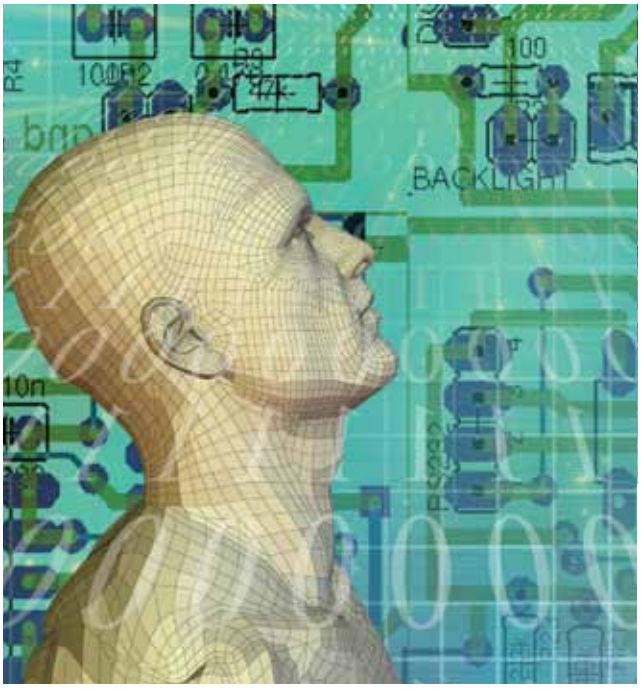

Bob Metcalfe: Ethernet at Forty

Charles Severance

Bob Metcalfe describes how the Ethernet local area network was created 40 years ago at Xerox Palo Alto Research Park.

It's pretty much impossible today to find computing technology that doesn't support Ethernet or that didn't evolve from it, such as Wi-Fi. We simply assume that everything from our phones to our laptops to our printers and backup systems come ready to plug into a high-speed wired or wireless network. In fact, many homes now have both a wired and wireless local area network.

But 40 years ago, LANs didn't exist. The typical approach to distributed computing was to connect terminals in offices throughout a building with serial cables that ran from the back of the terminal to the mainframe. Sometimes, this connection was done through phone lines and a dial-up modem.

I recently spoke with Bob Metcalfe, who described how the Ethernet local area network was "invented" 40 years ago at Xerox Palo Alto Research Center (PARC) on 22 May 1973. Visit computer.org/computingconversations to watch the full interview. Metcalfe is quick to point out that many brilliant engineers contributed to Ethernet and other popular forms of high-speed local area networking over the years. Although it's an oversimplification to give him sole credit for inventing it, Metcalfe was definitely on the front lines all those years ago.

PERSONAL COMPUTERS

In the quest to build the "office of the future" during the early 1970s, the creative people at PARC decided that instead of having terminals connected to a single central computer, they would give every person a "personal" computer and connect those computers together. According to Metcalfe,

> I happened to be at the Xerox Palo Alto Research Center when a problem evolved that had never before occurred—the problem of having a building full of personal computers. I was the networking guy, so they turned to me and said, "Network these puppies." We had just finished starting the ARPANET, which was packet switching, and it was pretty clear that we wanted this [personal computer] network to connect to the [not yet called the Internet] thing.

There was also a desire to connect the personal computers of the future with the printers of the future:

> Our first printer—whose name was EARS, and that is a whole other story—could do a page per second at 500 dots per inch. If you do the math, that's about 20 Mbits per second. Existing methods

 Published by the IEEE Computer Society

of interconnection had a lot of problems. First, they were all "home run," so all these wires, one from every desk, would come to this one place in the building. Second, the existing interconnects ran at 300 bits per second, 14,400 bits per second if you really revved them up, which wasn't even close to 20 Mbits per second. We wanted to keep the printer busy by sending documents to it from all these PCs that hadn't been built yet. We were literally building the printer and the PCs at the same time.

