CSC 458/2209 – Computer Networking Systems

Handout # 5: Interconnecting LANs; Internet Protocol (IP)

Professor Yashar Ganjali
Department of Computer Science
University of Toronto

ganjali7@cs.toronto.edu
http://www.cs.toronto.edu/~yganjali

Announcements

- Programming Assignment 1 out today (Jan. 20th).
  - Build part of a network stack: Ethernet packets from IP packets
    - Use ARP to figure out MAC address of next node.

- Logistics:
  - PA1 Handout: available on class website.
  - Due Friday Feb. 14th at 5pm.
  - Submit electronically on MarkUS.
    - NOTE. Information about submission in PA1 handout.
  - To be submitted *individually*.

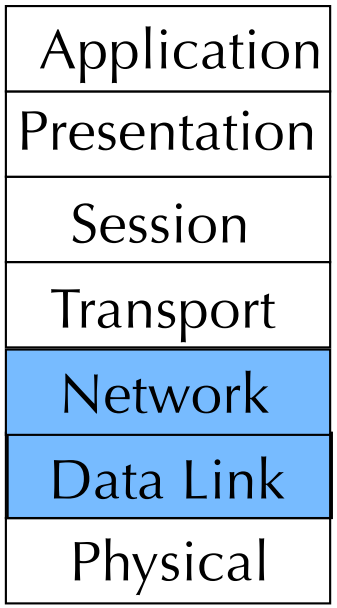
- Remember …
  - The late submission policy.
  - Academic integrity guidelines.
  - Start early! Start early! Start early! ☺

Announcements – Cont'd

- Reading for this week:
  - Chapter 3 of the textbook
  - Next week: Also, Chapter 3 (catch up if you are behind)
  - Note: only sections covered in class are required
    - But helpful to go through the rest.
- Problem Set 1: next week
  - Due on Friday the week after.
- This week's tutorial: Programming Assignment 1 (PA1) overview.
  - Next week: Problem Set 1 sample problems

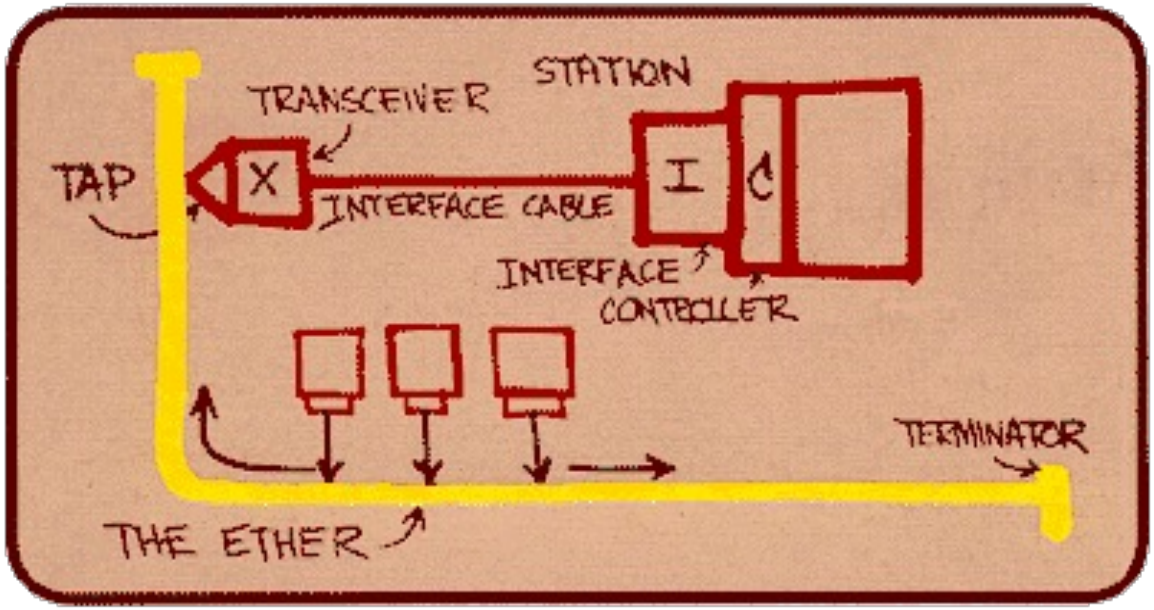
The Story

- So far …
  - Layers, and protocols
  - Link layer
    - Media type, encoding
    - Framing, link model
    - Error detection, correction
- This time
  - Interconnecting LANs
    - Hubs, switches, and bridges
  - The Internet Protocol



Ethernet

- First widely used LAN technology
- Dominant wired and wireless LAN technology
  - Significant improvements over time …
- Kept up with speed race: 10 Mbps – 400 Gbps
  - Even 800Gbps and 1.6 Tbps soon



Metcalfe's Ethernet sketch

Ethernet Uses CSMA/CD

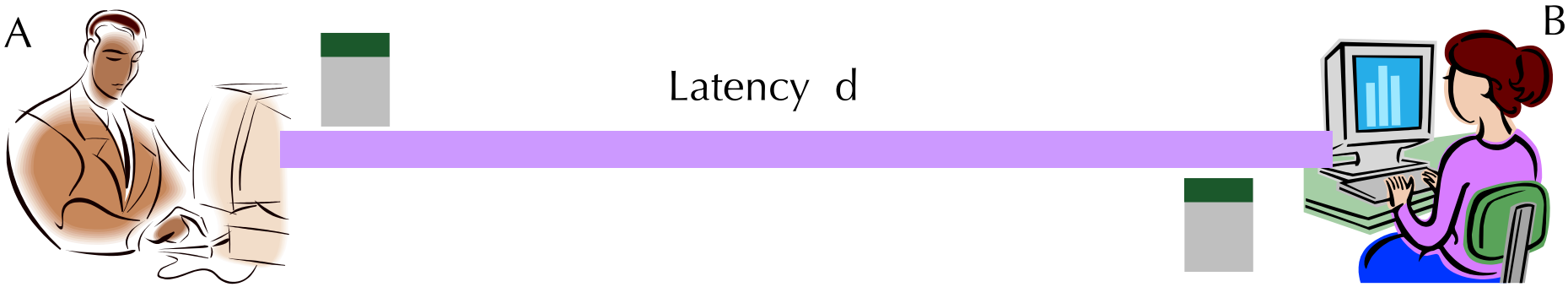
- Let us assume the link is shared among several nodes
  - We call this a collision domain: if two nodes talk at the same time, we will have a collision.
  - How can we ensure messages are not mixed up?

- Carrier sense: wait for link to be idle
  - Channel idle: start transmitting
  - Channel busy: wait until idle

- Collision detection: listen while transmitting
  - No collision: transmission is complete
  - Collision: abort transmission, and send jam signal

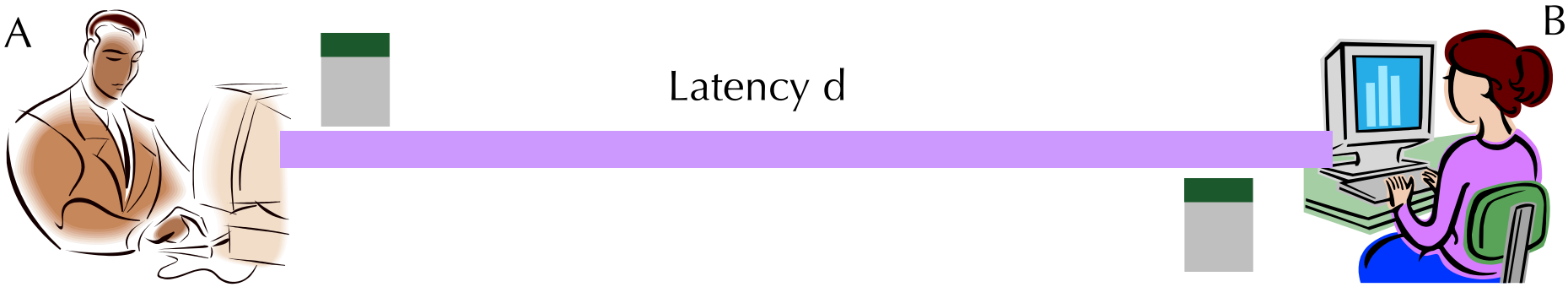
- Random access: exponential back-off
  - After collision, wait a random time before trying again
  - After the m^{th} collision, choose K randomly from $\{0, \ldots, 2^m-1\}$
  - … and wait for K*512 bit times before trying again

Limitations on Ethernet Length



- Latency depends on physical length of link
  - Time to propagate a packet from one end to the other
- Suppose A sends a packet at time t
  - And B sees an idle line at a time just before t+d
  - … so B happily starts transmitting a packet
- B detects a collision, and sends jamming signal
  - But A doesn't see collision till t+2d

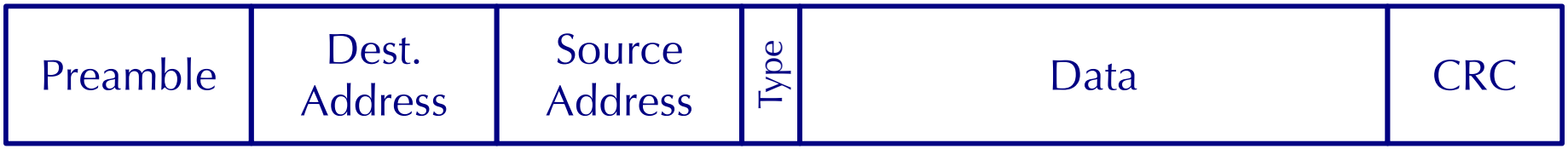
Limitations on Ethernet Length



- A needs to wait for time 2d to detect collision
  - So, A should keep transmitting during this period
  - … and keep an eye out for a possible collision
- Imposes restrictions on Ethernet
  - Maximum length of the wire: 2500 meters
  - Minimum length of the packet: 512 bits (64 bytes)

Ethernet Frame Structure

- Addresses: source and destination MAC addresses
  - Use ifconfig to get your machine's MAC address:
    - E.g., 36:45:74:cd:4b:00
  - Adaptor passes frame to network-level protocol
    - If destination address matches the adaptor
    - Or the destination address is the broadcast address
  - Otherwise, adapter discards frame
- Type: indicates the higher layer protocol
  - Usually IP
  - But also, Novell IPX, AppleTalk, … in early days
- CRC: cyclic redundancy check
  - Checked at receiver
  - If error is detected, the frame is simply dropped



