Multi-Access Media

Introduction

This chapter of Tanenbaum is called the “Medium Access Control Sublayer.” It is an odd choice of title. More precisely, this is a chapter about “multi-access technologies” and the technologies both wired and wireless that derive from them. For wired, there will be more, while there are only two “ends” but multiple stations attached to the single medium. Multi-access media were common in the telephone network from the beginning. They were called early “party lines.” Everyone could hear anyone who was attached to the line. To distinguish calls to different telephones, each one had a distinctive ring, so, many longs and shorts.

The Channel Allocation Problem

Static Channel Allocation

Assumptions for Dynamic Channel Allocation

Read

Read Tanenbaum Chapter 4, Section 4.1.2, pp. 268–271.

One note: We will see some use of queueing theory here. The reader should be aware that queuing theory relies on assuming that the traffic is Poisson. This is because Poisson is about the only distribution that queuing theory can solve for and it can only be solved for the steady-state, when it is the transients that we are interested in. It has been well-understood for decades that network traffic is not Poisson. Some later textbooks were taken by a flawed paper claiming that traffic is self-similar. It isn’t self-similar either.

Multiple Access Protocols

Aloha

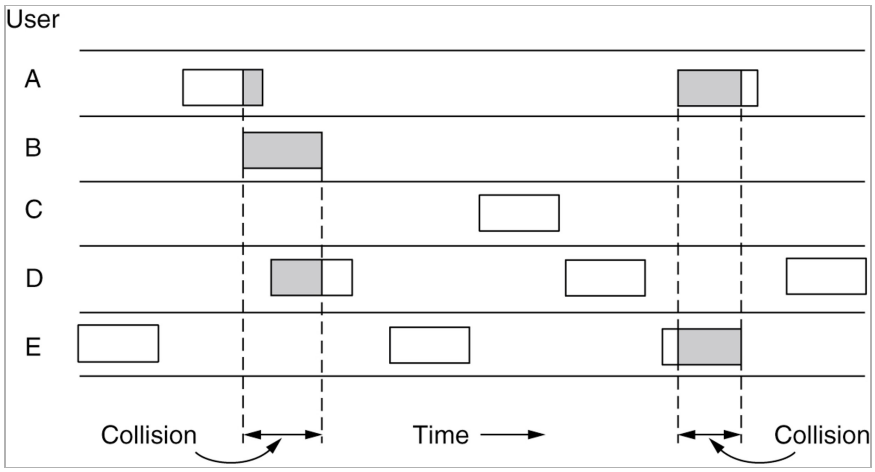
Read

Read Tanenbaum Chapter 4, Section 4.2.1, pp. pp. 272–274.

We will look at several different multi-access protocols, but we'll start with Aloha. Aloha was the first real, serious packet radio system developed at the University of Hawaii in 1970–71. They worked out much of the basics of packet radio. Everything starts with Aloha. The problem they had before them was that they wanted to be able to communicate with all the various parts of the university that were on different islands and running cable was not exactly an easy thing to do.

Pure Aloha

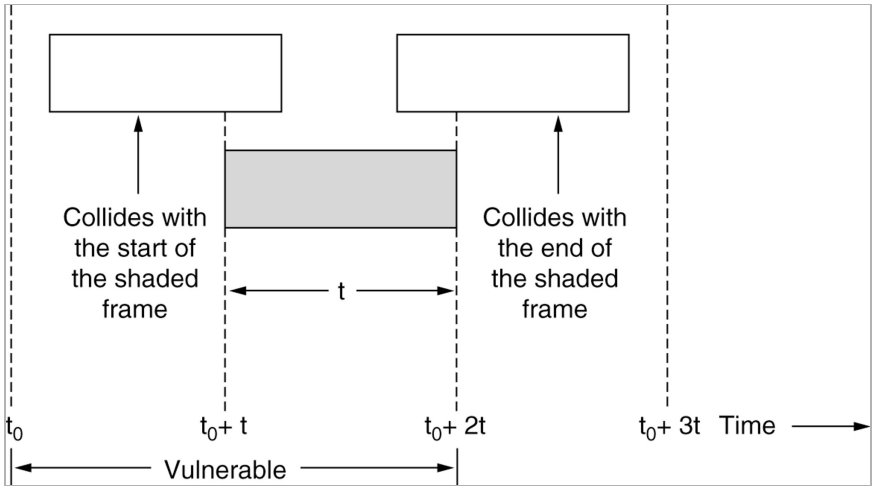
ALOHA



In pure ALOHA, frames are transmitted at completely arbitrary times.

Source: Tanenbaum, A., Wetherall, D., & Feamster, N. (2020). *Computer Networks* (6th ed.).

ALOHA



Vulnerable period for the shaded frame.

Source: Tanenbaum, A., Wetherall, D., & Feamster, N. (2020). *Computer Networks* (6th ed.).

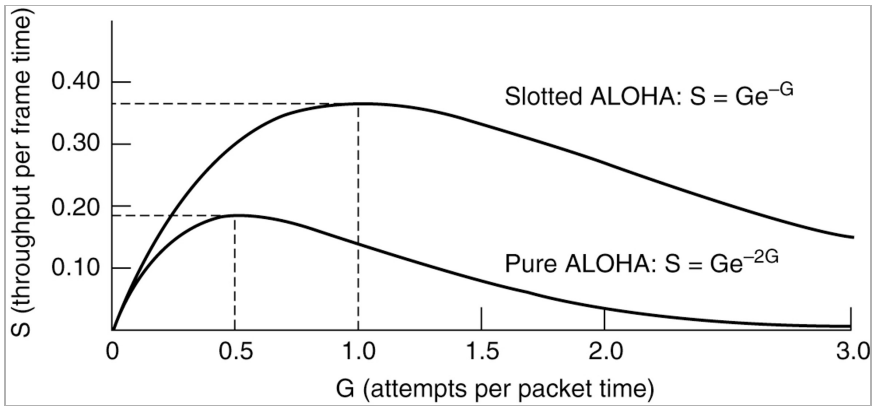
There is a very famous paper by Frank Kuo, who did the queuing analysis. What he found was that for pure Aloha, where anyone can transmit at any time, the maximum bandwidth you can get is 18.5%.

Slotted Aloha

Read

Read Tanenbaum Chapter 4, Section 4.2.1, pp. 275–276

ALOHA



Throughput versus offered traffic for ALOHA systems.

Source: Tanenbaum, A., Wetherall, D., & Feamster, N. (2020). *Computer Networks* (6th ed.).

However, for Slotted Aloha, in other words, if all the transmitters are synchronized to only send on specific time slots, then, if there's a collision, the collision is perfect. The two packets overlap entirely, not partially. The queuing analysis shows that it doubles Aloha over pure Aloha to 36% of the bandwidth. But that's the best you can do.

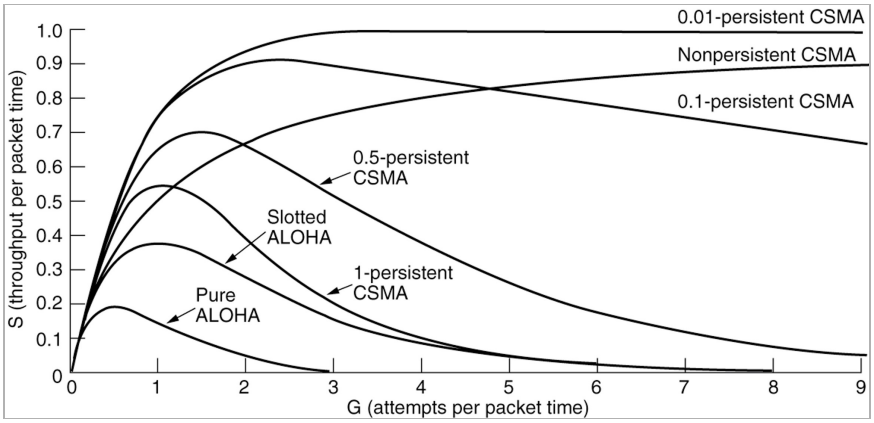
Carrier-Sense Multiple Access Protocols (CSMA)

Persistent and Non-Persistent CSMA & CSMA with Collision Detection

Read

Read Tanenbaum Chapter 4, Section 4.2.2, pp. 276–279.

