24 31

<table>
<tr><td>Version</td><td>IHL</td><td>Type of Service</td><td colspan="3">Total Length</td></tr>
<tr><td colspan="3">Identification</td><td>Flags</td><td colspan="2">Fragment Offset</td></tr>
<tr><td colspan="2">Time to Live</td><td>Protocol</td><td colspan="3">Header Checksum</td></tr>
<tr><td colspan="6">Source IP Address</td></tr>
<tr><td colspan="6">Destination IP Address</td></tr>
<tr><td colspan="5">Options</td><td>Padding</td></tr>
</table>

- Minimum 20 bytes
- Up to 40 bytes in options fields

## IP Packet Header

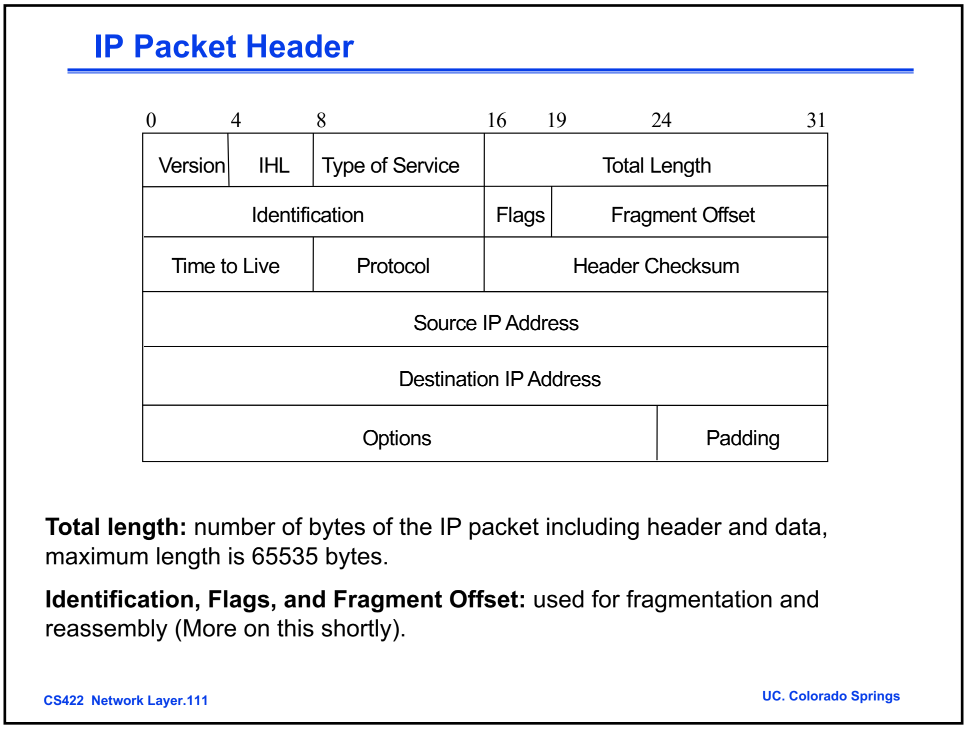
0 4 8 16 19 24 31

<table>
<tr><td>Version</td><td>IHL</td><td>Type of Service</td><td colspan="3">Total Length</td></tr>
<tr><td colspan="3">Identification</td><td>Flags</td><td colspan="2">Fragment Offset</td></tr>
<tr><td colspan="2">Time to Live</td><td>Protocol</td><td colspan="3">Header Checksum</td></tr>
<tr><td colspan="6">Source IP Address</td></tr>
<tr><td colspan="6">Destination IP Address</td></tr>
<tr><td colspan="5">Options</td><td>Padding</td></tr>
</table>

**Version:** current IP version is 4.

**Internet header length (IHL):** length of the header in 32-bit words.

**Type of service (TOS):** traditionally priority of packet at each router. Recent Differentiated Services redefines TOS field to include other services besides best effort.

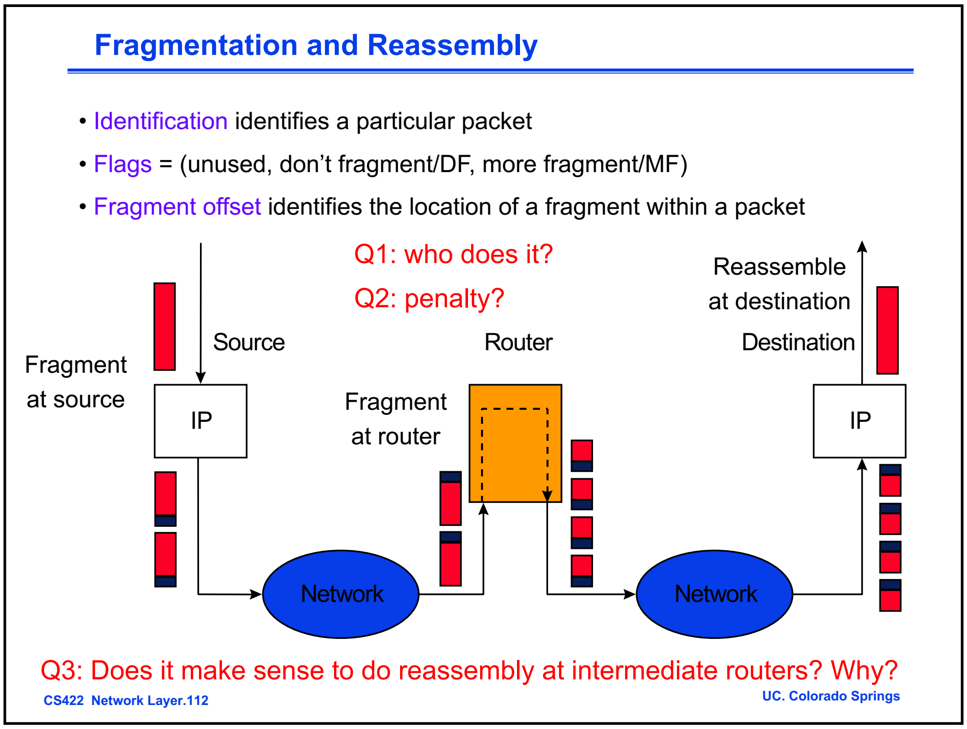


## IP Packet Header

<table>
<tr><th>0</th><th>4</th><th>8</th><th>16</th><th>19</th><th>24   31</th></tr>
<tr><td>Version</td><td>IHL</td><td>Type of Service</td><td colspan="3">Total Length</td></tr>
<tr><td colspan="3">Identification</td><td>Flags</td><td colspan="2">Fragment Offset</td></tr>
<tr><td colspan="2">Time to Live</td><td>Protocol</td><td colspan="3">Header Checksum</td></tr>
<tr><td colspan="6">Source IP Address</td></tr>
<tr><td colspan="6">Destination IP Address</td></tr>
<tr><td colspan="5">Options</td><td>Padding</td></tr>
</table>

**Total length:** number of bytes of the IP packet including header and data, maximum length is 65535 bytes.

**Identification, Flags, and Fragment Offset:** used for fragmentation and reassembly (More on this shortly).



## Fragmentation and Reassembly

- Identification identifies a particular packet
- Flags = (unused, don't fragment/DF, more fragment/MF)
- Fragment offset identifies the location of a fragment within a packet

Q1: who does it?

Q2: penalty?

Q3: Does it make sense to do reassembly at intermediate routers? Why?

## IP Packet Header

| 0 | 4 | 8 | 16 | 19 | 24 | 31 |
|---|---|---|---|---|---|---|
| Version | IHL | Type of Service | Total Length | | | |
| Identification | | | Flags | Fragment Offset | | |
| Time to Live | | Protocol | Header Checksum | | | |
| Source IP Address | | | | | | |
| Destination IP Address | | | | | | |
| Options | | | | | Padding | |

**Time to live (TTL):** number of hops packet is allowed to traverse in the network.

- Each router along the path to the destination decrements this value by one.
- If the value reaches zero before the packet reaches the destination, the router discards the packet and sends an error message back to the source.

**Why not use actual time in TTL?**

very large #;
more complex to track and update

## IP Packet Header

| 0 | 4 | 8 | 16 | 19 | 24 | 31 |
|---|---|---|---|---|---|---|
| Version | IHL | Type of Service | Total Length | | | |
| Identification | | | Flags | Fragment Offset | | |
| Time to Live | | Protocol | Header Checksum | | | |
| Source IP Address | | | | | | |
| Destination IP Address | | | | | | |
| Options | | | | | Padding | |

**Protocol:** specifies upper-layer protocol that is to receive IP data at the destination. Examples include TCP (protocol = 6), UDP (protocol = 17), and ICMP (protocol = 1).