Unreliable, Connectionless Service

- Connectionless
  - No handshaking between sending and receiving adapter.
- Unreliable
  - Receiving adapter doesn't send ACKs or NACKs
  - Packets passed to network layer can have gaps
  - Gaps will be filled if application is using TCP
  - Otherwise, the application will see the gaps

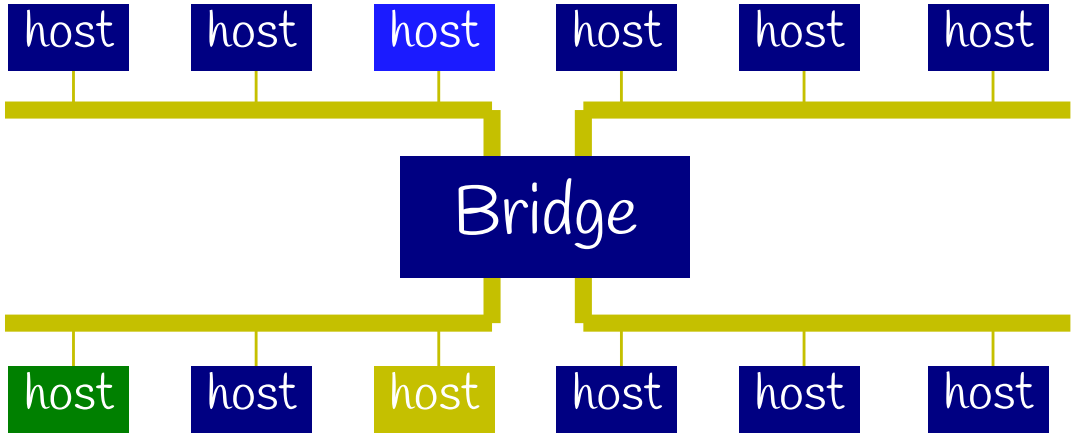
Link Layer: Switches vs. Bridges

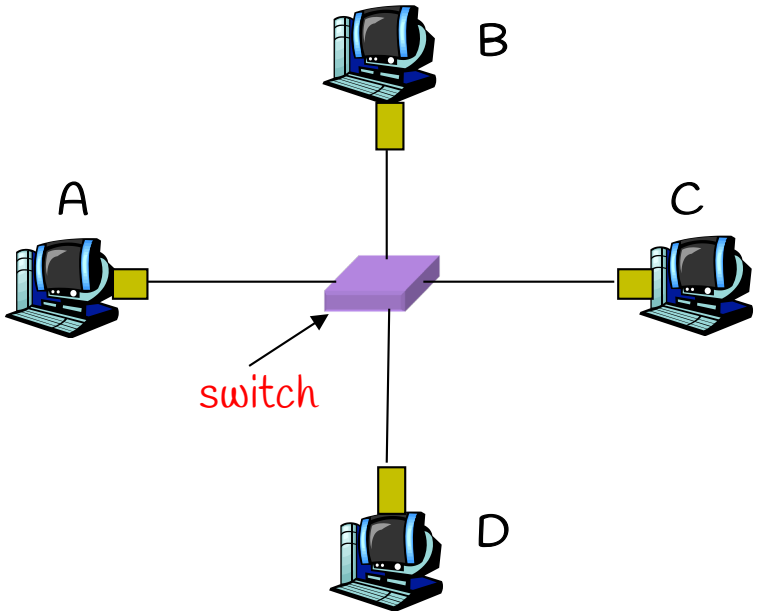
Bridge

- Connects two or more LANs at the link layer
  - Extracts destination address from the frame
  - Looks up the destination in a table
  - Forwards the frame to the appropriate LAN segment
- Each segment is its own collision domain

Switch

- Connects individual computers
  - Essentially the same as a bridge
  - … though typically used to connect hosts and switches
  - … not LANs
- Like bridges, support concurrent communication
  - A can talk to C, while B talks to D



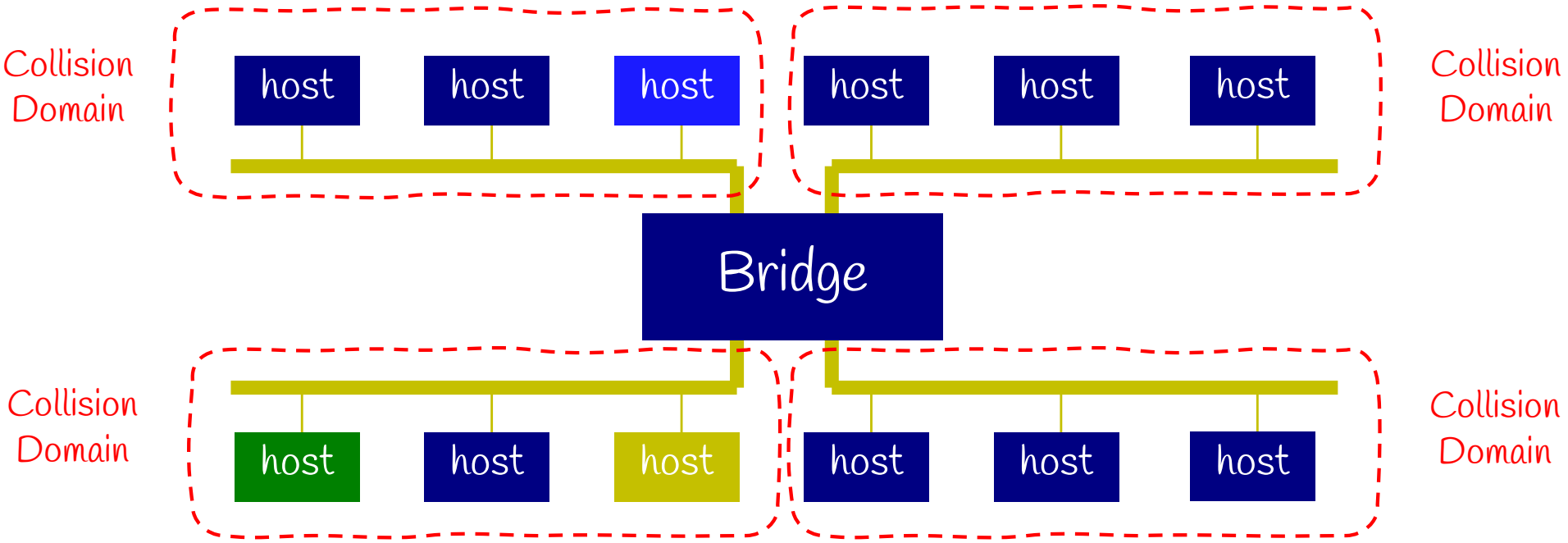


Switches: Dedicated Access and Full Duplex

- Dedicated access
  - Host has direct connection to the switch
  - … rather than a shared LAN connection
- Full duplex
  - Each connection can send in both directions
  - Host sending to switch, and host receiving from switch
- Completely avoids collisions
  - Each connection is a bidirectional point-to-point link
  - No need for carrier sense, collision detection, and so on

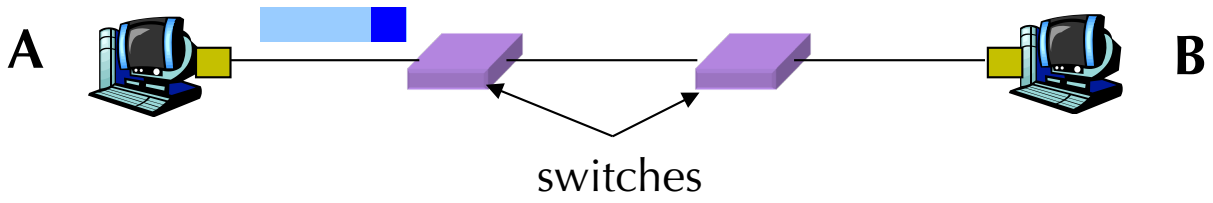
Bridges: Traffic Isolation

- Bridges break the network into separate collision domains (called LAN segments)
- Filter packets
  - Frame only forwarded to the necessary segments
  - Segments become separate collision domains



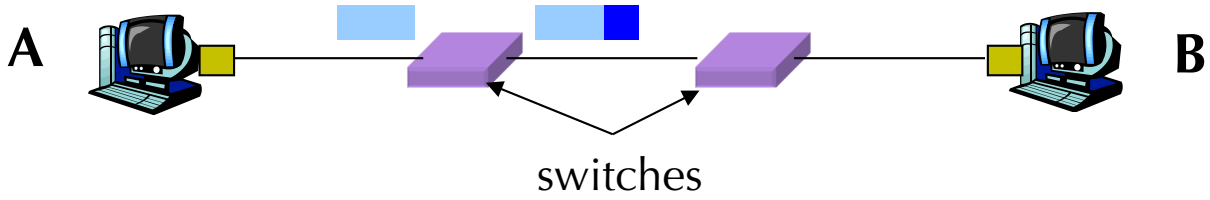
Motivation For Cut-Through Switching

- Buffering a frame takes time
  - Suppose L is the length of the frame
  - And R is the transmission rate of the links
  - Then, receiving the frame takes L/R time units
- Buffering delay can be a high fraction of total delay
  - Propagation delay is small over short distances
  - Making buffering delay a large fraction of total
  - Analogy: large group walking through NYC



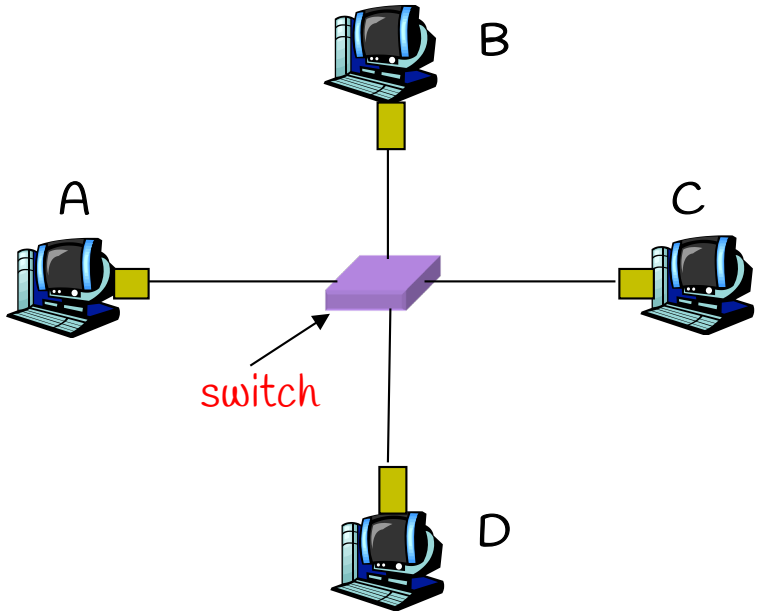
Cut-Through Switching

- Start transmitting as soon as possible
  - Inspect the frame header and do the look-up
  - If outgoing link is idle, start forwarding the frame
- Overlapping transmissions
  - Transmit the head of the packet via the outgoing link
  - … while still receiving the tail via the incoming link
  - Analogy: different folks crossing different intersections



