Bridging

Photo from Tourism NSW

Course outline

- Administration
- Introduction to Switching
- Discrete switches
- **Distributed switches**
  - **Bridges**
  - Asynchronous Transfer Mode
  - Modern reservation protocols: MPLS and Integrated Services (RSVP)
  - Differentiated Services
  - Caching

Context

So far, we've looked *inside discrete* switches (except week 4 <RH]):

- physically centralised
- haven't discussed interconnection with other switches

Now we'll look at *distributed* switching systems

How do devices acquire information about where to forward packets?

- **Bridges: Link layer devices. Learn by observing traffic**
  Being link layer devices, data units should be called "frames" but we&others often say "packets"
  Most products that are called "switches" are Ethernet switches and perform the bridging functions described in this lecture.
- **Routers**: Network layer devices. Learn by exchanging topology info
- **ATM/SS7/etc "switches"**†: Link or network layer devices. Explicitly told through signalling

† This is a very specific usage of the word, e.g. ATM switches, rather than the generic sense of the word used elsewhere in this course, e.g. "**Switch**ing Systems Design" covers bridges and routers as well as these "switches".

Resources

Varghese S 10.1 and 10.2
Keshav doesn't cover this topic.

The best book: R. Perlman: *Interconnections: Bridges, Routers, Switches and Internetworking Protocols*, 2nd edition, Addison-Wesley
Ch. 3: Bridges

A free book chapter: U. Black: *IP Routing Protocols*, Prentice-Hall
Ch. 4: http://vig.pearsoned.com/samplechapter/0130142484.pdf

R. Seifert: *The Switch Book: The Complete Guide to LAN Switching Technology*, Wiley
Ch. 2 (Transparent Bridges) and Ch. 5 (Loop Resolution)

The authoritative standard: IEEE: Standard 802.1d: Part 3: Media Access Control (MAC) Bridges

Cisco Nexus context

- "Existing Layer 2 networks based on **Spanning Tree Protocol** have a number of challenges to overcome: suboptimal path selection, underutilized network bandwidth, control-plane scalability, and slow convergence."
- "Cisco Nexus 5500 platform has the hardware capability to support Cisco ® FabricPath and **IETF Transparent Interconnection of Lots of Links (TRILL)** to build scalable and highly available Layer 2 networks." "For Cisco FabricPath or TRILL, the switch ID or egress **RBridge** is looked up in the 8000-entry Layer 2 Multipathing (L2MP) table for a forwarding decision."
- "It supports a set of network technologies known collectively as **IEEE Data Center Bridging (DCB)** that increases the reliability, efficiency, and scalability of Ethernet networks."

Outline

Overview

Solitary bridges

Spanning Tree Protocol: Coordinating multiple bridges

Limitations of bridged networks

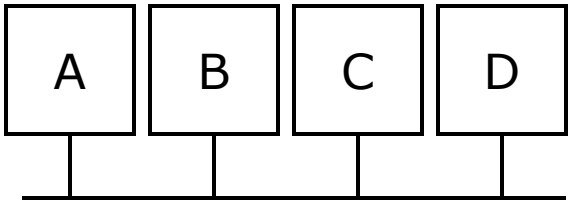
- Bridges vs routers
- Security

Overview

- Context
  - Evolution of LAN topologies
  - Evolution of LAN interconnection
- Terminology
- Goals
- Types

Evolution of LAN topologies

1980s

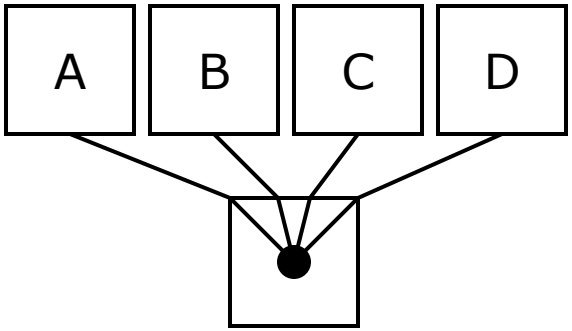


Coaxial cable snakes its way past computers.

× Maintenance is complicated by needing broad physical access.

× Data cable is distinct from twisted pair used for phones.

Early 1990s



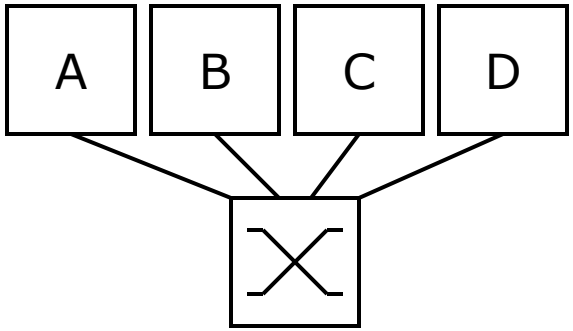
Computers connected

√ using twisted pair

√ to wiring closet – central point for maintenance.

× Hub [HE> in closet provides shared medium: Single "collision domain" limits scalability.

Late 1990s



Switches become cheaper, and replace hubs in closets, improving performance.

√ Separate "collision domain" (full duplex link) for each station.
Indeed, collisions won't occur on p2p link, so no longer even need MAC.

21st Century: Links become wireless and topology returns to original distributed shared medium!

Why interconnect?

Given that *Local* Area Networks already exist, we want to:

- Increase geographical span
- Increase number of stations
- Join LANs that use different technologies
  - old vs new
  - features needed for particular contexts, e.g. mobility, priorities
- Isolate stations and their traffic:
  - Performance
  - Fault tolerance
  - Security

The network that results from using bridges to interconnect multiple LANs, is sometimes called a “catenet” (concatenated network). c.f. “internet” for interconnection at the network layer.

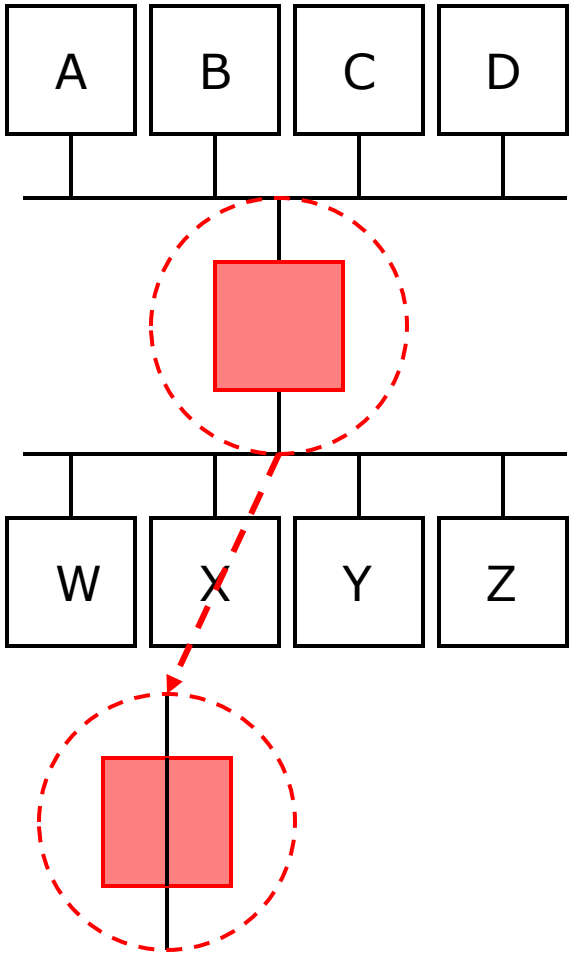
Hubs for physical interconnection

- Hubs provide *physical* interconnection, allowing original physical signal to propagate to all ports.
- Repeater may even regenerate signal, allowing geographical extension.

However,

× Barriers to increasing number of nodes:
  × All nodes share medium with Medium Access Control. For Ethernet, all part of same "Collision Domain"
  × All traffic goes everywhere

× Hub can't match LANs using different technology, e.g. stations on one fast LAN (100 Mb/s Ethernet) seeking to transmit to stations on another slow LAN (10Mb/s Ethernet) would need to detect that destination is slow before transmitting => barrier to incremental deployment of new technologies.