Charles Simonyi designed an earlier effort to network personal computers, called SIGNET (Simonyi's Infinitely Glorious Network), but Metcalfe felt it was too complex and wanted something simpler. He came across a wireless network in use at the University of Hawaii:

> In the course of investigating how to organize Ethernet, I ran into a packet radio network at the University of Hawaii called AlohaNet. What was beautiful about AlohaNet is that it solved a distributed problem. How could we share a radio channel back to the mainframe at the University of Hawaii if we're just a bunch of terminals scattered around the Hawaiian islands and can't easily talk to each other and get coordinated around the sharing of an inbound radio channel?
>
> Norm Abramson at the University of Hawaii devised a randomized retransmission procedure in which a person would type a card image 80 columns wide. After typing in your card image, you would hit "Send," and then your terminal would send it to the mainframe and wait a short time to see if it returned an acknowledgement on the outbound channel. If so, everything was fine, but if there was no acknowledgement, it probably meant that two terminals had sent at the same time.

If the stations detected that the data wasn't successfully sent, they would calculate a random time to wait before retransmitting in the hope that they wouldn't overlap when they retransmitted the data. This allowed multiple computers to share the same media—a radio frequency—by using randomized retransmission. This approach appealed to Metcalfe because he wanted to have multiple personal computers share a single Ethernet cable:

Metcalfe wanted to have multiple personal computers share a single Ethernet cable.

> I was trying to avoid this big rat's nest of wires—I only wanted one wire, not 16 or 32, and I wanted a distributed solution for how to share this single cable.

SHARED CABLE

If Ethernet was going to be a single, long, shared cable, it was important to see how data could be transmitted at high speeds over long distances:

> One of the first things I did was to buy a mile of cable. Then I hooked up a pulse generator to one end, hooked an oscilloscope to the other end, and started launching square waves down the cable to see what came out the other end. I figured this would be good preparation for building a network. But what came out the other side wasn't a square wave: it had a lazy rise time and lazy fall time. If you put a digital gate on the receiving end, you could recover the square wave. I had some confidence that if we could get the stations connected to the cable, they could inject their square waves, and the other stations could recover them.

Once Metcalfe was convinced that he could reliably send high-speed data over long distances using coax cable, it was time to define the details and build the hardware. He was joined by David Boggs, who had some experience working in a television studio. Boggs helped with both the hardware and software design.

The team decided to use Manchester encoding to send the bits over the cable. In Manchester encoding, the first half of the bit time is the opposite of the bit's value, and the second half of the bit time is the bit's actual value, guaranteeing a voltage transition in the middle of every bit:

> The beauty of Manchester encoding is that while you were sending a packet, you could tell whether it was going by so you didn't have to listen for it for long—usually, 340 nanoseconds.
>
> One of the first differences between the Ethernet and AlohaNet was this carrier sense. In AlohaNet, you couldn't tell if someone else was transmitting at the same time as you, but on the Ethernet, you could. By waiting, you avoided destroying each other's packets.
>
> The Ethernet rule was that before sending, a station would listen first to avoid stepping on ongoing packet transmissions. This meant that once you had been sending a packet for a short while, you would have "acquired" the Ether and could continue without interference. The maximum packet length was limited to ensure shared access to the Ether.

Another advantage of Manchester encoding was detecting collisions after you had started transmitting a packet:

> PARC's Ethernet could pull the cable up to a voltage or leave it open with no voltage. If you are leaving the cable open (half the time under Manchester encoding) and if you

detect the cable pulled up anyway, then you have detected a collision.

In addition to carrier sense and collision detection, each packet had a source and destination address so that each workstation or printer could identify the traffic being sent to it:

> The addresses were 8 bits, so on the backplane of these little personal computers, we would wire wrap in a code between zero and 255, and that would be the machine's serial number. You would read the address off the backplane and put it in the packet. Having two addresses was different from AlohaNet, which had one address because it had two one-way channels.
>
> We also added cyclic redundancy checksum (CRC) on the end of the packet, which we implemented in hardware so that you could tell if a packet had been damaged. If there was a collision, and the contending stations backed off, there would be a hunk of garbage on the cable. When it was received, the checksum wouldn't match, so you would throw the packet away.