**Header checksum:** verifies the integrity of the IP header.

**Source IP address** and **destination IP address:** contain the addresses of the source and destination hosts.

## IP Packet Header

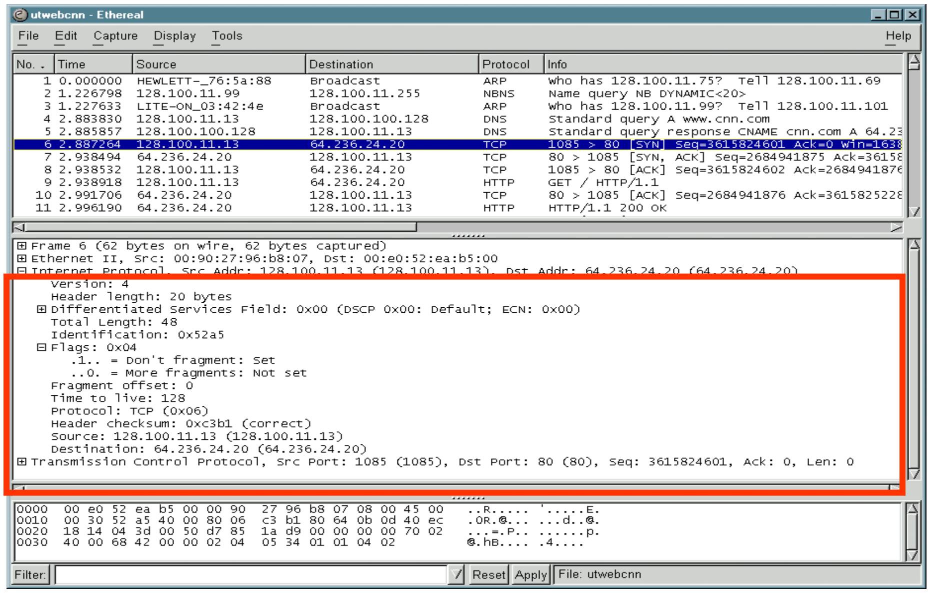
0 4 8 16 19 24 31

<table>
<tr><td>Version</td><td>IHL</td><td>Type of Service</td><td colspan="3">Total Length</td></tr>
<tr><td colspan="3">Identification</td><td>Flags</td><td colspan="2">Fragment Offset</td></tr>
<tr><td colspan="2">Time to Live</td><td>Protocol</td><td colspan="3">Header Checksum</td></tr>
<tr><td colspan="6">Source IP Address</td></tr>
<tr><td colspan="6">Destination IP Address</td></tr>
<tr><td colspan="5">Options</td><td>Padding</td></tr>
</table>

**Options:** Variable length field, allows packet to request special features such as security level, route to be taken by the packet, and timestamp at each router. Detailed descriptions of these options can be found in [RFC 791].

**Padding:** This field is used to make the header a multiple of 32-bit words.

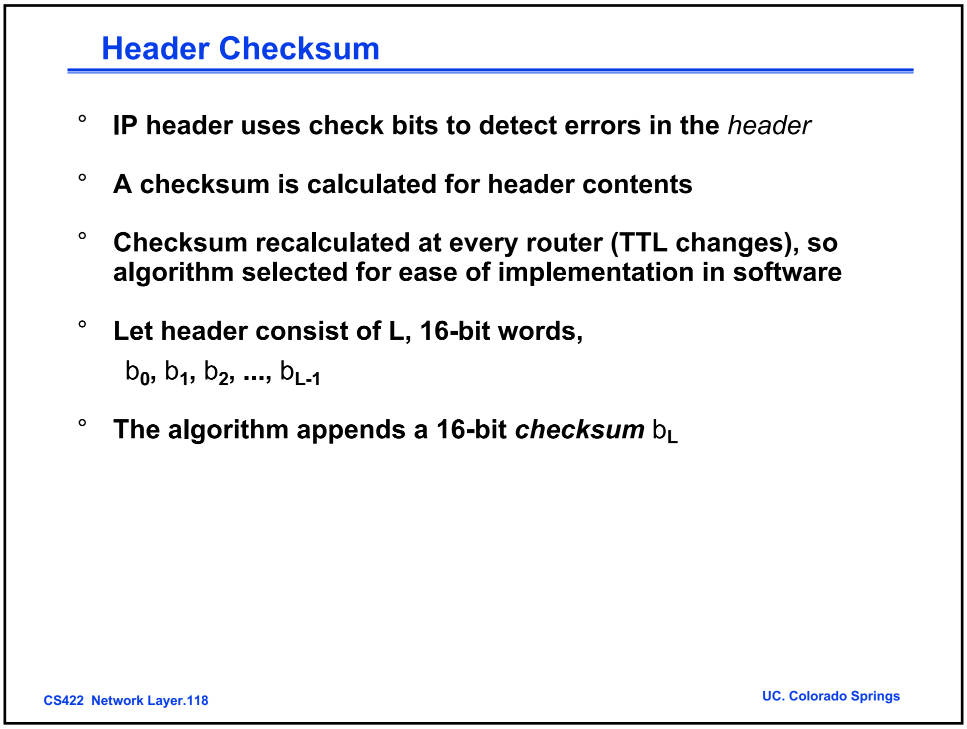
## Example of IP Header (Ethereal)



```
No. . Time       Source              Destination         Protocol  Info
    1 0.000000   HEWLETT-_76:5a:88   Broadcast           ARP       Who has 128.100.11.75?  Tell 128.100.11.69
    2 1.226798   128.100.11.99       128.100.11.255      NBNS      Name query NB DYNAMIC<20>
    3 1.227633   LITE-ON_03:42:4e    Broadcast           ARP       Who has 128.100.11.99?  Tell 128.100.11.101
    4 2.883830   128.100.11.13       128.100.100.128     DNS       Standard query A www.cnn.com
    5 2.885857   128.100.100.128     128.100.11.13       DNS       Standard query response CNAME cnn.com A 64.23
    6 2.887264   128.100.11.13       64.236.24.20        TCP       1085 > 80 [SYN] Seq=3615824601 Ack=0 Win=1638
    7 2.938494   64.236.24.20        128.100.11.13       TCP       80 > 1085 [SYN, ACK] Seq=2684941875 Ack=36158
    8 2.938532   128.100.11.13       64.236.24.20        TCP       1085 > 80 [ACK] Seq=3615824602 Ack=268494187
    9 2.938918   128.100.11.13       64.236.24.20        HTTP      GET / HTTP/1.1
   10 2.991706   64.236.24.20        128.100.11.13       TCP       80 > 1085 [ACK] Seq=2684941876 Ack=361582522
   11 2.996190   64.236.24.20        128.100.11.13       HTTP      HTTP/1.1 200 OK
```

```
⊞ Frame 6 (62 bytes on wire, 62 bytes captured)
⊞ Ethernet II, Src: 00:90:27:96:b8:07, Dst: 00:e0:52:ea:b5:00
⊟ Internet Protocol, Src Addr: 128.100.11.13 (128.100.11.13), Dst Addr: 64.236.24.20 (64.236.24.20)
     Version: 4
     Header length: 20 bytes
   ⊞ Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00)
     Total Length: 48
     Identification: 0x52a5
   ⊟ Flags: 0x04
       .1.. = Don't fragment: Set
       ..0. = More fragments: Not set
     Fragment offset: 0
     Time to live: 128
     Protocol: TCP (0x06)
     Header checksum: 0xc3b1 (correct)
     Source: 128.100.11.13 (128.100.11.13)
     Destination: 64.236.24.20 (64.236.24.20)
⊞ Transmission Control Protocol, Src Port: 1085 (1085), Dst Port: 80 (80), Seq: 3615824601, Ack: 0, Len: 0
```

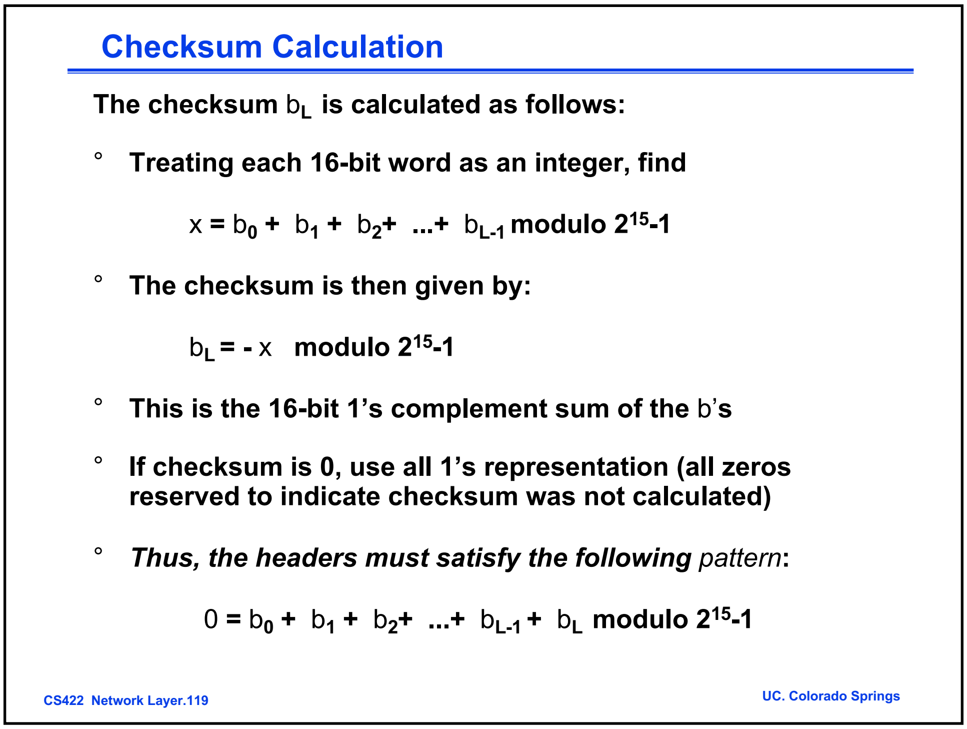
```
0000   00 e0 52 ea b5 00 00 90   27 96 b8 07 08 00 45 00    ..R..... '.....E.
0010   00 30 52 a5 40 00 80 06   c3 b1 80 64 0b 0d 40 ec    .0R.@... ...d..@.
0020   18 14 04 3d 00 50 d7 85   1a d9 00 00 00 00 70 02    ...=.P.. ......p.
0030   40 00 68 42 00 00 02 04   05 34 01 01 04 02          @.hB.... .4....
```