Persistent and Nonpersistent CSMA



Comparison of the channel utilization versus load for various random-access protocols.

Source: Tanenbaum, A., Wetherall, D., & Feamster, N. (2020). *Computer Networks* (6th ed.).

1-persistent waits until clear and sends. (Could have others waiting.) Propagation time is critical. Propagation delay creates a window for collisions.

Non-persistent detects the channel is busy and waits a random amount of time before trying to send. Non-persistent will lead to longer delays but fewer collisions.

P-persistent, either sends or waits with probability, p, or defers with probability $(1 - p)$.

CSMA/CD requires the transmitter to detect the collision. Hence it can be used for wired, but not wireless. Collisions are harder to detect with wireless since signal strength falls off as $1/r^2$.

Collision-Free Protocols

Read

Read Tanenbaum Chapter 4, Section 4.2.3, pp. 279–283.

Limited Contention Protocols

Read

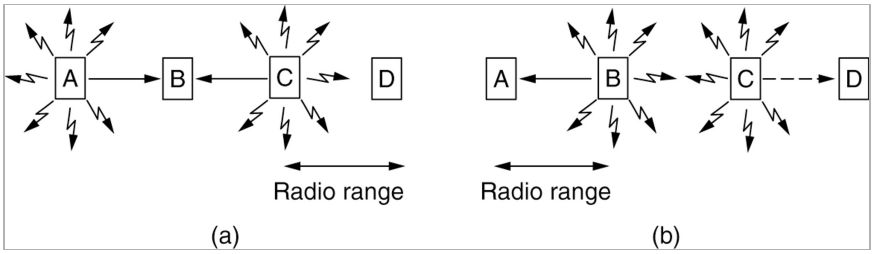
Read Tanenbaum Chapter 4, Section 4.2.4, pp. 283–287.

Wireless LAN Protocols

Read

Read Tanenbaum Chapter 4, Section 4.2.5, pp. 287–290.

Wireless LAN Protocols

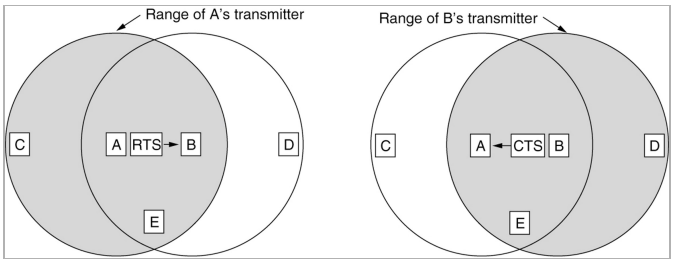


A wireless LAN. (a) *A* and *C* are hidden terminals when transmitting to *B*. (b) *B* and *C* are exposed terminals when transmitting to *A* and *D*.

Source: Tanenbaum, A., Wetherall, D., & Feamster, N. (2020). *Computer Networks* (6th ed.).

With wireless protocols, as we saw with Aloha, a big problem is the hidden transmitter or the range. Even if it's not hidden, there is always the problem of radios being out of range. Where B can hear both A and C, but A can't hear C and C can’t hear A. A can hear B but can't hear C. D can hear C but not B.

Wireless LAN Protocols



The MACA protocol. (a) *A* sending an RTS to *B*. (b) *B* responding with a CTS to *A*.

Source: Tanenbaum, A., Wetherall, D., & Feamster, N. (2020). *Computer Networks* (6th ed.).

So how can we resolve this? Almost all the wireless protocols are based on having two commands for mobile access collision avoidance: Request to Send (RTS) and Clear to Send (CTS). This of course, turns

the broadcast media into a half-duplex, point-to-point media.

What happens is A will send a Request to Send to B. A and B hear the RTS and so does C. C knows to be quiet, but D doesn't hear it. When B responds with the Clear to Send, C and D hear the CTS and know to be quiet for a while. This gives a free channel for A and B to exchange data. We will use this when we talk about 802.11 later in this lecture. This is the basis for a lot of these protocols.

Ethernet

Read

Read Tanenbaum Chapter 4, Section 4.3, pp. 290.

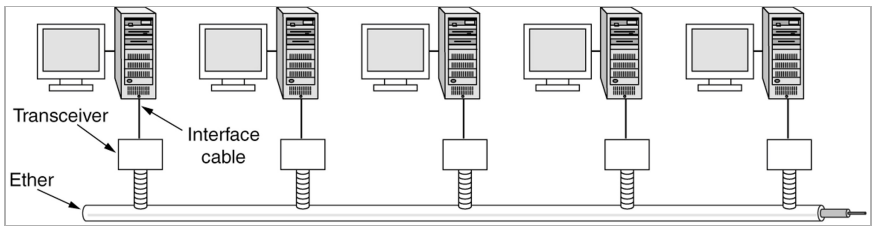
We have come to Ethernet, which is derived from Aloha. Bob Metcalfe was doing his PhD at Harvard in 1971, looking for a PhD topic. He was working on the ARPANET node at Harvard and MIT and learned about the ARPA ALOHA project at the University of is Hawaii. He saw the results from Frank Kuo's paper that one could only get 18.5% of the bandwidth because of collisions caused by the fact that all of the transmitters couldn't hear each other. Bob had a brilliant idea: suppose the "ether" is replaced by coax. Now, all the transmitters can hear each other and back off. Then one will get much better utilization of the wire and there can be much higher bandwidths. That's Ethernet: Aloha where the ether is replaced by coax!

Classic Ethernet Physical Layer

Read

Read Tanenbaum Chapter 4, Section 4.3.1, pp. 290–292.

Architecture of Classic Ethernet



Source: Tanenbaum, A., Wetherall, D., & Feamster, N. (2020). *Computer Networks* (6th ed.).

It is the classic thing one would expect from a graduate student. "Hey, we can replace the ether with coax

and run the coax around the lab. Whenever we go by a workstation, we can just tap into the coax! Neat!" That is literally what they did! To connect a workstation to the Ethernet, a cable from a device known as a vampire tap was used to connect to the coax. The vampire tap actually stuck a probe through the braiding on the coax and made contact with the conductor in the coax.

This was really neat because with coax you can have very high bandwidth. If you remember what I have said about typical speeds in 1972: dial up was 300 bits per second (you can type faster than that), and 9.6 kbps was typical in the lab. Ethernet started off at a megabit per second (two orders of magnitude greater!) and very quickly went up to 10 Mbps. This was a huge step function in performance! There were now all sorts of things that could be done. Bob finally finished his PhD and went off to work at Xerox PARC1 on the West Coast. In fact, Bob likes to point out the fact that Harvard never published his PhD but MIT did.

This opened the door for many distributed computing uses that people had been contemplating but were not possible with the technology of the time. Many people saw Ethernet as enabling distributed computing. Certainly, Xerox PARC did and had gone a long way toward making it a reality.

At PARC, they started developing Ethernet. Bob was joined by some very smart guys: John Shoch, Dave Boggs, Butler Lampson, and others. They developed an entire networking system built around Ethernet. Everybody was visiting Xerox PARC. Lots of people in the ARPANET knew what was going on at PARC. Xerox was developing this to become a product. At the time, Xerox was also in the computer business. The XDS 940 was actually pretty good system.

It was clear immediately that Ethernet was going to be at the center of a new market in local area networks and what they enabled. PARC had had an extensive network based on Ethernet running in the lab for six or eight years. It was a mature technology. Not only that they had developed a full complement of protocols on top of Ethernet for office automation, e.g., file servers, print servers, a distributed directory system called Grapevine. (Notice how this leveraged Xerox's dominant position in the copier market.) They were using this within Xerox fairly heavily. It should be noted that what PARC was doing was no secret. There was a lot information and idea flow among with the ARPANET and other minicomputer companies, at least the ones who had an idea of the future.

