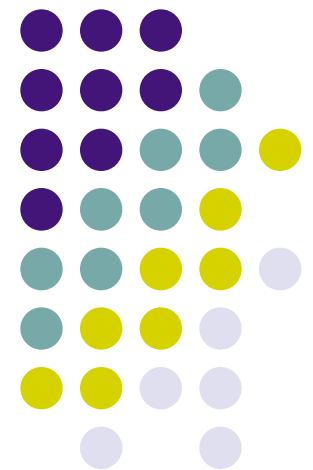


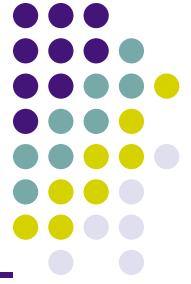
# Computer Networks

## CS 552

Badri Nath  
Rutgers University  
[badri@cs.rutgers.edu](mailto:badri@cs.rutgers.edu)

1. Link Layer, Multiple access, Bridges, Switching
2. IP addressing, CIDR, NAT
3. IP routing, OSPF (link state), RIP(DV), Issues

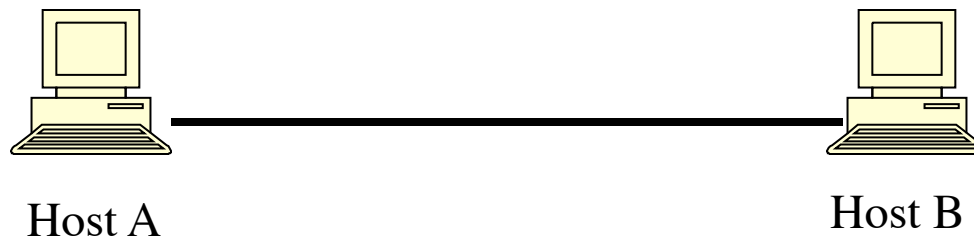




# Communication over a link

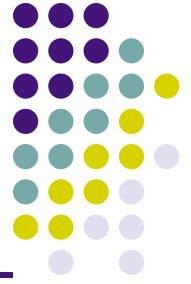
---

- Information needs to be converted to a signal appropriate to the link
  - Wired, optics, wireless
  - 1s and 0s to appropriate signal levels
  - Bits per second = baud  $\times \log_2(\text{levels})$



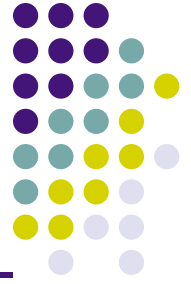
# Data Link Layer Functionality

---

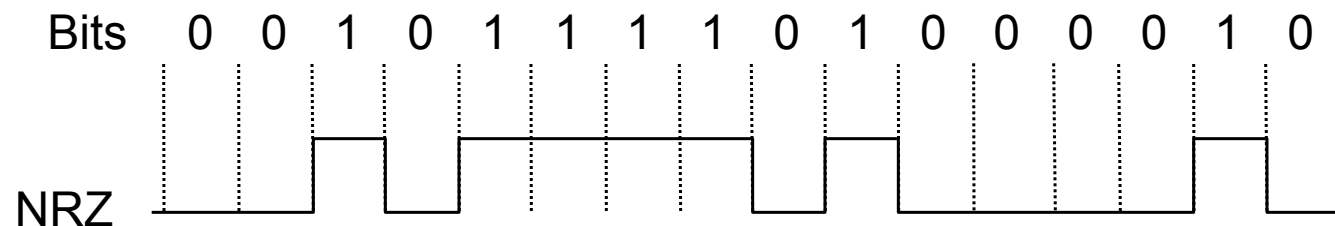


- Convert bits to signals and recover bits from received signals
  - Encoding
- Decide on a minimum unit for sending bits
  - Cannot send bit by bit (too much overhead)
  - Frame creation
- Error detection and/or correction of frames
  - Parity, CRC
- Flow control
  - ARQ, Sliding WINDOW
- Addressing
  - MAC address

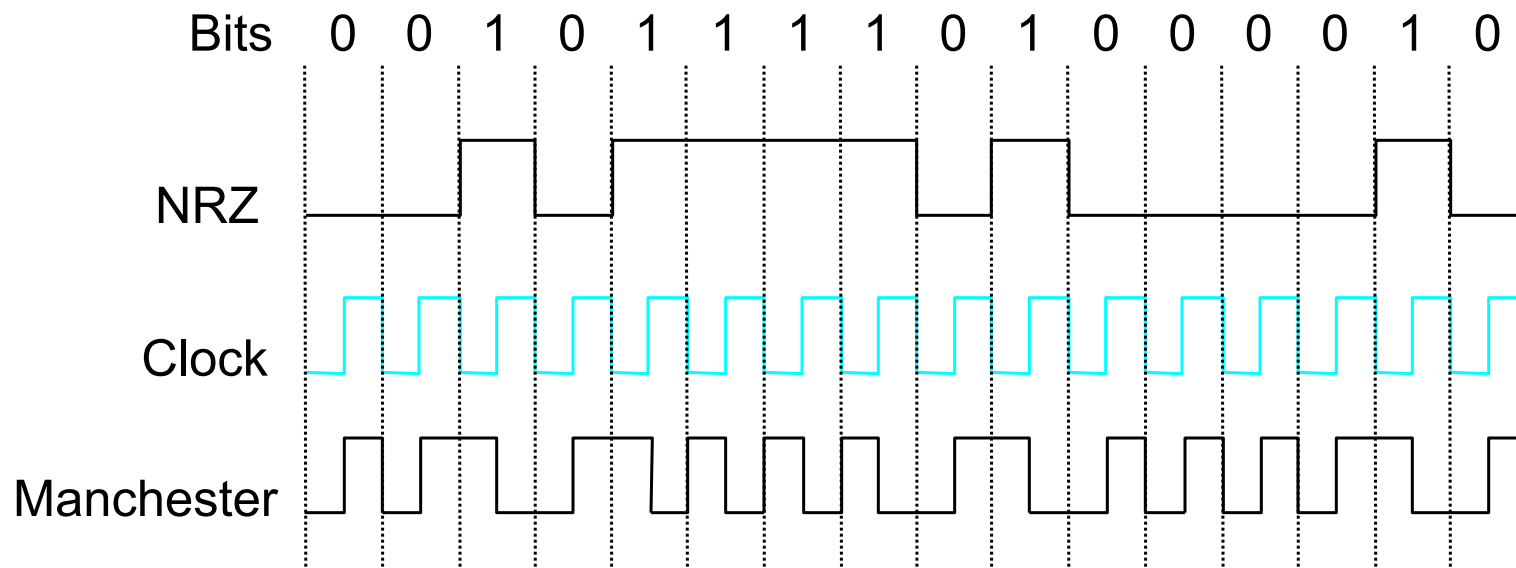
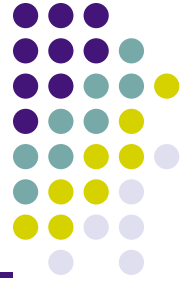
# Encoding



- Signals propagate over a physical medium
  - modulate electromagnetic waves
  - e.g., vary voltage
- Encode binary data onto signals
  - e.g., 0 as low signal and 1 as high signal
  - known as Non-Return to zero (NRZ)
  - Problem: consecutive 1s and 0s , noise levels



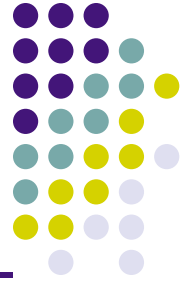
# Encodings (cont)



- Manchester encoding: +ve transition  $\rightarrow$  0; -ve transition  $\rightarrow$  1
- XOR(bit,clock)

# Framing

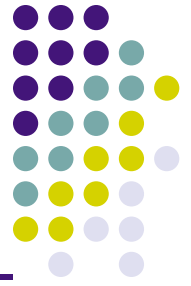
---



- The data unit at the data link layer is called a “frame”
- A frame is a group of bits, typically in sequence
- Issues:
  - Frame creation
    - How many bits (size of frame)
    - Overhead
  - Frame delineation
  - Have meta tags
    - start and stop characters or bit sequence
  - What if the meta tags appear in the message?

# stuffing

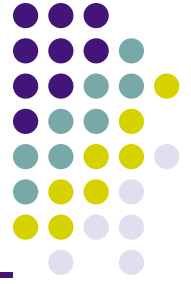
---



- Character stuffing (\$, #)
  - \$# this prof is good \$^
  - \$# this prof s\$^ks \$^ .. Meta tag in message
    - \$# this prof s\$\$^ks \$^ at sender
    - \$# this prof s\$^ks \$^ at receiver, remove stuffing
- Bit stuffing: have a unique bit sequence
  - 01111110 this prof is good 01111110
  - 01111110 this prof is 01111110 good 01111110
  - 01111110 this prof is 011111010 good 01111110 -- sender
  - Receiver checks for 5 1s, if next bit is 0 – stuff
  - If next bits are 10 end of frame else error

# Error Control

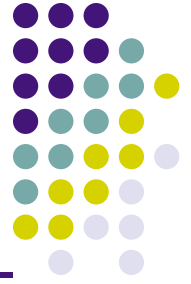
---



- No physical link is perfect
- Bits will be corrupted
- We can either:
  - detect errors and request retransmission
  - or correct errors without retransmission

# Error Detection

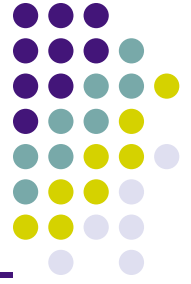
---



- Parity bits
  - For a fixed sequence, add a 1 bit (0/1) to have odd 1s (odd parity) or even 1s (even parity)
  - Extra bits are overhead, multiple bit errors
- Checksum
  - Divide msg into fixed size (16-bit) chunks
  - Add chunks (using 1s complement) and send check sum (complement of total ) with message
  - Receiver add msg + checksum
  - Complement (result) = 0 accept else reject
- Polynomial codes or CRC
  - Divide the MSG by polynomial, add R to get CRC bits
  - Receiver : divides MSG + R, check if zero

