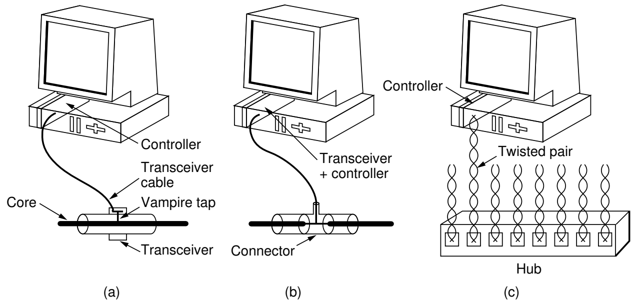
# 802.2 (or Ethernet)

**Ethernet** stands for a near implementation of the **IEEE 802.3** protocol. It is CSMA/CD based (sense the channel, wait until idle, and backoff when you detect a collision).

| Name | Cable | Max. dist. | Nodes/Seg. |
|---|---|---|---|
| 10Base5 | Thick coax | 500 m | 100 |
| 10Base2 | Thin coax | 200 m | 30 |
| 10Base-T | Twisted pair | 100 m | 1024 |
| 10Base-F | Fiber optics | 2000 m | 1024 |

**Note**: 10Base-T is popular as it can make use of telephone lines, and is easy to maintain when it comes to cable breaks (cables go to **hubs**). All four types are widely used.

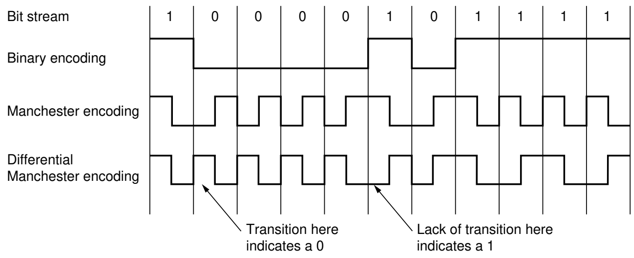





# Manchester Encoding

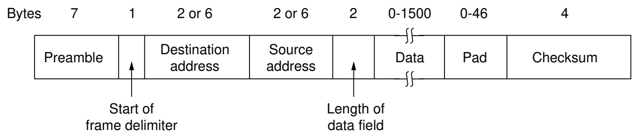
**Problem:** We can't just send straight binary codes across the wire, because stations can't distinguish a 0 from an idle line.

**Solution**: use an encoding scheme in which a voltage transition occurs during every bit time (Manchester Encoding):






# 802.3 Frame Layout




**Preamble:** Seven times 10101010 is used to synchronize the receiver's clock with that of the sender.

**Start:** Just a delimiter to tell that the real info is now coming.

**Address:** Generally 48-bit fields. Leftmost bit indicates ordinary or group addresses. Second bit indicates global or local address.

**Length:** Ranges from 0-1500. Frames should always be at least 64 bytes. See Tanenbaum for the details.

**Pad:** you got it...

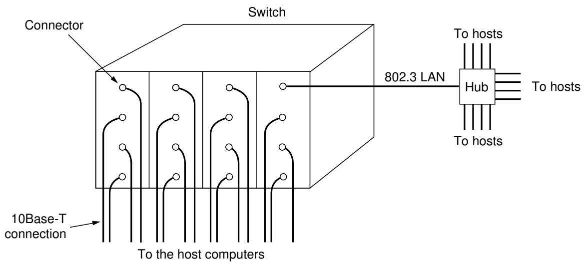
**Checksum:** Calculated over the data field. CRC-based.



# Switched 802.3 LANs

**Problem:** As more stations are added, traffic will go up, and so will the possibility of collisions ⇒ the network will saturate.

**Solution:** Divide the network into separate sub-LANs and connect them through a high-speed switch:




**Question**: Are we really improving things here?

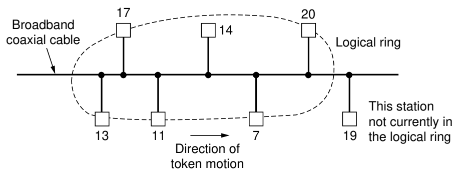
**Note**: Our ZOO works like this.