Most could see along with Xerox that while there was a lot of Ethernet equipment to sell, what was really important was what could be done with it. (But not everyone.) That would be an even bigger market with many more players. Hence in the late '70s, it became clear that key to Ethernet' success would be making it a standard. To prepare for that Xerox formed a collaboration with Intel and Digital Equipment Corporation (DEC), which was then the largest employer in the State of Massachusetts. (DEC had pioneered the mini-computer, with a series of very successful machines starting with the PDP-1, PDP-8, PDP-9, PDP-10 (a timesharing system) and the PDP-11 (which started out small and became a major timesharing system and ultimately a high-end workstation) The 3 companies formed what was known as *DIX*, the collaboration of Digital Intel and Xerox targeting the business of Local Area Networks. In 1980, they determined that they needed to make Ethernet into a standard.

In 1980, DIX approached IEEE about forming a standards committee to standardize Ethernet. They assumed that they could go to IEEE, form a standards committee for it, and, with all they had running, and given the maturity of the work, they would have a standard in three months. Not a problem.

In those days, most computer standards committees would be 10 to 15 people from different companies coming together to write a standard. This was true for the ASCII committee, Fortran, Algol, etc. These were not big events and didn't attract a lot of interest. This is what the DIX organizers expected.

The first day of the IEEE 802 meeting, 350 people showed up! It was complete bedlam. Building consensus with 350 people is impossible! The attendees had all sorts of ideas about what to do other than just Ethernet as well as changes to Ethernet! This was mortifying. The DIX group already had considerable product in use internally and with customers. There were groups that didn't like the non-deterministic nature of Ethernet and wanted something deterministic for process control, and then there was IBM, who wanted to do anything other than what everyone else was doing.

In addition, there were some interesting clashes of culture going on. James Pelkey, a friend, has just finished a book that he started 30 years ago on the business history of networking. In the 80s, he interviewed almost everybody in the networking business. This is really important now, because having that picture of what people were thinking before all this became really big is very interesting. Over the years, I have reviewed and edited chapters and interviews for him. One of the people he talked to was the president of Concord Data Systems, a modem company in the Boston area that had spun out of another company that made modems. The president was a real electrical engineer working at the physical layer, not really a computer person at all. He was at that first 802 meeting and his comment at the time was very telling. In his interview with Pelkey, his reaction to that first IEEE 802 meeting was, "What the heck is going on here? Why are all these computer people here? This is a data comm problem, not a computer problem!" For him, the only problem was how to build a cheap modem for those speeds. What he had missed was that the others saw was that Ethernet enabled distributed computing. (There were people at the time who became too enthralled with Ethernet and claimed it to be a whole new paradigm. As usual, it turned out to be just another data link technology.) There were two groups in that meeting: the ones who saw this purely in data comm terms and the ones who saw this as distributed computing.

There were several different physical layer proposals that finally boiled down to three. There was the standard Ethernet, which was essentially datagram packet switching on coax. There was a token bus proposal from the process control community who wanted a deterministic protocol. And, finally, in classic standards behavior, IBM proposed token ring, another deterministic approach but not the same as token bus.

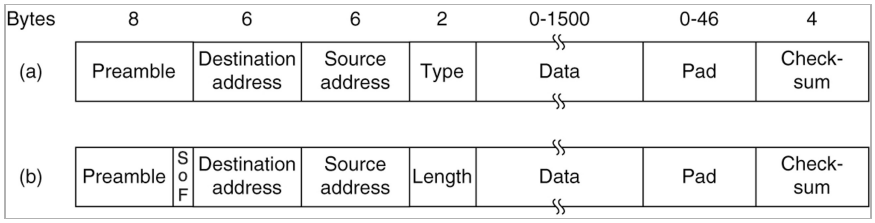
The IBM proposal represented a common stratagem. In standards, the major players in an industry will line up again around a standard, but the dominant player will do something different, just to be different. This has happened from the beginning in computing: the industry did ASCII, IBM did EBCDIC; the industry did Algol, IBM did PL/1; the industry did Ethernet, IBM did token ring; the industry did Java, Microsoft did .NET; and so on.

Classic Ethernet MAC Sublayer

Read

Read Tanenbaum Chapter 4, Section 4.3.2, pp. 292–296.

Classic Ethernet MAC Sublayer Protocol



Frame formats. (a) Ethernet (DIX). (b) IEEE 802.3.

Source: Tanenbaum, A., Wetherall, D., & Feamster, N. (2020). *Computer Networks* (6th ed.).

This is what an Ethernet header looks like. First of all, there is a preamble, 64 bits of alternating ones and zeros. This synchronizes the hardware to the Manchester encoding. We saw in the physical layer chapter that, in Manchester encoding a transition from high to low is a *one* and from low to high is a *zero*. By sending this pattern of alternating ones and zeroes, it trains the electronics to the width of a bit.

Then there are the two addresses, 48-bit MAC addresses. There is a type field that indicates what the syntax of the data is, i.e., what protocol is encapsulated. Then there is padding and a trailing checksum.

Most of you have heard of MAC addresses and that they are 48 bits. The designers wanted to avoid the problem of having to configure devices when they joined the network, especially if all it took was tapping into the cable. The problem of assigning addresses to devices on a multi-access media could get very involved. They wanted to make it as easy as possible. But the question was how to ensure the uniqueness of the address without configuration. To do that, they determined the best way would be to assign the addresses when the interfaces were manufactured. If the address was global unique, then someone could plug in on any Ethernet in the world and be up and running! Simplicity itself! But how long should the address be?

Up to this point, header overhead was always a critical factor. Address fields (if they existed) in other protocols were often 8 bits or even less. But with 10 Mbps Ethernet, longer headers were not the overhead they had been. It was possible to be generous!

The way they arrived at 48 bits was that some bright guy at Xerox observed that 248 kg was greater than the weight of Earth. Therefore, that should be enough addresses to name everything that would be on an Ethernet somewhere. A computer isn't going to weigh much less than a kilogram! Famous last words!

Okay, so we have 48-bit Ethernet addresses now. Strictly speaking, these are not really addresses because

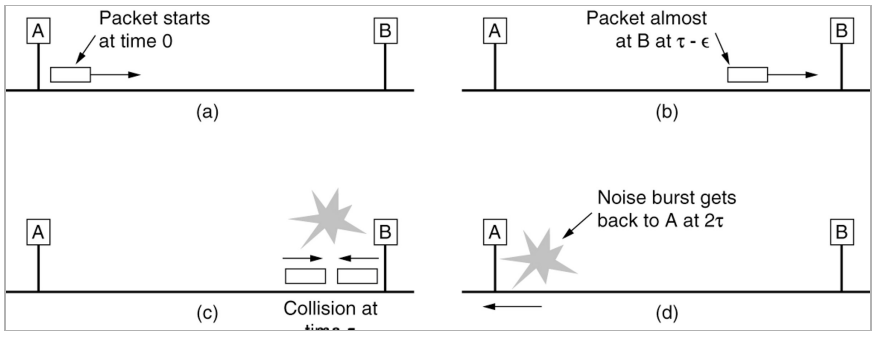
they're assigned at the factory. (Addresses should be location-dependent and route-independent. More on that later.) At the link layer (because of its small scope) and in small networks, enumeration is a perfectly reasonable way to assign addresses, which is basically what we have here. This was what Xerox was proposing.

Of course, standards committees couldn't leave their hands off it, so when the standard was finally approved, the "official" header had a Start of Header field marker for some reason, the 48-bit source and destination addresses, and the Type field became a length field. Otherwise, it was the same. (It is likely that there were more radical changes proposed, but there was considerable pressure not to vary too far from what DIX had done because there was a lot of product already sold that used the DIX header.) This standard was produced by IEEE 802.3; 802.4 produced Token Bus and 802.5 produced Token Ring. (802.1 was architecture, and 802.2 was Logical Link Control.)