# Flow Control

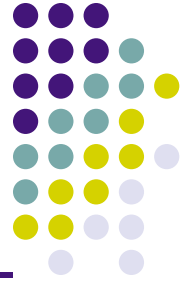
---



- What happens if the sender tries to transmit faster than the receiver can accept?
- Data will be lost unless flow control is implemented
- Has to be dynamic?
  - Can sender learn the receiver rate apriori

# Some Flow Control Algorithms

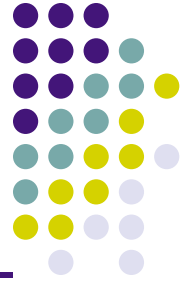
---



- Stop and wait
  - Send 1, wait for ack, then send next
  - Retransmit after timeout, use sequence numbers to detect duplicates
- Sliding window
  - Send  $W$ , wait for ack to advance the window
  - At any point in time  $\text{Max}(W)$  unacknowledged messages
- Sliding window with error control
  - Go Back  $N$
  - Selective Repeat

# Handling errors

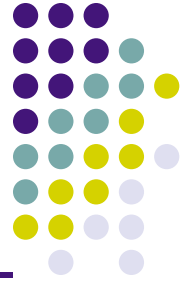
---



- GO back N
  - Receiver rejects any message in error or out of order
  - Only acks in-sequence
- Selective repeat
  - Receiver buffers correctly (out-of-sequence) received messages but acks only the last in-sequence message received correctly
  - Sender retransmits only the lost packet,

# Addressing

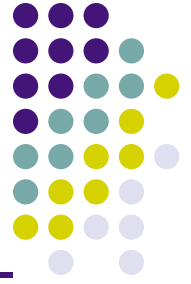
---



- Hosts need to be identified at the link layer
- MAC address
  - 48 bits unique address (permanent with adapter)
  - 24 bits: manufacturer; 24 bits Serial number
- No relationships between MAC addresses hosts connected by a link
  - No grouping or hierarchy possible
- Fixed length address
  - Look up is efficient but table size = number of hosts on the network
  - Scaling

# Connecting multiple hosts

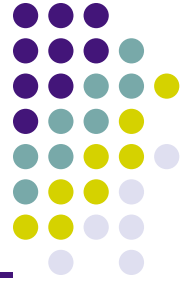
---



- All hosts share the same link
  - Simple, need to deal with contention
  - A pair at a time can communicate
- Each pair of hosts connected by separate link
  - Mesh connection required, complex
  - $N/2$  pairs can simultaneously communicate

# Contention Access Methods

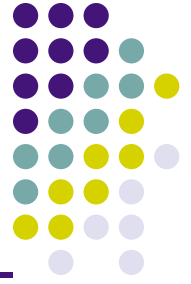
---



- Determine when to transmit, sense the channel
- CSMA
  - 1-Persistent CSMA
    - Transmit if idle, else wait until idle and then transmit
  - Non-Persistent CSMA
    - Transmit if idle, else wait for random time, and then repeat
    - Spreads arrival times
  - P-Persistent CSMA
  - Transmit with probability  $p$  if idle, else wait until idle

# CSMA/CD

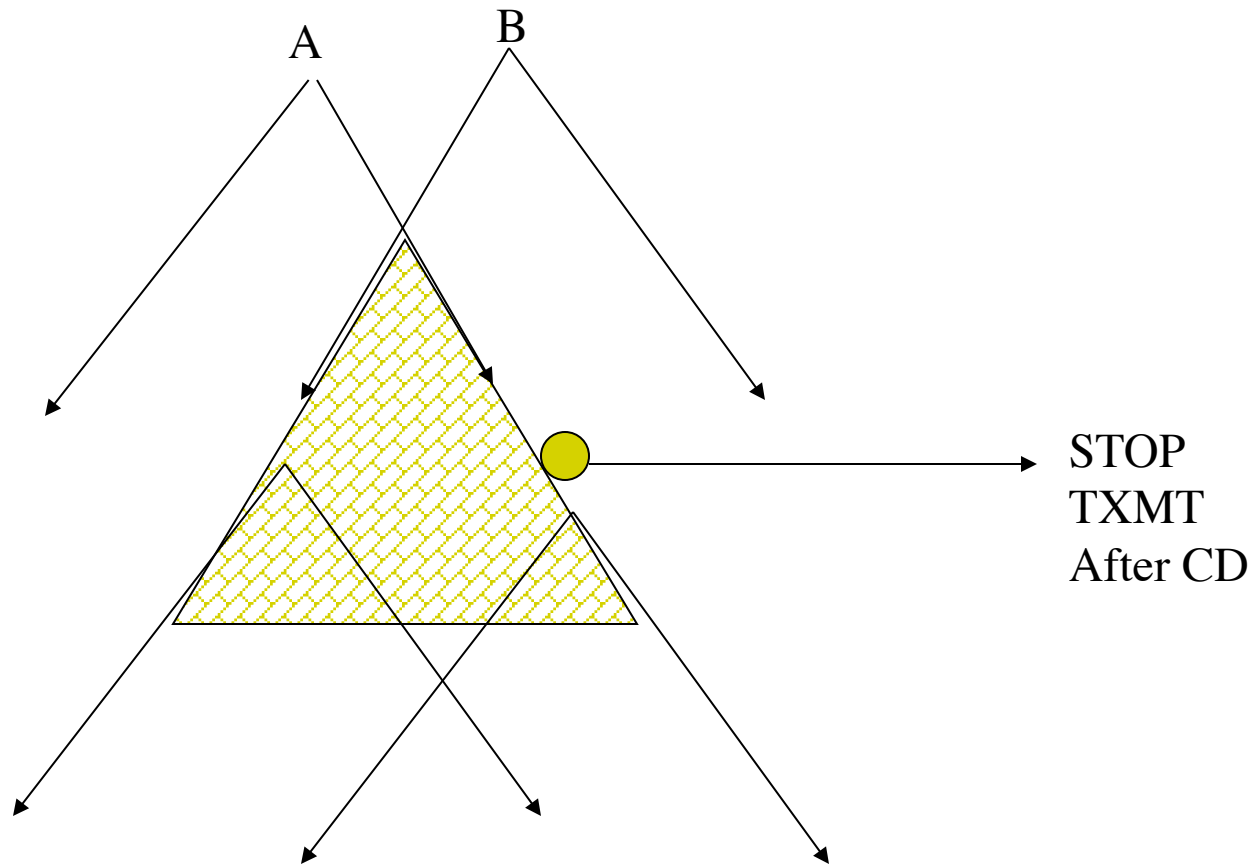
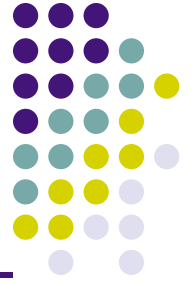
---



- In CSMA protocols
  - If two stations begin transmitting at the same time, each will transmit its complete packet, thus wasting the channel for an entire packet time
- In CSMA/CD protocols
  - The transmission is terminated immediately upon the detection of a collision
  - CD = Collision Detect
- In wired links, transreceiver can send and receive simultaneously

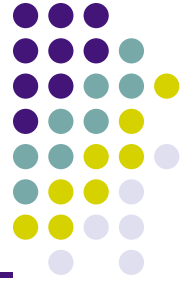
# Collision detection

---

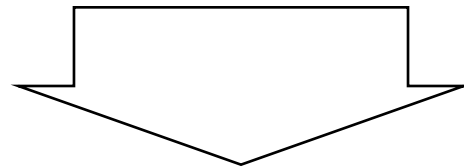


# Ethernet Backoff Algorithm: Binary Exponential Backoff

---



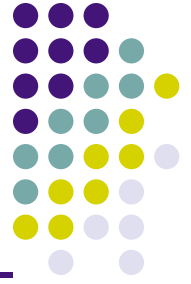
- If collision,
  - Choose one slot randomly from  $2^k$  slots, where  $k$  is the number of collisions the frame has suffered.
  - One contention slot length = 2 x end-to-end propagation delay



This algorithm can adapt to  
changes in network load.

# Algorithm (cont)

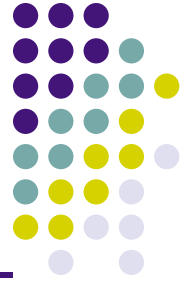
---



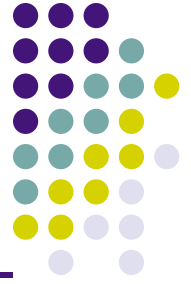
- If collision...
  - jam for 32 bits, then stop transmitting frame
  - minimum frame is 64 bytes (header + 46 bytes of data)
  - delay and try again
    - 1st time: 0 or 51.2us
    - 2nd time: 0, 51.2, or 102.4us
    - 3rd time: 51.2, 102.4, or 153.6us
    - *n*th time:  $k \times 51.2\text{us}$ , for randomly selected
      - $k=0 \dots 2^n - 1$
    - give up after several tries (usually 16)
    - exponential backoff

# CSMA/CD and Ethernet

---

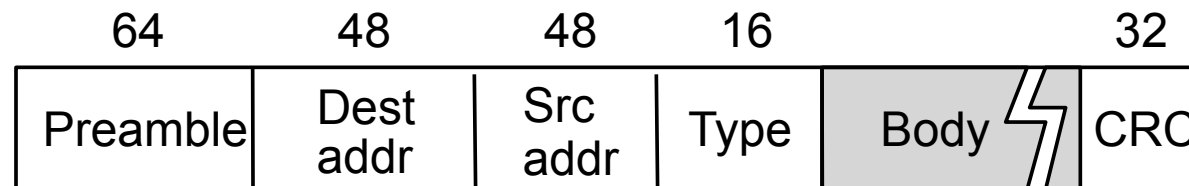


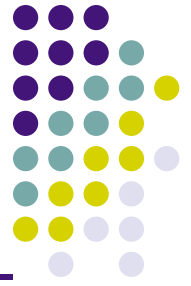
- Ethernet:
  - Short end-to-end propagation delay
  - Broadcast channel
- Ethernet access protocol:
  - 1-Persistent CSMA/CD
  - with Binary Exponential Backoff Algorithm



