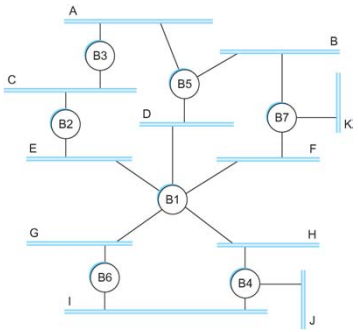
Bridges and LAN Switches

- Can the bridge learn this information by itself?
  - Yes – Learning Bridge
- How?
  - Each bridge inspects the source address in all the frames it receives
  - Record the information at the bridge and build the table
  - When a bridge first boots, this table is empty
  - Entries are added over time
  - A timeout is associated with each entry
  - The bridge discards the entry after a specified period of time
    - To protect against cases where a host moves from one network to another
- If the bridge receives a frame that is addressed to host not currently in the table
  - Forward the frame out on all other ports

Bridges and LAN Switches

- Strategy works fine if the extended LAN does not have a loop in it
- Why? Frames potentially loop through the extended LAN forever



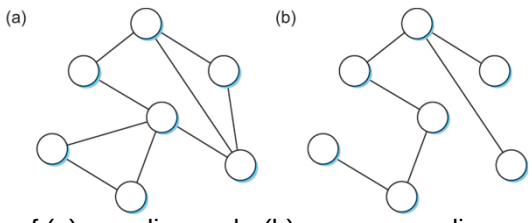
- Bridges B1, B4, and B6 form a loop

Bridges and LAN Switches

- How does an extended LAN come to have a loop in it?
  - Network is managed by more than one administrator
    - For example, it spans multiple departments in an organization
    - It is possible that no single person knows the entire configuration of the network
    - A bridge that closes a loop might be added without anyone knowing
  - Loops are built into the network to provide redundancy in case of failures
- Solution
  - Distributed Spanning Tree Algorithm

Spanning Tree Algorithm

- Think of the extended LAN as being represented by a graph that possibly has loops (cycles)
- A spanning tree is a sub-graph of this graph that covers all the vertices but contains no cycles
  - Spanning tree keeps all the vertices of the original graph but throws out some of the edges



Example of (a) a cyclic graph; (b) a corresponding spanning tree.

Spanning Tree Algorithm

- Developed by Radia Perlman at Digital
- A protocol used by a set of bridges to agree upon a spanning tree for a particular extended LAN
- IEEE 802.1 specification for LAN bridges is based on this algorithm
- Each bridge decides the ports over which it is and is not willing to forward frames
  - In a sense, it is by removing ports from the topology that the extended LAN is reduced to an acyclic tree
  - It is even possible that an entire bridge will not participate in forwarding frames

Spanning Tree Algorithm

- Algorithm is dynamic
  - The bridges are always prepared to reconfigure themselves into a new spanning tree if some bridges fail
- Main idea
  - Each bridge selects the ports over which they will forward the frames

Spanning Tree Algorithm

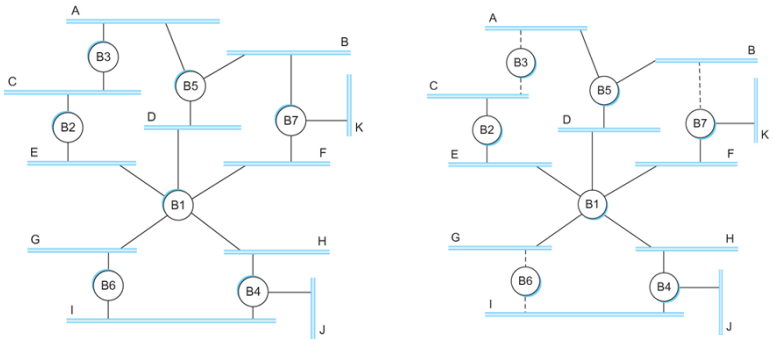
- Spanning Tree Algorithm selects ports as follows:
  - Each bridge has a unique identifier
    - B1, B2, B3,…and so on.
  - Elect the bridge with the smallest id as the root of the spanning tree
  - The root bridge always forwards frames out over all of its ports
  - Each bridge computes the shortest path to the root and notes which of its ports is on this path
    - This port is selected as the bridge's preferred path to the root
  - Finally, all the bridges connected to a given LAN elect a single *designated bridge* that will be responsible for forwarding frames toward the root bridge

