



## Computer Networks: Crash Course Computer Science #28

Crash Course: Computer Science

<https://youtube.com/watch?v=3QhU9jd03a0>

<https://nerdfighteria.info/v/3QhU9jd03a0>

Hi, I'm Carrie Anne, and welcome to Crash Course Computer Science.

The internet is amazing! In just a few key strokes, we can stream videos on YouTube (hello), read articles on Wikipedia, order supplies on Amazon, video chat with friends, and Tweet about the weather. Without a doubt, the ability for computers and their users to send and receive information over a global telecommunications network forever changed the world. 150 years ago, sending a letter from London to California would have taken two to three weeks, and that's if you paid for express mail. Today, that email takes a fraction of a second. This million-fold improvement in latency - that's the time it takes for a message to transfer - juiced up the global economy, helping the modern world to move at the speed of light or(?) fiber optic cables spanning the globe.

You might think that computers and networks always went hand in hand. But actually, most computers pre-1970 were humming away all alone. However, as big computers started popping up everywhere, and low-cost machines started to show up on people's desks, it became increasingly useful to share data and resources, and the first network of computers appeared. Today, we're going to start a three-episode arc on how computer networks came into being and the fundamental principles and techniques that power them.

\*Intro music\*

The first computer networks appeared in the 1950s and '60s. They were generally used within an organization, like a company or research lab to facilitate the exchange of information between different people and computers. This was faster and more reliable than the previous method of having someone walk a pile of punch cards or a reel of magnetic tape to a computer on the other side of the building, which was later dubbed a sneakernet.

A second benefit of networks was the ability to share physical resources. For example, instead of each computer having its own printer, everyone could share one attached to the network. It was also common on early networks to have large, shared storage drives, ones too expensive to have attached to every machine.

These relatively small networks of close-by computers are called local area networks, or LANs. A LAN could be as small as two machines in the same room or as large as a university campus with thousands of computers. Although many LAN technologies were developed and deployed, the most famous and successful was Ethernet, developed in the early 1970s at Xerox Park and still widely used today. In its simplest form, a series of computers are connected to a single, common Ethernet cable. When a computer wants to transmit data to another computer, it writes the data as an electrical signal onto the cable. Of course, because the cable is shared, every computer plugged into the network sees the transmission, but doesn't know if the data is intended for them or another computer.

To solve this problem, Ethernet requires that each computer has a unique media access control address, or MAC address. The unique address is put into a header. The prefix is any data sent over the network. So computers simply listen to the Ethernet cable and only process data when they see their address in the header. This works really well. Everywhere computer made today comes with its own unique MAC address for both Ethernet and WiFi. The general term for this approach is carrier sense multiple access, or CSMA for short. The "carrier" in this case is any shared transmission medium that carries data - copper wire in the case of Ethernet, and the air carrying radio waves for WiFi. Many computers can simultaneously sense the carrier, hence the sense in multiple access and the rate at which the carrier and transmit data is called its bandwidth.

Unfortunately, using a shared carrier has one big drawback. When network traffic is light, computers can simply wait for silence on the carrier and then transmit their data, but as network traffic increases, the probability that two computers will attempt to write data at the same time also increases. This is called a collision, and the data gets all gobbled up, like two people trying to talk on the phone at the same time. Fortunately, computers can detect these collisions by listening to the signal on the wire. The most obvious solution is for computers to stop transmitting, wait for silence, and then try again. The problem is the other computer is going to try that too, and other computers that have been waiting for the carrier to go silent will try to jump in during any pause. This just leads to more and more collisions. Soon, everyone is talking over one another and has a backlog of things they need to say, like breaking up with a boyfriend over a family holiday dinner - terrible idea!

Ethernet had a surprisingly simple and effective fix. When transmitting computers detect a collision, they wait for a brief period before attempting to retransmit. As an example, let's say one second. Of course, this doesn't work if all the computers use the same wait duration. They just collide again one second later, so a random period is added. One computer might wait 1.3 seconds, while another waits 1.5 seconds. With any luck, the computer that waited 1.3 seconds will wake up, find the carrier to be silent, and start transmitting. When the 1.5 second computer wakes up a moment later, it will see the carrier is in use, and will wait for the other computer to finish.

This definitely helps, but doesn't totally solve the problem, so an extra trick is used. As I just explained, if a computer detects a collision while transmitting, it will wait one second plus some random extra time. However, it collides again, which suggests network congestion, instead of waiting another one second, this time it will wait two seconds. If it collides again, it will wait four seconds, and then eight, and then sixteen, and so on until it's successful. With computers backing off, the rate of collision goes down, and data starts moving again, freeing up the network. Family dinner saved! This backing off behavior using an exponentially growing wait time is called exponential back-off. Both Ethernet and WiFi use it, and so do many transmission protocols.

But even with exponential back-off, you can never have an entire university's worth of computers on one shared Ethernet cable. To reduce collisions and improve efficiency, we need to shrink the number of devices on any given shared carrier, what's called the collision domain. Let's go back to our earlier Ethernet example, where we had six computers on one shared cable, aka one collision domain. To reduce the likelihood of collisions, we can break this network into two collision domains by using a network switch. It sits between our two smaller networks and only passes data between them if necessary. It does this by keeping a list of what MAC addresses are on what side of the network. So if A wants to transmit to C, the switch doesn't forward the data to the other network. There's no need. This means if E wants to transmit to F at the same time, the network is wide open, and two transmissions can happen at once. But if F wants to send data to A, then the switch passes it through, and the two networks are both briefly occupied. This is essentially how big computer networks are constructed, including the biggest one of all - the internet, which literally interconnects a bunch of smaller networks, allowing inter-network communication.