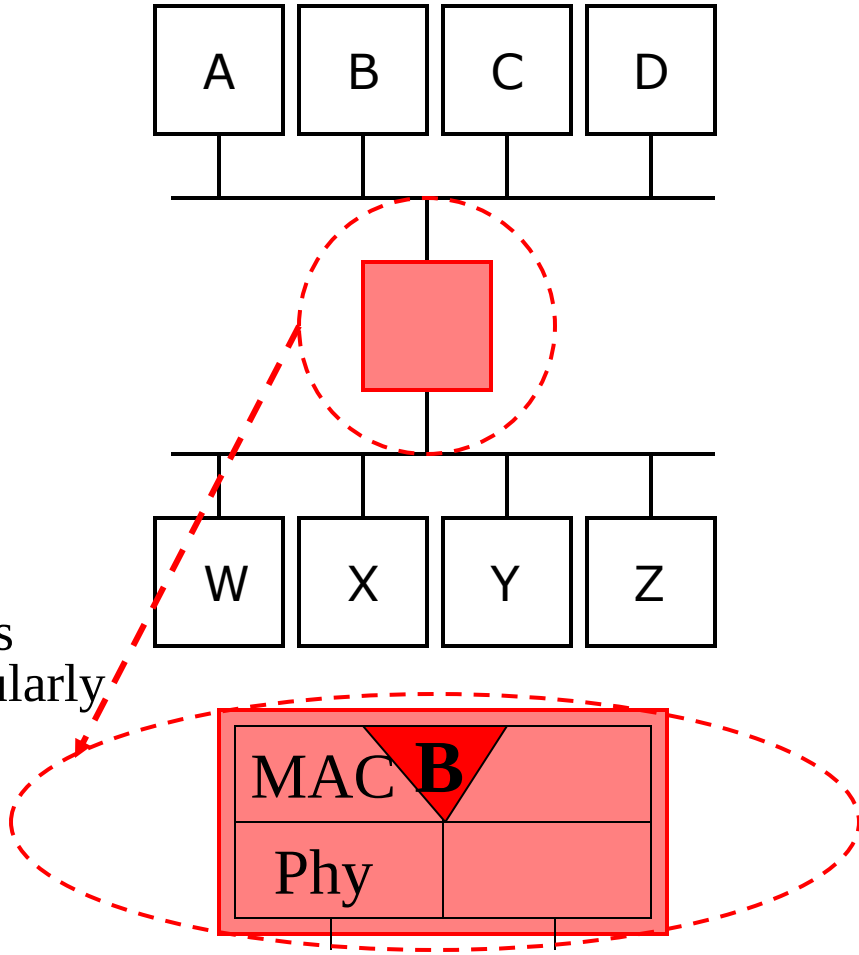


Bridges for link layer interconnection

Bridges operate on *link layer frames*

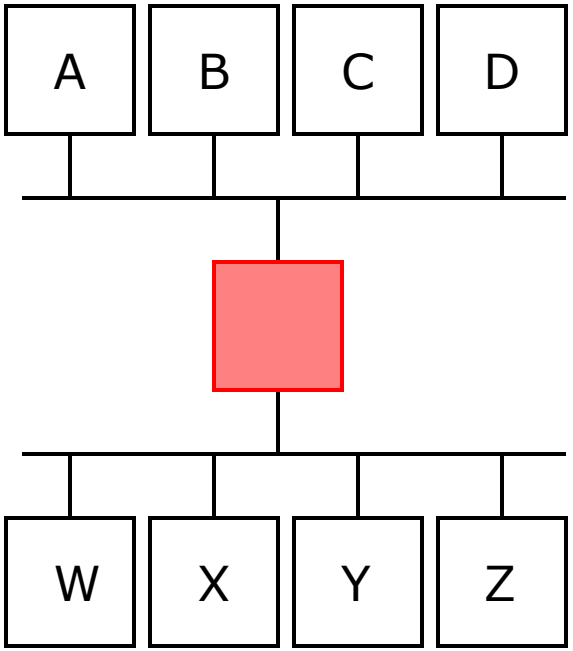
- √ Separate MAC process for each port

  For Ethernet:
  - This is called "segmentation" of the Collision Domain.
  - The extreme, when only one station is connected to each port, is called "microsegmentation".
- √ Can filter frames so that frames only propagate where required.
- √ Ports can use different technologies.
- √ Can operate with varied network layers (e.g. IPv4, v6, Appletalk, etc) – particularly important in 1980s when bridges first appeared.



Evolution of LAN interconnection

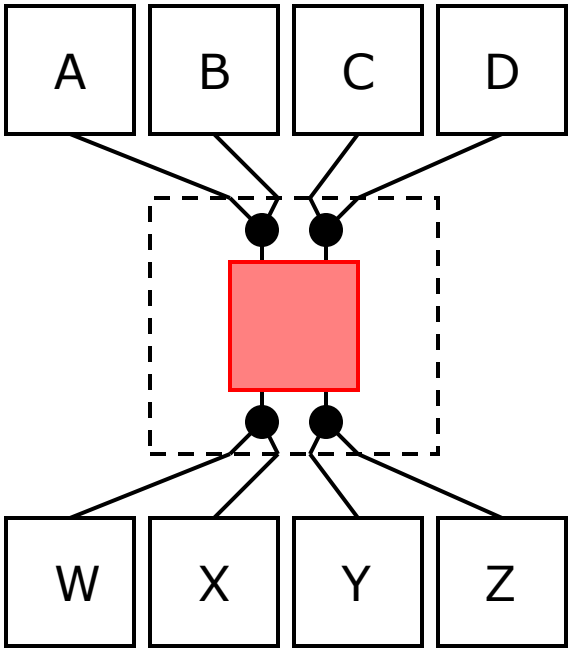
1980s



"Bridges" have few (e.g. 2) ports.

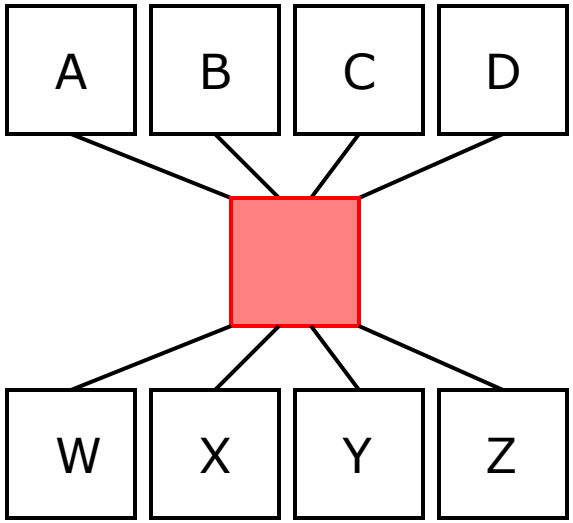
21st C: Wireless "Access Points" act as simple bridges, connecting wireless LAN to Ethernet.

Early 1990s



Bridge ports become cheaper, and hubs can be replaced by "multi-port bridges" in wiring closet, transparently wrt users.

Late 1990s



Ports become so cheap (relative to management costs) that all users can have their own port.

"Switches": Products with many high-speed ports

Summary of terminology

Marketing terms:

- 1980s: "Bridges": few ports
- Early 1990s: Multi-port bridges
- Late 1990s: "Switches" have even more ports, and operate at "wire speed"/"media rate"<1UK] (line interfaces are the bottleneck, rather than port processors or fabric)

All support shared media access to ports, so are functionally equivalent; only different in scale (# ports†, speed‡)

Our terms:

- Switch: Any device with multiple ports that aims to direct unicast traffic only to one output port that leads to the destination.
- Bridge: A type of "switch" implemented at the link layer; specific techniques for learning paths and forwarding appropriately.

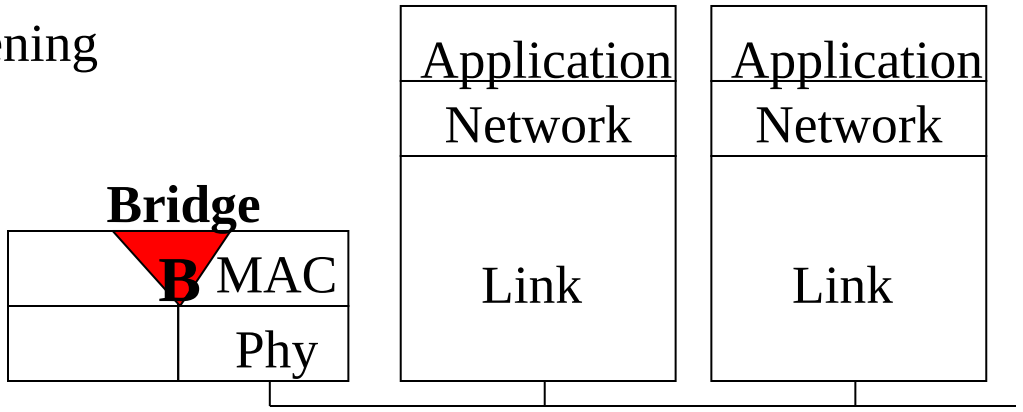
† "The most important difference between a bridge and switch is that bridges usually have a small number of interfaces (that is, 2-4), whereas switches may have dozens of interfaces." [Kurose & Ross]
‡ "Marketers chose to call their products switches primarily to differentiate them from the (more primitive) [non-wire speed] bridges of old." [Seifert, p. 150]
"bridges traditionally contained an actual CPU that did store-and-forward switching in software. But since all modern bridges and switches contain special integrated circuits for switching, the difference between a switch and bridge is more a marketing issue than a technical one." [Tanenbaum, p. 328]

Implications of shared media

Historically, bridges were designed to interconnect shared media.