Motivation For Self Learning

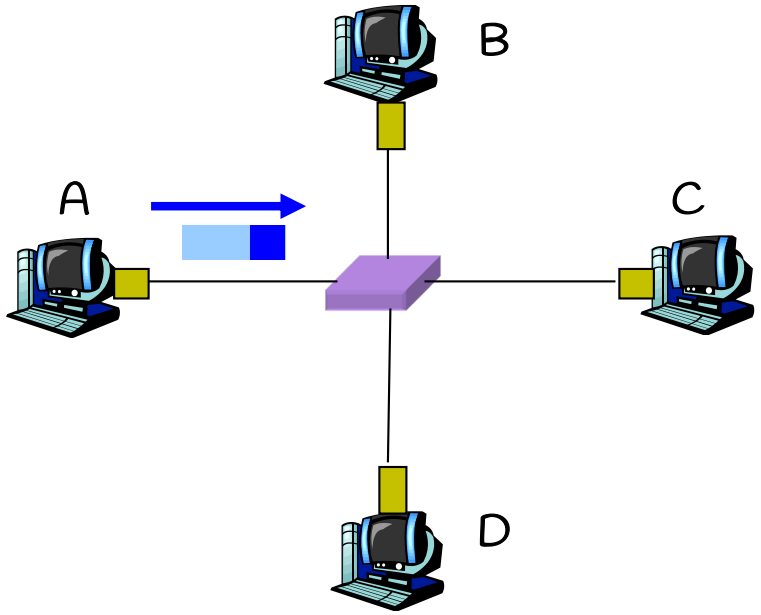
- Switches forward frames selectively
  - Forward frames only on segments that need them
- Switch table
  - Maps destination MAC address to outgoing interface
  - Goal: construct the switch table automatically



Self Learning: Building the Table

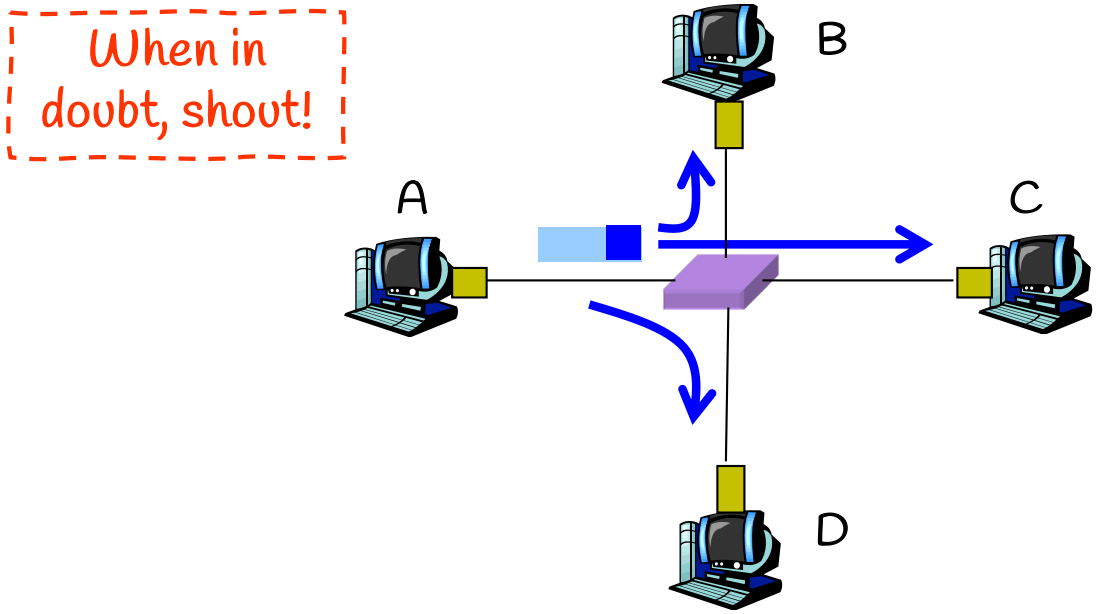
- When a frame arrives
  - Inspect the source MAC address
  - Associate the address with the incoming interface
  - Store the mapping in the switch table
  - Use a time-to-live field to eventually forget the mapping

Switch learns how to reach A.



Self Learning: Handling Misses

- When frame arrives with unfamiliar destination
  - Forward the frame out of all the interfaces
  - … except for the one where the frame arrived
  - Hopefully, this case won't happen very often



Switch Filtering/Forwarding

When switch receives a frame:

index switch table using MAC dest address
if entry found for destination
then {
if dest on segment from which frame arrived
then drop the frame
else forward the frame on interface indicated
}
else flood

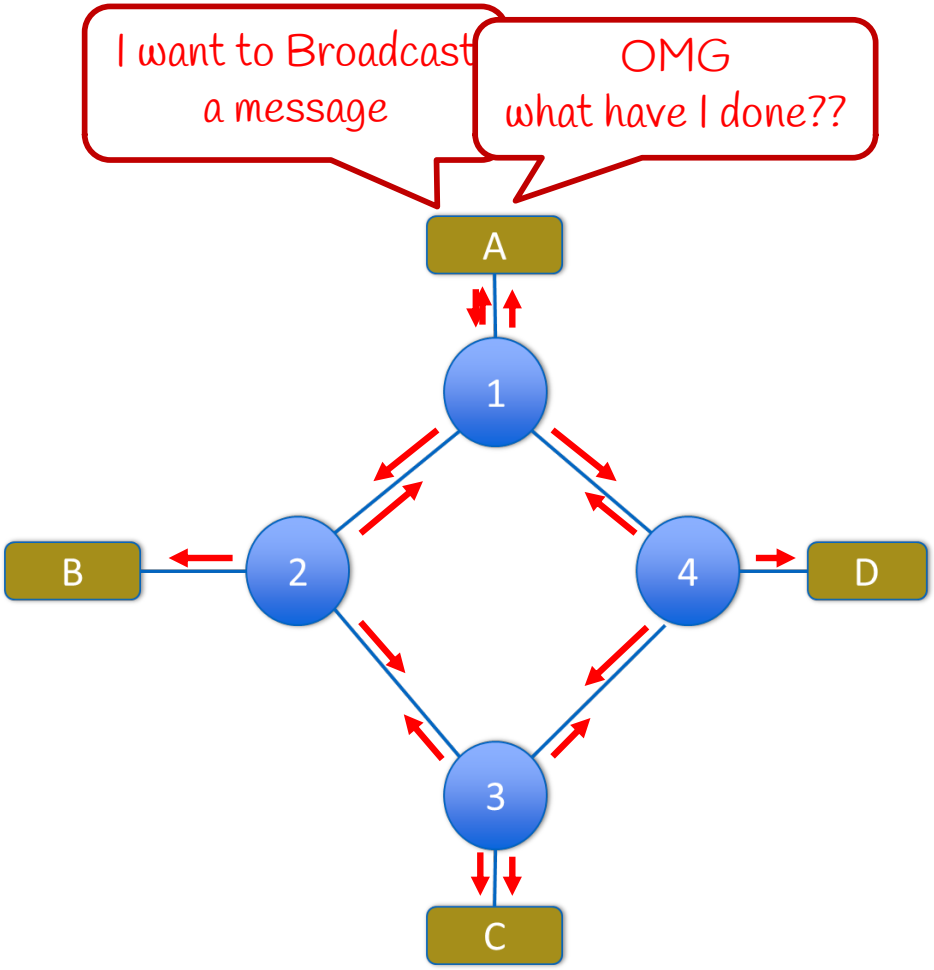
forward on all but the interface on which the frame arrived

Flooding Can Lead to Loops

- Switches sometimes need to broadcast frames
  - Upon receiving a frame with an unfamiliar destination
  - Upon receiving a frame sent to the broadcast address
- Broadcasting is implemented by flooding
  - Transmitting frame out every interface
  - … except the one where the frame arrived
- Flooding can lead to forwarding loops
  - E.g., if the network contains a cycle of switches
  - Either accidentally, or by design for higher reliability

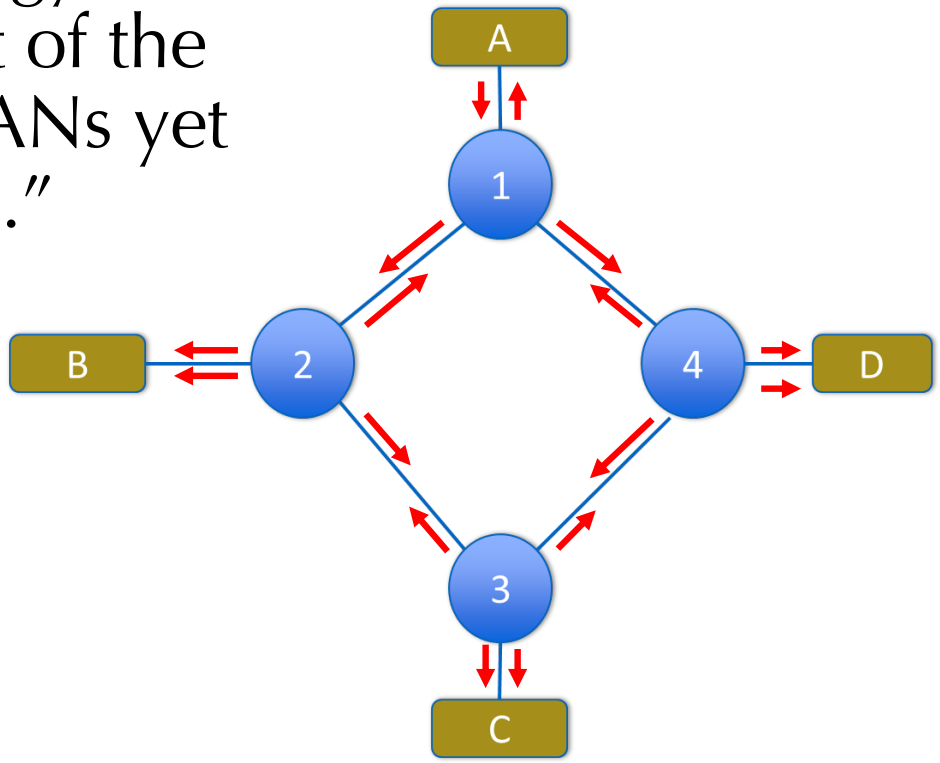
Flooding Can Lead to Loops

- Example: A wants to broadcast a message
  - A sends packet to 1
  - 1 Floods to 2 and 4
  - 2 Floods to B and 3
  - 4 Floods to D and 3
  - 3 Floods packet from 2 to C and 4
  - 3 Floods packet from 4 to C and 2
  - 4 Floods packet from 3 to D and 1
  - 2 Floods packet from 3 to B and 1
  - 1 Floods packet from 2 to A and 4
  - 1 Floods packet from 4 to B and 2
  - …