# 802.4: Token Bus

**Problem:** In 802.3 there is a possibility that you'll never be able to transmit due to (1) simultaneous transmission efforts, and (2) waiting time that increases with each failed attempt to start transmission (binary exponential back-off).

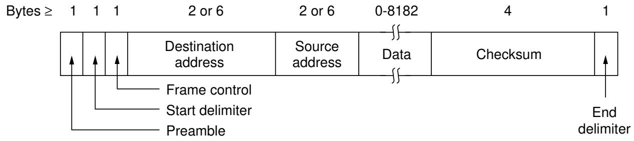
This is not acceptable for real-time, reliable, industrial systems:

- Make the network deterministic by **logically** organizing the stations in a ring.
- When a station wants to transmit, it requires a special control frame called a **token**.
- The token is handed from station to station.
- *Physically*, all stations are connected to a single bus (as with Ethernet). **WHY!?**






# Token Bus Frame




The only thing in common with 802.3 are the addresses (sigh...). The real important difference lies in the control frames (Ethernet doesn't have those):

- Claim the token during ring initialization.
- Allow new senders to enter the ring.
- Recover from token loss.
- Resolve contending stations for position in the ring.
- The actual token.
- Allow station to leave the ring.

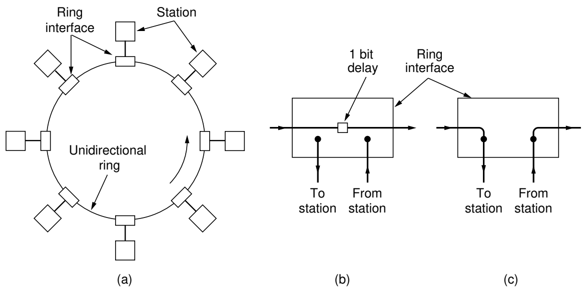
**Note:** stations maintain the ring in a logical fashion – it has nothing to do with their phyiscal position.

**Note:** token bus systems have a (rather intricate) priority scheme. This is absolutely necessary for handling real-time data.



# 802.5: Token Ring

**Basic idea:** Why make things so difficult? Just physically order a bunch of computers into a ring ⇒ we only have to use the well-understood (digital) point-to-point connections.

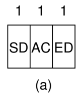


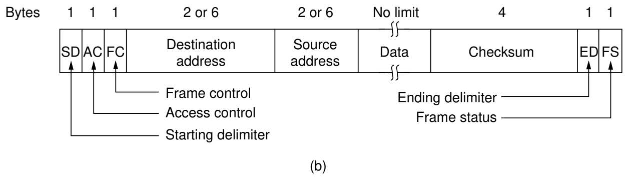

**Problem:** A bit has a *physical length*: If the ring has a bandwidth of $R$ Mbps, a bit is emitted every $1/R$ $\mu$sec. The propagation speed is about $200m/\mu$sec ⇒ a bit has a length of $200/R$ meters.

**Effect:** Take a 1 km ring operating at 4 Mbps – the ring can contain at most 20 bits of information. This may not be enough for a token (which may need the ring entirely when all stations are idle)!



# Token Ring Frame




- A full frame can generally not be held by the ring ⇒ the sender will have to transmit bits *and* remove those that have circulated the ring.
- Seizing the token means changing a bit in the AC field, after which it immediately changes to the first two bytes of a frame (**HUH!?**).
- The FC field a frame is used by the destination to indicate (1) its presence, and (2) frame acceptance ⇒ because the source has to remove the frame anyway, we have acknowledgments for free.
- There's a separate monitor station to maintain the ring, and every station can switch to play monitor.

# Comparison

| Issue | 802.3 | 802.4 | 802.5 |
|---|---|---|---|
| Performance | OK | OK | OK |
| Simple engineering? | yes | no | yes |
| Analog components | yes | a lot | hardly |
| Deterministic | no | yes | yes |
| Priorities | no | yes | yes (not fair) |
| High-load perf. | bad | good | good |
| Reliability | OK | very good | good (monitor) |
| Use-base | large | small | large |

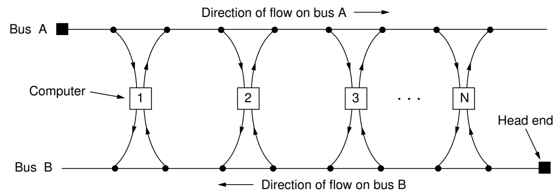
**Note:** Study Tanenbaum carefully – these are important protocols. Understanding the similarities and differences is important.