IEEE maintains a registry for MAC addresses. Manufacturers buy a block of MAC addresses. The first 24 bits is a manufacturer's code and then the manufacturer can do whatever they want with the other 24.

While it's understandable why they did it, the MAC address was a bad idea. Its scope is too large for the layers it is used in. It makes enrollment and initial configuration very easy. However, the 48-bit address was really a device ID. Consequently, using it for things other than addressing has made it a real security problem; it becomes a way of tracking traffic from a certain device. And some bright people had the incredibly dumb idea of putting MAC addresses in the base of IPv6 addresses, which would make all your traffic trackable. Not to mention that, from a naming and addressing point of view, it is architecturally a very bad thing to do. We will consider that in a future lecture.

Classic Ethernet MAC Sublayer Protocol



Collision detection can take as long as 2*t*

Source: Tanenbaum, A., Wetherall, D., & Feamster, N. (2020). *Computer Networks* (6th ed.).

Ethernet must be able to detect collisions. The whole idea of the cable as the Ether was that all of the transmitters could hear each other. To detect collisions, Ethernet has two requirements: the maximum length of the cable and the minimum PDU size. (This is why there is a Padding field in the PDU to be able to send less data than the minimum PDU size.) For Ethernet, the minimum PDU size was defined as 64 bytes and

the initial maximum cable length was 1 km. How this works is quite interesting.

They are chosen so that (see figure above) if a system at one end of the cable starts to transmit a PDU, and just before it reaches a system at the far end of the cable, that system starts to transmit a PDU, the first system has to still be transmitting when the PDU being sent by the second system reaches the first. (See figure above). Sending a PDU will raise the voltage, let's say 1 volt. When the second system starts to transmit, it will raise the voltage as well so there are 2 volts on the cable, and both systems will know that there is too much voltage and that there has been a collision. But the 2-volt value has to be on the line long enough for the other end to detect it. Hence the cable cannot be longer than the minimum PDU size. (Referring to the figure of the PCI above, there are 28 bytes that must be in every PDU, therefore if a station sends less than 36 bytes of data, padding will be required.)

Ethernet Performance

Read

Read Tanenbaum Chapter 4, Section 4.3.3, pp. 296–297.

Switched Ethernet

Read

Read Tanenbaum Chapter 4, Section 4.3.4 pp. 297–300.

Needless to say, Metcalfe's idea of snaking a cable around the lab and having vampire taps was never very reliable, as you can well imagine. Poking something through the cable to try to contact a conductor in the middle of the coax and make it continually have good contact is just not practical. Not only that, but running a cable around a graduate student lab is fine, but it doesn't work all that well for a real office environment. Very quickly, people started talking about hubs and Ethernet switches in a wiring closet. A hub is a pure electrical relay of one collision domain. With a switch, each port is a separate contention domain. As people have recognized, the cables must be centered at wiring closets. It makes it a whole lot easier, and so running cables back to a wiring closet with switches or hubs became common.

Fast Ethernet

Read

Read Tanenbaum Chapter 4, Section 4.3.5, pp. 300–302.

Gigabit Ethernet

Read

Read Tanenbaum Chapter 4, Section 4.3.6, pp. 302–306.

10-Gigabit Ethernet and Above

Read

Read Tanenbaum Chapter 4, Sections 4.3.7-9, pp. 306–309.

Read Tanenbaum's Retrospective on Ethernet. There are some good observations there.

---

1. Palo Alto Research Center

Wireless LANs

This brings us to 802.11. We have come full circle. The wireless Aloha network technology inspired Ethernet in the early 1970s as a multi-access wired technology, and that was so successful that by 2000 it had inspired a new wireless technology called WiFi.

802.11 Architecture and Protocols

Read

Read Tanenbaum Chapter 4, Section 4.4.1, pp. 309–311.

There are basically two configurations: 1) a star network to an access point connected by a wired Ethernet network, and 2) a wireless ad hoc network where the clients are all talking among each other.

802.11 Physical Layers

Read

Read Tanenbaum Chapter 4, Section 4.4.2, pp. 311–314.

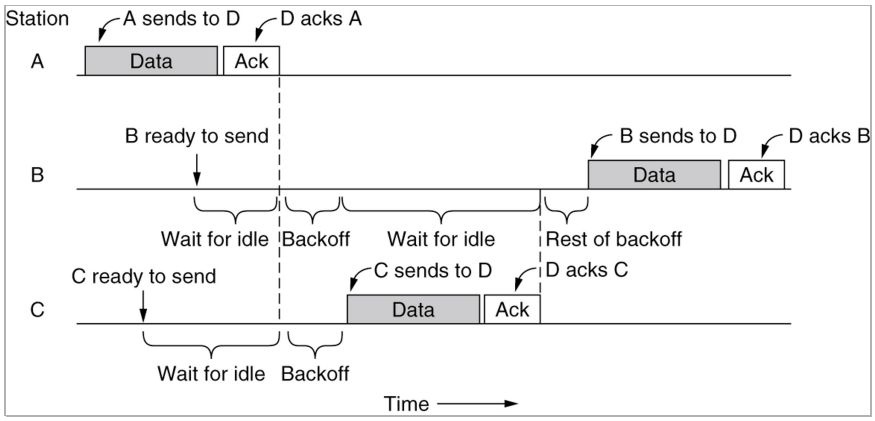
802.11 MAC Sublayer

Read

Read Tanenbaum Chapter 4, Section 4.4.3, pp. 311–321.

How WiFi works is very interesting. There is some very good engineering going on here. It is impressive how they use very tight timing in the protocol.

The 802.11 MAC Sublayer Protocol (1 of 4)



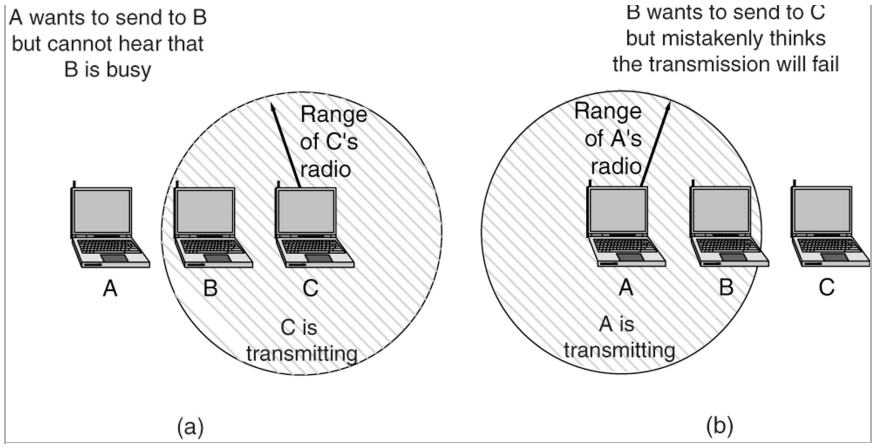
Sending a frame with CSMA/CA.

Source: Tanenbaum, A., Wetherall, D., & Feamster, N. (2020). *Computer Networks* (6th ed.).

As we saw earlier with wireless, we can have collisions and the hidden transmitter problem, so WiFi uses RTS/CTS. We can't do collision detection because the signals are too weak. In the figure, A has been sending data. During that time, B and C become ready to send. When A receives the Ack, the channel is idle. Rather than trying to send immediately, both B and C do an exponential back off. C picks a smaller time, and C sends first. B suspends its back off countdown, when it senses C has the channel. When C finishes, B resumes its countdown. Note that this means that others will start a back off, and its likely B has a shorter countdown to do and will thus get the channel next, thus providing fairness and avoiding starving B.

There are two modes of the Coordination Function: DCF (distributed) and PCF (polled). PCF isn't used much because of potential interference from other nearby APs. The assumption in 802.11 is that if the sender doesn't get an Ack, then the data transfer never took place.