## Header Checksum

- IP header uses check bits to detect errors in the *header*
- A checksum is calculated for header contents
- Checksum recalculated at every router (TTL changes), so algorithm selected for ease of implementation in software
- Let header consist of L, 16-bit words,

  $b_0, b_1, b_2, \ldots, b_{L-1}$
- The algorithm appends a 16-bit ***checksum*** $b_L$



## Checksum Calculation

The checksum $b_L$ is calculated as follows:

- Treating each 16-bit word as an integer, find

$$x = b_0 + b_1 + b_2 + \ldots + b_{L-1} \text{ modulo } 2^{15}-1$$

- The checksum is then given by:

$$b_L = -x \text{ modulo } 2^{15}-1$$

- This is the 16-bit 1's complement sum of the b's
- If checksum is 0, use all 1's representation (all zeros reserved to indicate checksum was not calculated)
- ***Thus, the headers must satisfy the following*** *pattern*:

$$0 = b_0 + b_1 + b_2 + \ldots + b_{L-1} + b_L \text{ modulo } 2^{15}-1$$

## Internet Checksum Example

**Use Modulo Arithmetic**

- **Assume 4-bit words**
- **Use mod $2^4$-1 arithmetic**
- **$\underline{b}_0$=1100 = 12**
- **$\underline{b}_1$=1010 = 10**
- **$\underline{b}_0$+$\underline{b}_1$=12+10=7 mod15**
- **$\underline{b}_2$ = -7 = 8 mod15**
- **Therefore**
- **$\underline{b}_2$=1000**

**Use Binary Arithmetic**

- **Note 16 mod15 =1**
- **So: 10000 mod15 = 0001**
- **leading bit wraps around**

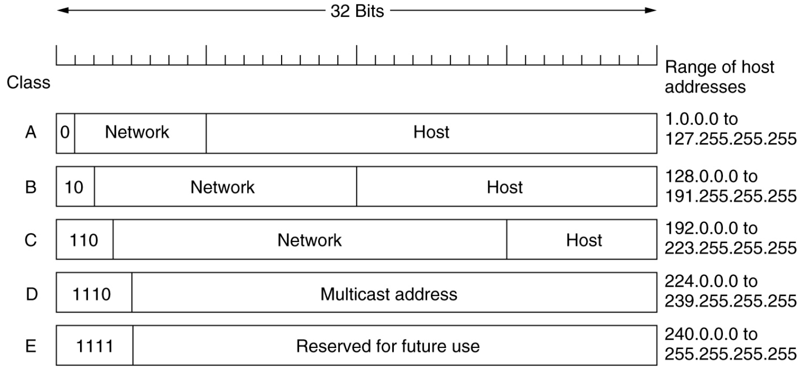
$$\begin{aligned} b_0 + b_1 &= 1100+1010 \\ &=10110 \\ &=10000+0110 \\ &=0001+0110 \\ &=0111 \\ &=7 \end{aligned}$$

*Take 1s complement*

$b_2$ = -0111 =1000

## IP Addresses

**IP address formats based on classful addressing (ICANN).**



| Class | | | | Range of host addresses |
|---|---|---|---|---|
| A | 0 | Network | Host | 1.0.0.0 to<br>127.255.255.255 |
| B | 10 | Network | Host | 128.0.0.0 to<br>191.255.255.255 |
| C | 110 | Network | Host | 192.0.0.0 to<br>223.255.255.255 |
| D | 1110 | Multicast address | | 224.0.0.0 to<br>239.255.255.255 |
| E | 1111 | Reserved for future use | | 240.0.0.0 to<br>255.255.255.255 |

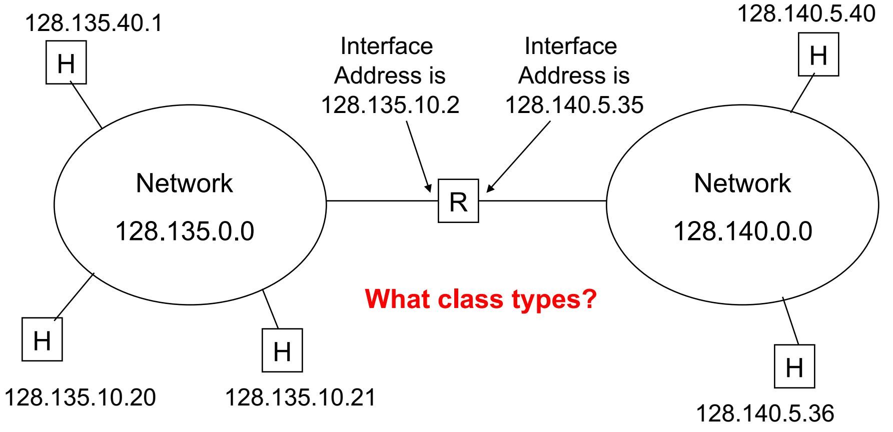
- **Dotted decimal notation**
  - **C0290614 -> 192.41.6.20**

## IP Addresses (2)

**Special IP addresses.**

| | | |
|---|---|---|
| 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 | | This host |
| 0 0 . . . 0 0 | Host | A host on this network |
| 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 | | Broadcast on the local network |
| Network | 1 1 1 1 . . . 1 1 1 1 | Broadcast on a distant network |
| 127 | (Anything) | Loopback |

## Example of IP Addressing



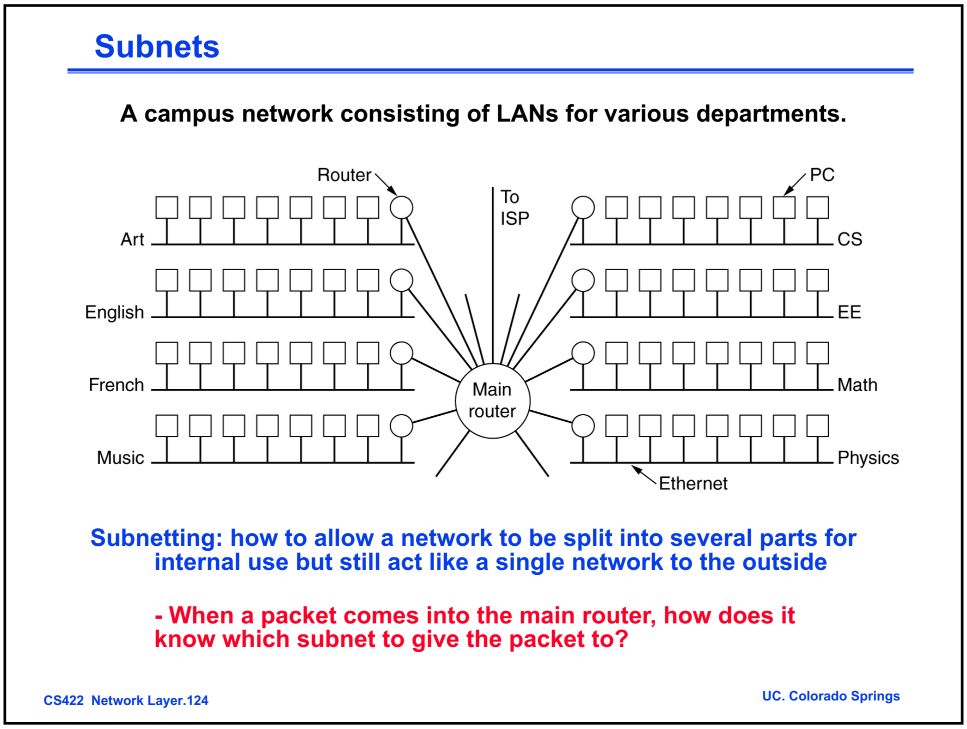
What class types?