Dealing with Loops

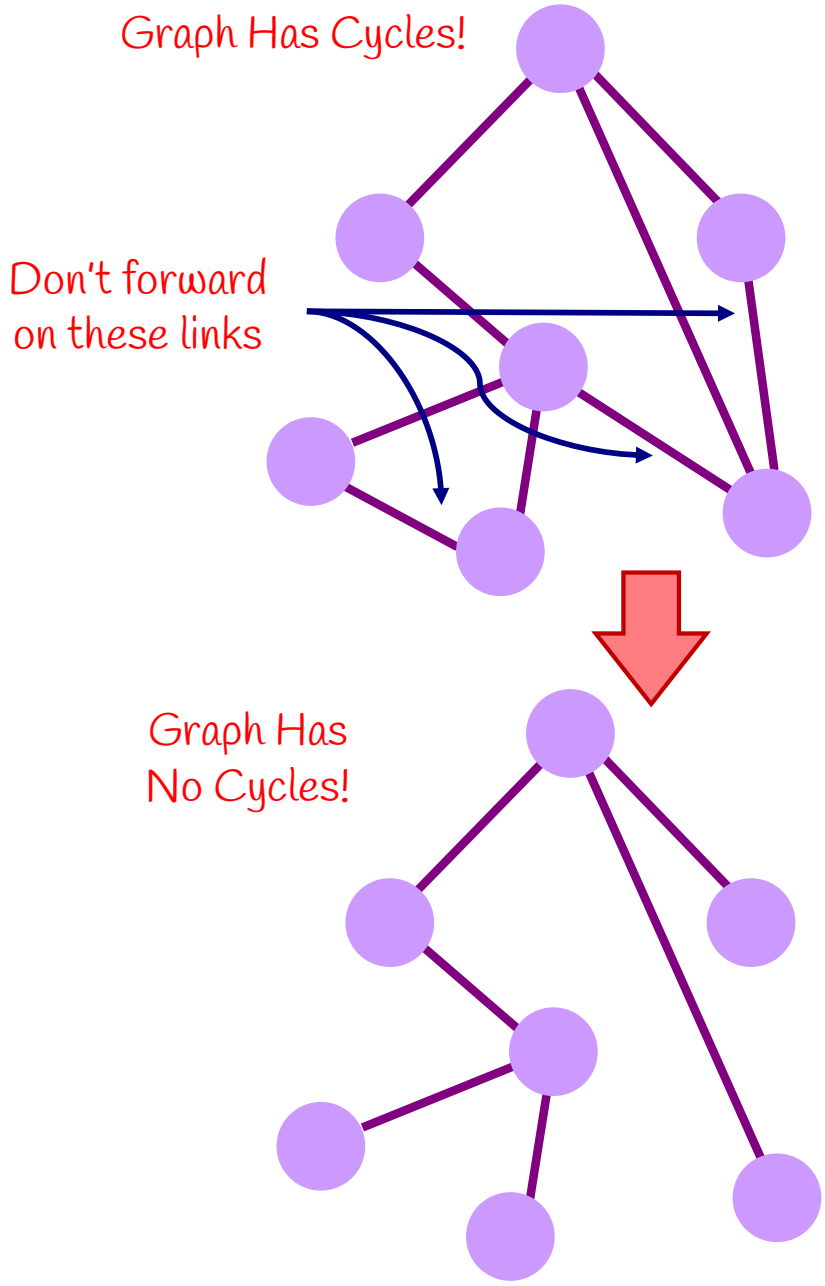
- Enters Radia Perlman
- "we have designed an algorithm that allows the extended network to consist of an arbitrary topology. The algorithm computes a subset of the topology that connects all LANs yet is loop-free (a spanning tree)."



Perlman, Radia. "An Algorithm for Distributed Computation of a Spanning Tree in an Extended LAN", *ACM SIGCOMM 1985*

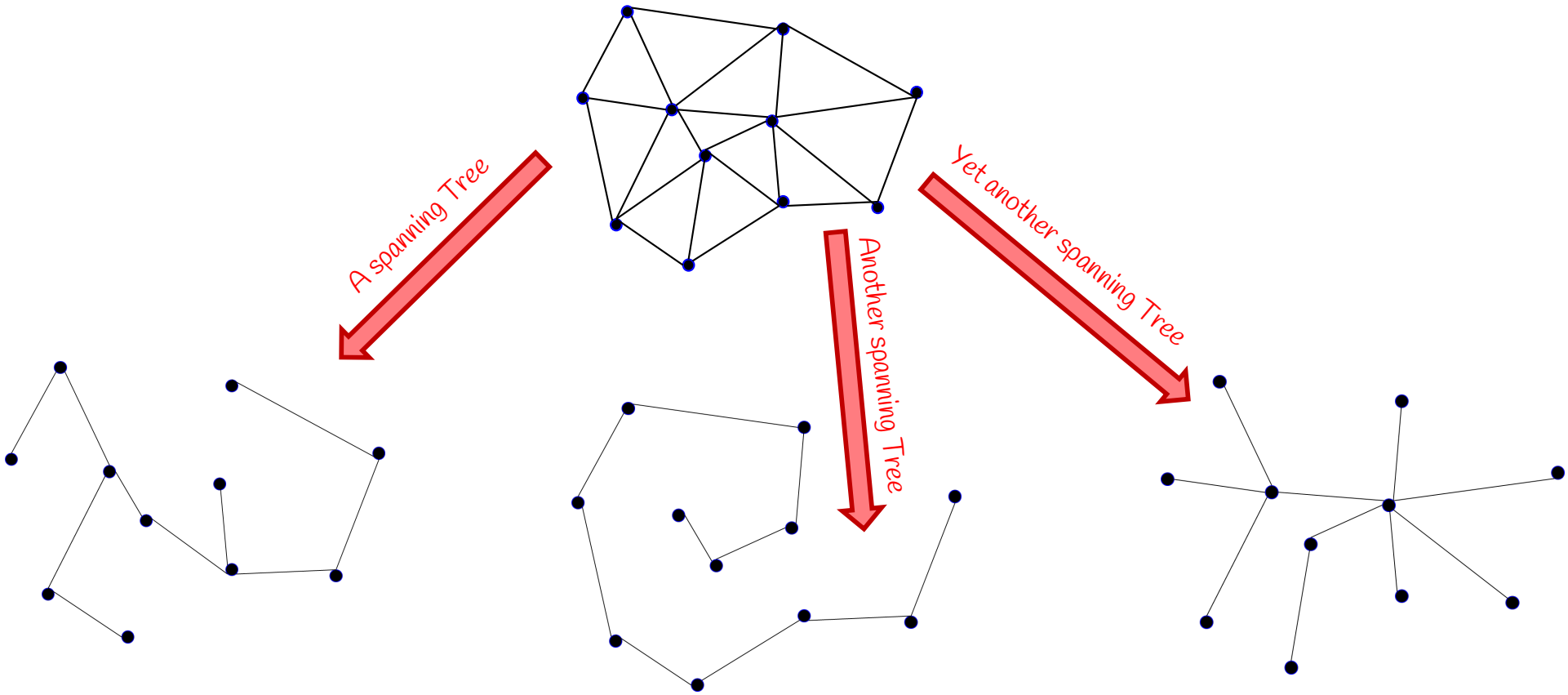
Avoiding Loops

- Ensure the topology has no loops
  - Avoid using some of the links when flooding to avoid forming a loop
- Build spanning tree
  - A sub-graph that covers all vertices but contains no cycles
  - Links not in the spanning tree are not used in forwarding frames
- Only one path to destinations on spanning trees
  - So don't have to worry about loops!



Multiple Spanning Trees

- Spanning Tree: A subgraph that includes all vertices but contains no cycles
- We can have multiple spanning trees

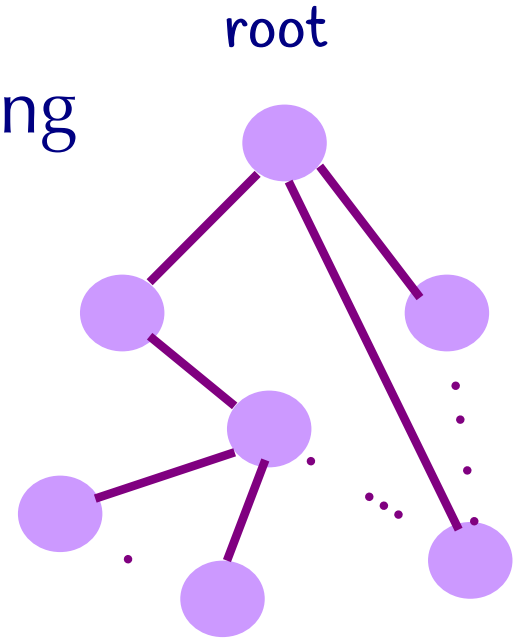


Spanning Tree Protocol (STP)

- Protocol by which switches/bridges construct a spanning tree
- Nice properties
  - Zero configuration (by operators or users)
  - Self healing
- STP should consider some constraints for backwards compatibility
  - No changes to end-hosts
  - Maintain plug-n-play aspect

Constructing a Spanning Tree

- Need a distributed algorithm
  - Switches cooperate to build the spanning tree, and adapt automatically when failures occur



- Key ingredients of the algorithm
  - Switches need to elect a "root"
  - Each switch must identify the interfaces that are on the shortest path from the root
  - And exclude interfaces that are not on the shortest path from the tree

Algorithm has Two Aspects

- Pick a root:
  - Destination to which the shortest paths go
  - Pick the one with the smallest identifier (MAC name/address)
- Compute the shortest paths to the root
  - No shortest path can have a cycle
  - Only keep the links on the shortest path
  - Break ties in some way
    - So that we only keep one shortest path from each node
- STP does both with a single algorithm

Perlman Describes Her Algorithm

- Algorhyme, by Radia Perlman:

I think that I shall never see
A graph more lovely than a tree.