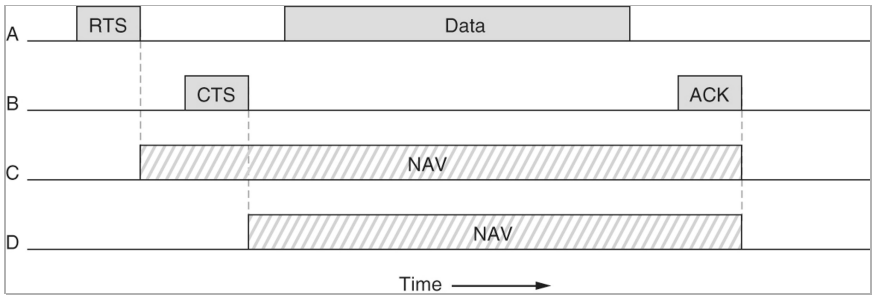
The 802.11 MAC Sublayer Protocol (2 of 4)



(a) The hidden terminal problem; (b) The exposed terminal problem

Source: Tanenbaum, A., Wetherall, D., & Feamster, N. (2020). *Computer Networks* (6th ed.).

The 802.11 MAC Sublayer Protocol (3 of 4)



Virtual channel sensing using CSMA/CA

Source: Tanenbaum, A., Wetherall, D., & Feamster, N. (2020). *Computer Networks* (6th ed.).

To avoid the hidden transmitter problem and that not everyone can hear everyone else, 802.11 does an RTS/CTS, just as we talked about earlier. 802.11 does exactly what we talked about before: the sender will send out an RTS to clear the channel and wait for the CTS. Given the variable length of PDUs, 802.11 does something very interesting. There is a field in the RTS called the *NAV*. Its value is the number of microseconds it will take to send the frame and receive an Ack. The CTS contains a NAV reduced not just by the amount of time that has passed, but the time it took the PDU to get to the receiving station and send a CTS. The channel is now reserved for this long.

Now, all stations in range of A and B have heard the RTS and know not to send. Now B needs to send a CTS to notify all stations in the range of B, which may include stations not in the range of A that it is reserving the media. When A gets the CTS it sends the data packet and waits to get an Ack. If, for some reason, no Ack arrives, A acts as if it never happened and tries again. This is a stop-and-wait protocol.

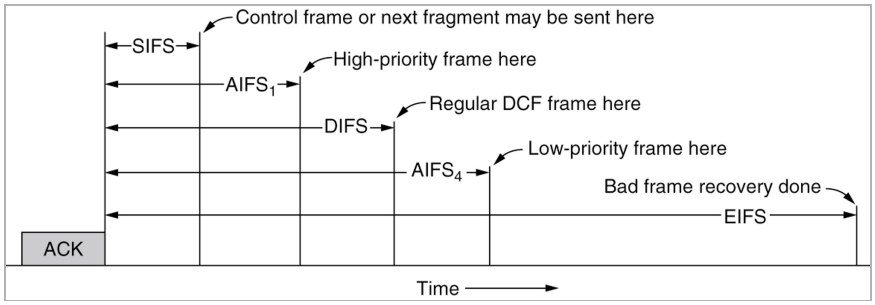
Previously, we have seen that stop-and-wait is slow. But everyone is using WiFi, but it doesn't seem slow. What is going on? Stop and think. What is the range of WiFi? About 300 meters.

With WiFi, the signal strength is such that it is guaranteed that a 1500 byte Ethernet packet was essentially the maximum distance of the access point. If one computes the physical length of a 1500 byte WiFi PDU at initial WiFi data rate, one finds that it is approximately 300 meters, which is about the limit of WiFi.

There is a more general result here. No matter how fast the medium is, if time for a PDU to get to the destination is roughly the length of a PDU, in other words if the first bit is arriving at the destination just as the last bit is being sent, then stop-and-wait is the best that can be done.

As WiFi data rates have gone up, there's room for more than one packet in flight in time.

The 802.11 MAC Sublayer Protocol (4 of 4)



Virtual channel sensing using CSMA/CA.

Source: Tanenbaum, A., Wetherall, D., & Feamster, N. (2020). *Computer Networks* (6th ed.).

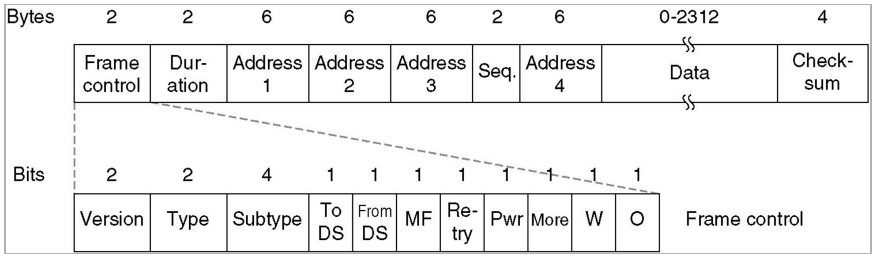
After each transfer is there is a contention period. (One problem was how to allow both Polled and Distributed Coordination to operate at the same time. The contention period resolves that.) What happens in the contention period is that there is a period of time after the Ack when others can attempt to get the channel.

802.11 Frame Structure

Read

Read Tanenbaum Chapter 4, Section 4.4.4, pp. 321–322.

The 802.11 Frame Structure



Source: Tanenbaum, A., Wetherall, D., & Feamster, N. (2020). *Computer Networks* (6th ed.).

Let's take a look at what a WiFi packet looks like, which is very different. There are a number of fields in frame control. Duration is where the NAV value goes. There are three MAC addresses! That is pretty strange! (Keep that in mind.) There is a sequence number and possibly a fourth MAC address! That's even stranger. Then we have up to 2312 bytes of data and a 4-byte checksum, a CRC32.

The frame control contains a version field (2 bits), a type field, which indicates what kind of command, a Subtype (4 bits) that indicates the particular command, To DS, From DS as to whether it's to or from the AP, a More Frames bit, a Retry bit, a Power Saving bit, whether it is on, a More Data bit, Protected, and Order.

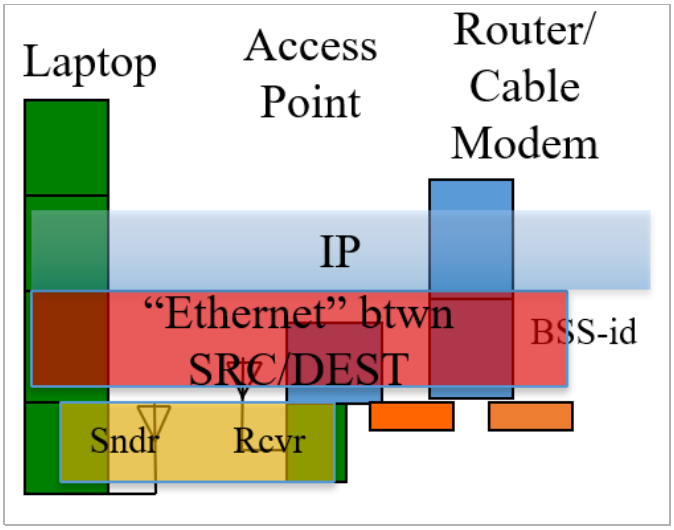
The peculiar thing here is multiple addresses; why would there be four addresses? Think about why there are addresses in the first place. Generally they occur in pairs: destination and source. So we have two pairs of addresses here. Why might one want two pairs?

Let's take a look.

Why So Many Addresses?
met_cs535_mod2_lec4_slide42_animation video cannot be displayed at the printable page. Please go to the module page to watch.

Why are there two pairs of addresses? Because it is two layers. Two pairs are needed when the network configuration is relaying from one access point to another access point before reaching a router on the wired network. One needs to specify the relay points and the original source and final destination of the PDU. In essence, two addresses define a "virtual" Ethernet between the laptop and the access point, SRC/DEST; then, one must specify the next hop, labeled RCVR/SNDR. SRC/DEST would remain the same, but RCVR/SNDR would change at each relay.