Address with host ID=all 0s refers to the network

Address with host ID=all 1s refers to a broadcast packet

R = router

H = host

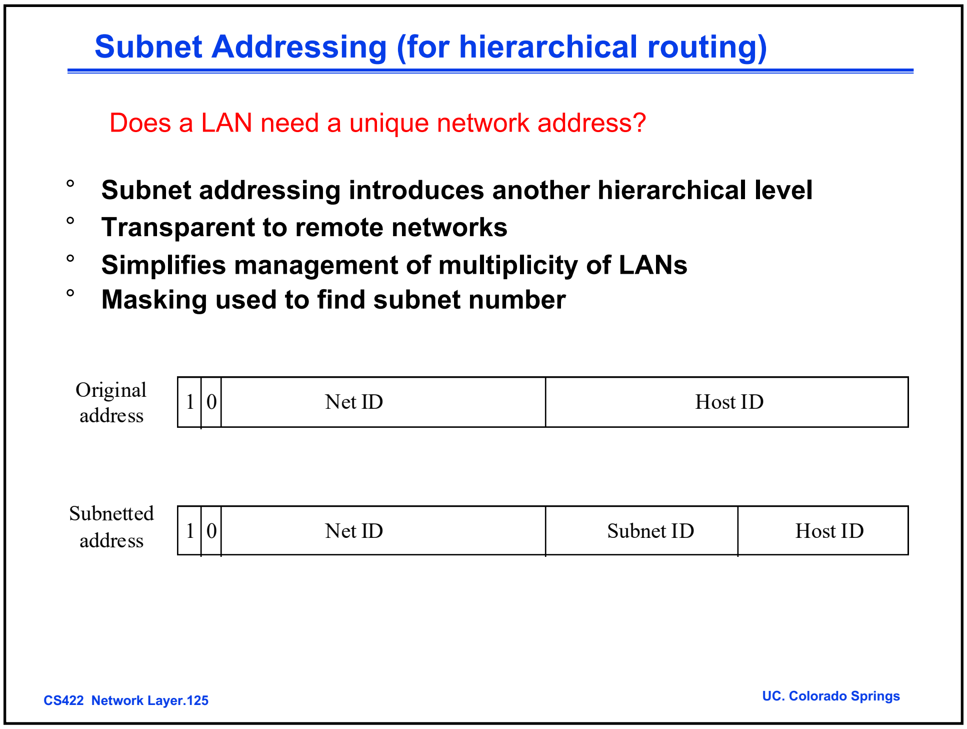


## Subnets

A campus network consisting of LANs for various departments.

Subnetting: how to allow a network to be split into several parts for internal use but still act like a single network to the outside

- When a packet comes into the main router, how does it know which subnet to give the packet to?



## Subnet Addressing (for hierarchical routing)

Does a LAN need a unique network address?

- Subnet addressing introduces another hierarchical level
- Transparent to remote networks
- Simplifies management of multiplicity of LANs
- Masking used to find subnet number

| | | | | |
|---|---|---|---|---|
| Original address | 1 | 0 | Net ID | Host ID |

| | | | | | |
|---|---|---|---|---|---|
| Subnetted address | 1 | 0 | Net ID | Subnet ID | Host ID |

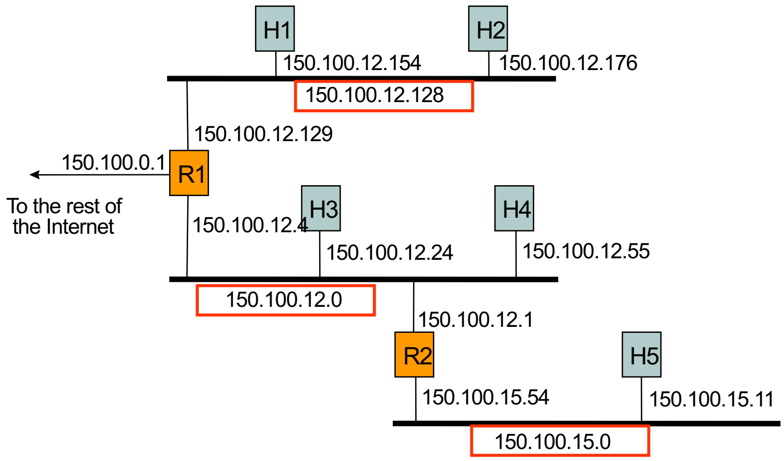
## Subnetting Example 1

- Organization has Class B address (16 host ID bits) with network ID: 150.100.0.0
- Create subnets with up to 250 hosts each
  - 8 bits sufficient for each subnet
  - 16 – 8 = 8 bits for subnet ID
- Apply subnet mask to IP addresses to find corresponding subnet
  - Example:  Find subnet for 150.100.12.176
  - IP add = 10010110 01100100 00001100 10110000
  - Mask   = 11111111 11111111 11111111 00000000
  - AND    = 10010110 01100100 00001100 00000000
  - Subnet = 150.100.12.0
  - Subnet address used by routers within organization

## Subnetting Example 2

- Organization has Class B address (16 host ID bits) with network ID: 150.100.0.0
- Create subnets with up to 100 hosts each
  - 7 bits sufficient for each subnet
  - 16 – 7 = 9 bits for subnet ID
- Apply subnet mask to IP addresses to find corresponding subnet
  - Example:  Find subnet for 150.100.12.176
  - IP add = 10010110 01100100 00001100 10110000
  - Mask   = 11111111 11111111 11111111 10000000
  - AND    = 10010110 01100100 00001100 10000000
  - Subnet = 150.100.12.128
  - Subnet address used by routers within organization

## Subnet Example



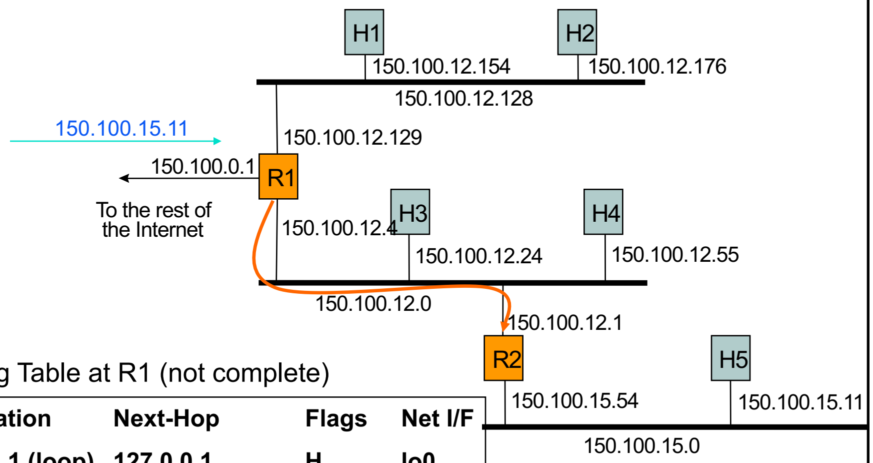
## Routing with Sub-networks

- IP layer in hosts and routers maintain a routing table
- Originating host: To send an IP packet, consult routing table
  - If destination host is in same network, send packet *directly* using appropriate network interface
  - Otherwise, send packet indirectly; typically, routing table indicates a default router
- Router: Examine IP destination address in arriving packet
  - If dest IP address not own, router consults routing table to determine next-hop and associated network interface & forwards packet

## Routing Table

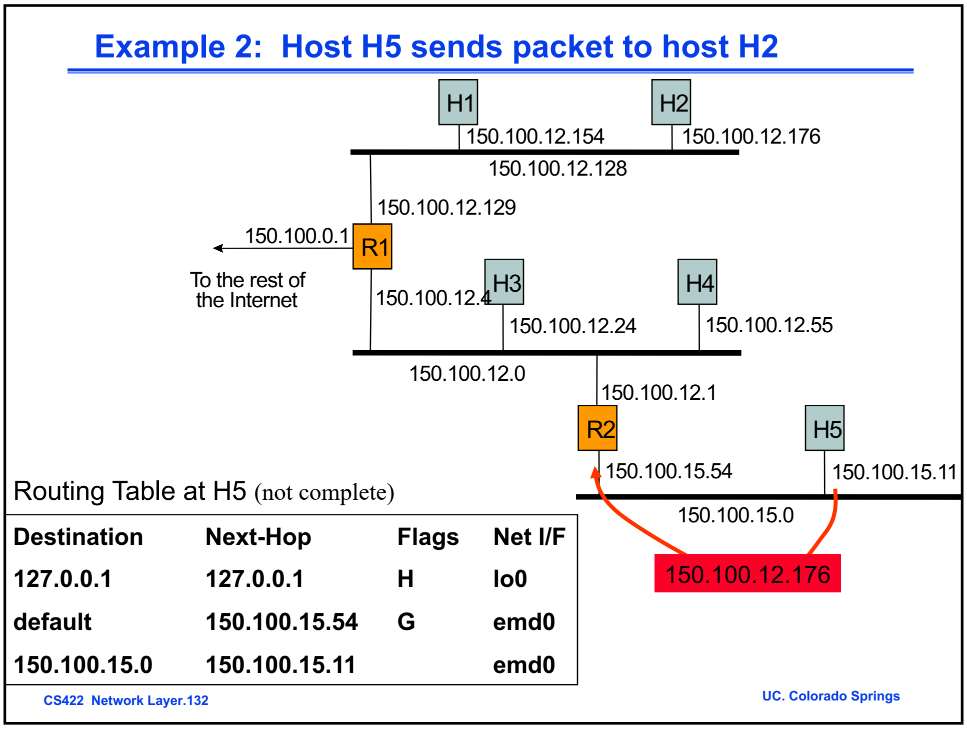
- Each row in routing table contains:
  - Destination IP address
  - IP address of next-hop router
  - Physical address
  - Statistics information
  - Flags
    - H=1 (0) indicates route is to a host (network)
    - G=1 (0) indicates route is to a router (directly connected destination)
- Routing table search order & action
  - Complete destination address; send as per next-hop & G flag
  - Destination network ID; send as per next-hop & G flag
  - Default router entry; send as per next-hop
  - Declare packet undeliverable; send ICMP “host unreachable error” packet to originating host