A frame that a bridge receives may have already been received by its destination on the shared medium.
=> filtering

Multiple stations on shared medium & bridge can learn addresses by listening to local communication.
=> promiscuous listening



Types of bridges

1. Need end-stations be aware of bridges?

Transparent bridges:

- **End-stations don't need to know about bridges.**
- **Bridges act independently of end-stations.**
- **Most common form, especially with Ethernet – our focus.**

Source routing bridges:

- Non-transparent: end-stations need to know about bridges in order to specify route through the "catenet" <DC].
- Less common, used with Token Ring.

2. Is the bridge centralised or distributed?

"Remote bridges" may be distributed, with ports widely separated (e.g. different office locations) and interconnected by proprietary link

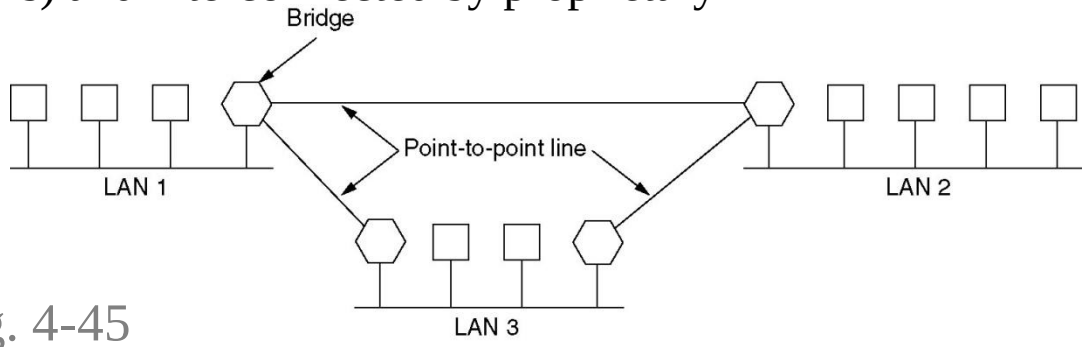


Figure from Tanenbaum Fig. 4-45

More on bridge transparency

Being transparent to stations† is desirable because end-stations:

- can continue using legacy protocols that may not be designed for interconnection (e.g. Ethernet has no TTL field; apps may rely on LAN delivering frames in sequence)
- don't have to be configured – save time.

To achieve transparency:

- End-stations shouldn't participate in bridging
- Bridges must:
  - Forward broadcast/multicast traffic.
    Since a LAN does just that.
    => Achilles heel of bridging, and why routers can be better [LE>.
- Preserve 6 aspects of reliable transfer <6Q], in particular:
  - **Order**: Don't change order of frames flowing between a given source/destination (at a specified priority)
  - **Uniqueness**: => Spanning Tree to prevent loops [HF>

† "Bridges are not only transparent to end stations, they are transparent to each other." [Seifert, p. 71]

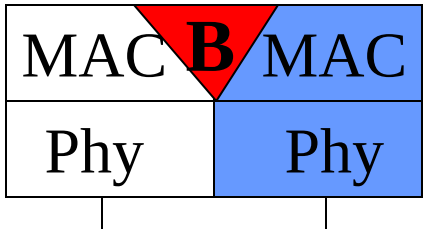
Limitations to transparency

While bridges are transparent in terms of functionality, they *can* affect performance:

- Degrade reliability: loss within bridge
- Increase throughput :-)
- Affect delay
  - ↓ access delay: fewer stations contending for access in each LAN
  - ↑ "propagation" delay: frames get delayed while waiting in bridge buffers.

Transparency when bridging heterogeneous LANs

A bridge *can*† join different LAN types
e.g. Ethernet and 802.11



How LANs may functionally differ:

Frame length requirements

- Minimum: 64B Ethernet, 34B 802.11
- Maximum: 1522B Ethernet, 2346B 802.11
  Bridge *discards* frames with inappropriate length. Either
  - Configure all source MAC layers to use lowest common denominator
  - Higher layers learn, e.g. TCP MTU discovery

Services, e.g. may lose information about priority

- when bridging from LAN that supports priority (e.g. WiFi) to one that doesn't (e.g. Ethernet).

† Despite the claims of those with a vested interest in selling competing products, e.g. "if you need … communication between dissimilar LANs, routers are necessary."
[www.**Cisco**.com/univercd/cc/td/doc/cisintwk/idg4/nd2012.htm]

Other bridge requirements

(Other than interconnection, and transparency)

Efficiency: Conserve transmission capacity, try to send unicast traffic only where needed. → learning

Automation: Learn where stations are, without manual programming. "plug and play" → learning

Robust: to physical mis-configuration, e.g. two bridges in parallel with each other → Spanning Tree

Responsive to station movement → learning and ageing

Outline

Solitary bridges

- Learning
  - Learning by source lookup
  - Flooding
  - Replacement policy
- Ageing
- Forwarding
  - not forwarding: Filtering
  - Forwarding after learning
- Complete Bridge process

Bridge process (Synopsis)

Whenever a frame arrives:

1. Update database to record direction of source
2. Filter out certain frames
3. Forward remaining frames

Database used for bridging

Each database entry contains the following information:

Address (48b)	Know? (1b)	Port ($\log_2 P$ b)	Used? (2b)	Seen? (2b)

- Address of a station
- Know port to reach that address?
  - Port to reach that address

} "Forwarding database"

- Recently used as a destination address?
  - "Recent" relative to timing epochs (e.g. every 5 minutes) when these flags get reset. 2b = 1b for current epoch + 1b for previous epoch. Neither set => no recent activity.
- Recently seen as a source address?

+ possibly VLAN & priority information

The database is implemented using standard packet classifiers <16U*]

Learning by source lookup

Process:

1. **Listen "promiscuously"** (indiscriminately) to *all* frames, not just those traversing the bridge or destined to the bridge (e.g. management)
2. **Observe the *Source Address*** of these frames. Assuming that the links are bidirectional, source station should be reachable through port from which its traffic arrives.

Shouldn't learn SA from erroneous frames, since SA may be errored, leading to later mis-direction of frames.

=> when using cut-through forwarding <V0], switch should defer learning until it has verified the frame CRC.

Flooding

A bridge may not know directions to all stations (wrt ports), e.g.:

- when first started, and automated mechanisms for learning haven't completed.
- because of limited database size: it may not be able to remember *all* stations in a large LAN.

When a bridge receives a frame destined to a station with unknown direction, the bridge will "**flood**" the frame: Forward it on all active ports other than that from which it was received.

- Sufficient: One of these ports will lead to the station (if it is reachable)
- Undesirable: Loads ports and networks that don't lead to the destination.
- Rare -> [KZ>

Learning assumes that receivers transmit

Bridges assume that receiving stations also transmit.

They *usually* do:

- With IPv4, first step in communication may be to resolve IP address into link layer address: sending station broadcasts ARP request, then receiver replies. Broadcast ARP would be flooded anyway; response will teach switch right direction, avoiding need for extra flooding.
- Destination will usually reply with feedback for error or flow control. so the bridge can learn its direction and avoid subsequent flooding.

But may not, e.g.

- multicast
- station hidden behind a "data diode"†

A silent high-volume receiver may degrade bridged LAN performance (since its traffic will be flooded)

† A security device that only allows information to flow one way. e.g. separating a monitor from a network so that it can collect frames from the network but can't be detected.

Replacement policy

The database will have limited capacity because <16U]:

- It isn't feasible to construct a database of all 2^{47}† possible stations
- Classification may be faster with fewer addresses, e.g. if implemented using a linked list

When the bridge learns that a new address is reachable through a certain port ...