The 802.11 Frame Structure



But many (if not most) WiFi networks are just devices connecting to an access point, a star network. In that case, two of the addresses would always be the same. So, there is no reason to have the fourth address. One can also interpret that third address as identifying the network. They call this the *BSS-ID*.

We haven't had to identify the network before. But think about it; how many WiFi networks can you see? Five, 10, 20? Those are designated by the BSS-ID. The 32-character string you see on your laptop is called an SS-ID and represents the BSS-ID. So, it is the name of the WiFi network. A laptop or other device joins one of these networks, which we have called Enrollment; 802.11 calls it *association*.

802.11 Services

Read

Read Tanenbaum Chapter 4, Section 4.4.5, pp 321–324.

Security and Priority/Power Saving

Power saving in 802.11 is interesting. They recognize that turning on a transmitter requires more power than turning on a receiver. Therefore, power saving is driven by the access point. At precise intervals, the access point sends out a Beacon command with a bitmap for the stations for which it has PDUs waiting. Wireless devices have to turn on their receivers at precisely the right time to receive the Beacon command, determine if its bit in the bit map is set and if it has turned on the transmitter, and receive the PDUs, or if the bit is not set, it goes back to sleep.

Bluetooth

Bluetooth Architecture and Bluetooth Protocols

Read

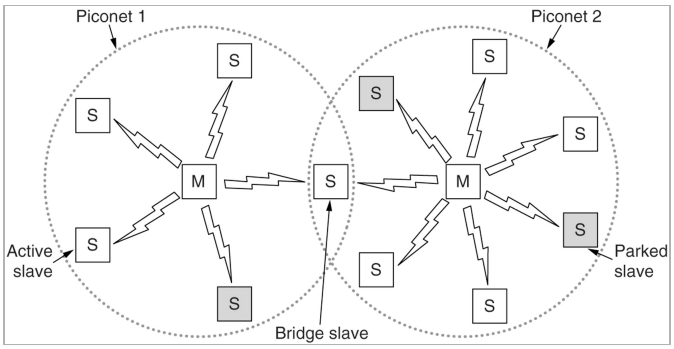
Read Tanenbaum Chapter 4, Sections 4.5.1-2, pp. 325-327.

Bluetooth has had an interesting lifetime.

You all know about Bluetooth. You have all seen it. Many or most of you probably use. I know I use it, much to my chagrin. Bluetooth was developed to clean up the rat's nest of wires around your machine. Developed by Ericsson, they thought that there would be a market for a wireless system with an even shorter range than WiFi (~10m). The advantages of Bluetooth were supposed to be that it would be less expensive and low power, longer battery life. When I first heard about it, my response was that WiFi was further down the cost curve and if the antenna gain is turned down, so it has less range and uses less power, then what is the point of Bluetooth? Actually, Bluetooth was and probably still is more complex than WiFi. As noted above, the market for Bluetooth was to replace the wires connecting all the peripherals around a person's desk. But what they failed to recognize was that once you got rid of the wires, people didn't want those devices near the desk at all. Before WiFi and Bluetooth, my printer had to have a cable running to it, which kept it near my desk. Without the wire, it could be in the next room.

Bluetooth had to find another use. It had a little bit of trouble figuring out what the market was before it really settled into cars and specialized peripherals, such as keyboards, mice, trackpads, headphones, etc.

Bluetooth Architecture



Two piconets can be connected to form a scatternet.

Source: Tanenbaum, A., Wetherall, D., & Feamster, N. (2020). *Computer Networks* (6th ed.).

The architecture left a bit to be desired. It is nothing but a heap of special cases. It can't really be called a

design. It was a star network with a master/slave relationship between the devices. There was an upper bound of seven active devices on a Bluetooth network, called a *piconet*.

Of course, once you hear that you think, “Wait a minute! Don't tell me some idiot put a three-bit field in the protocol!” Because you know there will be pressure to increase that number. (Another example of not doing a management-proof design.)

Then, just to show you how smart these guys were, the initial press releases talked about how piconets could be connected into Scatternets. Of course, you are thinking: How can you have a master/slave pico-net and connect them as in the figure above? How can it belong to two piconets at the same time? How can one “slave” device have two masters? In fact, they didn't discover the contradiction in their design until a group of Bluetooth vendors were trying to do a DEMO at a major trade show! How embarrassing!!

When you open the first official Bluetooth overview document it shows this figure as their architecture. There was basically a different profile for every application. This was done by Ericsson, which was originally a telephone company. To support headphones, there is a purely analog stack that goes straight from the radio up to the device.

Everything is a special case. There is a stack for each application. I have talked to people who were on the committee who confirm that it's just a wealth of patches and kludges.

But what really made me crazy while reading the Bluetooth specifications is that I found that, yes, there was a 3-bit field for the identifiers for the active devices in a piconet, but there was also a 6-bit field for 64 *non-active* devices, and there was a MAC address. We know MAC addresses are globally unique. If there is a MAC address, what are these identifiers for active and non-active devices for?

They are really more a mode rather than something that needs to be identified! They have the MAC address to identify them. They could have had a mode defined as active or passive and a bit somewhere indicating which type it was. Perhaps for performance reasons, it is desirable to have an upper bound on the number of active ones. There is a variable somewhere, probably a byte to hold the current upper bound on the number of active devices. If the upper bound is 7, then store 7 in the variable and the master can check it to make sure it doesn’t activate more than 7. If a second generation of the hardware can handle more, then increase the value of the variable. They didn't need identifiers for it because they had MAC addresses! They built constraints into the protocol that were unnecessary and would be very disruptive to change. This kind of thing was common in Bluetooth. It was just one thing like this after another.

There were other things that gave one pause to worry about what one hadn’t yet found. Reading the Bluetooth specifications, I came across a section on naming and addressing, but one in which the identifiers it covered didn’t make any sense. What was it identifying? Why is this naming addressing? I read and reread it. Finally, I work it out: This identifier is the flag character for delimiting the PDUs!

This isn't naming addressing at all, this is delimiting!

It is a little thing, but it makes you wonder, if they don’t know the difference between addressing and

delimiting, what else don't they understand!?

Bluetooth Frame Structure

Read

Read Tanenbaum Chapter 4, Sections 4.5.3-6, pp. 327–331.

Bluetooth 5

Read

Read Tanenbaum Chapter 4, Section 4.5.7, pp. 331–332.

I have to say Bluetooth 5 and Bluetooth LTE have really been simplified and are much better than Bluetooth used to be. There are still a lot of kludges in it (probably unavoidable given where it was), but the protocol is a lot more reliable and lot easier to use than it used to be.

DOCSIS

Read

Read Tanenbaum Chapter 4, Section 4.6, pp. 332–334.

Data Link Layer Switching

Uses of Bridges

Read

Read Tanenbaum Chapter 4, Section 4.7.1, pp. 335–336.

As the uses of Ethernet grew, various issues arose. One of the rules interfered with these issues: There was

a rather strong belief in Dijkstra's observation that once a function was done in a layer, it didn't have to be repeated in another layer. This became not only an architecture rule—a rule about efficiency—but also a claim to turf that is jealously guarded. In this case, while the data link layer is clearly relaying, it shouldn't do routing because we all know routing belongs in layer 3. Or is that true?

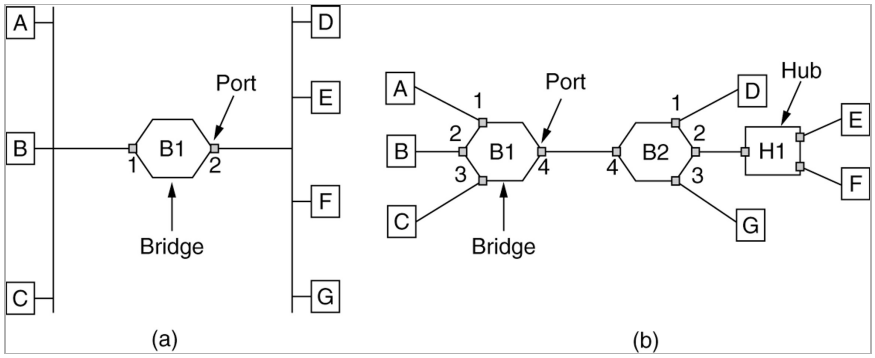
(As we have seen, they were too focused in applying Dijkstra's observation. Functions don't repeat in layers of the same scope.)