## Example 1: A packet with 150.100.15.11 arrives at R1



Routing Table at R1 (not complete)

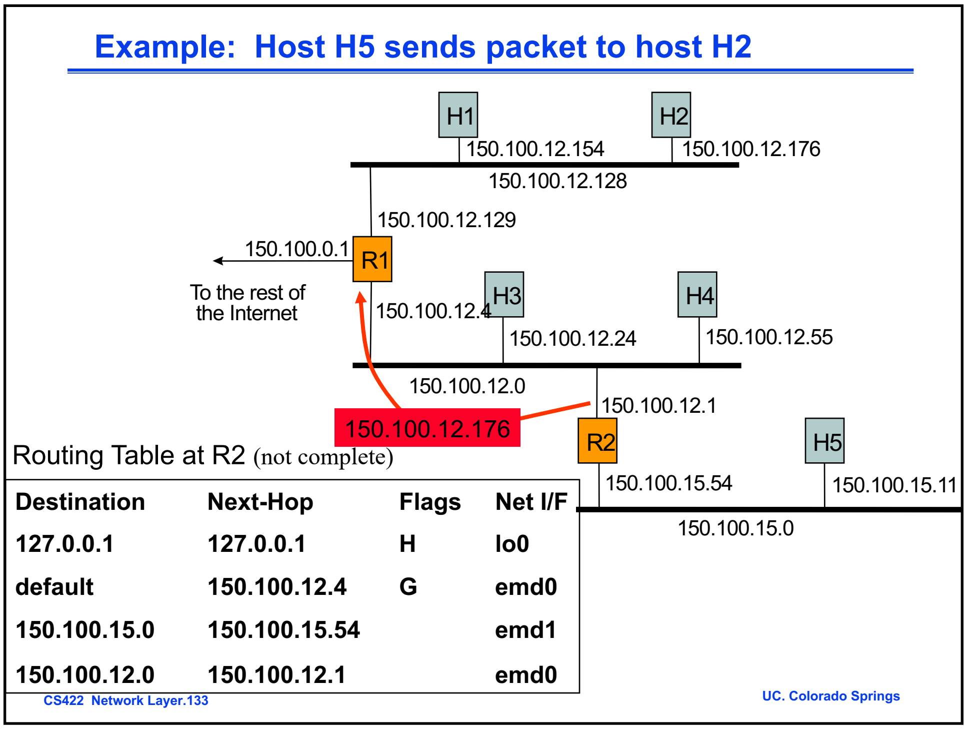
| Destination | Next-Hop | Flags | Net I/F |
|---|---|---|---|
| 127.0.0.1 (loop) | 127.0.0.1 | H | lo0 |
| 150.100.12.176 | 150.100.12.176 |  | emd0 |
| 150.100.12.0 | 150.100.12.4 |  | emd1 |
| 150.100.15.0 | 150.100.12.1 | G | emd1 |



## Example 2: Host H5 sends packet to host H2

Routing Table at H5 (not complete)

| Destination | Next-Hop | Flags | Net I/F |
|---|---|---|---|
| 127.0.0.1 | 127.0.0.1 | H | lo0 |
| default | 150.100.15.54 | G | emd0 |
| 150.100.15.0 | 150.100.15.11 | | emd0 |

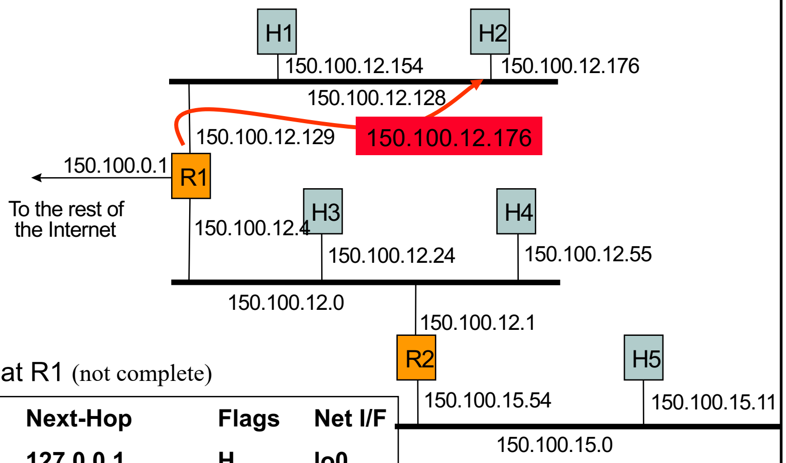


## Example: Host H5 sends packet to host H2

Routing Table at R2 (not complete)

| Destination | Next-Hop | Flags | Net I/F |
|---|---|---|---|
| 127.0.0.1 | 127.0.0.1 | H | lo0 |
| default | 150.100.12.4 | G | emd0 |
| 150.100.15.0 | 150.100.15.54 | | emd1 |
| 150.100.12.0 | 150.100.12.1 | | emd0 |

## Example:  Host H5 sends packet to host H2

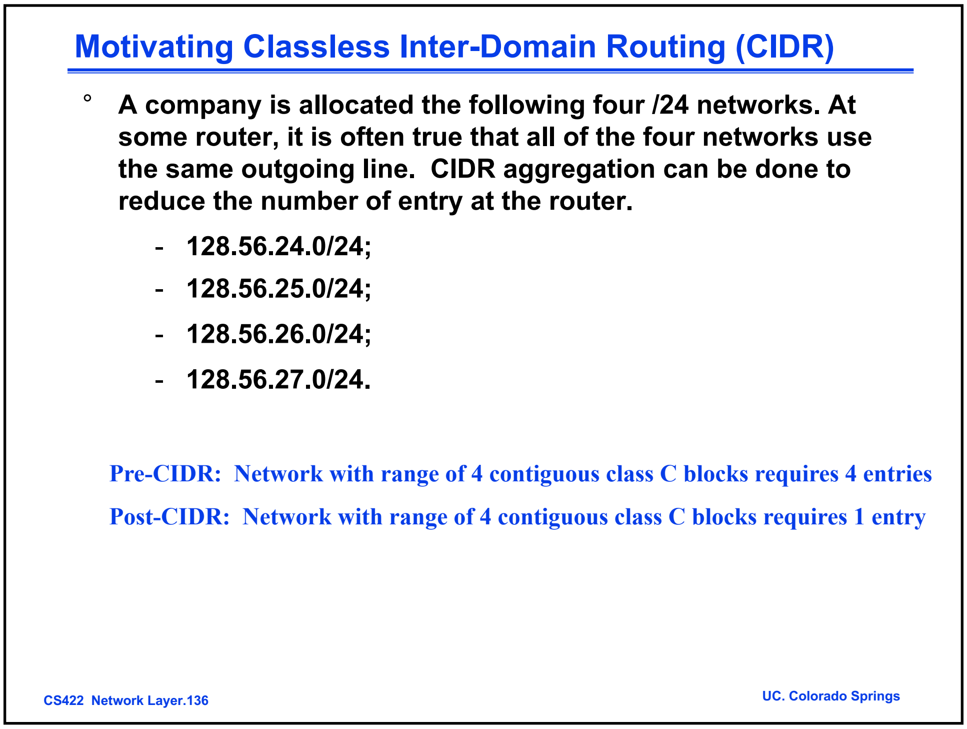


Routing Table at R1 (not complete)

| Destination | Next-Hop | Flags | Net I/F |
|---|---|---|---|
| 127.0.0.1 | 127.0.0.1 | H | lo0 |
| 150.100.12.176 | 150.100.12.176 | | emd0 |
| 150.100.12.0 | 150.100.12.4 | | emd1 |
| 150.100.15.0 | 150.100.12.1 | G | emd1 |

## IP Address Problems

- In the 1990, two problems became apparent
  - IP addresses were being exhausted
  - IP routing tables were growing very large
- IP Address Exhaustion
  - Class A, B, and C address structure inefficient
    - Class B too large for most organizations
    - Class C too small
    - Rate of class B allocation implied exhaustion by 1994
- IP routing table size
  - Growth in number of networks in Internet reflected in # of table entries
    - From 1991 to 1995, routing tables doubled in size every 10 months
    - Stress on router processing power and memory allocation
- Short-term solution:
- Classless Inter-domain Routing (CIDR), RFC 1518
- New allocation policy (RFC 2050)
- Private IP Addresses set aside for intranets (NAT)
- Long-term solution:   IPv6 with much bigger address space