... and the database is full

... then, replace entries of least recently used *destination* addresses (little penalty in flooding their information)

† 2^{47}, not 2^{48}, because 1 address bit is used to indicate multicast & multicast is often flooded

Implementing the replacement policy

- Whenever an address in the database is used as a *destination*, set its "Recently Used" flag.
- Periodically, scan the database to determine which addresses haven't been recently used. Add them to the list of entries to be replaced if the bridge learns a new address when the database is full.

Outline

Ageing†: Motivation

Even if the database could record information about all addresses in the "catenet" <DC], it shouldn't retain learned information forever:

- **Station mobility**: Mobile station may move, directing packets in old direction could create unnecessary loss & the station won't reveal its new direction by responding.
- **Topological changes**: links/bridges go down/up

† "a parameter known as *aging time* ... (Actually, the 802 spec spells the parameter "ageing" because the editor of 802.1d is British and the country seems to have a surplus of vowels.)" [Perlman, p. 50]

Ageing: Implementation

To implement ageing:

- Whenever an address in the database is matched to a *source* address, set the "Recently Seen" flag.
- Periodically, scan the database to determine which addresses haven't been recently observed. *Delete them*.
- Standard ageing period = 5 minutes.

Replacement and Ageing summary

Element:

- Which address to look up in database?
- Treatment of addresses not recently used?
- Purpose?

Replacement:

- Destination
- Add to preferential discard list. (do not delete immediately)
- Optimise performance by prioritising recently / frequently used addresses

Ageing:

- Source
- Delete entry
- Respond to topological/station location changes.

Outline

Filtering

A bridge can filter frames based on multiple criteria:

- **If destination is known to be reachable** through port from which frame arrived, then discard the frame (e.g. shared media LAN).
- **Custom filtering rules**: May discard frames in order to enforce security/policy/etc, e.g.
  - Wireless LAN: only forward traffic from known link layer addresses.
  - Prevent specific users
  - Control propagation of sensitive traffic
- **Don't forward traffic to certain multicast addresses** used to disseminate Spanning Tree info [W6>.

Forwarding

[after previous DA filtering]

if(unicast && destination is known)

Forward on port leading to destination (from database)

else

Forward on all *outgoing* ports ("flooding")

Use of addresses when forwarding

Outgoing frame retains original SA (doesn't use a bridge address) since bridge is transparent

Bridge only needs its own link layer address for:

- **MAC purposes**, e.g. token-passing protocols
- **Spanning Tree Protocol** (coming soon…). Bridge may have a link layer address for each interface; use the smallest of these as a Bridge Identifier.
- **Communication with the bridge itself**, e.g. management

Outline

Bridge process (detail – part 1)

Whenever a frame arrives:

1. **Update database to record direction of source**
   - Lookup source in database
   - If it exists
     - record the port of arrival
     - mark it as Recently Seen
   - else, create a new database entry (using free space, or replacing address least recently used as destination)
2. **Filter out certain frames**
3. **Forward remaining frames**

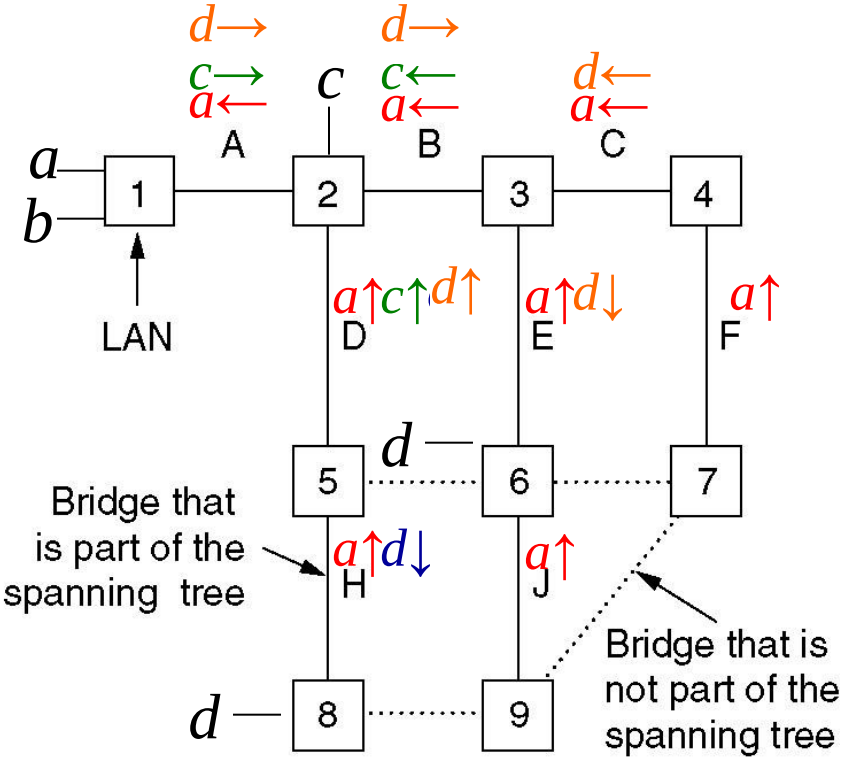
} Details on next slide

Bridge process (detail – parts 2,3)

Whenever a frame arrives:

1. **Update database to record direction of source**
2. **Filter out certain frames**
   - according to policy
   - multicast traffic used for the Spanning Tree Protocol
   - Lookup destination in database
     - Discard if output port = arrival port
3. **Forward remaining frames**
   - if multicast or output port is not known, then flood
   - else send to output port

Example of the learning process



Stations *a*, *b*, *c*, *d* are spread about.

1. Bridge databases are initially empty.
2. *a* sends to *d*.
   - Flood frame across whole tree.
   - Bridges learn direction to *a*.
3. *c* sends to *a*.
   - Only A forwards the frame.
   - Bridges A B D learn about *c*.
4. *d* sends to *c*. Bridges A and B know not to forward the frame.
5. *d* moves to LAN 6.
6. *d* sends to *a*. Ageing will remove H's stale knowledge of *d*

[Tanenbaum Fig. 4-44]

Beware of the unusual naming in this figure: squares are LANs, uppercase letters are nodes (bridges)

Q5

Outline

Spanning Tree Protocol

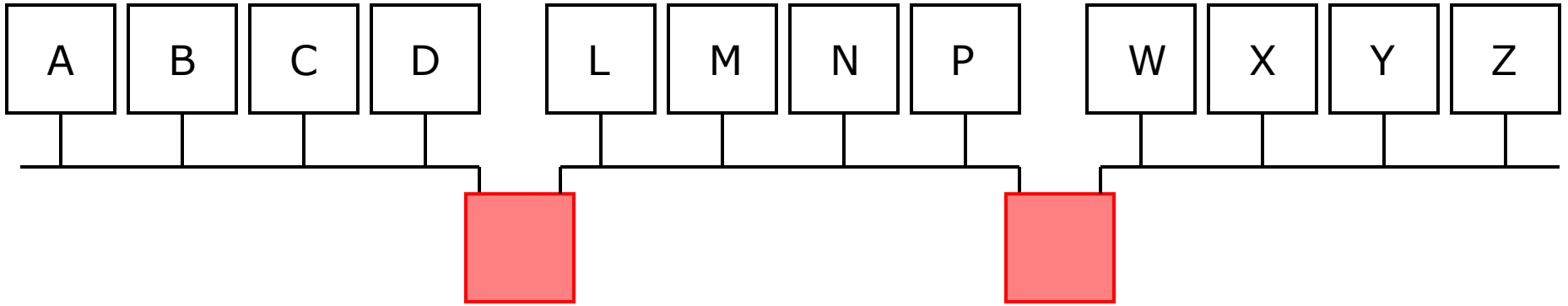
Why loops arise
- Interconnecting multiple LANs
- Other reasons (Why physical connections often loop)

Loopy problems
- What's wrong with loops?
- Some techniques for *handling* loops
- *Preventing* loops with Spanning Trees

Spanning Tree Protocol
- Info used to create a Spanning Tree
  - Comparing messages; Bridge Identifiers; Link cost; Port identifiers
- Electing a Root
- Implementation
  - An actual Message; Other Message fields; Topology Change notifications; Avoiding temporary loops

Wrapping up Spanning Tree Protocol
- Algorhyme; Bridging without the Spanning Tree Protocol