Spanning Tree Algorithm

- Each LAN's designated bridge is the one that is closest to the root
- If two or more bridges are equally close to the root,
  - Then select bridge with the smallest id
- Each bridge is connected to more than one LAN
  - So it participates in the election of a designated bridge for each LAN it is connected to.
  - Each bridge decides if it is the designated bridge relative to each of its ports
  - The bridge forwards frames over those ports for which it is the designated bridge

Spanning Tree Algorithm

- B1 is the root bridge
- B3 and B5 are connected to LAN A, and B5 is the designated bridge
- B5 and B7 are connected to LAN B, and B5 is the designated bridge
- B3 and B2 are connected to LAN C, and B2 is the designated bridge



Spanning Tree Algorithm

- Initially each bridge thinks it is the root, so it sends a configuration message on each of its ports identifying itself as the root and giving a distance to the root of 0
- Upon receiving a configuration message over a particular port, the bridge checks to see if the new message is *better* than the current best configuration message recorded for that port
- The new configuration is better than the currently recorded information if
  - It identifies a root with a smaller id or
  - It identifies a root with an equal id but with a shorter distance or
  - The root id and distance are equal, but the sending bridge has a smaller id

Spanning Tree Algorithm

- If the new message is better than the currently recorded one,
  - The bridge discards the old information and saves the new information
  - It adds 1 to the distance-to-root field for the new information
- When a bridge receives a configuration message indicating that it is not the root bridge (that is, a message from a bridge with smaller id)
  - The bridge stops generating configuration messages on its own
  - Only forwards configuration messages from other bridges after adding 1 to the distance field

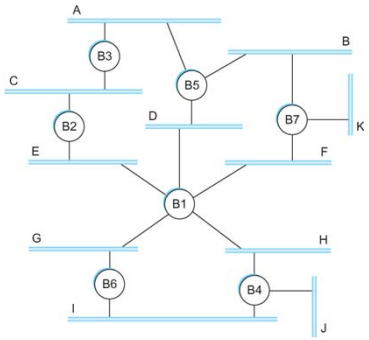
Spanning Tree Algorithm

- When a bridge receives a configuration message that indicates it is not the designated bridge for that port

  => a message from a bridge that is closer to the root or equally far from the root but with a smaller id
  - The bridge stops sending configuration messages over that port
- When the system stabilizes,
  - Only the root bridge is still generating configuration messages.
  - Other bridges are forwarding these messages only over ports for which they are the designated bridge

Spanning Tree Algorithm

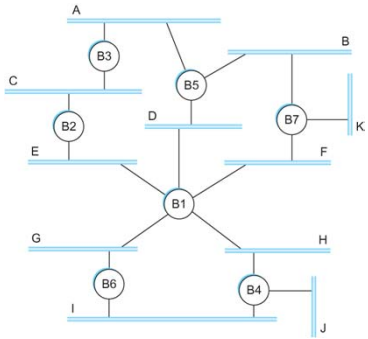
- Consider the situation when the power had just been restored to the building housing the following network



- All bridges would start off by claiming to be the root

Spanning Tree Algorithm

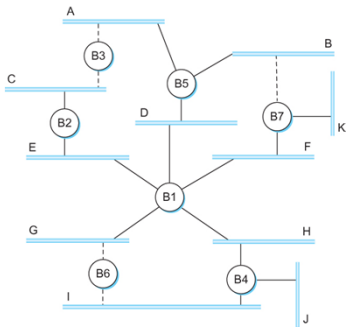
- Denote a configuration message from node X in which it claims to be distance d from the root node Y as (Y, d, X)



- Consider the activity at node B3

Spanning Tree Algorithm

- B3 receives (B2, 0, B2)
- Since 2 < 3, B3 accepts B2 as root
- B3 adds 1 to the distance advertised by B2 and sends (B2, 1, B3) to B5
- Meanwhile B2 accepts B1 as root because it has the lower id and it sends (B1, 1, B2) toward B3
- B5 accepts B1 as root and sends (B1, 1, B5) to B3
- B3 accepts B1 as root and it notes that both B2 and B5 are closer to the root than it is.
  - Thus B3 stops forwarding messages on both its interfaces
  - This leaves B3 with both ports not selected