# Distributed Queue Dual Bus

**Problem:** The 802.3–802.5 networks are only suited for **local** area networks. They cannot be used for very large, non-wide area networks.

**Solution:** The IEEE defined 802.6 as a standard for MANs. The basis is to use two buses:

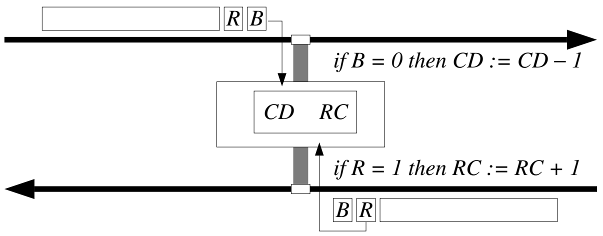



**Principal:** When you want to send something to one of your *right-hand* neighbors, use the upper bus, otherwise the lower one.

**Note:** DQDB uses fixed-sized 53-byte cells instead of (variable-length) frames. This is the same as ATM technology. Each cell carries a **busy bit**, and a **request bit**.



# Distributed Queue Dual Bus II




- *RC* counts the number of requests from downstream neighbors.
- *CD* counts the number of *outstanding* requests issued before your request.
- If you want to send something downstream, set the first available request bit in a cell on the B bus to 1, and copy *RC* to *CD*.
- You can have at most one request transmission at a time.
- When *CD* drops to zero, take the first non-busy cell on the A bus.

**Question:** what happens when hardly any station wants to send? And what if everyone wants to send?

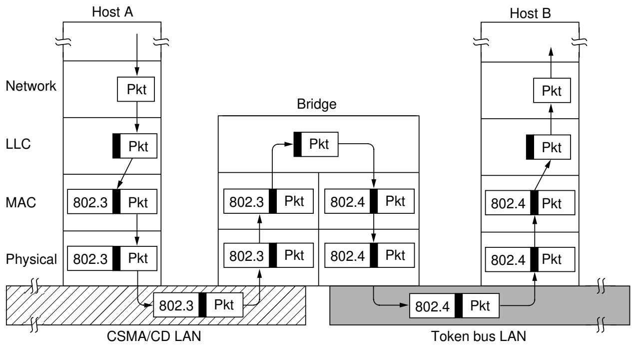


# Bridges

**Basic idea:** we want to interconnect a number of LANs, rather than having one big one. Two LANs are connected through a **bridge**.

- You want to let existing LANs (in departments, buildings, etc.) as they are. On the other hand, you do want to connect them.
- When an organization is spread over several buildings, it may be cheaper to have a different interconnect (e.g., infrared) than coax cacble. You may also have no choice.
- Splitting things up (rather than just tying things together) may be good for load balancing. There can be a file server per LAN.
- Physical distance sometimes precludes building one big LAN. For example, Ethernet can only handle cables up to 2.5 km. Not very much.
- The reliability can be improved: if one part goes down, the other LAN segments may still operate.
- For security reasons, bridges can check frames, and refuse to forward those that seem suspicious.

# Principal Operation

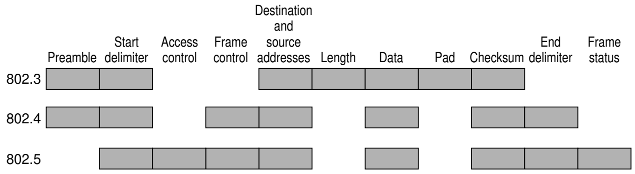



- A packet is passed to the data link layer (LLC part)
- It is then passed to the MAC layer (specific access strategy)
- A bridge **converts** the stuff above the MAC layer, in the LLC layer



# Bridging Problems

- Really great – the three committees for 802.x invented three different frame formats:




- We also have the problem of different date rates. If a higher speed LAN starts pumping frames on a lower speed one, we've got a problem.
- What should we do if the target LAN can't accept the frame length of the source LAN? Splitting a frame into pieces is out of the question! (**WHY?**)



# Bridging Problems II

| | | Destination LAN | | |
|---|---|---|---|---|
| | | 802.3 (CSMA/CD) | 802.4 (Token bus) | 802.4 (Token ring) |
| Source LAN | 802.3 | | 1, 4 | 1, 2, 4, 8 |
| | 802.4 | 1, 5, 8, 9, 10 | 9 | 1, 2, 3, 8, 9, 10 |
| | 802.5 | 1, 2, 5, 6, 7, 10 | 1, 2, 3, 6, 7 | 6, 7 |

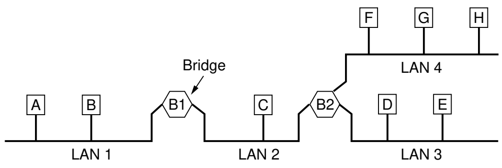
Actions:
1. Reformat the frame and compute new checksum
2. Reverse the bit order.
3. Copy the priority, meaningful or not.
4. Generate a ficticious priority.
5. Discard priority.
6. Drain the ring (somehow).
7. Set A and C bits (by lying).
8. Worry about congestion (fast LAN to slow LAN).
9. Worry about token handoff ACK being delayed or impossible.
10. Panic if frame is too long for destination LAN.

Parameters assumed:
802.3: 1500-byte frames, 10 Mbps (minus collisions)
802.4: 8191-byte frames 10 Mbps
802.5: 5000-byte frames 4 Mbps



# Transparent Bridges

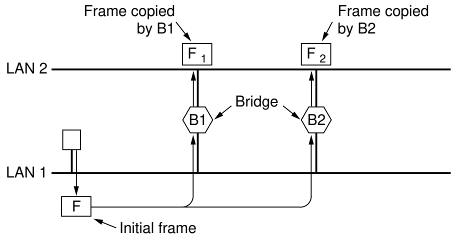
**Main issue:** Can we develop a bridge that interconnects LANs in a completely transparent way, i.e. seems to turn it into one big LAN?




- An incoming frame is simply forwarded to all other LAN segments connected to the bridge.
- Because an incoming frame contains the source address, a bridge can gradually get to know through which interface it can reach a host.
- Use a timeout mechanism to flush all knowledge a bridge has ⇒ it will gradually build up a fresh view again

# Parallel Transparent Bridges

**Problem:** Sometimes LANs are connected by multiple bridges in such a way that we no longer have a tree, but a graph containing cycles ⇒ we can't just forward frames anymore.

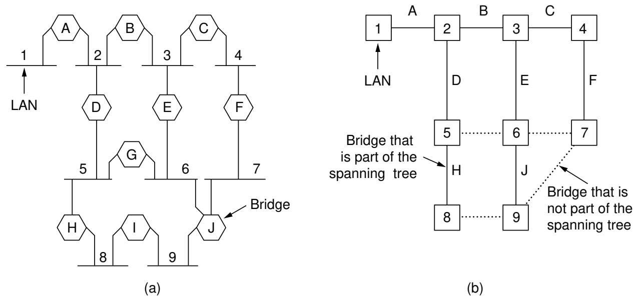



**Question:** why would we use multiple bridges in such a way that we don't have a tree anymore?



# Parallel Transparent Bridges II

**Solution:** let the bridges construct a **spanning tree** on their own.




- Each bridge broadcasts its ID across the attached LAN segments. The lowest numbered bridge becomes root for that segment.
- A root bridge for a segment knows it can never be the root for the tree, if it finds out there's a bridge with a lower number.
- Bridges advertise their distance to the "real" root ⇒ that's how we build a spanning tree.



# Source Routing Bridges

**Principal:** The source is assumed to provide a route through the interconnected network towards the destination:

LAN-1, BRIDGE-1, LAN-2, BRIDGE-4, LAN-5

The problem is the routing ⇒ **route directives**:

- **null directive:** deliver a frame to all stations on the same segment as the source.
- **non-broadcast directive:** specifies a unique route that is to be followed.
- **all-routes broadcast:** forward the frame over all possible routes in the network (explodes!) Is used to *discover routes.*
- **single-route broadcast:** the frame is to be sent to each station, but only once (spanning tree).