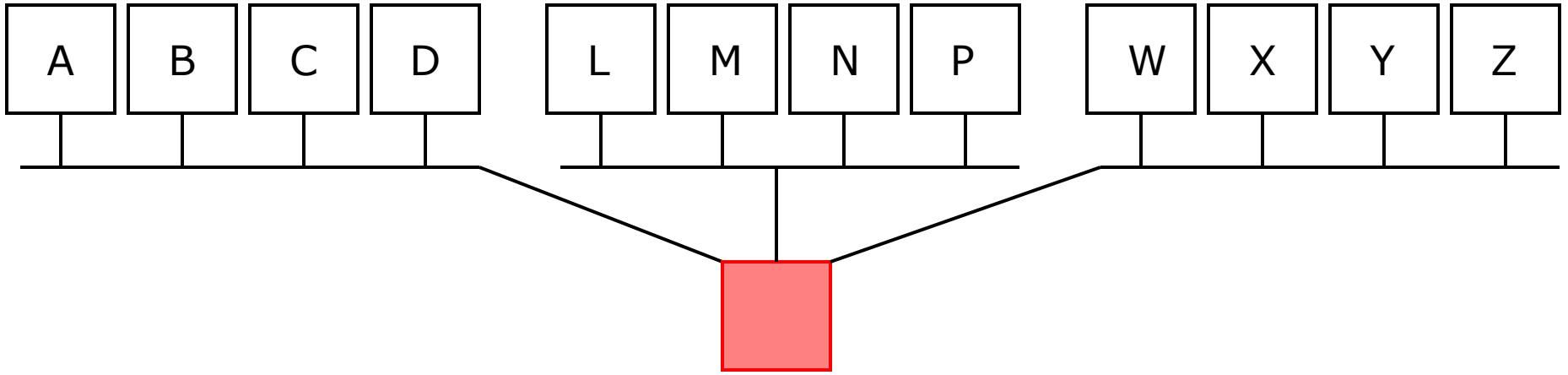
Interconnecting multiple LANs



In series: Traffic between left and right LANs must traverse central LAN, making central LAN:

- A single point of failure, also affecting left & right LANs
- A performance bottleneck
- A security risk

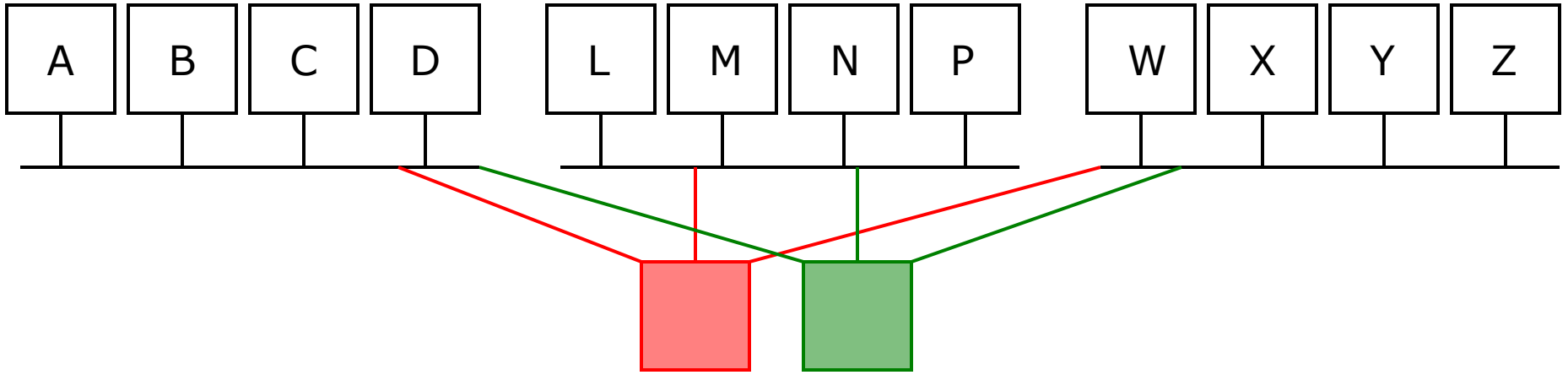
Interconnecting multiple LANs



In a hierarchy:

- Bridge replaces central LAN as a single point of failure and performance bottleneck.

Interconnecting multiple LANs



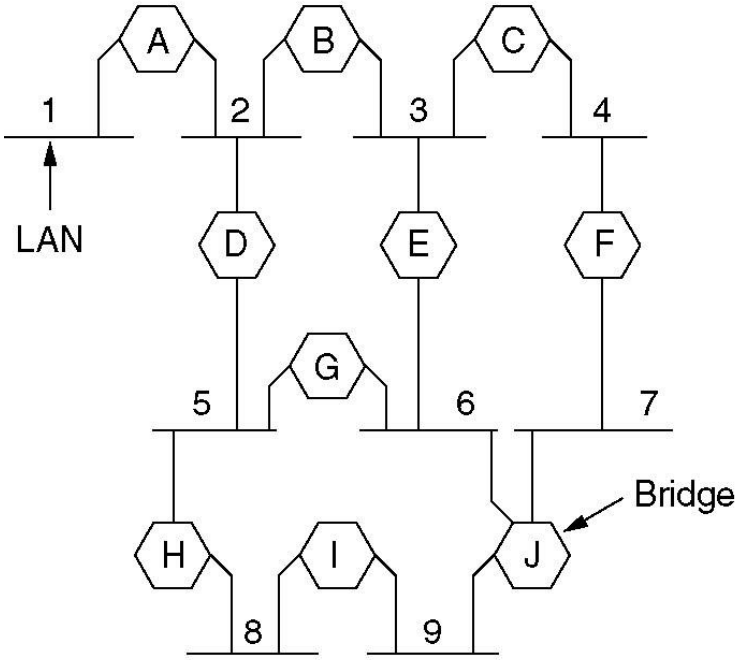
In a hierarchy with multiple paths:

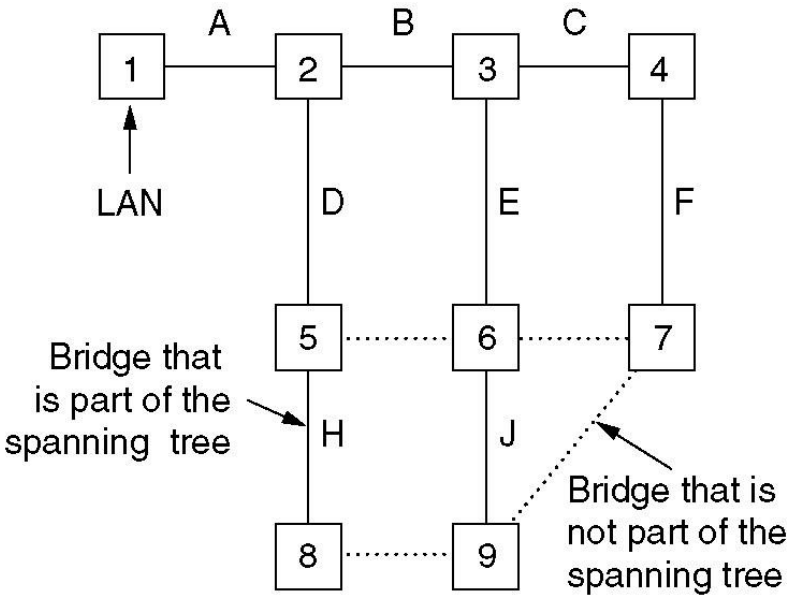
- Need to deal with potential loops → Spanning Trees

Physical connections often loop

Problem: Network may not be physically connected as a tree

- Want redundancy, in case of failure
- Want auto-configuration – arbitrary physical connectivity used sensibly.
- May be mis-configured





[Tanenbaum Fig. 4-44]

Loops are bad

- × Duplicate receipt, which breaks transparency goal.
  - Source sends frame
  - Both bridges (B1 and B2) receive it and forward it.
  - × D receives 2 copies. (so far!)
- × Frames proliferate and loop endlessly
  - Bridges have learned that S is on the lower LAN, but try to accommodate changes (e.g. mobility) by updating knowledge.

B1 forwards frame, but does so *transparently*

=> D and B2 "think" that S is on upper LAN.