What's interesting about these bigger networks is that there's often multiple paths to get data from one location to another. And this brings us to another fundamental networking topic - routing. The simplest way to connect two distinct computers or networks is by allocating a communication line for their exclusive use. This is how early telephone systems worked. For example, there might be five telephone lines running between Indianapolis and Missoula. If John



## Computer Networks: Crash Course Computer Science #28

Crash Course: Computer Science

<https://youtube.com/watch?v=3QhU9jd03a0>

<https://nerdfighteria.info/v/3QhU9jd03a0>

picked up the phone wanting to call Hank in the 1910s, John would tell a human operator where he wanted to call, and they'd physically connect John's phone line into an unused line running to Missoula. For the length of that call, that line was occupied, and if all five lines were already in use, John would have to wait for one to become free. This approach is called circuit switching because you're literally switching whole circuits to route traffic to the correct destination. It works fine, but it's relatively inflexible and expensive because there's often unused capacity. On the upside, once you have a line to yourself or you have the money to buy one for your private use, you can use it to its full capacity without having to share. For this reason, the military, banks, and other high-importance operations still buy dedicated circuits to connect their data centers.

Another approach for getting data from one place to another is message switching, which is sort of like how the postal system works. Instead of a dedicated route from A to B, messages are passed through several stops. So if John writes a letter to Hank, it might go from Indianapolis to Chicago, and then hop to Minneapolis, then Billings, and then finally make it to Missoula. Each stop knows where to send it next because they keep a table of where to pass letters given a destination address. What's neat about message switching is that it can use different routes, making communication more reliable and fault-tolerant. Sticking with our mail example, if there's a blizzard in Minneapolis grinding things to a halt, the Chicago mail hop can decide to route the letter through Omaha instead. In our example, cities are acting like network routers.

The number of hops a message takes along its route is called the hop count. Keeping track of the hop count is useful because it can help identify routing problems. For example, let's say Chicago thinks the fastest route to Missoula is through Omaha, but Omaha thinks the fastest route is through Chicago. That's bad because both cities are going to look at the destination address, Missoula, and end up passing the message back and forth between them endlessly. Not only is this wasting bandwidth but it's a routing error that needs to get fixed. This kind of error can be detected because the hop count is stored with the message and updated along its journey. If you start seeing messages with high hop counts, you can bet something has gone awry in the routing. This threshold is called the hop limit.

A downside to message switching is that messages are sometimes big, so they can clog up the network because the whole message has to be transmitted from one stop to the next before continuing on its way. While a big file is transferring, that whole link is tied up. Even if you have a tiny one kilobyte email trying to get through it either has to wait for the big file transfer to get finished or take a less efficient route. That's bad. The solution is to chop up big transmissions into many small pieces called packets. Just like with message switching, each packet contains a destination address on the network, so routers know where to forward them. This format is defined by the "internet protocol", or IP for short, a standard created in the 1970s.

Every computer connected to a network gets an IP address. You've probably seen these four eight-bit numbers written with dots in between. For example, 172.217.7.238 is an IP address for one of Google's servers. With millions of computers online exchanging data, bottlenecks can appear and disappear in milliseconds. Network routers are constantly trying to balance the load across whatever routes they know to ensure speedy and reliable delivery, which is called congestion control.

Sometimes different packets from the same message take different routes through a network. This opens the possibility of packets arriving at their destination out of order, which is a problem for

some applications. Fortunately, there are protocols that run on top of IP, like TCP/IP, that handle this issue. We'll talk more about that next week. Chopping up data into small packets and passing these along flexible routes with spare capacity is so efficient and fault-tolerant, it's what the whole internet runs on today. This routing approach is called packet switching. It also has the nice property of being decentralized, with no central authority or single point of failure. In fact, the threat of nuclear attack is why packet switching was developed during the cold war. Today, routers all over the globe work cooperatively to find efficient routings, exchanging information with each other using special protocols like the Internet Control Message Protocol (ICMP) and the Border Gateway Protocol (BGP).

The world's first packet switch network and the ancestor of the modern internet was the ARPANET, named after the U.S. agency that funded it, the Advanced Research Projects Agency. Here's what the entire ARPANET looked like in 1974. Each smaller circle is a location like a university or research lab that operated a router. They also plugged in one or more computers. You can see PDP-1s, IBM's System 360, and even an Atlas in London connected over satellite link. Obviously, the internet has grown in leaps and bounds in the decades since. Today, instead of a few dozen computers online, it's estimated to be nearing 10 billion, and it continues to grow rapidly, especially with the advent of WiFi-connected refrigerators, thermostats, and other smart appliances forming an internet of things.

So that's part one, an overview of networks. Is it a series of tubes? Well, sort of. Next week, we'll tackle some higher level transmission protocols, slowly working our way up to the world wide web. I'll see you then.

\*outro music\*

Crash Course Computer Science is produced in association with PBS Digital Studios. At their channel, you can check out a playlist of shows like BrainCraft, Kominidi(?), and PBS Infinite Series. This episode was filmed in The Chad & Stacey Emigholz Studio in Indianapolis, Indiana and was made with the help of all these nice people and our wonderful graphics team, Thought Cafe. Thanks for the random access memories, I'll see you next time.