Spanning Tree Algorithm

- Even after the system has stabilized, the root bridge continues to send configuration messages periodically
  - Other bridges continue to forward these messages
- When a bridge fails, the downstream bridges will not receive the configuration messages
- After waiting a specified period of time, they will once again claim to be the root and the algorithm starts again
- Note: Although the algorithm is able to reconfigure the spanning tree whenever a bridge fails, it is not able to forward frames over alternative paths for the sake of routing around a congested bridge

Spanning Tree Algorithm

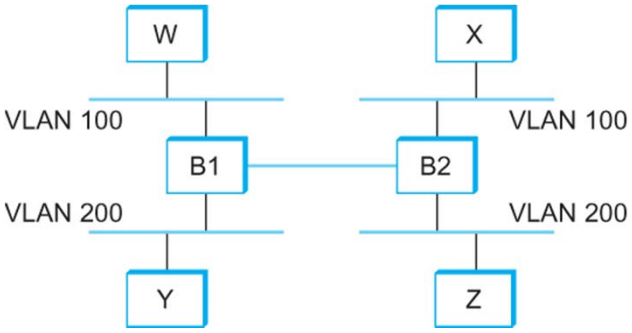
- Broadcast and Multicast
  - Forward all broadcast/multicast frames
    - Current practice
  - Learn when no group members downstream
  - Accomplished by having each member of group G send a frame to bridge multicast address with G in source field

Spanning Tree Algorithm

- Limitation of Bridges
  - Do not scale
    - Spanning tree algorithm scales linearly- no provision for a hierarchical structure
    - Broadcast does not scale
  - Do not accommodate heterogeneity
    - Networks must all have the same format for addresses

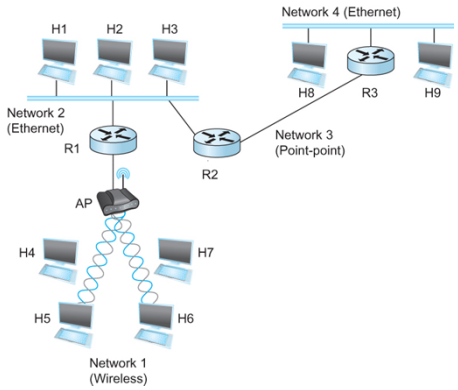
Spanning Tree Algorithm

- Virtual LAN-increases scalability
  - Broadcast packets are given a VLAN ID



Internetworking

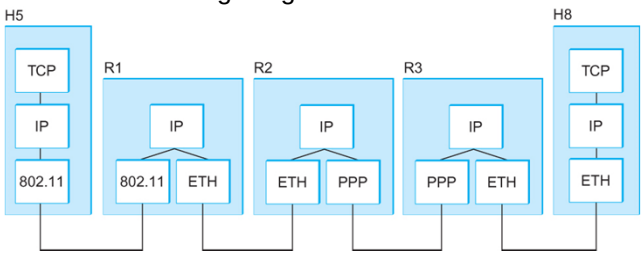
- What is internetwork
  - An arbitrary collection of networks interconnected to provide some sort of host-host to packet delivery service



A simple internetwork where H represents hosts and R represents routers

Internetworking

- What is IP
  - IP stands for Internet Protocol
  - Key tool used today to build scalable, heterogeneous internetworks
  - It runs on all the nodes in a collection of networks and defines the infrastructure that allows these nodes and networks to function as a single logical internetwork



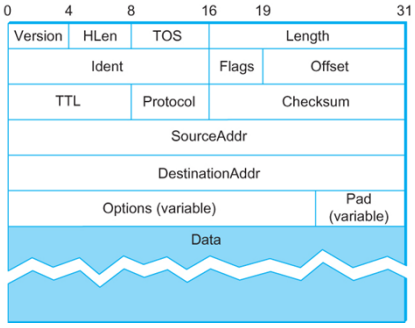
A simple internetwork showing the protocol layers

IP Service Model

- Packet Delivery Model
  - Connectionless model for data delivery
  - Best-effort delivery (unreliable service)
    - packets are lost
    - packets are delivered out of order
    - duplicate copies of a packet are delivered
    - packets can be delayed for a long time
- Global Addressing Scheme
  - Provides a way to identify all hosts in the network

Packet Format (IPv4)