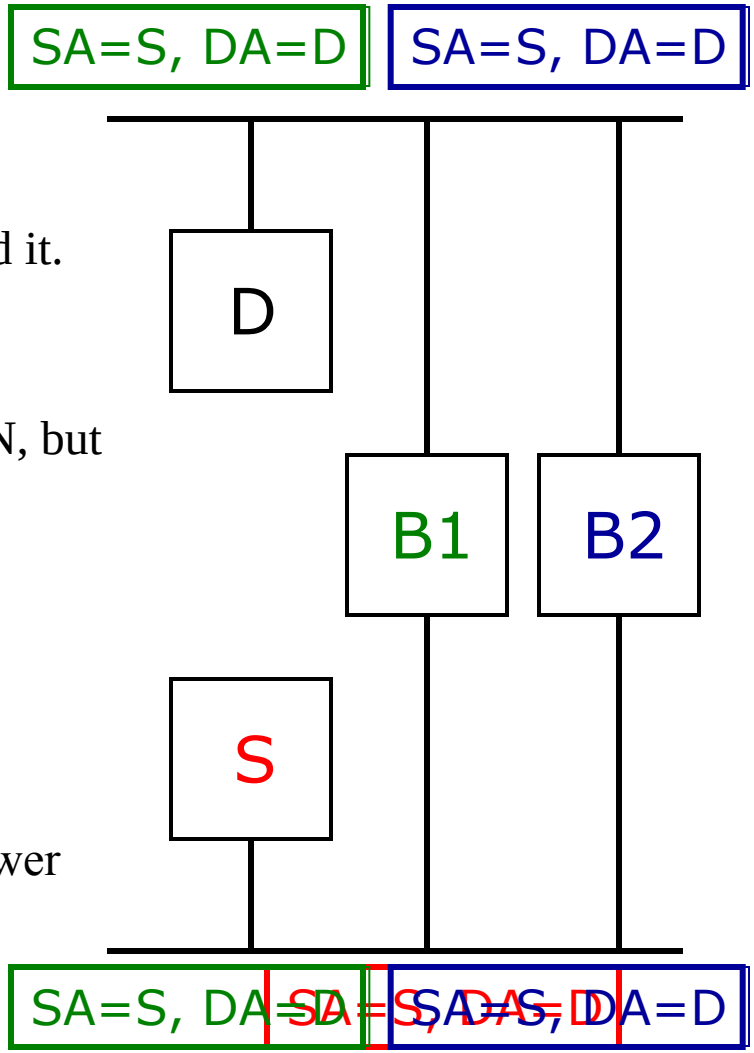
B2 doesn't know D's direction

=> forwards B1's transmission on lower LAN.

Similarly B1 forwards B2's transmission on the lower LAN.

Now 2 frames on the lower LAN,

and the process repeats...



Outline

Some techniques for *handling* loops

Limit the propagation of each packet by adding new header fields
× new header fields violate transparency goal.

e.g.:

Time To Live (TTL) field
- TTL is decremented after each hop, and packet discarded when decremented to 0.
- Used by the Internet Protocol (called a "Hop count" for IPv6)

× Prevents *indefinite* looping, but allows limited duplication (until TTL=0). Even limited duplication violates transparency goal.

"Unique" packet identifier field – intermediate systems record packets that have been recently seen and discard.

√ Packet is discarded after one loop
- Used in peer-to-peer systems (e.g. Gnutella) and ad hoc networks (e.g. DSDV and AODV)

× Intermediate systems need state information <Y4]

× How to choose unique IDs? Large random number: Probably unique => unlikely (though possible) that two packets will have same ID and one will be unnecessarily discarded. Large => transmission capacity cost.

Preventing loops with Spanning Trees

Bridging idea: Allow *physical* loops, but create a *loop-free logical topology* (defining how frames can propagate) by disabling certain ports.

Bridge ports can now be in one of several modes: Initialising, Enabled, Disabled†

In particular, desire a Spanning Tree:

- **Spanning**: Provides a path between each pair of nodes
- **Tree**: Loop-free

Spanning trees have uses other than in bridging, e.g. for multicast distribution & for routing in ad hoc networks.

† Other terms also get used, e.g. Active, Blocking, Listening, Forwarding, ...

Outline

Info used to create a Spanning Tree

Bridges periodically send "Configuration Bridge Protocol Data Units" ("Messages") to a multicast address to which all bridges listen.

Core fields of Messages: (others described later):

- Root ID [1A8>: ID of bridge assumed to be root by bridge sending this Message.
- Cost [HD>: Transmitting Bridge's estimate of the cost of the least cost path from the Root to itself.
- Transmitting Bridge ID: Identifies which bridge sent this Message.
- Port ID [PY>: Which port of the bridge was this Message sent through?

The following slides show how Messages are compared, before detailing these fields.

Comparing messages

Messages must often be compared to determine which is "better", e.g. to elect root [1A8> or designated bridge [WK>

Choose message with:

1. lowest Root ID [1A8>
2. lowest Cost [HD>
3. lowest Transmitting Bridge ID [1A8>
4. lowest Port ID [PY>

i.e. the lower the value, the better.

The ordering (1-4) is important: Later comparisons are only made if earlier comparisons fail to differentiate.

Bridge Identifiers

Bridge Identifier = 48b link layer address + 16b "priority"

- Usually the bridge has a link layer address for each port, and the Bridge ID is one of these addresses (e.g. the lowest).
- Bridge with the smallest Bridge Identifier is elected as **root**.

Why add a priority?:

- A link layer address consists of:
  - Organizationally Unique Identifier (OUI)
  - Device identifier
- Using link layer address alone would give preference to manufacturers with numerically small OUIs.
- Preface by "priority" in more-significant bits. Priority can be set by administrator on each managed device, e.g. to locate root on bridge with high capacity (e.g. packets/second forwarding rate).

Link cost

lower=better, e.g. faster link (lower opportunity cost to use)

For manageable switches, 802.1d recommends (but does not require):

Table 8-5—Path Cost Parameter Values

Of ANSI/IEEE Std 802.1D, 1998 Edition, p. 109

Parameter	Link Speed	Recommended value	Recommended range	Range
Path Cost	4 Mb/s	250	100–1000	1–65 535
Path Cost	10 Mb/s	100	50–600	1–65 535
Path Cost	16 Mb/s	62	40–400	1–65 535
Path Cost	100 Mb/s	19	10–60	1–65 535
Path Cost	1 Gb/s	4	3–10	1–65 535
Path Cost	10 Gb/s	2	1–5	1–65 535

Port identifiers

Locally unique to a bridge

- 8b port # + 8b priority

  (Bridges with 2^8=256 or more ports act as a set of multiple smaller bridges for Spanning Tree purposes. There are proposals to use part of the priority field to extend the port number field.)

- Used in case the bridge has multiple connections to one LAN:
  - Intentionally: e.g. for fault tolerance
  - Effectively: Bridge may connect to "different" LANs that are unintentionally interconnected elsewhere by a repeater/hub.

Spanning Tree Protocol: Synopsis

1. Elect a root
2. For each link/LAN, elect a **Designated Bridge** that has the lowest cost to reach the root.

   "Designated Port" connects Designated Bridge to this link.
3. Each bridge identifies a Root Port that leads towards the root. Bridges enable Root and Designated ports, but disable all other ports
4. Maintain the topology through periodic advertisements

Elections

Each Bridge initially assumes that it is the Root
i.e. Root ID = Transmitting Bridge ID
and sends Messages on LANs to which it is connected with a cost=0

If the Bridge observes a "better" Message <7K] (e.g. lower Root ID) on any port, then it uses that Message as the basis for subsequent Messages that it sends.

Bridge continues sending Messages on ports until it receives one that is better than what it could send.

e.g.