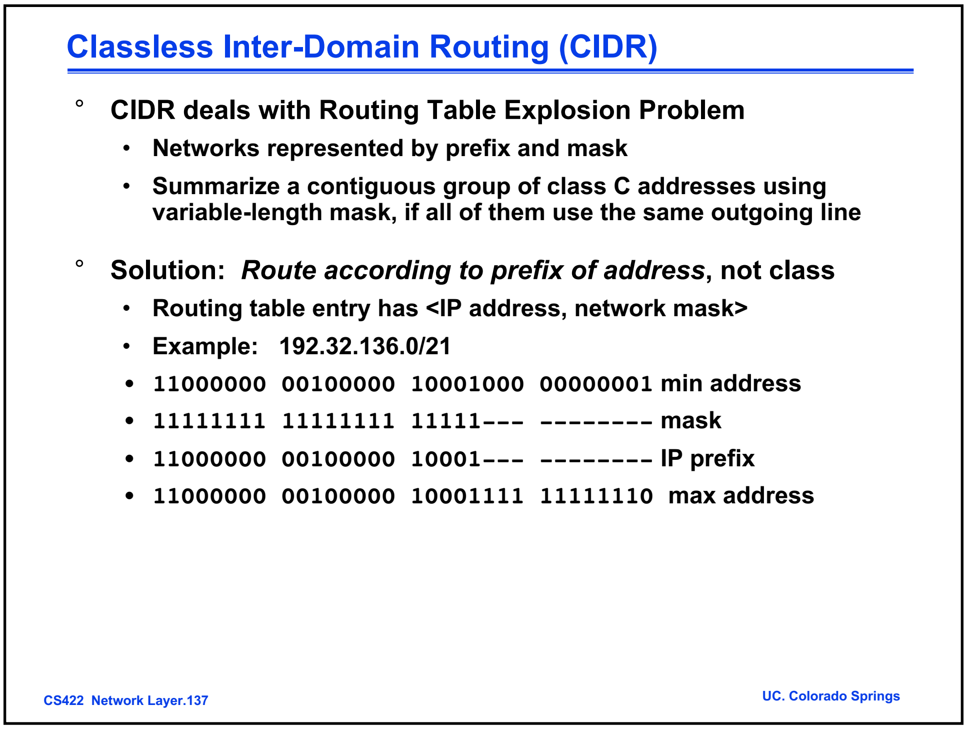
## Motivating Classless Inter-Domain Routing (CIDR)

° A company is allocated the following four /24 networks. At some router, it is often true that all of the four networks use the same outgoing line. CIDR aggregation can be done to reduce the number of entry at the router.

- 128.56.24.0/24;
- 128.56.25.0/24;
- 128.56.26.0/24;
- 128.56.27.0/24.

Pre-CIDR: Network with range of 4 contiguous class C blocks requires 4 entries

Post-CIDR: Network with range of 4 contiguous class C blocks requires 1 entry



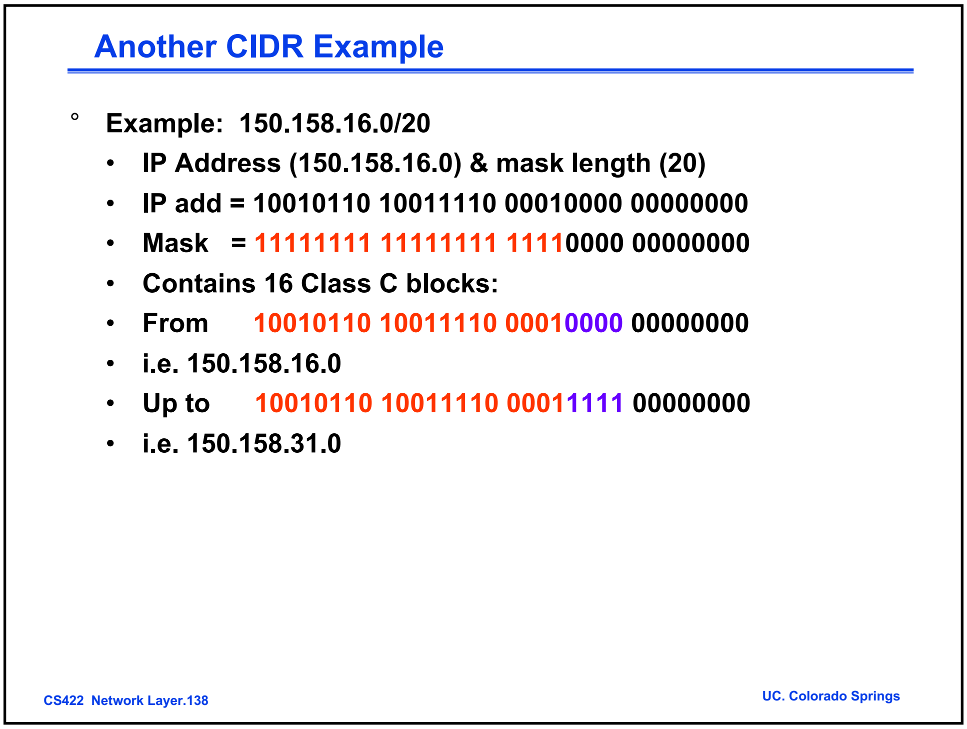
## Classless Inter-Domain Routing (CIDR)

° CIDR deals with Routing Table Explosion Problem

- Networks represented by prefix and mask
- Summarize a contiguous group of class C addresses using variable-length mask, if all of them use the same outgoing line

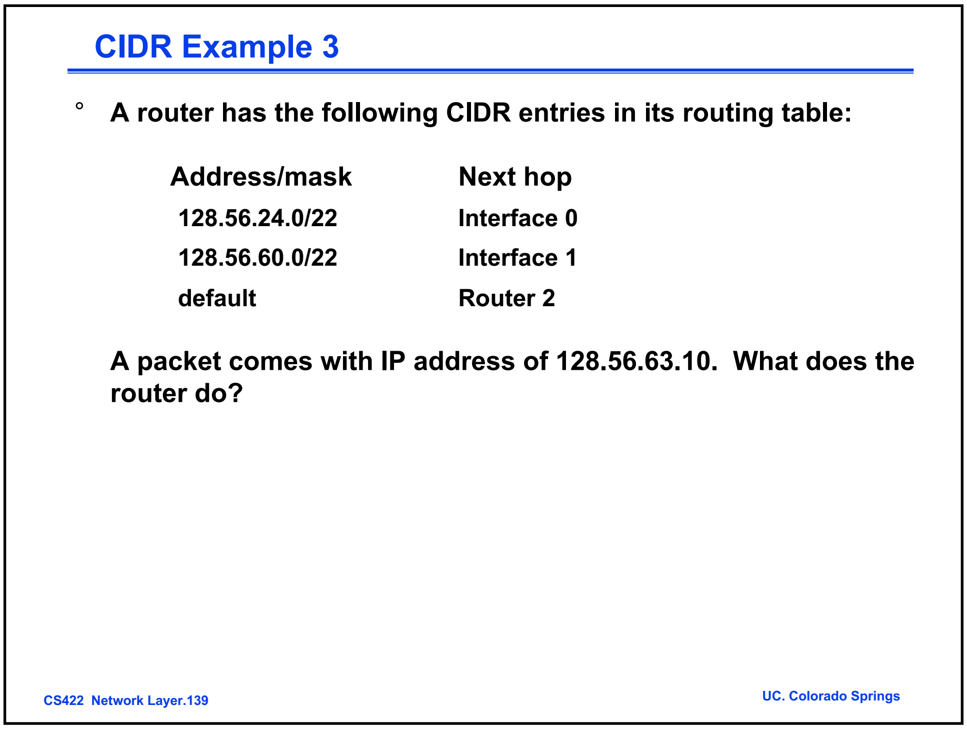
° Solution: *Route according to prefix of address*, not class

- Routing table entry has <IP address, network mask>
- Example: 192.32.136.0/21
- 11000000 00100000 10001000 00000001 min address
- 11111111 11111111 11111--- -------- mask
- 11000000 00100000 10001--- -------- IP prefix
- 11000000 00100000 10001111 11111110 max address



## Another CIDR Example

- Example:  150.158.16.0/20
  - IP Address (150.158.16.0) & mask length (20)
  - IP add = 10010110 10011110 00010000 00000000
  - Mask   = 11111111 11111111 11110000 00000000
  - Contains 16 Class C blocks:
  - From      10010110 10011110 00010000 00000000
  - i.e. 150.158.16.0
  - Up to      10010110 10011110 00011111 00000000
  - i.e. 150.158.31.0



## CIDR Example 3

- A router has the following CIDR entries in its routing table:

| Address/mask | Next hop |
|---|---|
| 128.56.24.0/22 | Interface 0 |
| 128.56.60.0/22 | Interface 1 |
| default | Router 2 |

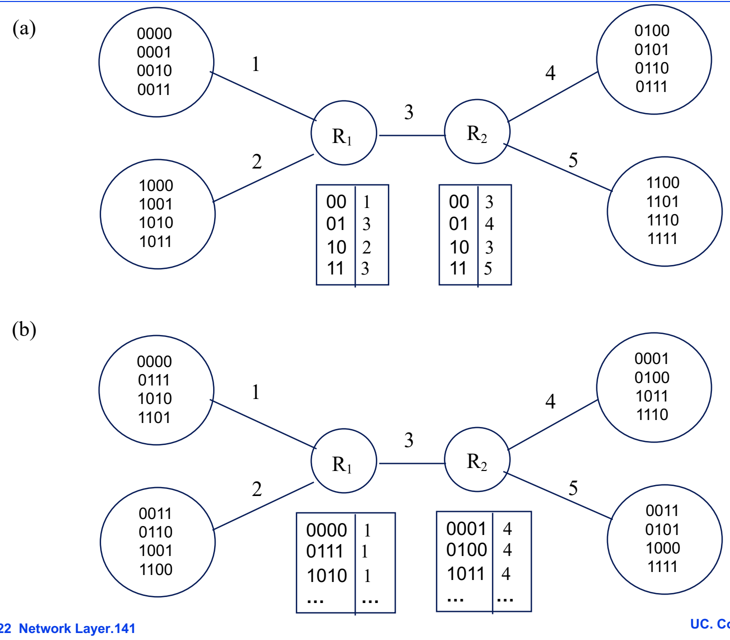
A packet comes with IP address of 128.56.63.10.  What does the router do?