Learning Bridges

Read

Read Tanenbaum Chapter 4, Section 4.7.2, pp. 336–339.

Learning Bridges



(a) Bridge connecting two multidrop LANs. (b) Bridges (and a hub) connecting seven point-to-point stations.

Source: Tanenbaum, A., Wetherall, D., & Feamster, N. (2020). *Computer Networks* (6th ed.).

With bridges, whenever a packet is sent, every station on the bridged Ethernet would see it. This becomes a problem. Even if a station is sending to another station on the same Ethernet segment, the PDU will be sent on every Ethernet segment in the bridged LAN. Not every station in the network has to see every PDU. This is not only unnecessary load, but it also increases contention on the other segments. Hence, learning bridges don't actually route PDUs but do avoid sending them where they are not needed.

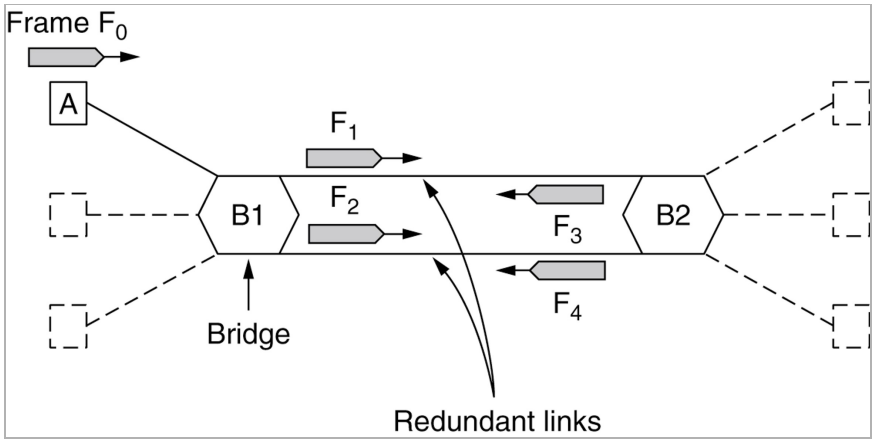
In addition, we learned the hard way not to build large bridged Ethernets. If an Ethernet board failed, it could jam the entire network, which made finding it very difficult. A partial solution to this was learning bridges. The real solution was to break up the bridged LAN with routers to contain the jamming. One might be tempted to call this "passive routing," but don't tell the Layer 3 people!

Spanning-Tree Bridges

Read

Insert Tanenbaum Chapter 4, Section 4.7.3, pp. 340–342.

Spanning-Tree Bridges (1 of 2)



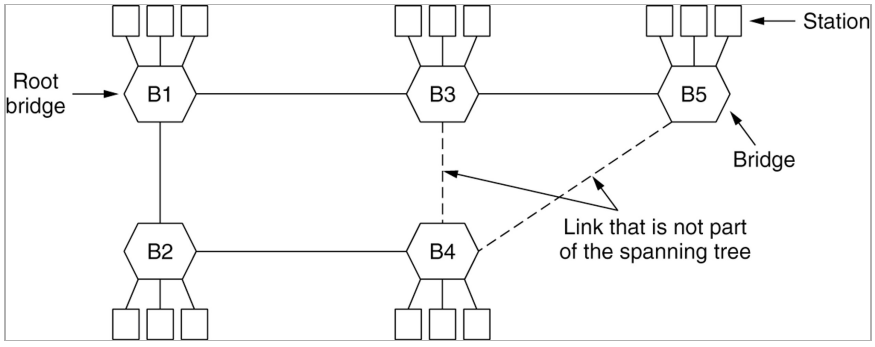
Bridges with two parallel links.

Source: Tanenbaum, A., Wetherall, D., & Feamster, N. (2020). *Computer Networks* (6th ed.).

A spanning tree is a subgraph of a graph that includes all nodes with the minimum number of edges. Hence a spanning tree has no loops.

Redundancy was often introduced into bridged Ethernets so that a single failure wouldn't partition the network. However, this could lead to loops in the network causing packets to loop indefinitely and causing packet storms. To counter this, spanning-tree algorithms were introduced to avoid loops.

Spanning-Tree Bridges (2 of 2)



A spanning tree connecting five bridges. The dashed lines are links that are not part of the spanning tree.

Source: Tanenbaum, A., Wetherall, D., & Feamster, N. (2020). *Computer Networks* (6th ed.).

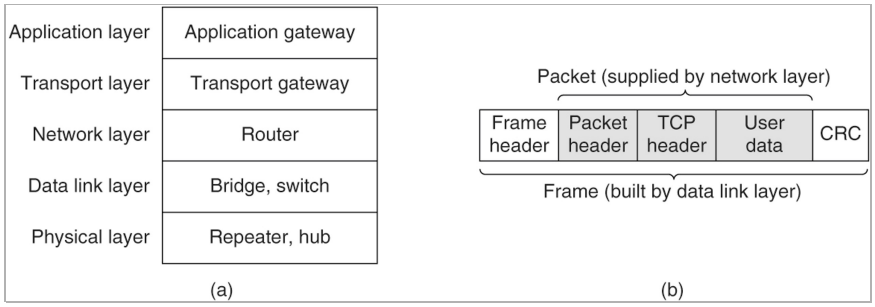
What isn't said here is that this is routing. As we will see in the next module, one of the common routing algorithms works by using an algorithm to generate an optimal spanning tree rooted at each node in the graph. However, most of the people working on this don't understand that is how it works. If IEEE 802 had called this routing, there would have been a big turf battle. Instead, IEEE used the same algorithm to generate a spanning tree from one node, so that all PDUs sent by any station in the Ethernet follow this single tree. By calling it a spanning-tree algorithm, IEEE was lauded for their sophistication in using graph theory to solve the problems and no one noticed (at least no one who spoke up) that it was the same algorithm used for Layer 3 routing. Once we have covered the Dijkstra optimal path algorithm in the next module, come back and re-read this section again and notice how similar the descriptions are. Tanenbaum is very careful not to give away the secret. (This isn't the only place where the ignorance of experts is exploited to do what needs to be done, but it contradicts the "belief system," which some might call religion, and is be prohibited.)

Repeaters, Hubs, Bridges, Routers, and Gateways

Read

Read Tanenbaum Chapter 4, Section 4.7.4, pp. 342–345.

Repeaters, Hubs, Bridges, Switches, Routers, and Gateways



(a) Which device is in which layer. (b) Frames, packets, and headers.

Source: Tanenbaum, A., Wetherall, D., & Feamster, N. (2020). *Computer Networks* (6th ed.).

Repeaters and *hubs* are at the physical layer and simply amplify the signal and re-transmit the PDUs. They don't interpret the PDU at all. They do not terminate a contention domain.

Bridges are at the Data Link Layer and actually do interpret the MAC addresses and make forwarding decisions.

Routers are at the network layer, one layer up, where they terminate a contention domain at the data link layer and contain the scope of a LAN. Routers interpret IP addresses and make forwarding decisions.

Transport layer gateways are often mentioned and sometimes implemented, but they are not recommended. Any relaying can incur congestion and the loss of PDUs. At the Physical Layer, this is avoided by the physics; at the Data Link Layer, by the contention for the media; but it occurs at the Network Layer, and the primary purpose of the Transport Layer is to recover from losses due to congestion at the Network Layer. A transport layer gateway introduces relaying at the Transport Layer and the possibility of congestion and loss with no recovery at the layer above.