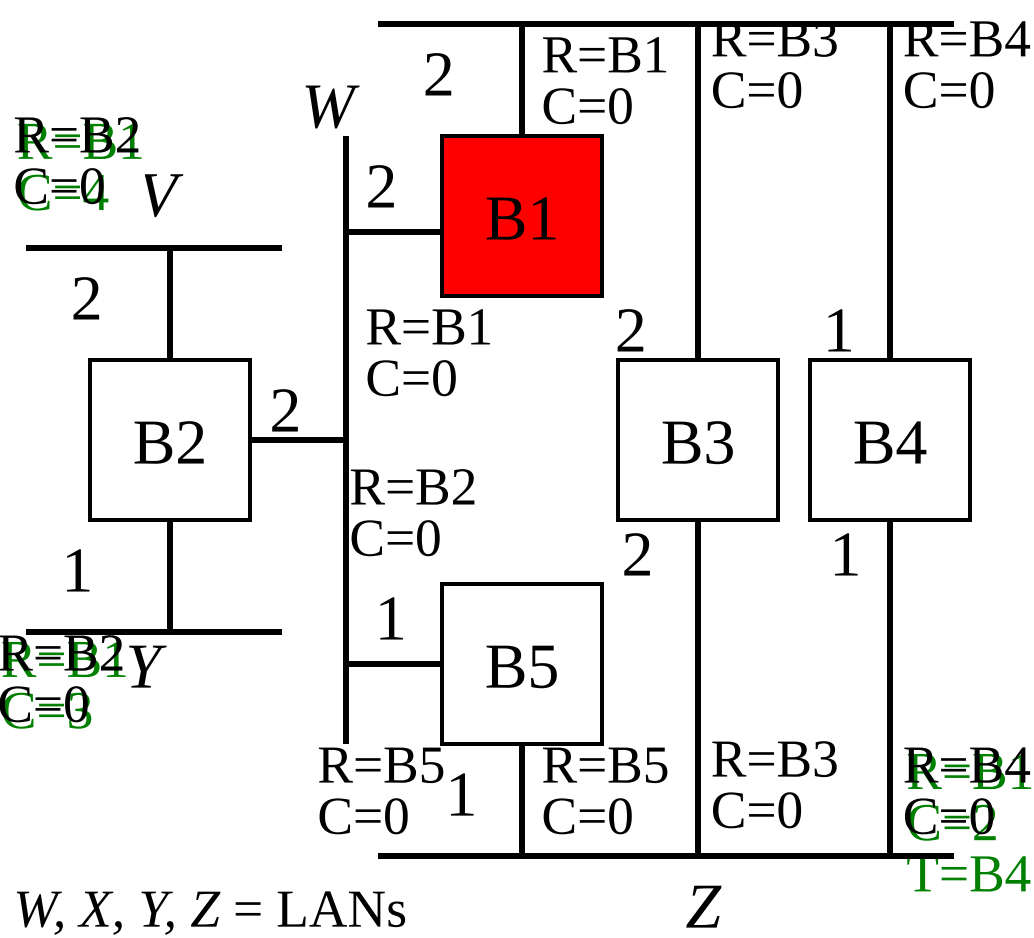
B3 on LANs *L* (cost=1) and *M* (cost=2), and thinks that it is Root
B3 receives a Message from LAN *L* indicating that another bridge (B2) knows of a root B1 which B2 can reach with cost 2.
B3 now thinks root is B1 with cost of 3 (B2's cost to B1+B3's cost to LAN *L*)

While not used for bridging, other election algorithms also exist. e.g. see section 5.4 of A. Tanenbaum and M. v. Steen: 'Distributed Systems: Principles and Paradigms', 2002

Example of electing the Root

- Each bridge thinks it is the Root.
- B2, B3, B4, B5 surrender to B1, and may forward Messages indicating B1 is the Root.

  Cost in forwarded messages = cost of path to Root *except* Root's port (Root sends with cost=0), e.g. B4→*Z* = 1 (B4.*Z*)+1(B4.*X*) but not + 2 (B1.*X*)
- Bridges (B3 and B5) stop sending Messages on LANs from which they receive Messages better than what they could send.



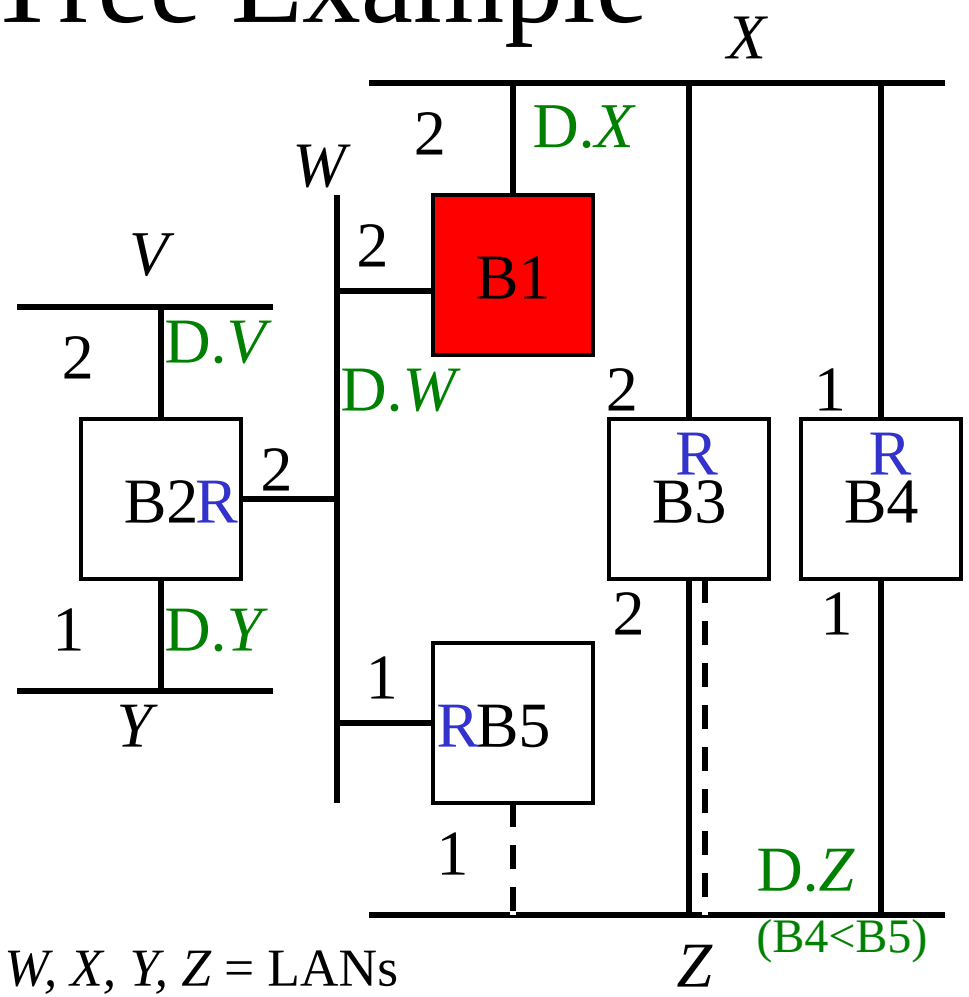
V, W, X, Y, Z = LANs
B1, B2 ... = Bridge Identifiers
1, 2 = costs of using ports

Spanning Tree Example

1. Elect a **root**
2. For each link, **elect a Designated Bridge** that has the lowest cost to reach the root. (& "Designated Port" connects that bridge to that link)

   V,*Y*:B2, *W*,*X*: B1, *Z*: B4 or B5

   For ties, choose bridge/port with the smallest identifier (Z: B4)
3. Each bridge identifies a **Root Port** that leads towards the root.
4. Bridges enable Root and Designated Ports, but disable all other ports (e.g. B3.*Z*, B5.*Z*)
5. Maintain the topology through periodic advertisements



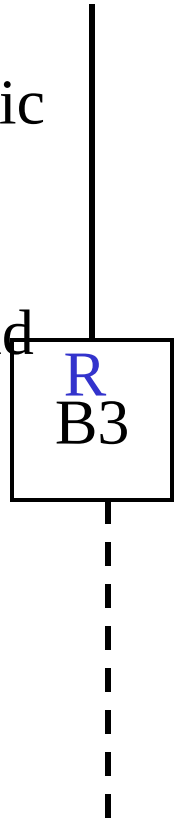
V, *W*, *X*, *Y*, *Z* = LANs
B1, B2 ... = Bridge Identifiers
1, 2 = costs of using ports

Bridges with only one port enabled?

A bridge that has no Designated Port will not forward traffic from one link to another

However, it keeps its Root Port active so that:

- it can receive management traffic being sent to itself, and
- to monitor the network to see if the root changes.



Algorhyme†

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 which must be sure to span
So packets 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.

Youtube

† By Radia Perlman, for those who use rhymes etc to facilitate memorisation

Bridging without the Spanning Tree Protocol

The Spanning Tree Protocol is not always used (with consequent risk of loops forming):

- Cheap switches often lack STP: STP adds complexity (=> cost), both for the protocol itself and management interface needed to allow configuration (e.g. to set priority of bridge to control whether it becomes root)
- Some operators disable STP since they prefer to manually control the network.
  - e.g. network won't automatically reconfigure when a user inserts a bridge with low priority.