# Ethernet Overview

- History
  - developed by Xerox PARC in mid-1970s
  - roots in Aloha packet-radio network
  - standardized by Xerox, DEC, and Intel in 1978
  - similar to IEEE 802.3 standard
- CSMA/CD
  - Available in 10 Mbps, 100 Mbps, 1 Gbps
  - Coax or twisted pair
- Frame Format

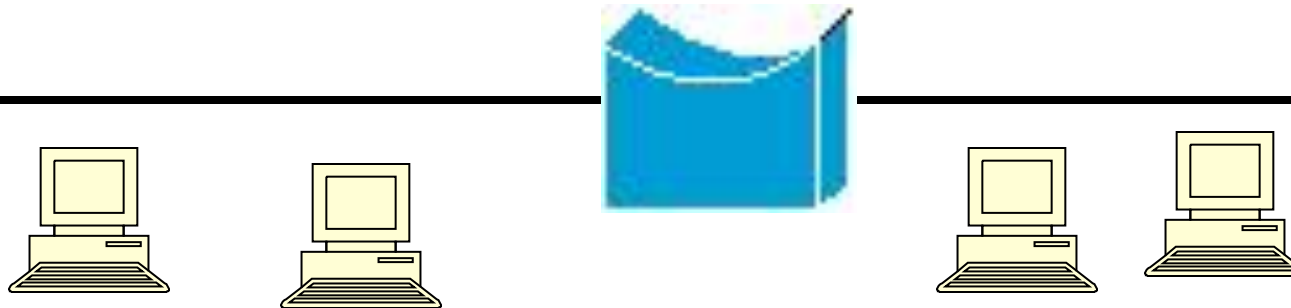




# Interconnecting LANS

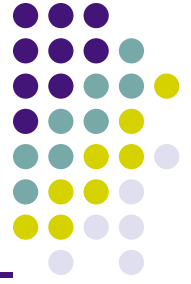
---

- Why not just one big LAN?
  - Limited amount of supportable traffic: on single LAN, all stations must share bandwidth
  - Single collision domain
- Physical layer extension
  - Repeaters
  - copies (amplifies, regenerates) bits between LAN segments
- Link Layer extensions
  - Bridges - connects (2) LAN segments
  - Each segment is its own collision domain
  - receives, stores, forward (when appropriate) packets between LANs
  - Learn which host is connected on which interface
  - Forget about the mapping after certain TTL – soft state



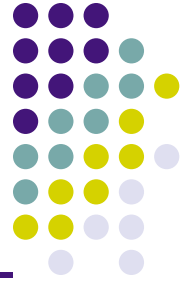
# Forwarding Algorithm

---

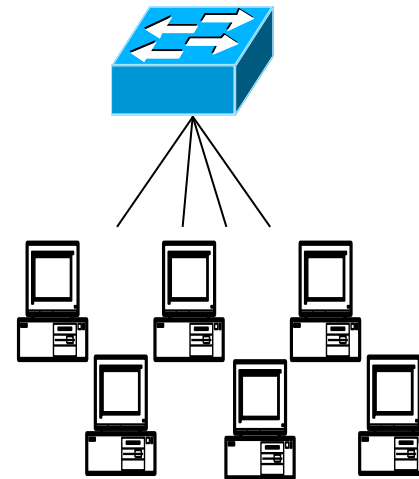


1. bridge receives every packet transmitted on every attached LAN
2. bridge stores for each packet
  - physical address of sender
  - port (incoming LAN segment) on which pkt was received
3. for each packet received on any port: lookup dest. physical address in table
  - if not found, flood onto all attached LANs
  - if found, forward only out to specified LAN
4. forwarding table deleted if not refreshed

# Switches



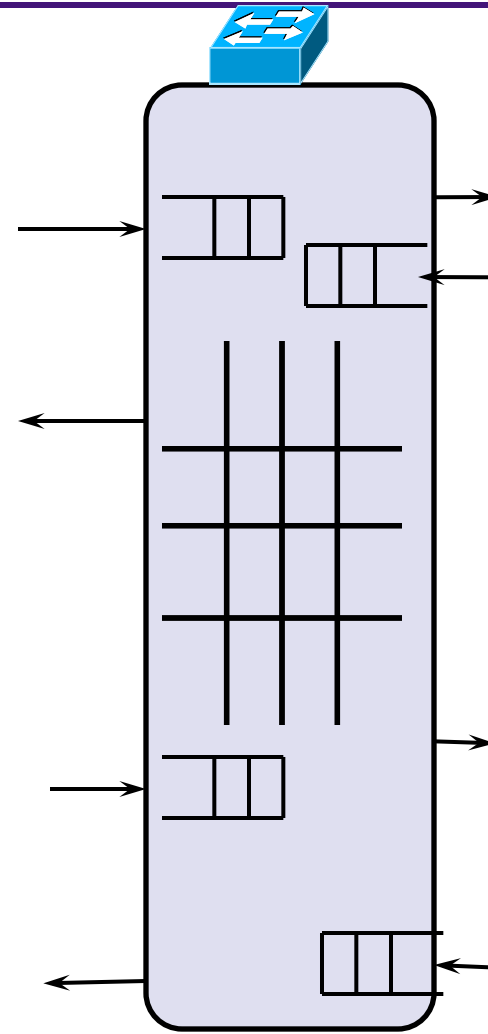
- Segments LANs (multi-port Bridge)
- Layer 2 Processing
- Directly connect hosts
- Multiport bridge
- Directly forward frames
- Cut-through switching
  - Forward as soon as header is processed
- Store-and forward
  - Buffer entire packet, check integrity (CRC)
  - Discard erroneous packets



# Switches

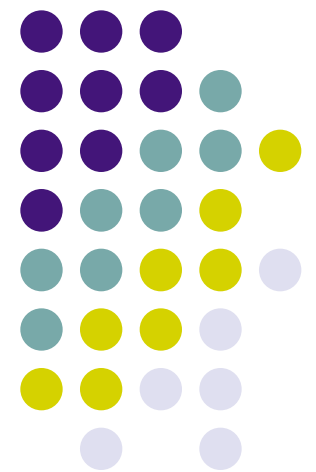


- Can connect a large number of ports/interfaces
- Switching fabric can send frames from any input to any output
- $N/2$  simultaneous transfers
- $C = N/2 * \text{line speed}$



# IP address hierarchy

---



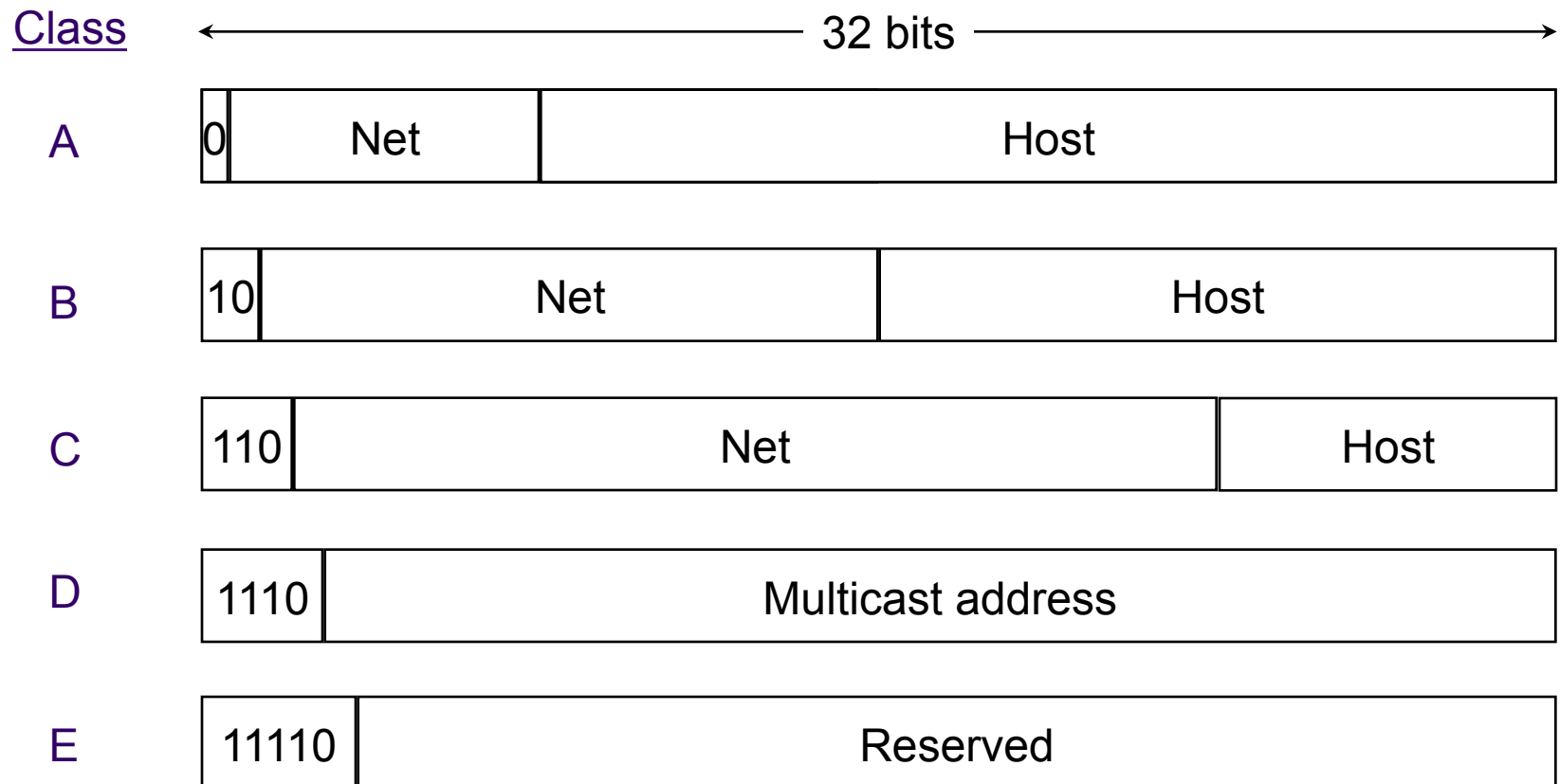
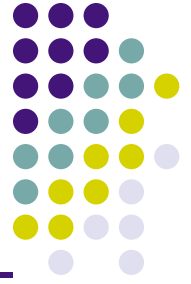
# IP Addresses