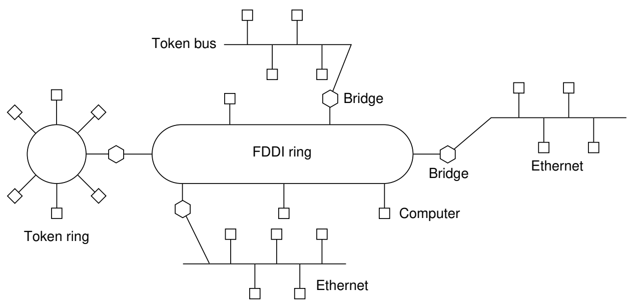
# Comparison of 802 Bridges

| Issue | Transparent bridge | Source routing |
|---|---|---|
| Orientation | Connectionless | Connection-oriented |
| Transparency | Fully transparent | Not transparent |
| Configuration | Automatic | Manually |
| Routing | Suboptimal | Optimal |
| Locating | Backward learning | Discovery frames |
| Failures | Handled by bridges | Handled by hosts |
| Complexity | In the bridges | In the hosts |

**Question:** which one do you prefer?

# FDDI

**Main issue:** We want to have a number of high-speed LANs, often to be used as a **backbone**. Fiber optic networks are really good here.

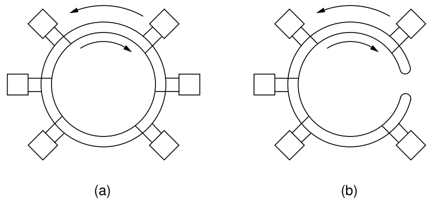



- FDDI (Fiber Distributed Data Interface) operates at 100 Mbps.
- It supports **synchronous** and **asynchronous** data transmission.
- It has an error rate of less than 1 per $2.5 \times 10^{10}$ bits. Pretty good, and practice is even better.



# FDDI II

For fault tolerance reasons, FDDI networks have two rings:




- A token can be inserted just after a frame has been sent: you don't have to wait until the frame comes back to the sender. *Why would they have done that?*
- A station is allowed to transmit synchronous data during the first time slots. If it has slots over, it can transmit asynchronous data. Time is determined through a **token holding timer**.
- If the token is "too late", a station can only transmit synchronous data (if it has any). Asynchronous data cannot be transmitted until lost time has been made up for.



# Fast Ethernet

**Problem:** FDDI is just plain expensive, so it's okay for backbones, but not for basic LAN segments.

**Solution:** Upgrade existing base of LANs (i.e. Ethernets) in such a way that the interfaces remain the same, but the capacity goes up $\Rightarrow$ 100 Mbps Ethernet.

Data formats, interfaces, and protocols are all the same.

That means that we can only drop the bit time from 100 nsec to 10 nsec. Just telling everyone to shorten their wires won't really do.

The solution is to use only hubs (10Base-T) and combine it with switching technology. (10Base-T can make use of telephone lines, and is easy to maintain when it comes to cable breaks.)



# Fast Ethernet II

**So:** the cabling turns out to be pretty important:

| **Name** | **Cable** | **Length** |
|---|---|---|
| 100Base-T4 | Twisted pair | 100 m |
| 100Base-TX | Twisted pair | 100 m |
| 100Base-F | Fiber optics | 2000 m |

**To illustrate:** T4 uses 4 twisted pairs, one to the hub, one from the hub, and two switchable to current direction (half-duplex).

Rather than binary signalling, we use ternary signalling $\Rightarrow$ three twisted pairs in the right direction gives $3^3$ possible signals, which can encode strings of 4 bits.

With a signalling speed of 25 Mhz, we then get 100 Mbps across the category 3 UTP wires. Not really bad...

**Note:** If we then subsequently add frame-buffering capabilities to the hub, combine that with a high-speed backplane, we can have stations transmit at the same time. The buffering at the hub can avoid a lot of collisions.