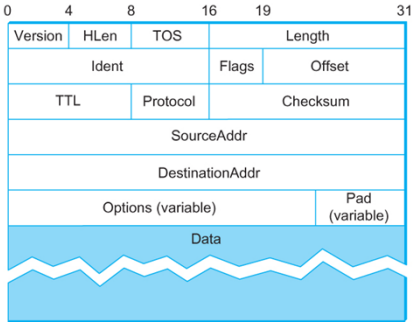
- Version (4 bits): currently 4
- Hlen (4 bits): number of 32-bit words in header. 5 words typically without Options
- TOS (8 bits): type of service (not widely used)
- Length (16 bits): number of bytes in this datagram – maximum datagram size is 65535 Bytes
- Ident (16 bits): used by fragmentation
- Flags/Offset (16 bits): used by fragmentation



<Return>

Packet Format (IPv4)

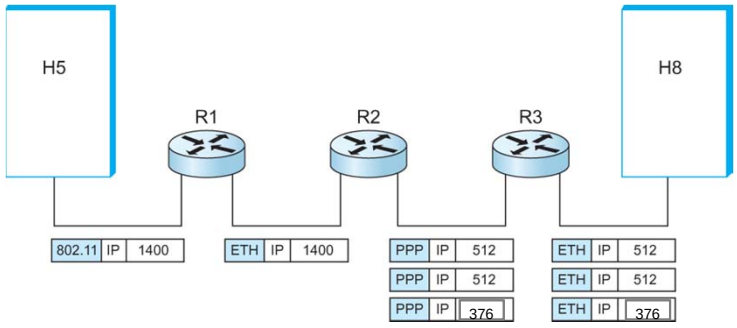
- TTL (8 bits): number of hops this datagram has traveled – typically it is a countdown timer
- Protocol (8): demux key (TCP=6, UDP=17)-higher level protocol using the packet info
- Checksum (16 bits): of the header only (IP checksum algorithm)
- DestAddr & SrcAddr (32 bits each, IPv6 has 128)
- Options/Pad (32 bits) – rarely used



IP Fragmentation and Reassembly

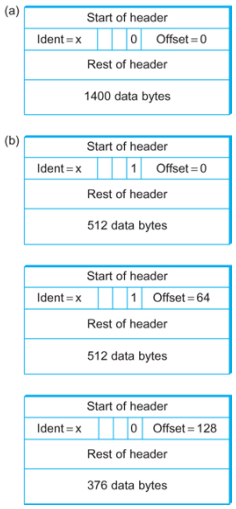
- Each network has some MTU (Maximum Transmission Unit)
  - Ethernet (1500 bytes), FDDI (4500 bytes)
- Strategy
  - Fragmentation occurs in a router when it receives a datagram that it wants to forward over a network which has (MTU < received datagram)
  - Reassembly is done at the receiving host
  - All the fragments carry the same identifier in the *Ident* field
  - Fragments are self-contained datagrams
  - Fragments re-encapsulate each IP datagram
  - IP does not recover from missing fragments

IP Fragmentation and Reassembly



IP datagrams traversing the sequence of physical networks

IP Fragmentation and Reassembly



(a) Total packet size = 1420 bytes
(20 bytes for header, 1400 for data)

(b) Total size of packets = 20*3 + 1400 = 1460 Bytes
(3 headers @ 20 bytes each plus data)

Header fields used in IP fragmentation. (a) Unfragmented packet; (b) fragmented packets.

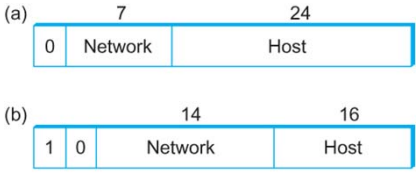
IP Fragmentation and Reassembly

- Offset is measured in 8 byte chunks
  - Offset field has 13 bits – measured in 8 byte chunks
  - Length field has 16 bits – measured in Bytes
  - Fragmented data has to be done in factors of 8 bytes
- For fragmentation, the IP-header is reproduced for each of the fragmented packets
  - Data is split between the fragments
  - Ident, Flags and Offset fields are used with appropriate information
  - Length and Checksum are re-calculated based on the fragment size and IP information.
- IP fragmentation is best avoided
  - Best-effort delivery has more chances of losing part of a packet
  - Use path MTU discovery to find the smallest MTU along the path – use the smallest MTU as the size for all packets

Slide 24>

Global Addresses

- Properties
  - globally unique
  - hierarchical: network + host
  - 4 Billion IP address, half are A type, ¼ is B type, and 1/8 is C type
- Format



(a) Class A, (b) Class B, (c) Class C

- Dot notation
  - 10.3.2.4
  - 128.96.33.81
  - 192.12.69.77