A tree whose crucial property
Is loop-free connectivity.

A tree that must be sure to span
So packet can reach every LAN.

First the root must be selected.
By ID it is elected.

Least cost paths from root are traced.
In the tree, these paths are placed.

A mesh is made by folks like me,
Then bridges find a spanning tree.

Constructing a Spanning Tree

- Messages (Y, d, X)
  - Proposing Y as the root
  - From node X
  - And advertising a distance d between X and Y
- Switches elect the node with smallest identifier as root
  - Y in messages
- Each switch determines if a link is on its shortest path to the root
  - If not, excludes it from the tree

Steps in Spanning Tree Algorithm

- Initially, each switch proposes itself as the root
  - Example: switch X announces (X, 0, X)

- Switches update their view of the root and their distance to the root
  - At each switch Z, whenever a message (Y, d, X) is received from X
    - If Y's id < current root:
      - set root = Y;
      - set shortest-distance-to-root = d + 1
    - If Y's id = current root and shortest-distance-to-root > d + 1:
      - set shortest-distance-to-root = d + 1

- If root changed or shortest-distance-to-root changed:
  - Send all neighbors message (Y, shortest-distance-to-root, Z)

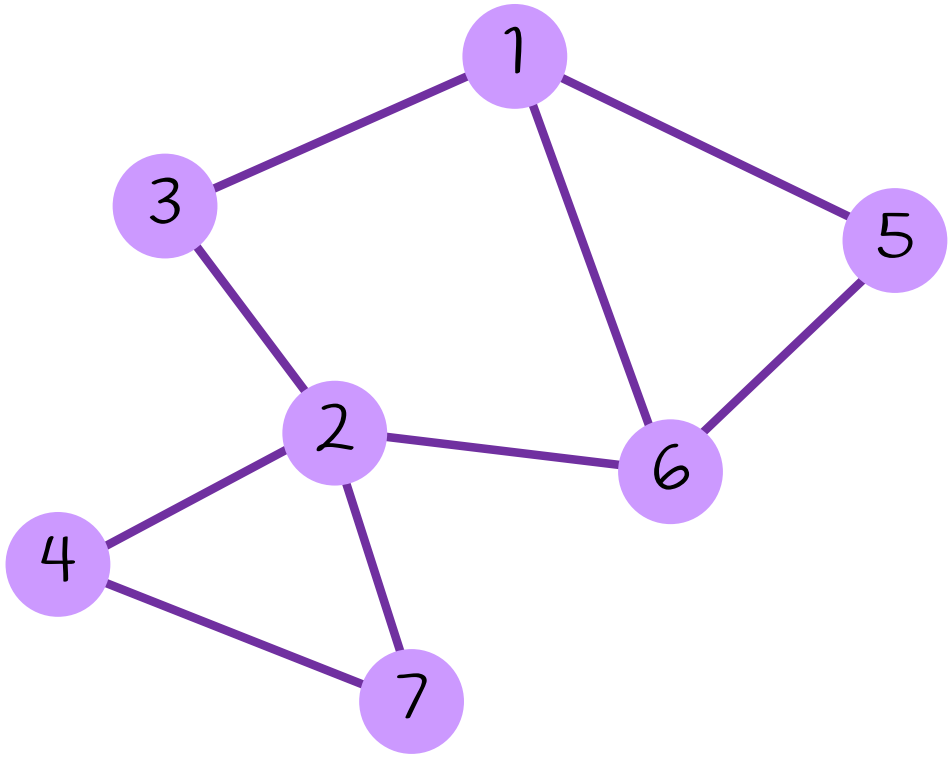
- Repeat until no more change
  - Identify interfaces not on a shortest path to the root
  - … and exclude them from the spanning tree

Breaking Ties

- When there are multiple shortest paths to the root:
  - Choose the path via neighbor switch with the smallest identifier
- One could use any tie-breaking system
  - This is just an easy one to remember and implement

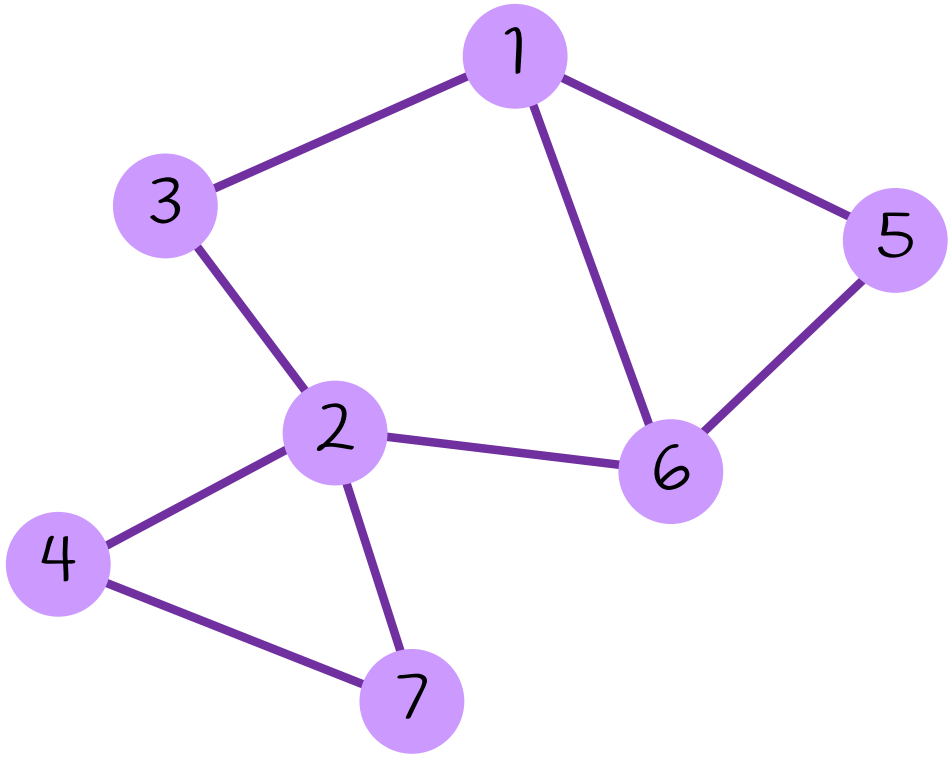
STP Example

- Let's run STP on this example
  - We assume all links have "distance" 1
  - Easy to generalize



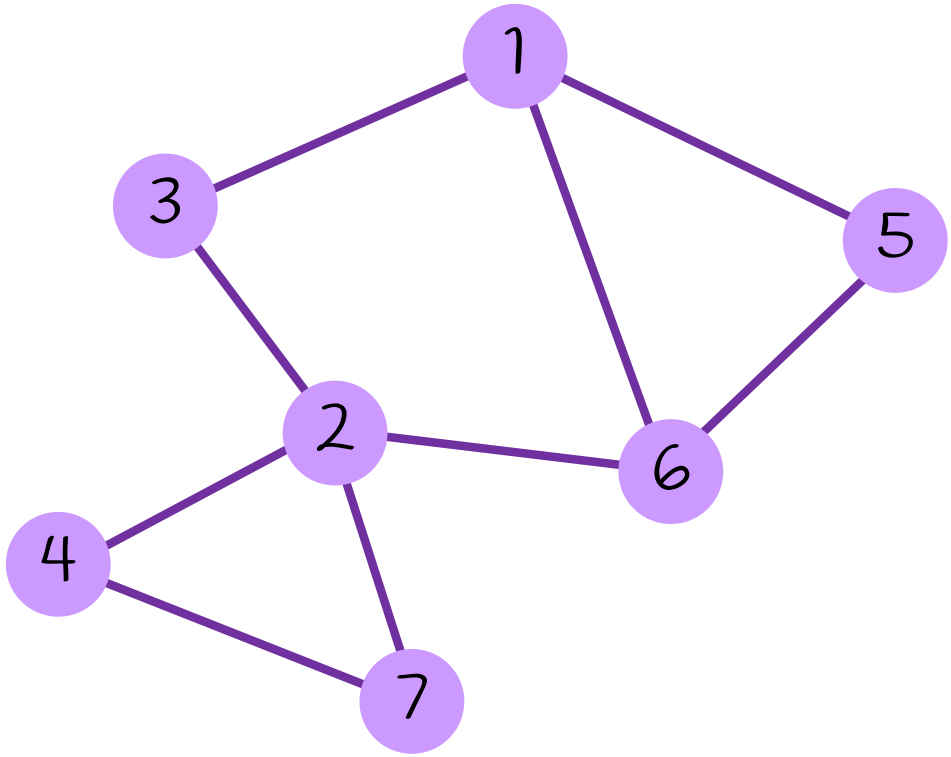
STP Example, Round 1

Current view	Receive	Send
1 (1, 0, 1)		(1, 0, 1)
2 (2, 0, 2)		(2, 0, 2)
3 (3, 0, 3)		(3, 0, 3)
4 (4, 0, 4)		(4, 0, 4)
5 (5, 0, 5)		(5, 0, 5)
6 (6, 0, 6)		(6, 0, 6)
7 (7, 0, 7)		(7, 0, 7)