Outline

An actual Message

IEEE 802.3 Ethernet
- Destination: 01:80:c2:00:00:00 (01:80:c2:00:00:00)
- Source: 00:30:24:c3:53:b2 (Cisco_c3:53:b2)

multicast to bridges for port leading to station receiving this frame

Logical-Link Control
- DSAP: Spanning Tree BPDU (0x42)
- SSAP: Spanning Tree BPDU (0x42)

Spanning Tree Protocol
- Protocol Identifier: Spanning Tree Protocol (0x0000)
- Protocol Version Identifier: 0
- BPDU Type: Configuration (0x00)
- BPDU flags: 0x00
  - 0... .... = Topology Change Acknowledgment: No
  - ....0 = Topology Change: No
- **Root Identifier: 8192 / 00:10:ff:2a:53:72**
- **Root Path Cost: 8**
- **Bridge Identifier: 32768 / 00:30:24:c3:50:72**
- **Port identifier: 0x2055**
- Message Age: 2
- Max Age: 20
- Hello Time: 2
- Forward Delay: 15

***Boldface** highlights core fields*

Bridge evidently has a smaller address on another port => BID doesn't match Eth source

More examples @ http://uluru.ee.unsw.edu.au/~tim/zoo/index.html#STP

Other Message fields

Data from Messages should be aged.

Root Bridge periodically sends a new Message which indicates:

Hello Time: How often root sends Messages
Max Age: Maximum age of a message before it is discarded.
Message Age=0: Age of this message.

Designated Bridges emit Messages on Designated Ports:

- After receiving one from Root Port, emit using same Message Age.
- If no such message received (e.g. due to transmission error) before a timeout (Hello Time + tolerance) forward a previous one with **Message Age** adjusted to indicate duration of storage in the Designated Bridge.

Forward Delay: Delays changing of port states [ZG> after receipt of a Topology Change Notification [next slide: MD>, giving TCN time to propagate & disable ports before new ports are enabled.

Topology Change Notifications

Strictly speaking, what we've referred to as "Messages" are "Configuration Messages". There is another message type: **Topology Change Notifications**.

Problem: Bridges are so transparent that a topology change may occur and only be seen by nearby bridges. However, distant bridges unaware of the change may keep using wrong port to reach previously learned stations. 5 minute wait for ageing makes for a long outage!

Solution:

1. The bridge that notices the change sends a Topology Change Notification message towards the root. Designated bridges forward it towards the root.
2. To ensure reliable transfer, next bridge to receive the TC message sets a **TC Acknowledgement flag** in the next Configuration Message that it sends. If the TCN sending bridge does not receive such an ack after a Hello period, then it retransmits.
3. When the root receives the TC notification, it then sets the **TC flag** in ensuing Configuration Messages, which causes bridges to use short ageing lifetimes (=> flush their caches <R8]) until they receive a Configuration Message without the TC flag set.

For details, see Perlman pp. 69-70, 78-79

Avoiding temporary loops

Topology Change Notifications take time to spread, during which time bridges may act inconsistently, causing temporary

- disconnection – undesirable
- looping – horrible!

=> Intermediate "preforwarding" state in transition from port being blocked to forwarding, giving topology change time to spread.

- In this state, the bridge sends Messages but doesn't forward frames.
- Preforwarding state lasts "twice the maximum transit time across the network" [Perlman p. 66]

Implementation: 802.1 has *two* intermediate states: Switch doesn't forward during either, which last 15 seconds == Forward Delay parameter:

1. Listening: Don't learn (directions may not work after change)
2. Learning: Do learn.

=> Disconnection may last 30 seconds = slow "convergence"

Rapid Spanning-Tree Protocol (802.1w, incorporated into 802.1D-2004) tries to address this.

Outline

Advantages of bridges

- √ Simple to use (=> minimal labour and disruption):
  - √ Transparent => require no configuration of existing nodes.
  - √ Self-learning => require minimal configuration of bridges (e.g. might set priority)
- √ Simple to implement (=> fast):
  - √ Only process MAC layer headers
  - √ Don't modify frames => no need to recalculate CRCs etc

Disadvantages of bridges

- Large addresses to classify [CJ>
- Implications of broadcasting [LE>
- Sub-optimal routing [K2>

Details on the following slides...

Bottom line: Bridges for networking hundreds of nodes, routers for more.

Disadvantages 1: Classification

- "Flat" addresses => Large addresses to classify:
  - Large (48b) keys
  - Either:
    - Large database – difficult to aggregate knowledge of multiple destinations reachable through same port.
    - Waste transmission capacity: Small database may overflow, and traffic to less-frequently used addresses is inefficiently flooded.

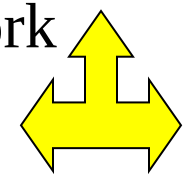
Disadvantages 2: Broadcasting

- Broadcast propagation (from transparency goal)
  - much traffic is broadcast when there are many nodes, and each periodically broadcasts a frame (e.g. to discover local printers, renew ARP bindings, etc).
  - Uncontrolled broadcast may lead to a "broadcast storm"
- Multicast may be flooded
  - i.e. to reach a subset of the nodes, send it to all nodes.
  - might prefer to "prune the spanning tree", i.e. block multicast from propagating on tree where it is not needed. (IGMP snooping)
- Flooding caused by cache misses (database full) is effectively broadcasting.

† "Broadcast storms" occur when large volumes of traffic result from broadcasting.
e.g. A misbehaving host continuously transmits broadcast traffic.
e.g. Poor protocols that broadcast responses to broadcast traffic, leading to escalation.
e.g. Many nodes that receive a broadcast packet send a reply (e.g. ICMP error)

Disadvantages 3: Sub-optimal paths

- All traffic is confined to the Spanning Tree:
  - May be forced to use indirect paths – may load network more than necessary.
  - Traffic tends to congregate near the root => assign priorities s.t. root is a high capacity switch
  - Can't use parallel paths => no load balancing
- Slow convergence



Routing overcomes these disadvantages, but requires higher-layer processing (more complicated)

Bridge security: Strengths

- √ Switched networks provide security by directing traffic towards the destination, rather than allowing it to diffuse across the network, and be accessible by eavesdroppers.

- √ Some switches disable a port if they notice a change in link layer address on that port => prevent people plugging their own computers in (e.g. to library or computing lab Ethernet outlets)

Bridge security: Weaknesses

Some network cards allow programming of the link layer address, e.g. using the Unix ifconfig command.

Attack 1 (e.g. "macof" tool):
Exploits: If destination address isn't in database, then forward on all ports.

Attacker can send many frames with different source and destination addresses to fill bridge's database. Real traffic will then be flooded =>

- Reduced performance
- No security advantages of switching

Attack 2: Exploits: Frames will be forwarded in the direction of the most recent observation of an address.
=> Deny service to station with a known link layer address by transmitting frames with that link layer address from another part of the network to replace legit station's database entry.

Attack 3: Exploits: Bridges disable ports based on information from external nodes. Attacker can advertise low Root ID to force tree change and focus traffic on itself (sniff traffic, discard and deny service, clog links close to self).
=> Network administrators may configure switches to disable ports that receive STP configuration messages when supposed to lead to end-system.

Final aside: Bridging *within* end-systems

Traditional end-systems have only one network interface.

Exception: "multi-homed" end-systems used where fault tolerance or throughput are critical.

Traditional view of multiple interfaces is that each has a distinct IP address.

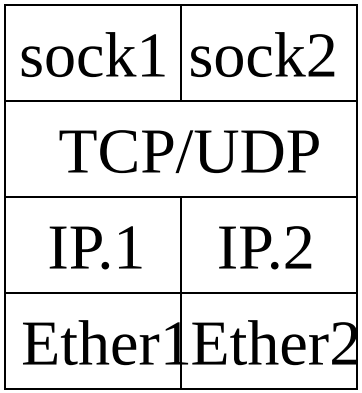
× Double the configuration

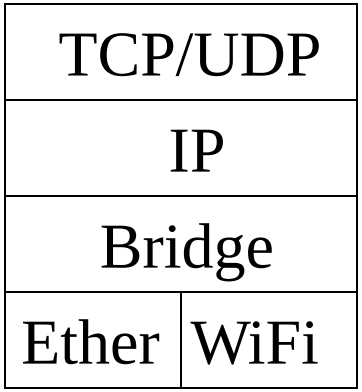
× Handover from one interface to another requires change of socket addresses, etc

Modern end-systems may have multiple interfaces, e.g. Ethernet and WiFi built-in (particularly on laptops)

Windows XP provided a "Network Bridge" interface

- Still have to rectify loops => Participates in Spanning Tree construction process
- Some net administrators (e.g. UNSW Comms Unit in 2004) configure switches to disable port if they receive Bridge PDUs from that port => Windows XP machines with multiple interfaces disconnected :-(





This essentially creates a tiny 3-port switch, as in <X2]

See http://www.microsoft.com/resources/documentation/windows/xp/all/proddocs/en-us/hnw_bridge_enable.mspx

The end of bridging