---



- 32 bits long
- Identifier for host, router *interface*
- Notation:
  - Each byte is written in decimal in MSB order, separated by dots
  - Example: 128.195.1.80

# IP Address Classes (old)



# IP Address Classes (old way)

---



- Class A:
  - For very large organizations
  - 16 million hosts allowed
  - 0.\*.\*.\* to 127.\*.\*.\*
- Class B:
  - For large organizations
  - 65 thousand hosts allowed
  - 128.\*.\*.\* to 191.\*.\*.\*
- Class C
  - For small organizations
  - 255 hosts allowed
  - 192.\*.\*.\* to 223.\*.\*.\*
- Class D
  - Multicast addresses
  - No network/host hierarchy
  - 224.\*.\*.\* to 239.\*.\*.\*
- Class E
  - Reserved, not used
  - 240.\*.\*.\* to 255.\*.\*.\*

# IP Address Hierarchy

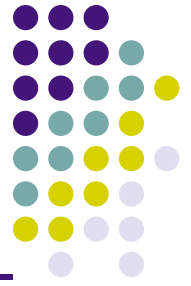
---



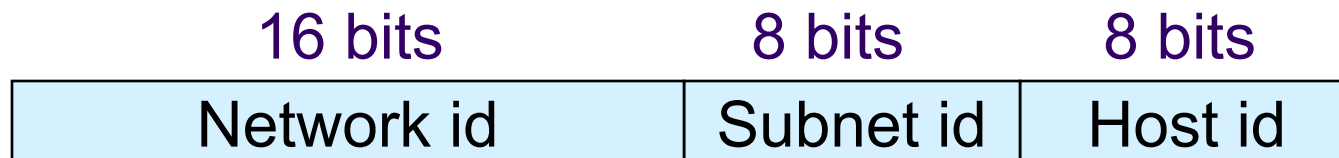
- Class A, B, C addresses support two levels of hierarchy
- However, the host portion can be further split into “subnets” by the address class owner
  - more than 2 levels of hierarchy

# Subnetting

---



Example: Class B address with 8-bit subnetting



Example  
Address:

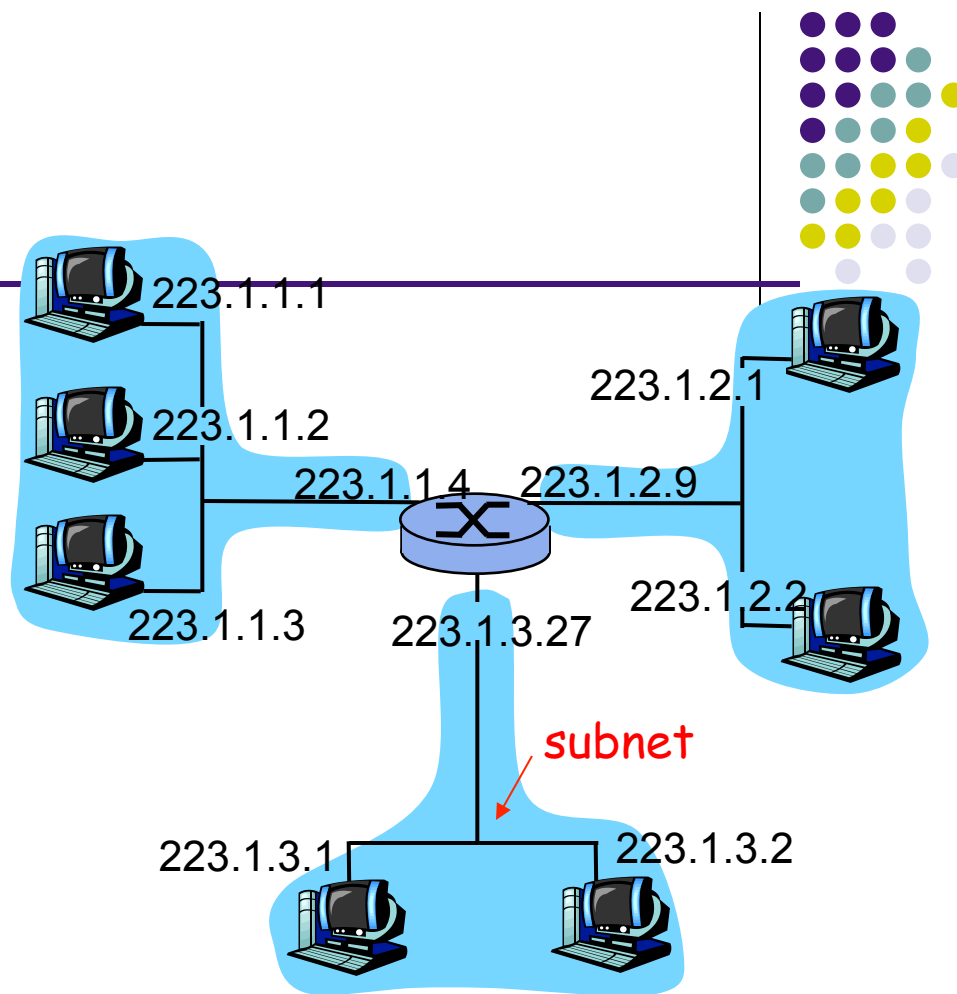
165.230

.24

.8

# Subnets

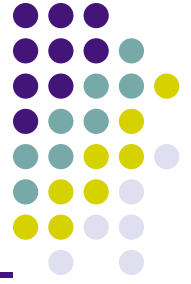
- IP address:
  - subnet part (high order bits)
  - host part (low order bits)
- *What's a subnet ?*
  - device interfaces with same subnet part of IP address
  - can physically reach each other without intervening router



network consisting of 3 subnets

# Subnet Masks

---

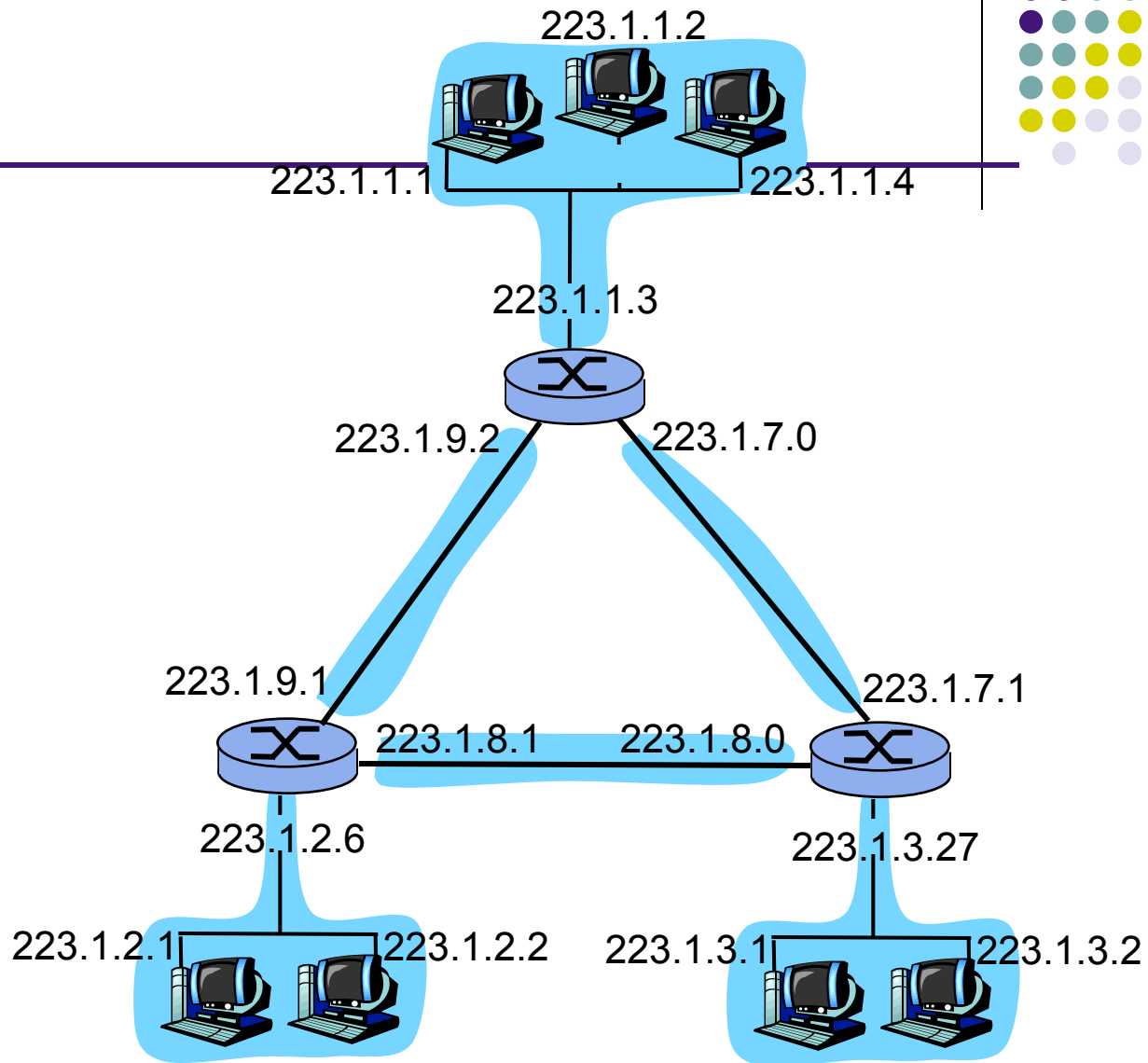


Subnet masks allow hosts to determine if another IP address is on the same subnet or the same network

	16 bits	8 bits	8 bits
	Network id	Subnet id	Host id
Mask:	1111111111111111	11111111	00000000
	255.255	.255	.0

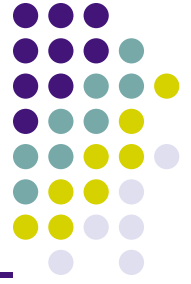
# Subnets

How many?