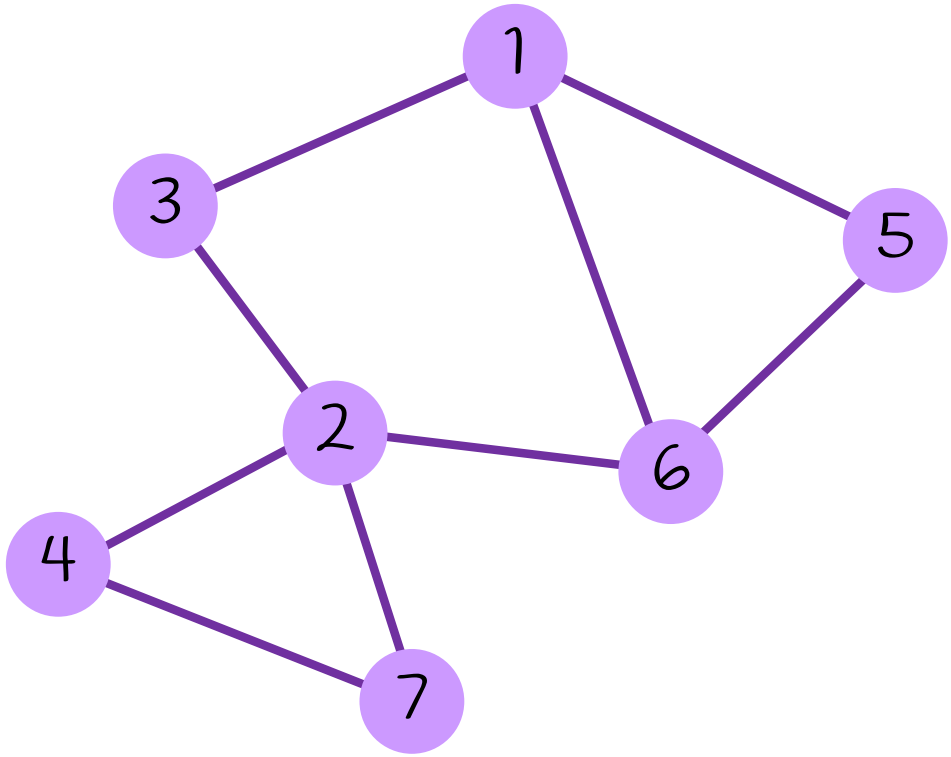
STP Example, Round 2

Current view	Receive	Send
1 (1, 0, 1)	(3, 0, 3), (5, 0, 5), (6, 0, 6)	
2 (2, 0, 2)	(3, 0, 3), (4, 0, 4), (6, 0, 6), (7, 0, 7)	
3 (3, 0, 3)	(1, 0, 1), (2, 0, 2)	**(1, 1, 3)**
4 (4, 0, 4)	(2, 0, 2), (7, 0, 7)	**(2, 1, 4)**
5 (5, 0, 5)	(1, 0, 1), (6, 0, 6)	**(1, 1, 5)**
6 (6, 0, 6)	(1, 0, 1), (2, 0, 2), (5, 0, 5)	**(1, 1, 6)**
7 (7, 0, 7)	(2, 0, 2), (4, 0, 4)	**(2, 1, 7)**



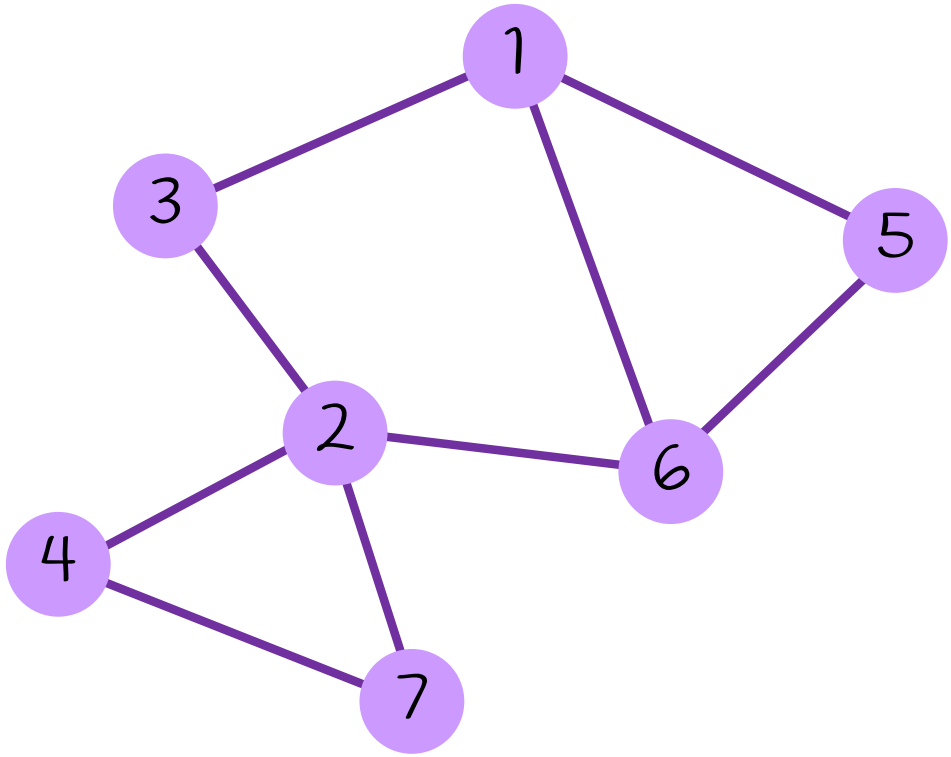
STP Example, Round 3

Current view	Receive	Send
1 (1, 0, 1)	(1, 1, 3), (1, 1, 5), (1, 1, 6)	
2 (2, 0, 2)	(1, 1, 3), (2, 1, 4), (1, 1, 6), (2, 1, 7)	**(1, 2, 2)**
3 (1, 1, 3)		
4 (2, 1, 4)	(2, 1, 7)	
5 (1, 1, 5)	(1, 1, 6)	
6 (1, 1, 6)	(1, 1, 5)	
7 (2, 1, 7)	(2,1, 4)	



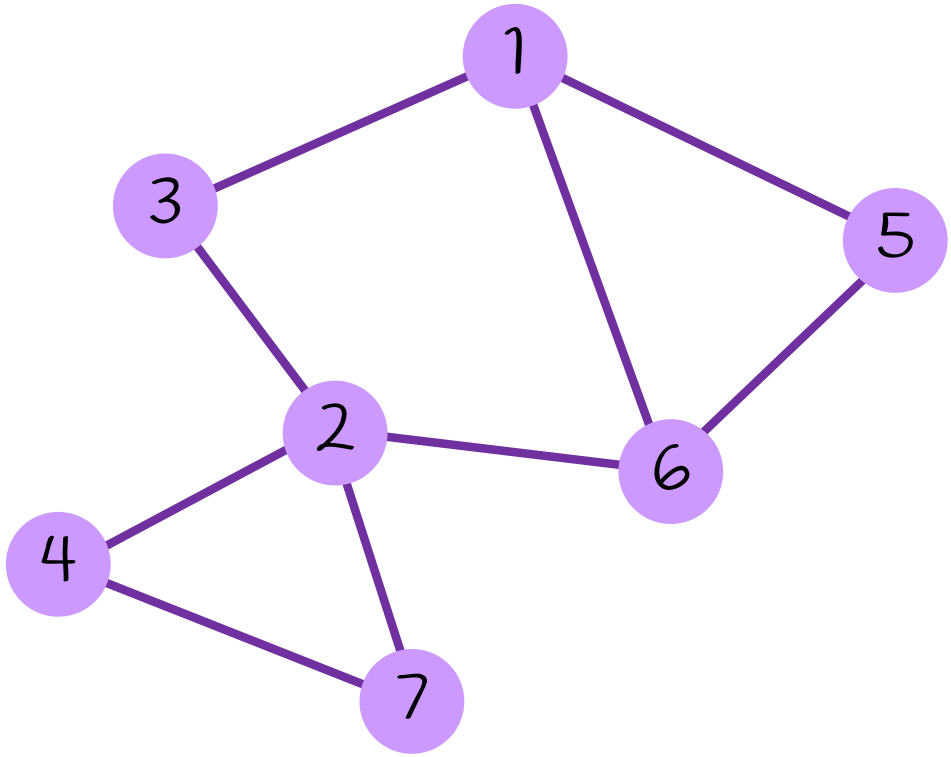
STP Example, Round 4

Current view	Receive	Send
1 (1, 0, 1)		
2 (1, 2, 2)		
3 (1, 1, 3)	(1, 2, 2)	
4 (2, 1, 4)	(1, 2, 2)	**(1, 3, 4)**
5 (1, 1, 5)		
6 (1, 1, 6)	(1, 2, 2)	
7 (2, 1, 7)	(1, 2, 2)	**(1, 3, 7)**

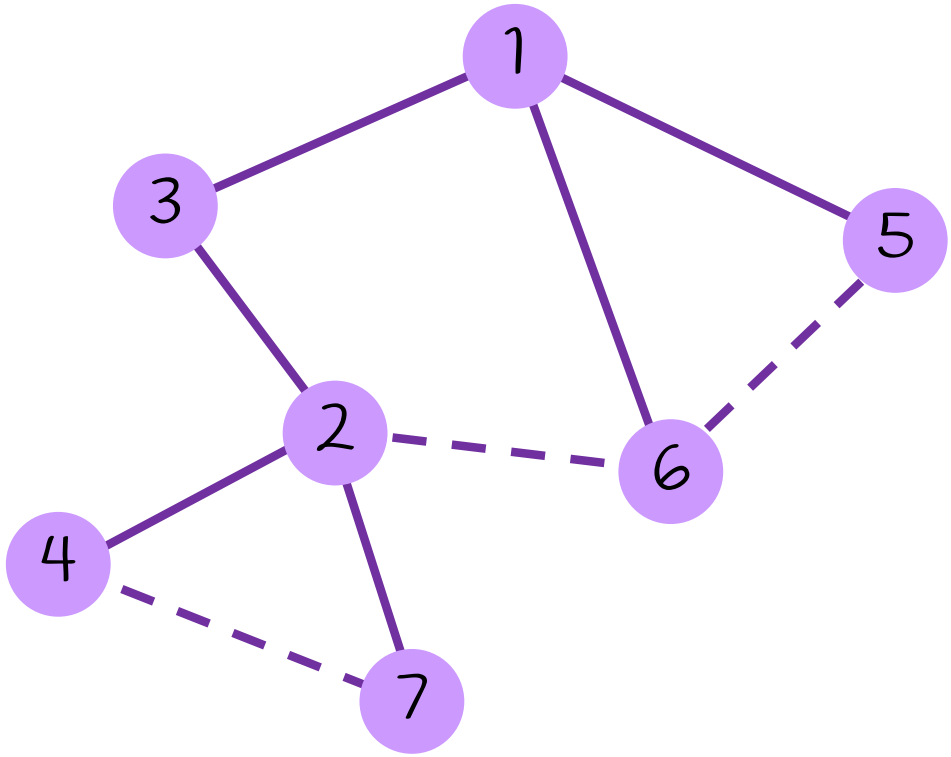


STP Example, Round 5

Current view	Receive	Send
1 (1, 0, 1)		
2 (1, 2, 2)	(1, 3, 4), (1, 3, 7)	
3 (1, 1, 3)		
4 (1, 3, 4)	(1, 3, 7)	
5 (1, 1, 5)		
6 (1, 1, 6)		
7 (1, 3, 7)	(1, 3, 4)	



After Round 5: We have our Spanning Tree



Robust Spanning Tree Algorithm

- Algorithm must react to failures
  - **Failure of the root node**: need to elect a new root, with the next lowest identifier
  - **Failure of other switches and links**: Need to recompute the spanning tree
- Root switch continues sending messages
  - Periodically reannouncing itself as the root
  - Other switches continue forwarding messages
- Detecting failures through timeout (soft state)
  - Switch waits to hear from others
  - Eventually times out and claims to be the root

See textbook for details and other examples.