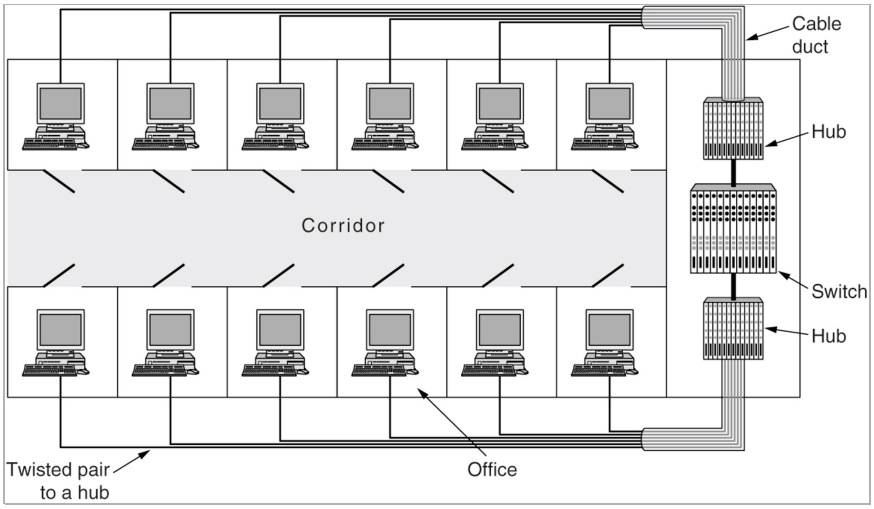
The same can be said about *application layer gateways*. Although, an application layer gateway over a full end-to-end Transport Layer may have such a low error rate as to be assumed to be reliable.

Virtual LANs

Read

Read Tanenbaum Chapter 4, Section 4.7.5, pp. 345–351.

Virtual LANs



A building with centralized wiring using hubs and a switch.

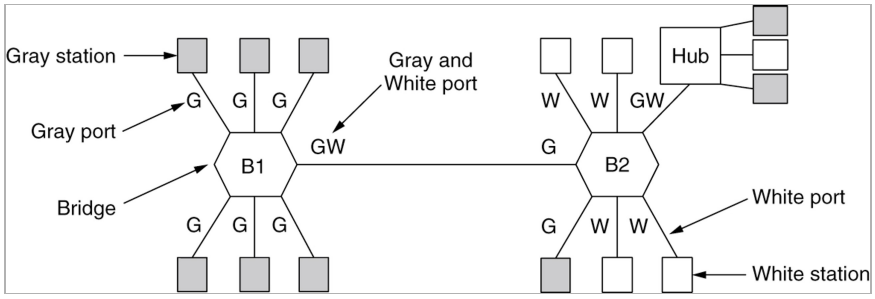
Source: Tanenbaum, A., Wetherall, D., & Feamster, N. (2020). *Computer Networks* (6th ed.).

The other big improvement in Local Area Networks was in the area of Virtual LANs or VLANs. Earlier we saw that people realized that snaking a cable around the office was a pain in the neck and a maintenance nightmare. It was much better to run cables back to a wiring closet.

It is common in offices to want separate LANs for different departments: Accounting, Development,

Marketing, etc. But people move around, people leave, new hires come in, etc., and the logical arrangement of offices starts to disperse. When a new hire arrives, there is the question of where there is an empty office. The office may be over in Marketing, but the new hire needs to be on the development LAN, etc. The equipment in the wiring closet has to be reconfigured to get everyone on the right network. Some tech has to spend (sometimes) days in a hot, poorly-ventilated wiring closet with a list of changes to make, moving cables around. Of course, the cables are not tagged so the tech knows which is which (because we don't have time to do that!). So, this is very error prone. (And to make it worse, since there is a chance it will interrupt people working, it has to be done afterhours!) A lot of reasons to be able to do this remotely. Not to mention isolating networks from each other for security reasons. So why not make it electronic so that bridges can be reconfigured from the tech's desk. VLANs solve this problem by creating logical LANs over the physical LANs.

Virtual LANs

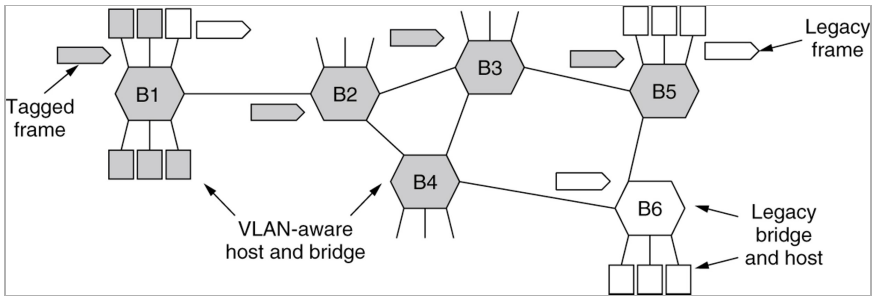


Two VLANs, gray and white, on a bridged LAN.

Source: Tanenbaum, A., Wetherall, D., & Feamster, N. (2020). *Computer Networks* (6th ed.).

As Tanenbaum describes it, VLANs are LANs of different colors, and the packets would be forwarded only to the stations that have that VLAN ID. This also has to be backward compatible.

The IEEE 802.1Q Standard



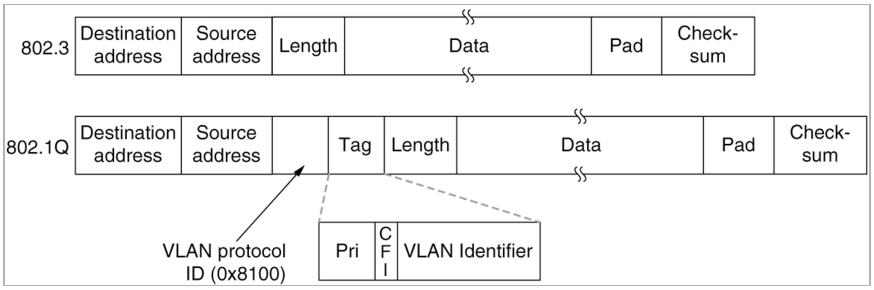
Two VLANs, gray and white, on a bridged LAN.

Source: Tanenbaum, A., Wetherall, D., & Feamster, N. (2020). *Computer Networks* (6th ed.).

For (legacy) equipment that didn't know about VLANs, the first VLAN-aware bridge a normal MAC PDU

encounters modifies the PDU to be VLAN-compatible and assigns it to the appropriate VLAN. To do this, they had to make changes to the Ethernet header. This was huge! If anything was considered written in stone, it was the Ethernet header. The Ethernet header had been unchanged since the 1980s, made a standard in the early 1980s, and now in the 2000s they were proposing to change it!! This was a big deal!

The IEEE 802.1Q Standard



The 802.3 (legacy) and 802.1Q Ethernet frame formats

Source: Tanenbaum, A., Wetherall, D., & Feamster, N. (2020). *Computer Networks* (6th ed.).

What they did, and it was very controversial, is they decided to take the Type field or the Length field (one of the differences between the DIX and 802.3 formats) and make it a VLAN tag. Since VLAN tag is only used by switches, the end equipment doesn't need it (backward compatibility). The first VLAN-aware device adds the tag; the last one removes it.

This increased the MTU from 1518 to 1522 bytes. If the VLAN Type is greater than 1500, then it indicates it is a Type and not a Length. The lower 12 bits are the VLAN-id, Pri is priority, and CFI (believe it or not) indicates the payload is 802.5! So, 12 bits for the tag allows for 4096 VLANs, which sounds like more than enough.

Very quickly, VLANs were put to use in datacenters and virtual machines and suddenly 4096 looked very small. Consequently, the lack of VLAN-ids has become crazy. They are standards for Q-in-Q and MAC-in-MAC. In other words, standards that allow the VLAN-tags to be encapsulated in VLANs and MAC headers be encapsulated in MAC headers. The standards are known as Q and Q, because the LAN is 802.1Q. What the standard is doing recursing, it is Q and Q; and recursing the MAC layer is MAC and MAC.