# Subnet Masks *(cont'd)*

---



Assume IP addresses A and B share subnet mask M.

Are IP addresses A and B on the same subnet?

1. Compute (A and M).
2. Compute (B and M).
3. If (A and M) = (B and M) then A and B are on the same subnet.

Example: A and B are class B addresses

A = 165.230.82.52

B = 165.230.24.93

M = 255.255.255.0

Same network?

Same subnet?

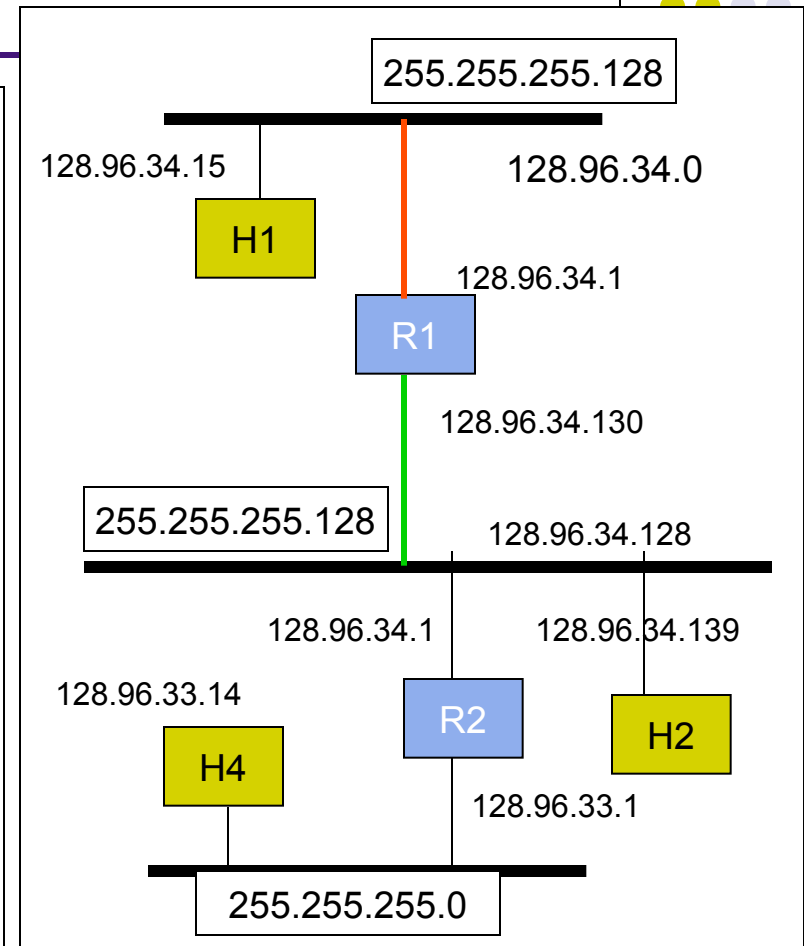
# Routing across Subnet



- Packet to H2
- 128.96.34.139

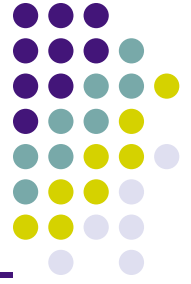
## Routing table at R1

Subnet number	Subnet mask	nexthop
128.96.34.0	255.255.255.128	Interface 0
128.96.34.128	255.255.255.128	Interface 1
128.96.33.0	255.255.255.0	R2
10.1.2.0/23	10.1.2.1	10.1.2.1
10.1.0.0/23	10.1.2.2	10.1.2.1

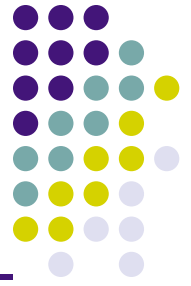


# Problems with Class-based Routing

---



- Too many small networks requiring multiple class C addresses
- Running out of class B addresses, not enough nets in class A
- Addressing strategy must allow for greater diversity of network sizes



# IP addressing: CIDR

---

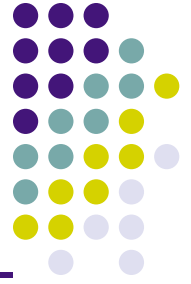
## CIDR: Classless InterDomain Routing

- subnet portion of address of arbitrary length
- address format: **a.b.c.d/x**, where x is # bits in subnet portion of address

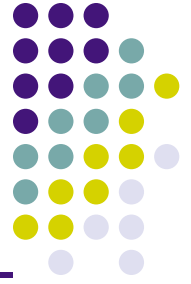


# CIDR

---

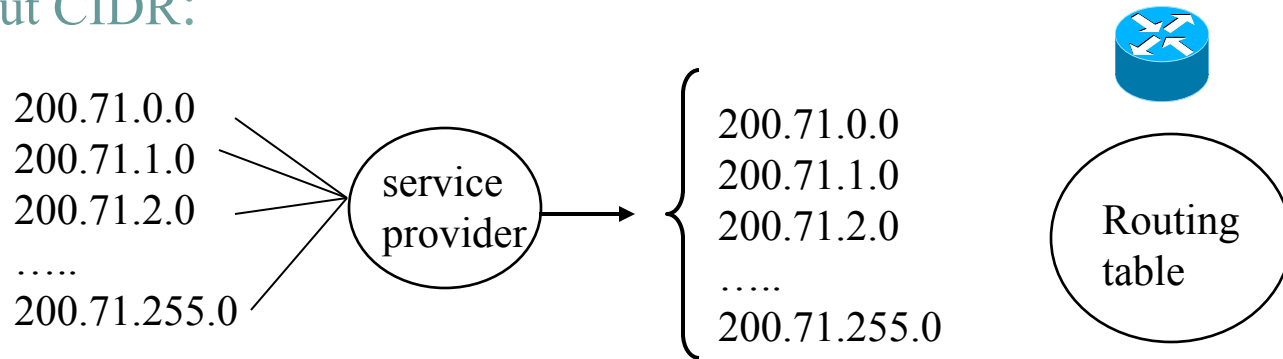


- An ISP can obtain a block of addresses and partition this further to its customers
  - Say an ISP has 200.8.4/24 address (256 addresses). He has another customer who needs only 4 addresses
  - from 200.8.4/24 a block of 4 addresses can be specified as 200.8.4.24/30
- Customer buys or is allocated a prefix by ISP
  - Addresses assigned to hosts by customer

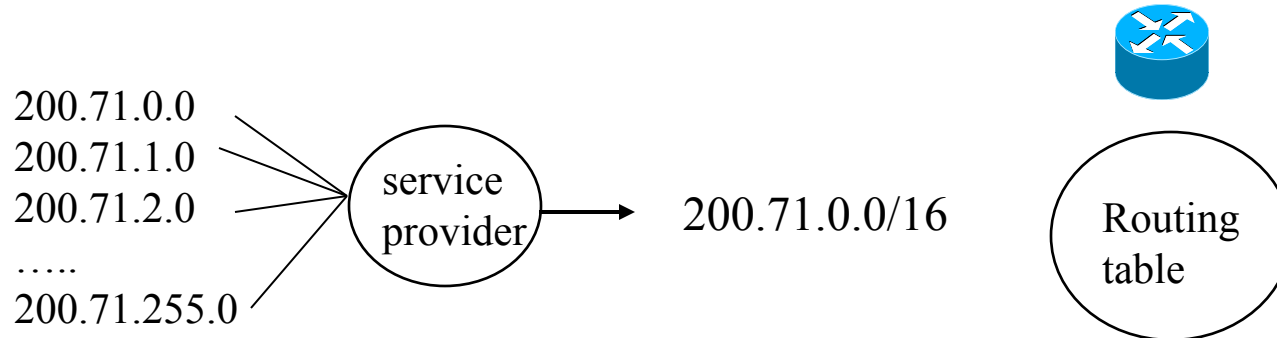


# Reducing Routing Table Size

Without CIDR:



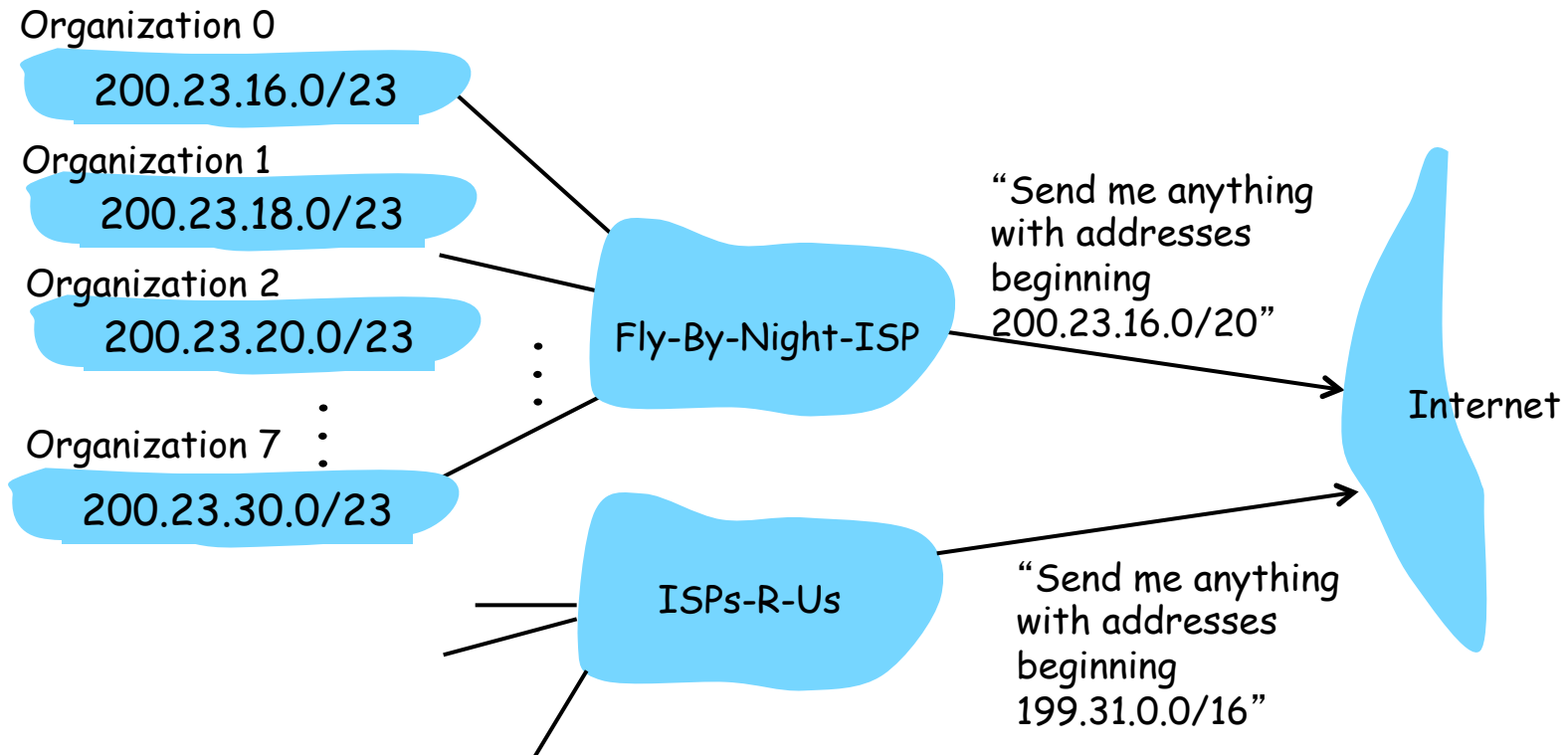
With CIDR:





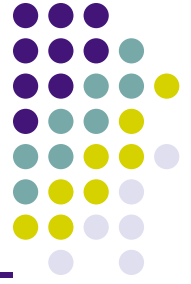
# Hierarchical addressing: route aggregation

Hierarchical addressing allows efficient advertisement of routing information:



# IPv4 Addresses may not suffice ... If free?

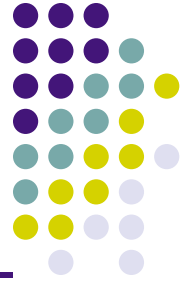
---



- $2^{32}$  is 4G
  - Not all that many unique addresses
  - Plus, some are reserved for special purposes
  - And, addresses are allocated in larger blocks
- And, many more devices need IP addresses
  - Computers, smart phones, routers, sensors, , ...
- Long-term solution: a larger address space
  - IPv6 has 128-bit addresses ( $2^{128} = 3.403 \times 10^{38}$ )
- Short-term solution: maybe winning
  - Use Private addresses with NAT

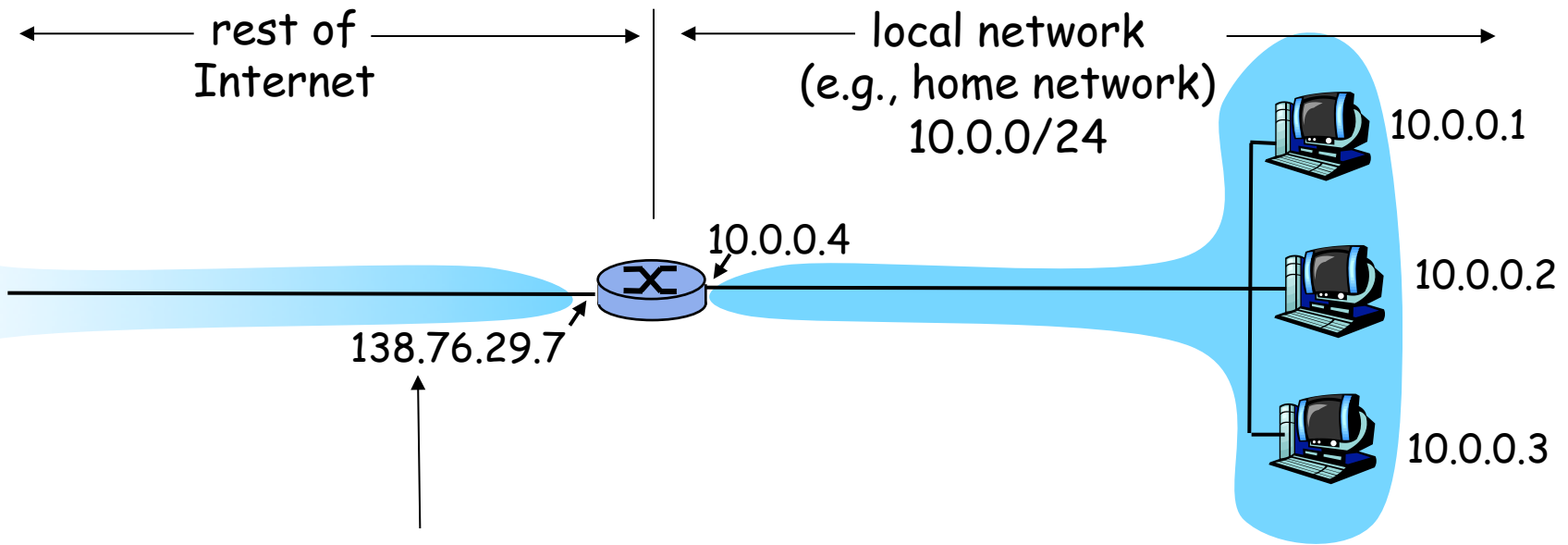
# Private IP addresses

---



- **A block of addresses reserved for internal routing**
- **10.0.0.0 to 10.255.255.255 → 10.0.0.0/8**
- **172.16.0.0 to 172.31.255.255 → 172.16/12**
- **192.168.0.0 to 192.168.255 → 192.168/16**

# NAT: Network Address Translation



*All* datagrams *leaving* local network have *same* single source NAT IP address: 138.76.29.7, different source port numbers

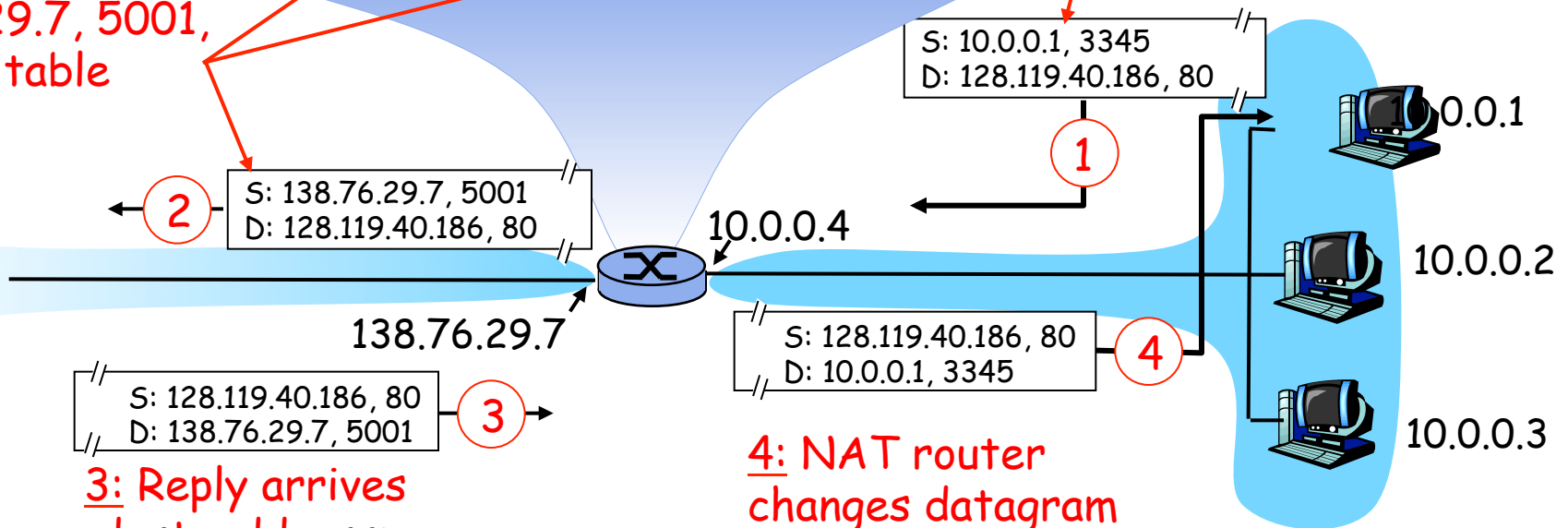


# NAT: Network Address Translation

NAT translation table	
WAN side addr	LAN side addr
138.76.29.7, 5001	10.0.0.1, 3345
.....	.....

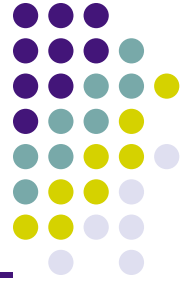
**2:** NAT router changes datagram source addr from 10.0.0.1, 3345 to 138.76.29.7, 5001, updates table

**1:** host 10.0.0.1 sends datagram to 128.119.40.186, 80




**3:** Reply arrives  
dest. address:  
138.76.29.7, 5001

**4:** NAT router  
changes datagram  
dest addr from  
138.76.29.7, 5001 to 10.0.0.1, 3345




# Types of NAT


- Full cone NAT

- Aa:Ap  Na:Np Bb:Bp  
● ← Na:Np;\*.\*


- IP restricted

- Aa:Ap  Na:Np Bb:Bp  
● ← Na:Np;Bb.\*

- Port Restricted

- Aa:Ap  Na:Np Bb:Bp  
● ← Na:Np:\*.Bp

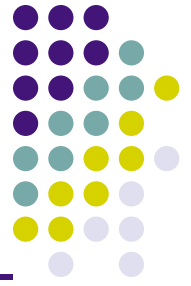
- Symmetric NAT

- Aa:Ap  Na:Np Bb:Bp  
● ← Na:Np:Ba:Bp

# NAT: Network Address Translation



- **Features:** local network uses just one IP address as far as outside world is concerned:
  - range of addresses not needed from ISP: just one IP address for all devices
    - Major cost consideration
  - can change addresses of devices in local network without notifying outside world
  - can change ISP without changing addresses of devices in local network
  - devices inside local net not explicitly addressable, visible by outside world (a security plus).