## New Address Allocation Policy

- Class A & B assigned only for clearly demonstrated need
- *Consecutive* blocks of class C assigned (up to 64 blocks)
  - All IP addresses in the range have a common prefix, and every address with that prefix is within the range
  - Arbitrary prefix length for network ID improves efficiency
- Lower half of class C space assigned to regional authorities
  - More hierarchical allocation of addresses
  - Service provider to customer

| Address Requirement | Address Allocation |
|---|---|
| < 256 | 1 Class C |
| 256<,<512 | 2 Class C |
| 512<,<1024 | 4 Class C |
| 1024<,<2048 | 8 Class C |
| 2048<,<4096 | 16 Class C |
| 4096<,<8192 | 32 Class C |
| 8192<,<16384 | 64 Class C |

## Recap: Hierarchical Routing & Table Efficiency



(a)

| | |
|---|---|
| 00 | 1 |
| 01 | 3 |
| 10 | 2 |
| 11 | 3 |

| | |
|---|---|
| 00 | 3 |
| 01 | 4 |
| 10 | 3 |
| 11 | 5 |

(b)

| | |
|---|---|
| 0000 | 1 |
| 0111 | 1 |
| 1010 | 1 |
| … | … |

| | |
|---|---|
| 0001 | 4 |
| 0100 | 4 |
| 1011 | 4 |
| … | … |

## CIDR Allocation Principles (RFC 1518-1520)

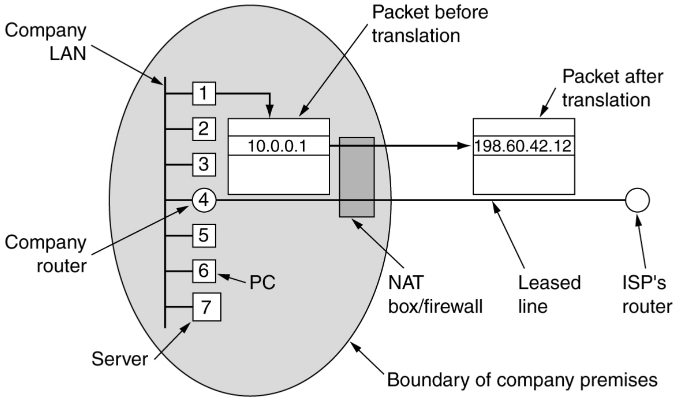
- IP address assignment reflects physical topology of network
- Network topology follows continental/national boundaries
  - IP addresses should be assigned on this basis
- Transit routing domains (TRDs) have unique IP prefix
  - carry traffic between routing domains
  - interconnected non-hierarchically, cross national boundaries
  - Most routing domains single-homed:  attached to a single TRD
  - Such domains assigned addresses with TRD's IP prefix
  - All of the addresses attached to a TRD aggregated into 1 table entry
- Implementation primarily through BGPv4 (RFC 1520)

## Longest Prefix Match

- CIDR impacts routing & forwarding
- Routing tables and protocols must carry IP address and mask
- Multiple entries may match a given IP destination address
- Example: perform CIDR on the following three /24 IP addresses (but 128.56.24.0/24 to a different port)
  - 128.56.25.0/24;
  - 128.56.26.0/24;
  - 128.56.27.0/24;
  - What if a packet with dest. IP address 128.56.24.0 comes?
- Packet must be routed using the *more specific route*, that is, the longest prefix match
- Several fast longest-prefix matching algorithms are available

## NAT – Network Address Translation

- What is the problem of the on-the-fly IP address assignment?
- NAT: pubic IP addresses and private IP addresses



Placement and operation of a NAT box (supporting class-B size).

How to translate when the reply comes back? What are its problems?

## Private IP Addresses

- Specific ranges of IP addresses set aside for use in private networks (RFC 1918)
- Use restricted to private internets; routers in public Internet discard packets with these addresses
- Range 1: 10.0.0.0 to 10.255.255.255
- Range 2: 172.16.0.0 to 172.31.255.255
- Range 3: 192.168.0.0 to 192.168.255.255
- Network Address Translation (NAT) used to convert between private & global IP addresses
  - Able to support about 64K interval distinct IP addresses

## Internet Control Message Protocol

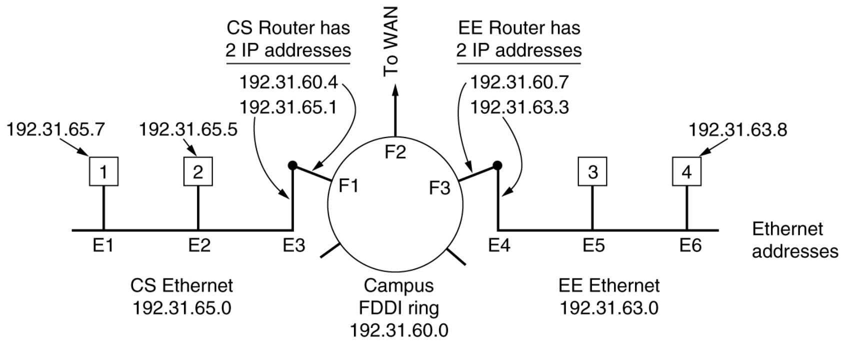
- ICMP reports unexpected operations and test Internet

| 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 request | Ask a machine if it is alive |
| Echo reply | Yes, I am alive |
| Timestamp request | Same as Echo request, but with timestamp |
| Timestamp reply | Same as Echo reply, but with timestamp |

**The principal ICMP message types.**

## ARP– The Address Resolution Protocol

- **How to map IP addresses to data link layer addresses since data link layer hardware does not understand IP addresses?**
- **Simplicity: a configuration file -> ARP using broadcast**



**Three interconnected /24 networks: two Ethernets and an FDDI ring.**

**How to make ARP work more efficiently? Caching**

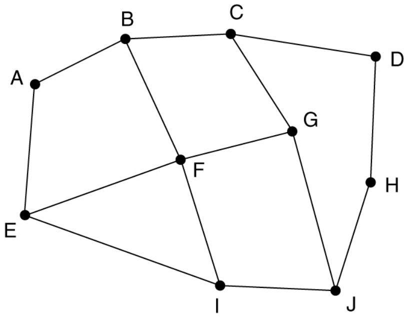
## OSPF – The Interior Gateway Routing Protocol

- How to route packets within an AS (autonomous system)?
  - RIP -> link state routing > OSPF (open shortest path first)
- What are important requirements for OSPF?
  - Openness
  - Variety of distance metrics
  - Dynamic
  - TOS support
  - Load balancing
  - Support hierarchical routing
  - security

## BGP – The Exterior Gateway Routing Protocol

What OSPF concerns most? Efficiency!

What OSPF does not care but BGP does? Politics?



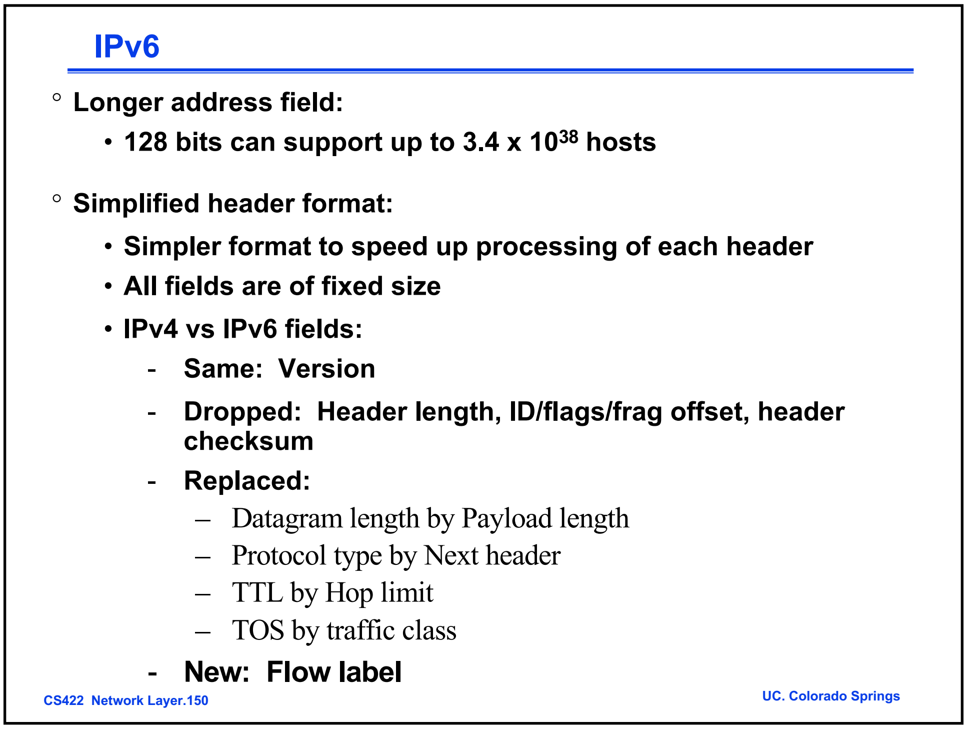
(a)