Strengths and Weaknesses

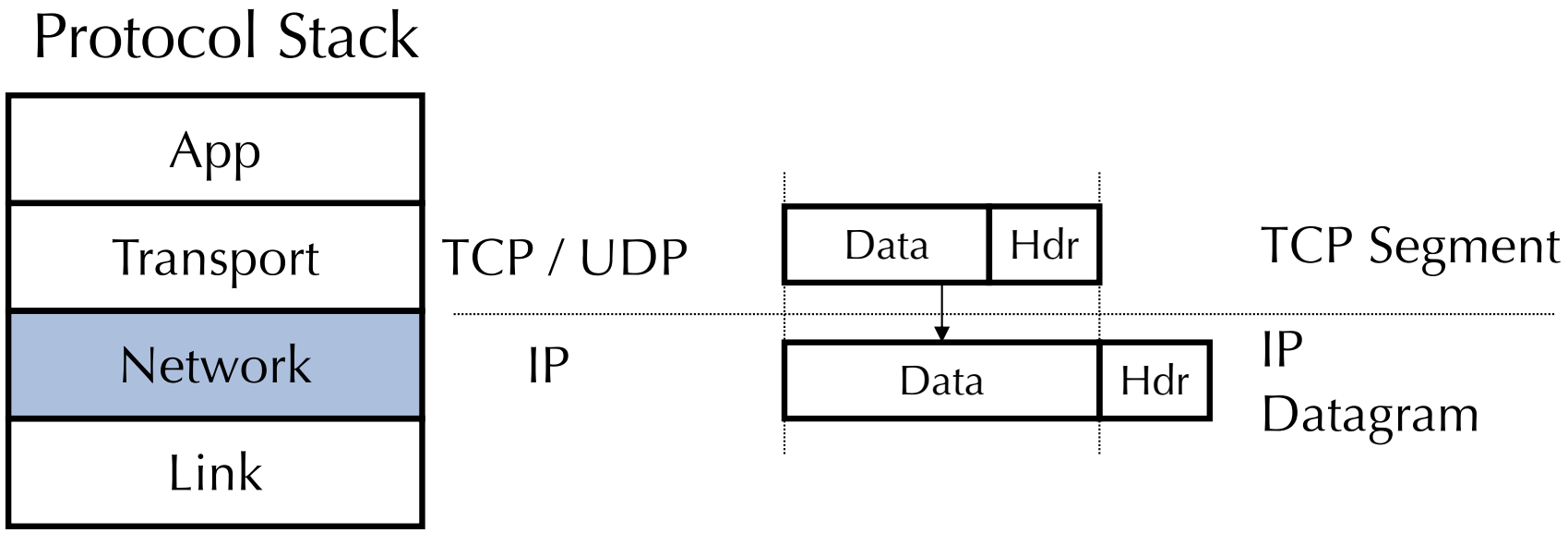
- Strengths:
  - Plug-n-Play: zero-configuration
  - Simple
  - Cheap

- Weaknesses:
  - Wasted bandwidth
  - Delay in reestablishing spanning tree
  - Slow to react to host movement (need timeout)
  - Poor predictability: depending on location of root and traffic pattern

Part II – The Internet Protocol (IP)

- IP: The Internet Protocol
  - Service characteristics
  - The IP Datagram format
  - IP addresses
  - Classless Inter-Domain Routing (CIDR)
  - An aside: Turning names into addresses (DNS)

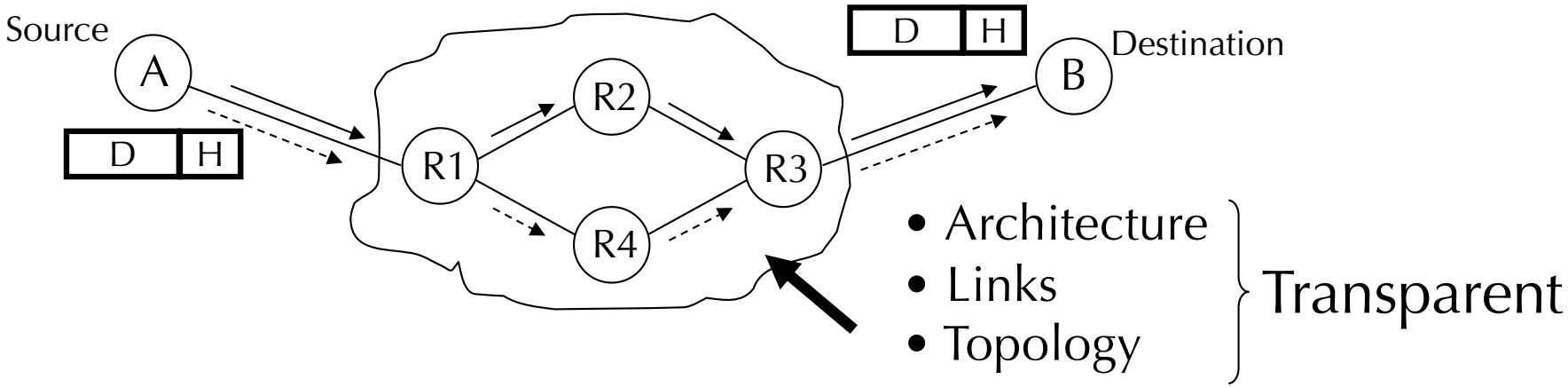
The Internet Protocol (IP)



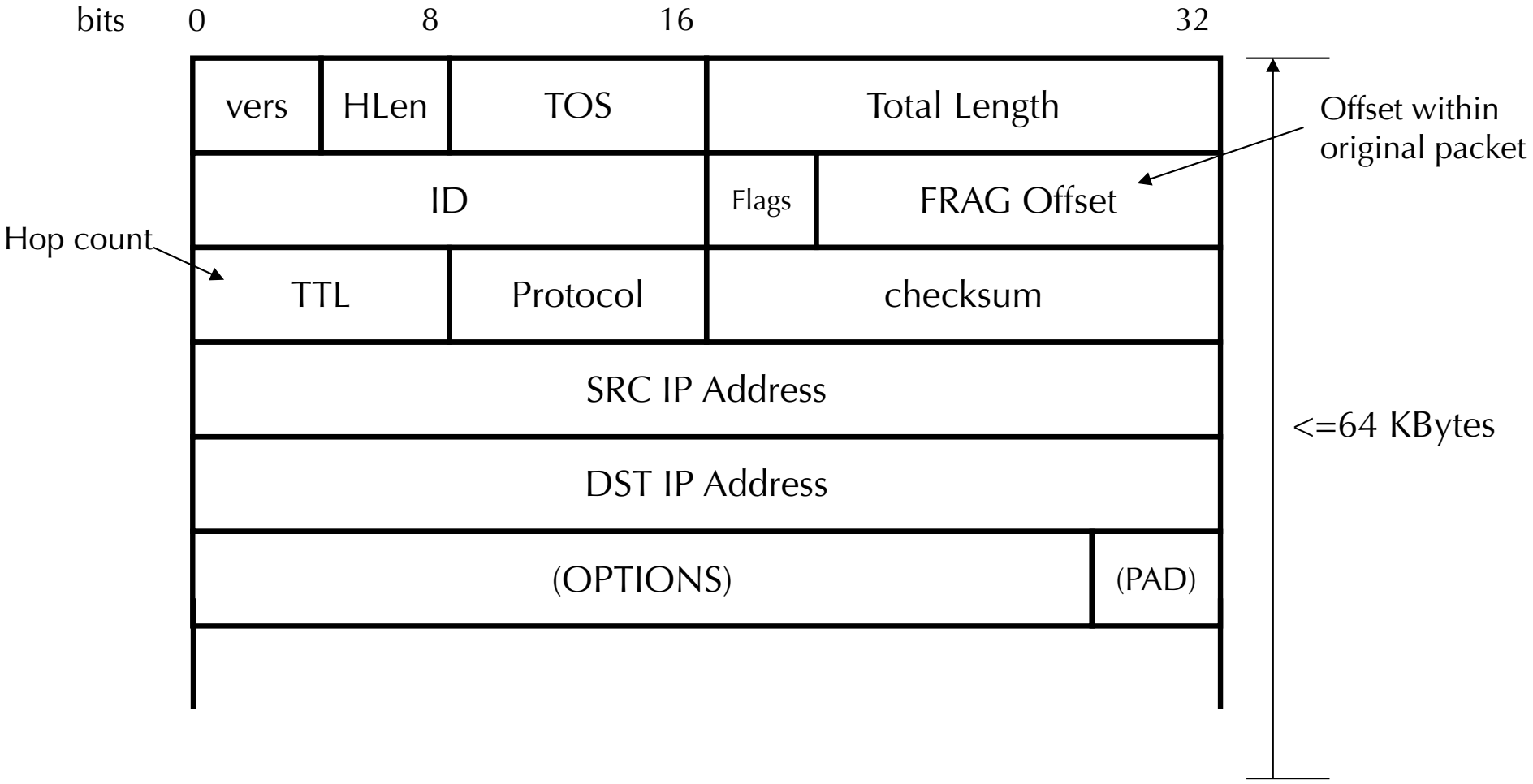
The Internet Protocol (IP)

Characteristics of IP

- CONNECTIONLESS: mis-sequencing
- UNRELIABLE: may drop packets…
- BEST EFFORT: … but only if necessary
- DATAGRAM: individually routed



The IP Datagram



IP Addresses

- IP (Version 4) addresses are 32 bits long
  - Usually represented in a dotted decimal notation
    - 1.2.3.4 (each 8 bits are represented as a decimal number)
- Every interface has a unique IP address:
  - A computer might have two or more IP addresses
  - A router has many IP addresses
- IP addresses are assigned statically or dynamically (e.g., DHCP)
- IP (Version 6) addresses are 128 bits long

Are Hosts on the Same Network?