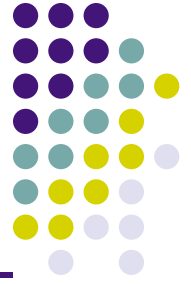
# NAT: Network Address Translation

---

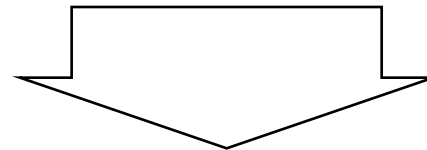
- 16-bit port-number field:
  - 60,000 simultaneous connections with a single LAN-side address!
- NAT is controversial:
  - routers should only process up to layer 3
  - violates end-to-end argument
    - NAT possibility must be taken into account by app designers, eg, VOIP, icoming notification applications
    - Pushing FB status updates on to device!!!
  - address shortage should instead be solved by IPv6
  - NAT is here to stay

# Recent Developments: IPv6

---



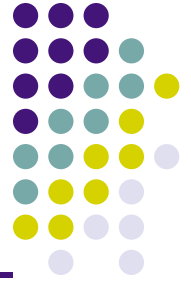
- IPv4 (the standard IP protocol) has limited address space
- Most importantly, IP is running out of addresses. 32 bits is not enough.
- Real-time traffic and mobile users are also becoming more common



IP version 6  
(Also called IPng, or IP next generation)

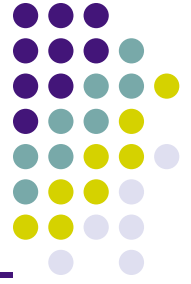
# IPv6: The Changes

---



- Large address space:
  - 128-bit addresses (16 bytes)
  - Allows up to 340,282,366,920,938,463,463,374,607,431,768,211,456 unique addresses
- Fixed length headers (40 bytes)
  - Improves the speed of packet processing in routers

# IPv6 header

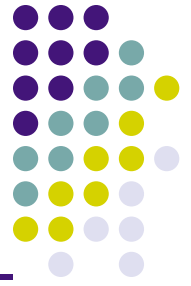


<b>Version (4)</b>	<b>TrafficClass (8)</b>	<b>Flow Label (20)</b>	4
<b>PayloadLen (16)</b>	<b>Next Header (8)</b>	<b>Hop Limit (8)</b>	4
<b>Source Address</b>			16
<b>Destination Address</b>			16

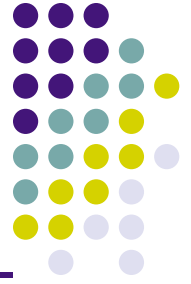
- Version field set to 6
- PayloadLen field gives the length in bytes of the packet excluding the header
- Next Header value gives the header (if any ) that follows the IPv6 header

# IPv6: The Changes *(cont'd)*

---



- Support for “flows”
  - Flows help support real-time service in the Internet
  - A “flow” is a number in the IPv6 header that can be used by routers to see which packets belong to the same stream
  - Guarantees can then be assigned to certain flows
  - Example:
    - Packets from flow 10 should receive rapid delivery
    - Packets from flow 12 should receive reliable delivery



# IPv6 Addresses

---

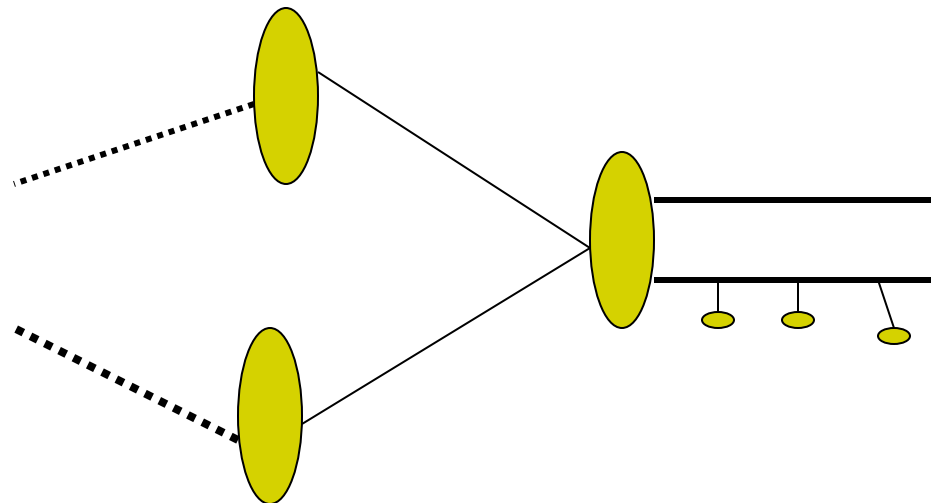
- Classless addressing/routing (similar to CIDR)
- Notation: x:x:x:x:x:x:x:x (x = 16-bit hex number)
  - contiguous 0s are compressed:  
47CD::A456:0124
  - IPv6 compatible IPv4 address: ::128.64.18.87



# Simple routing

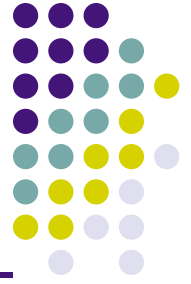
---

- Routing has 3 parameters
  - Name, address, route
- Destination directly connected?
  - If yes, forward
- If not send packet to default router



# Internet Routing

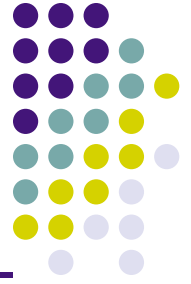
---



- Internet hierarchy
- Networks/organization belong to AS or Autonomous systems
- Two types of routing
  - Routing within AS (intradomain routing)
    - Link state (OSPF) or Distance Vector (RIP)
  - Routing between ASes (interdomain routing)
    - BGP

# Shortest Path Routing

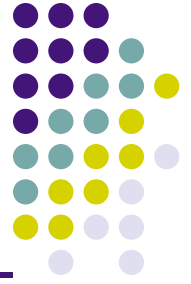
---



- For a pair of communicating hosts, there is a shortest path between them
- Shortness may be defined by:
  - Number of router/switch hops
  - Geographic distance
  - Link delay
  - Cost

# Routing Algorithms

---



- Two types of routing algorithms:
  - Global Routing Algorithms
    - Complete state information is used in routing decisions
  - Decentralized Routing Algorithms
    - Local/neighborhood state
- Hierarchical Routing is used to make these algorithms scale to large networks

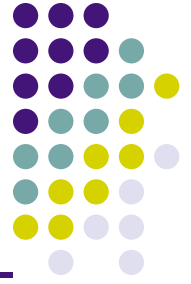
# Routing Algorithms

---

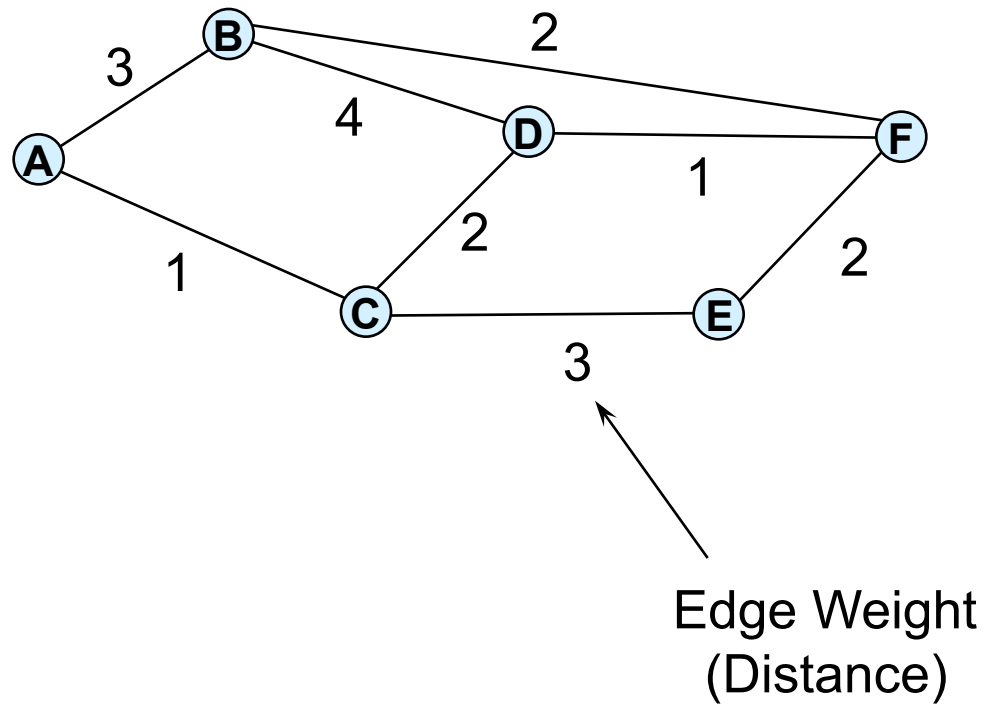


- Static routing algorithms do not base their routing decisions on the current state of the network
  - Flooding
- Dynamic routing
  - Route path changes as link cost changes

# Shortest Path

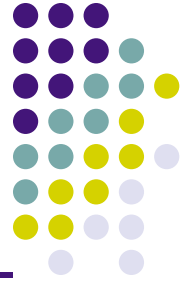


What is the shortest path between A and F?