Information F receives
from its neighbors about D

From B: "I use BCD"
From G: "I use GCD"
From I: "I use IFGCD"
From E: "I use EFGCD"

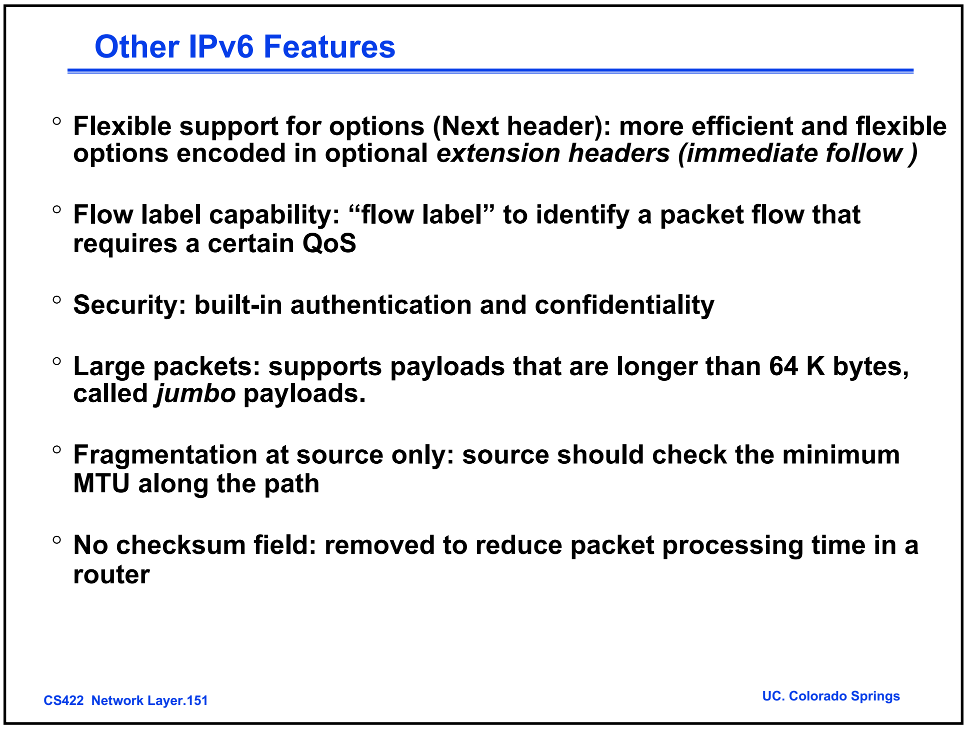
(b)

**(a) A set of BGP routers. (b) Information sent to F.**



## IPv6

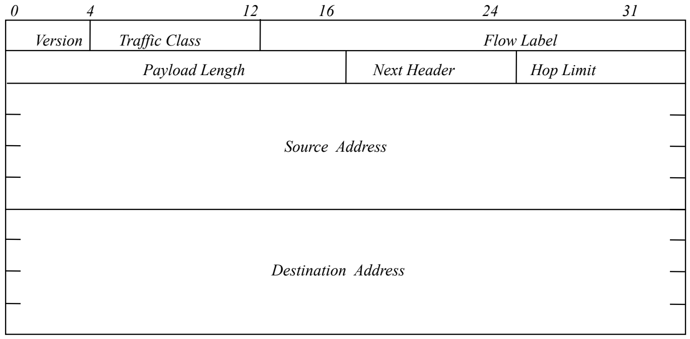
- Longer address field:
  - 128 bits can support up to 3.4 x $10^{38}$ hosts
- Simplified header format:
  - Simpler format to speed up processing of each header
  - All fields are of fixed size
  - IPv4 vs IPv6 fields:
    - Same: Version
    - Dropped: Header length, ID/flags/frag offset, header checksum
    - Replaced:
      - Datagram length by Payload length
      - Protocol type by Next header
      - TTL by Hop limit
      - TOS by traffic class
    - New: Flow label



## Other IPv6 Features

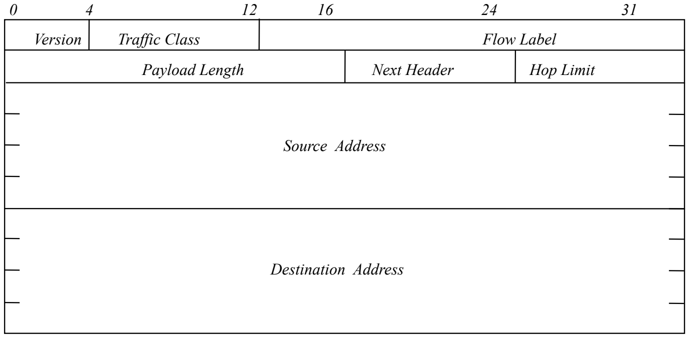
- Flexible support for options (Next header): more efficient and flexible options encoded in optional *extension headers (immediate follow )*
- Flow label capability: “flow label” to identify a packet flow that requires a certain QoS
- Security: built-in authentication and confidentiality
- Large packets: supports payloads that are longer than 64 K bytes, called *jumbo* payloads.
- Fragmentation at source only: source should check the minimum MTU along the path
- No checksum field: removed to reduce packet processing time in a router

## IPv6 Header Format



- Version field same size, same location
- Traffic class to support differentiated services
- Flow: sequence of packets from particular source to particular destination for which source requires special handling

## IPv6 Basic Header Format



- Payload length: length of data excluding header, up to 65535 B
- Next header: type of extension header that follows basic header
- Hop limit: # hops packet can travel before being dropped by a router

*Why fragmentation at source only?*

*Relieving load at routers*

## Extension Headers

- Allows an arbitrary number of extension headers be placed between the basic header and the payload (the extension headers are chained by the next header field)
- Large Packet: payload>64K (extension header)

| 0 | 8 | 16 | 24 31 |
|---|---|---|---|
| Next header | 0 | 194 | Opt len = 4 |
| Jumbo payload length | | | |

- Fragmentation: at source only (extension header)
  - Source performs "path MTU discovery" (a fragment extension header for each packet fragment)

| 0 | 8 | 16 | 29 | 31 |
|---|---|---|---|---|
| Next header | Reserved | Fragment offset | Res | M |
| Identification | | | | |

## Extension Headers

- IPv6 supports Source Routing

| 0 | 8 | 16 | 24 31 |
|---|---|---|---|
| Next header | Header length | Routing type = 0 | Segment left |
| Reserved | Strict/loose bit mask | | |
| Address 1 | | | |
| Address 2 | | | |
| • • • | | | |
| Address 24 | | | |

## IPv6 Addressing

° Address Categories

- Unicast: single network interface
- Multicast: group of network interfaces, typically at different locations. Packet sent to all.
- Anycast: group of network interfaces. Packet sent to only one interface in group, e.g. nearest.

° Hexadecimal notation

- Groups of 16 bits represented by 4 hex digits
- Separated by colons
  - 4BF5:AA12:0216:FEBC:BA5F:039A:BE9A:2176
- Shortened forms:
  - 4BF5:0000:0000:0000:BA5F:039A:000A:2176
  - To 4BF5:0:0:0:BA5F:39A:A:2176
  - To 4BF5::BA5F:39A:A:2176
- Mixed notation:
  - ::FFFF:128.155.12.198

## Migration from IPv4 to IPv6

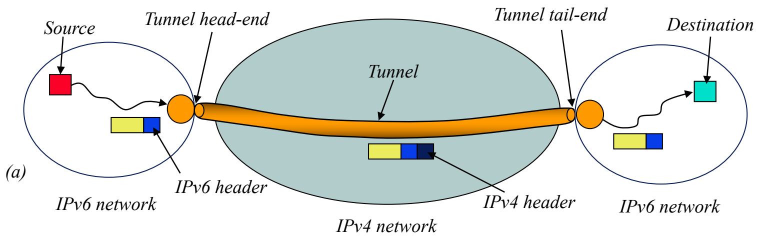
° Gradual transition from IPv4 to IPv6

° Dual IP stacks: routers run IPv4 & IPv6

- Type field used to direct packet to IP version

° IPv6 islands can tunnel across IPv4 networks

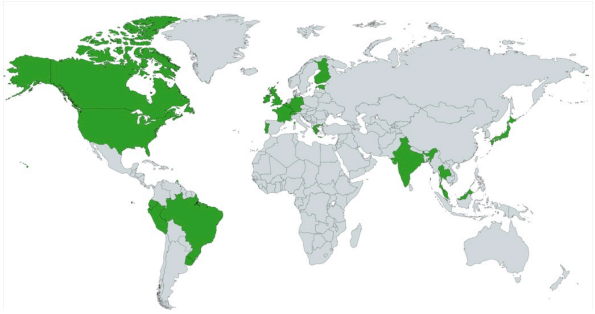
- Encapsulate user packet insider IPv4 packet



## Migration from IPv4 to IPv6 (Cont.)

° Transition is slow
- technical and cost challenges
- DHCP, NAT, Subnetting, and CIDR prolong IPv4 life

° 24 countries IPv6 volume over 15% (June 2018, Internet Society)



## Reading and Homework

° Chapter 5 of the textbook

° Homework Assignment
- See website