- Assume we want to send a packet from 1.2.3.4 (A) to 1.2.10.2 (B).

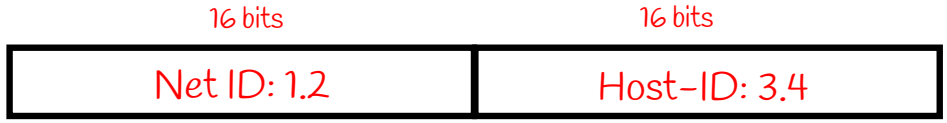
- Two possibilities:
  - If A and B are on the same network:
    - No need to route, A can directly send the packet over link layer to B
  - If A and B are NOT on the same network:
    - A must send the packet to a router that can forward it towards B.

Question.
How can we tell if A and B are on the same network?

Are Hosts on the Same Network?

- Each IP address is divided into two parts:
  - Network ID
  - Host ID

- E.g., in 1.2.3.4, we can have
  - Net-ID: 1, Host-ID: 2.3.4
  - Net-ID: 1.2, Host-ID: 3.4
  - …



Question.
How can we tell the boundary between Net-ID and Host-ID?

IP Addressing

- We In early days of the Internet
  - We had predefined classes (Class-based Addressing)
  - E.g., Class A had 8 bits of network ID, 24 bits of host ID
    - How to identify?
      - Class A addresses always started with 0.

- Very rigid and inflexible
  - We had 128 Class A network (each with 2^{24} hosts)

- Classless Inter-Domain Routing (CIDR) was introduced to provide more flexibility
  - No rigid classes
  - Identify Net-ID and Host-ID using a network mask

Network Mask (NetMask)

- A NetMask is a 32-bit number used to divide an IP address into Net-ID and Host-ID
  - A netmask consists of a sequence of 1s (representing the network part) followed by 0s (representing the host part).
- The netmask is typically shown as a dotted decimal notation
  - 255.255.255.0, meaning the first 24 bits are the Net-ID and remaining are Host-ID
- Can also be represented by a "prefix + length", e.g. 128.100.3.0/24, or just 128.100.3/24.
- Example:

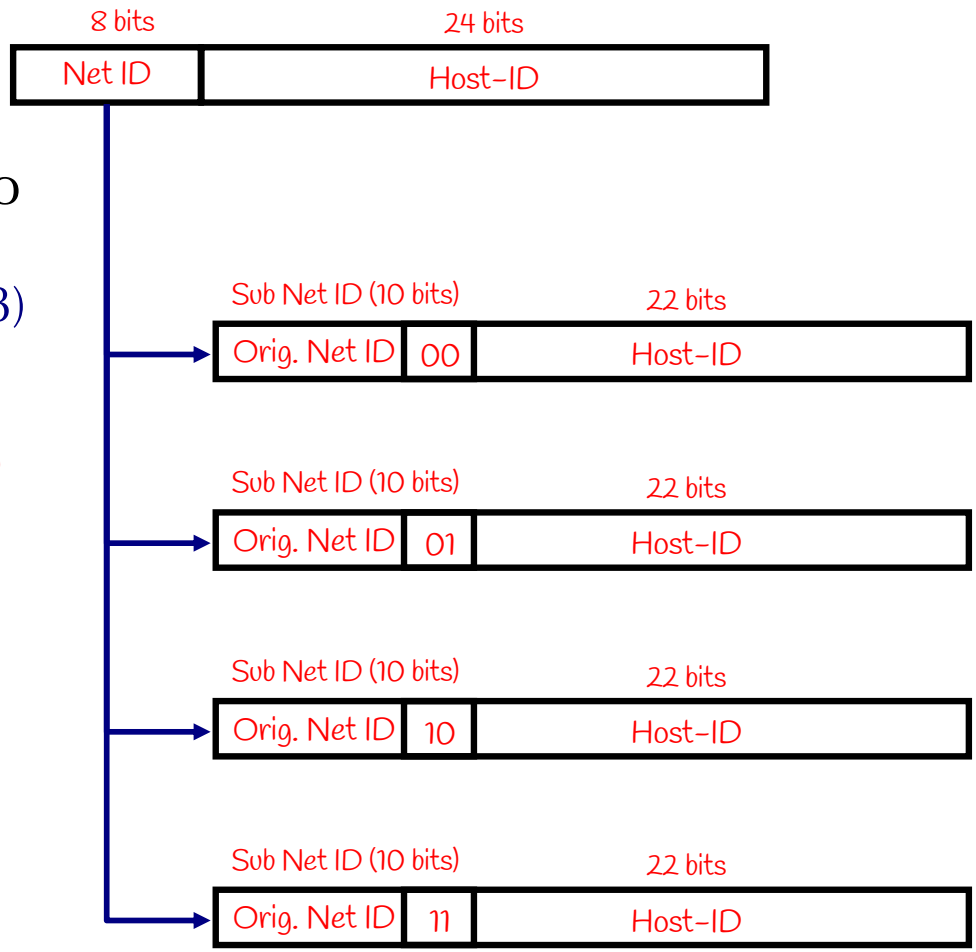
```
yganjali@apps0.cs.toronto.edu > ifconfig eth0
     Link encap:Ethernet  HWaddr 00:15:17:1C:85:30
     inet addr:128.100.3.40  Bcast:128.100.3.255  Mask:ffffff00
     UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
```

Subnetting and Supernetting

- Using flexible Net-ID vs. Host-ID gives flexibility in networks.

- **Subnetting**: break your network into smaller networks:
  - E.g., UofT can break its address (1/8) to multiple subnets: one subnet for DCS, another for ECE, …
  - 1/8 → 1.0/10 + 1.64/10 + 1.128/10 + 1.192/10

- **Supernetting**: combine networks into bigger ones:
  - E.g., DCS can combine the IP addresses of Systems Lab and Graphics Lab departments into one larger network
  - 1.2.128/17 + 1.2.0/17 → 1.2/16



Subnetting and supernetting allow hiding details of internal networks from the rest of the world. → simplify the network

Detour: Map Computer Names to IP addresses

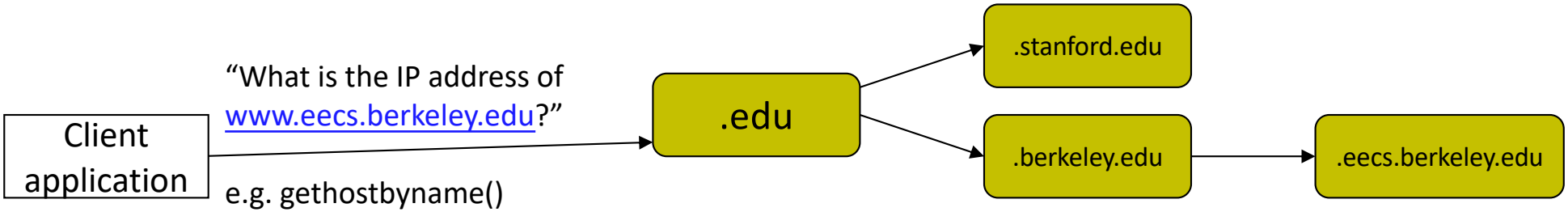
The Domain Naming System (DNS)

- Names are hierarchical and belong to a domain:
  - e.g. apps0.cs.utoronto.ca
  - Common domain names: .com, .edu, .io, .org, .net, .ca (other country-specific domain,).
  - Top-level names are assigned by the Internet Corporation for Assigned Names and Numbers (ICANN).
  - A unique name is assigned to each organization.
- DNS Client-Server Model
  - DNS maintains a hierarchical, distributed database of names.
  - Servers are arranged in a hierarchy.
  - Each domain has a “root” server.
  - An application needing an IP address is a DNS client.

Mapping Computer Names to IP addresses

The Domain Naming System (DNS)

- A DNS Query
  - Client asks local server.
  - If local server does not have address, it asks the root server of the requested domain.
  - Addresses are cached in case they are requested again.



Example: On CDF machines, try "host www.eecs.berkeley.edu"

An Aside – Error Reporting (ICMP) and traceroute

On DCS servers try: *traceroute* www.google.com

```
 1  router.cs.toronto.edu (128.100.3.254)  0.372 ms  0.369 ms  0.367 ms
 2  10.9.128.5 (10.9.128.5)  0.411 ms  0.508 ms  0.507 ms
 3  10.9.128.13 (10.9.128.13)  0.624 ms  0.637 ms  0.731 ms
 4  10.96.4.25 (10.96.4.25)  0.852 ms  0.843 ms 10.96.7.33 (10.96.7.33)  0.684 ms
 5  10.96.7.30 (10.96.7.30)  0.536 ms 10.96.4.30 (10.96.4.30)  0.521 ms  0.637 ms
 6  10.16.128.10 (10.16.128.10)  0.919 ms 10.17.128.10 (10.17.128.10)  1.078 ms  0.977 ms
 7  google.ip4.torontointernetxchange.net (206.108.34.6)  1.175 ms ut-hub-utoronto2-if-
    internet.gtanet.ca (205.211.94.133)  2.232 ms  1.830 ms
 8  205.211.94.25 (205.211.94.25)  2.330 ms  2.319 ms *
 9  google.ip4.torontointernetxchange.net (206.108.34.6)  1.463 ms 192.178.98.189
    (192.178.98.189)  1.464 ms  1.163 ms
10  172.253.69.115 (172.253.69.115)  1.164 ms 172.253.69.113 (172.253.69.113)  1.153 ms
    192.178.98.45 (192.178.98.45)  2.134 ms
11  172.253.69.113 (172.253.69.113)  1.418 ms 172.253.69.115 (172.253.69.115)  1.088 ms
    yyz10s17-in-f4.1e100.net (142.251.33.164)  1.179 ms
```

An Aside – Error Reporting (ICMP) and traceroute

Internet Control Message Protocol

- Used by a router/end-host to report some types of error:
  - E.g. Destination Unreachable: packet can't be forwarded to/towards its destination.
  - E.g. Time Exceeded: TTL reached zero, or fragment didn't arrive in time. traceroute uses this error to its advantage.
- An ICMP message is an IP datagram, and is sent back to the source of the packet that caused the error.

Summary

- Shuttling data from one link to another
  - Bits, frames, packets, …
  - Repeaters/hubs, bridges/switches, routers, …
- Key ideas in switches
  - Cut-through switching
  - Self learning of the switch table
  - Spanning trees
- Internet Protocol
  - Addresses, subnets, CIDR
  - DNS, Traceroute, ICMP