# Computing the Shortest Path

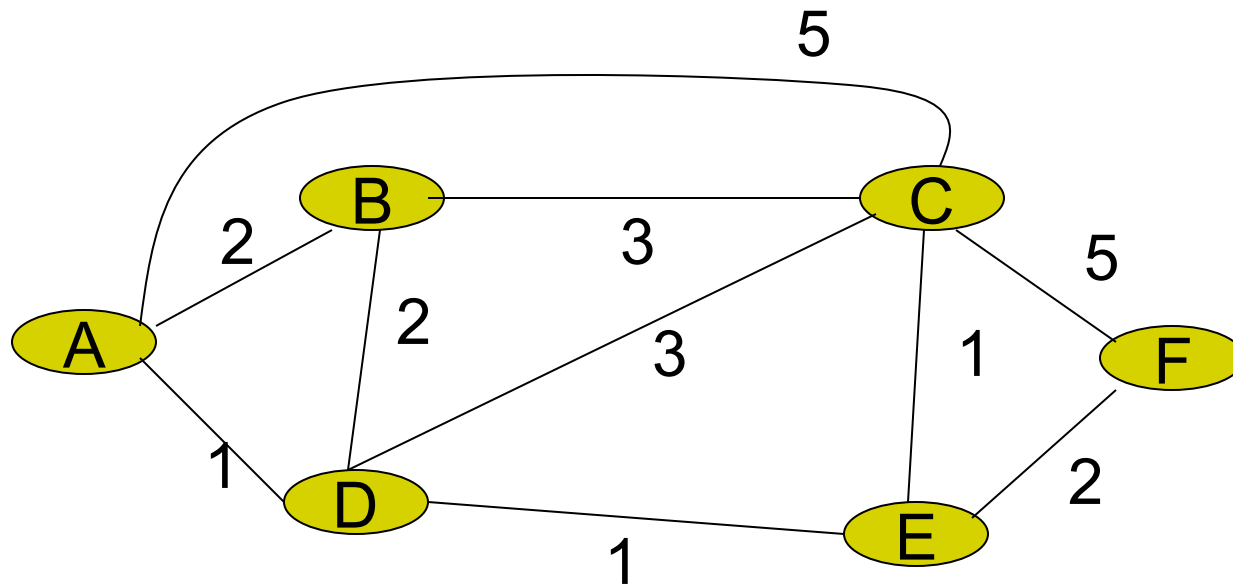
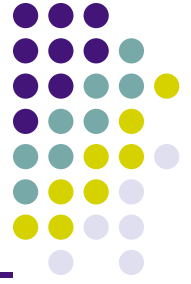
---



- Dijkstra's Shortest Path Algorithm:
  - Step 1: Draw nodes as circles. Fill in a circle to mark it as a "permanent node."
  - Step 2: Set the current node equal to the source node
  - Step 3: For the current node:
    - Mark the cumulative distance from the current node to each non-permanent adjacent node. Also mark the name of the current node. Erase this marking if the adjacent node already has a shorter cumulative distance marked
    - Mark the non-permanent node with the shortest listed cumulative distance as permanent and set the current node equal to it. Repeat step 3 until all nodes are marked permanent.

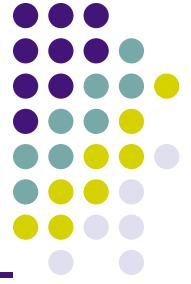
# Graph model of a network

---



# Link State Routing

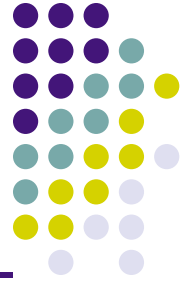
---



- Each router measures the distance (in delay, hop count, etc.) between itself and its adjacent routers
- The router builds a packet containing all these distances. The packet also contains a sequence number and an age field.
- Each router distributes these packets using flooding
- Each node builds a complete copy of graph
- Every node computes routes to every other node
  - Using single-source, shortest-path algorithm
- Process performed as needed
  - When connections die / reappear

# Link State Routing (cont'd)

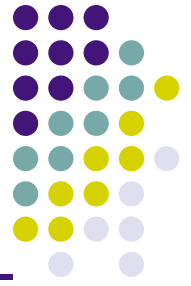
---



- To control flooding, the sequence numbers are used by routers to discard flood packets they have already seen from a given router
- The age field in the packet is an expiration date. It specifies how long the information in the packet is good for.
- Once a router receives all the link state packets from the network, it can reconstruct the complete topology and compute a shortest path between itself and any other node using Dijkstra's algorithm.

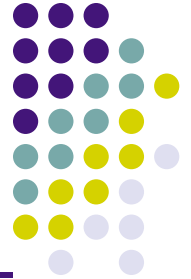
# Distance Vectors

---

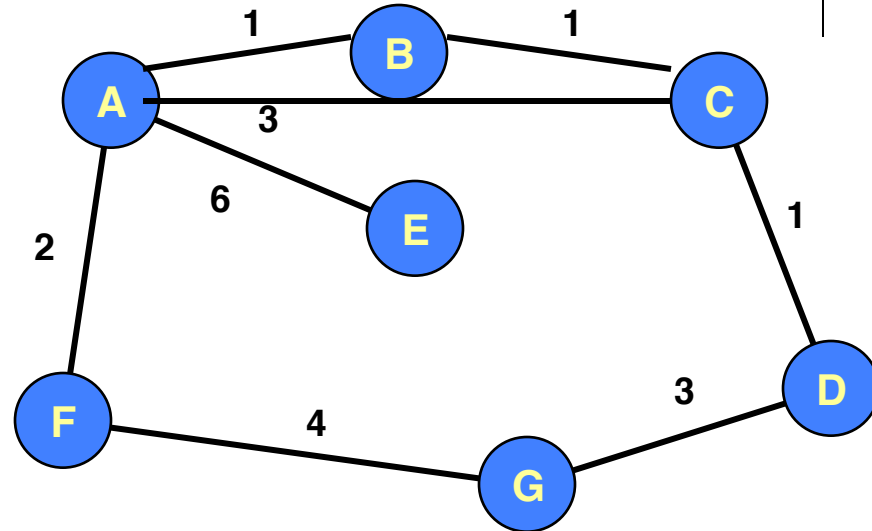


- Each router maintains lists of best-known distances to all other known routers. These lists are called “vectors.”
- Each router is assumed to know the exact distance (in delay, hop count, etc.) to other routers directly connected to it.
- Periodically, vectors are exchanged between adjacent routers, and each router updates its vectors.

# Distance-Vector Routing



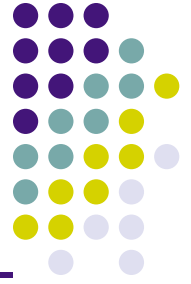
Initial Table for A		
Dest	Cost	Next Hop
A	0	A
B	1	B
C	3	C
D	$\infty$	-
E	6	E
F	2	F
G	$\infty$	-



- Initially
  - Only have entries for directly connected nodes
  - Send DV table to all neighbors, update cost with local table
  - transitive
    - Transitive

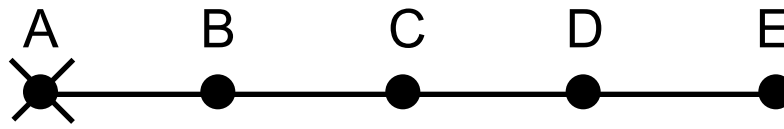
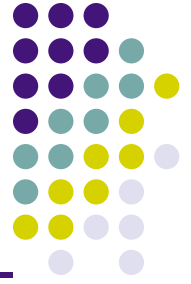
# Problem: Count-to-Infinity

---



- With distance vector routing, good news travels fast, but bad news travels slowly
- When a router goes down, it takes a really long time before all the other routers become aware of it

# Count-to-Infinity

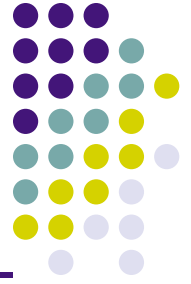


					Initially
	1	2	3	4	
	3	2	3	4	After 1 exchange
	3	4	3	4	After 2 exchanges
	5	4	5	4	After 3 exchanges
	5	6	5	6	After 4 exchanges
	7	6	7	6	After 5 exchanges

etc... to infinity

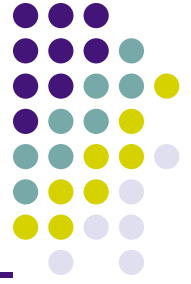
# Improvements

---

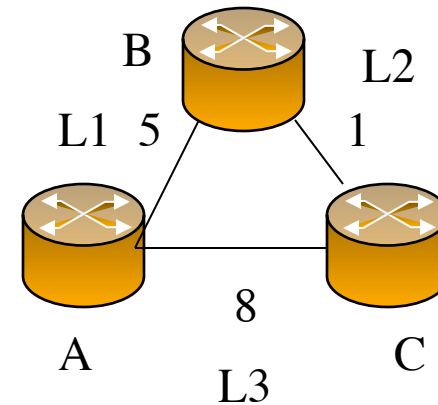


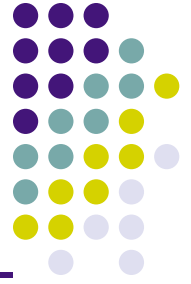
- Split Horizon
  - Don't tell neighbor about routes obtained from it
  - Or never advertise a route out of the interface you learnt it
- Poison reverse
  - Advertise network unreachable back through the same interface you learnt about it
  - advertising all network IDs, but those network IDs learned in a given direction are advertised with a metric of 16, indicating that the network is unavailable.
- Triggered updates as opposed to periodic updates
- Path vectors, Store vectors or complete path as opposed to just next hop

# Split-horizon



- C routes to A via B
- C does not advertise to B its cost to A
- If L1 breaks or cost increases to 50
- C does not announce to B the cost to A





# Poison-reverse

- C routes to A via B
- C tells B its (C's) distance to A is infinite
- You know better
- Update table is fixed length
- If L1 breaks or cost increases to 50
- C announce to B the cost to A is infinite