In addition to designing the protocol to put the bits onto the wire, the team also looked for a device to allow adding new workstations to the network without taking the network down:

> We didn't have to run a cable through the building and back to the rat's nest every time we installed a new PC. We wanted to put one cable down the middle of the corridor, and every time you wanted to add a PC, you just ran the cable and tapped into the coax. We didn't want the network to go down while tapping into it because we wanted 24/7 access to the network.
>
> This requirement led to a device we found in the cable television industry called the Gerald tap.
>
> David Liddle did cable television installations when he was in grad school in Toledo, and he suggested that we use the Gerald tap because it was already being made in volume and worked just fine. You would drill a little hole in the outer casing of the coax, screw in this tap, and it would puncture the insulation and go right to the copper and tap in.

LAN WARS

Other computing companies became interested in using Ethernet-like approaches and started working with Metcalfe, who decided that the best way to ensure interoperability among the various implementations was to develop a standard, which led to the formation of the IEEE 802 working group. Digital Equipment Corporation, Xerox, and Intel submitted the "Blue Book" Ethernet specification in 1980.

After a long battle, the IEEE 802 working group standardized all three approaches as IEEE 802.3 (Ethernet), IEEE 802.4 (token bus), and IEEE 802.5 (token ring) and let the market work out which technology it would adopt.

But once word got out that the IEEE 802 working group would be developing a LAN standard, several Ethernet alternatives were quickly put forward. IBM claimed its token ring approach was superior, and General Motors championed a token bus as the best approach. The early efforts of the IEEE 802 working group were fraught with politics as the three solutions fought for supremacy. Ultimately, after a long battle, the working group standardized all three approaches as IEEE 802.3 (Ethernet), IEEE 802.4 (token bus), and IEEE 802.5 (token ring) and let the market work out which technology it would adopt.

Given the slow process, DEC, Intel, Xerox, and 3Com (Metcalfe's newly formed company) decided not to wait and simply started building and shipping interoperable Ethernet hardware to an eager marketplace. One of the keys to 3Com's rapid success was that personal computer vendors didn't want to build network hardware onto the motherboards until the IEEE process had reached a conclusion. This meant that for many years, the only way to get Ethernet support for a personal computer was to purchase and install an expansion card. For a while, 3Com was selling well over a million Ethernet expansion cards per month.

While 10 Mbits seemed fast enough for personal computers in the mid to late 1980s, the Ethernet community always felt the need to go faster. According to Metcalfe:

> In 1992, I was involved in Grand Junction Networks, a company that would introduce the 100-Mbit Ethernet. I remember a group of us at my home trying to think of how we would make a faster Ethernet. Efficiency depends on the diameter of the network in bit times, and as you go faster and faster, the efficiency goes down. We realized that since the market had switched to using hubs, we could assume a maximum cable length of 100 meters instead of 1,000. And that was the factor of 10 that we needed! By changing the collision interval, you can maintain the same theoretical efficiencies by assuming that you're going 100 meters instead of a kilometer. That got us to 100 Mbits per second.

Later, the IEEE 802.11 (Wi-Fi) standard implemented an Ethernet-like protocol using wireless transmission. Over the years, there

have been improved versions of IEEE 802.11 with increased speeds. But even 100 Mbits wasn't fast enough for the Ethernet community:

> Then we went to gigabits, followed by 10 Gbit, which is the mainstream now. You can't be a computer scientist and build that kind of hardware now—you need to be a real hardware engineer. But after 100 Gbits, we'll want terabits, and I've already begun giving talks about terabit Ethernet.

Ethernet used AlohaNet as a starting point and built on the concept of a shared transmission medium and randomized retransmission when data was lost. But a few design innovations from Bob Metcalfe, David Boggs, and others who built that first Ethernet at PARC form the foundation of nearly all modern LAN technologies: adding source and destination addresses to every packet, carrier sense, collision detection, and CRCs. These patterns led to relatively simple LAN hardware solutions that are inexpensive to make and scale to very high levels of performance, while making efficient use of the medium's available bandwidth. These patterns have served us well over the past 40 years.

Ethernet's 40th birthday will be celebrated in style on 22-23 May 2013 at the Computer History Museum in Mountain View, California. It will be a gala event with industry briefings and all the many Ethernet inventors invited to come and share in the festivities and tell their stories.

***Charles Severance**, Computing Conversations column editor and Computer's multimedia editor, is a clinical associate professor and teaches in the School of Information at the University of Michigan. Follow him on Twitter @drchuck or contact him at csev@umich.edu.*