Early Token Ring Work at MIT

J. Noel Chiappa

Editor: David Walden

Token ring local area networks (LANs) are now obsolete technologies, but early work on them at the Massachusetts Institute of Technology produced networks that have left their mark on today's LANs. I start by telling the story of early work at the Laboratory for Computer Science (LCS) at MIT on token ring LANs (which we usually simply called "rings," not "token rings," as was later usual industry practice). Some of the more technical content below may bring back memories for people who lived through the era of this technology.

Background and Beginnings

LCS got involved in rings almost by accident, and I too wound up involved in the ring work (and much else besides!) similarly almost by accident.

In the mid-1970s, high-speed LANs were an obvious area of work, connecting up the various mainframes and mid-size machines found in a typical computing site of the day. Such LANs were designed to provide relatively high throughputs—those days 1 Mbit per second was very fast—to modest numbers of computers over a geographically fairly restricted scope, such as one building or a small group of them.

A particular LAN concept can be characterized along two main axes: (1) the method used to pass data to the interface stations connecting user computers to the LAN and (2) the access control method used to mediate access to the LAN's resources. The former was often indicated by what people called the "topology" of the network, such as a "bus" or "ring," which usually implied a particular data-distribution technique.

In the case of token ring LANs, they had point-point links between pairs of stations that joined them all in a ring, with each station repeating the data to its "downstream" neighbor. They also had a "token" (a special bit pattern) that circulated around the ring, and a station that wanted to send a packet waited to see the token, removed it from the ring, and then sent its packet, finally emitting a replacement token to indicate that the ring was free.[1]

Ethernet was a rival LAN technology being developed at the same time. Both the 3-Mbit/second Ethernet prototype done by PARC and the follow-on Dec-Intel-Xerox (DIX) 10-Mbit/second commercial version used a single wire bus to which all the stations were attached and used Carrier Sense Multiple Access with Collision Detection (CSMA-CD) as the control method. Very briefly, this meant that a station listened for quiet before sending (carrier sense) and stopped if its transmission collided with another (collision detection), retrying after a short (random) back off—similar to having a conversation at a crowded dinner table.

However, in the late 1970s, LANs weren't available off the shelf. Anyone who wanted one had to build their own. So the Computer Systems Research (CSR) group in LCS applied to DARPA (then LCS's main funding source) for money to build and deploy a LAN. DARPA was already funding a group at the University of California, Irvine (UCI) under Dave Farber to build a 1 Mbit/second ring, a development of a previous generation ring called the Distributed Computing System. Naturally enough, DARPA said, "Why don't you go in with their project?" So MIT, unwisely perhaps, agreed.

I came on board the project around then, in the fall of 1977, via a fortuitous (for me!) set of circumstances. As a computer science student at MIT, I had some ideas for an operating system I wanted to work on, and I thought Jerry Saltzer (who I had gotten to know fairly well via taking a course where he was a recitation instructor) might be able to help me find a way to try out my (crazy!) ideas.

Jerry, it turned out, was off on sabbatical at IBM, so I was shown in to the acting group leader, a person I'd never heard of before, David Clark. Dave listened with some interest as I sketched out my operating system ideas, and as it turned out, they were very similar to some work he'd done.

After hearing me out, he had a proposal: they were about to take delivery of the prototype ring interface called the Local Network Interface (LNI), and DARPA had provided them with a PDP-11/40 with which to test the LNI. The LNI was designed to plug into a Unibus, initially the standard I/O bus for the PDP-11 family of computers. The PDP-11/40 was a medium-powered minicomputer. At that time, it was the smallest PDP-11 available that had the capability of running timesharing.

CSR had a lead role in developing Multics,[2] and they had a whole flock of people who knew how to program Multics, but none of them knew anything about PDP-11s. As luck had it (for me), at that point in time, the MIT Computer Science Department's introductory course, 6.031, taught people to code in PDP-11 assembler in the first part of the course. So although I didn't know much about PDP-11s, I knew more than anyone else in CSR.

Dave offered to let me use the PDP-11/40 to work on my operating system ideas if I helped with the LNI by writing diagnostics for it. We were both pretty enthusiastic about the former, but it turned out that those ideas never went anywhere. Instead, I got completely involved with all the networking stuff, including shortly thereafter, working on the Internet; even then, networking was obviously going to be a really big deal.

 Published by the IEEE Computer Society 1058-6180/14/$31.00

Anyway, I signed on for this deal and started to learn more about our PDP-11, which was running Unix. I'd never worked with Unix, although a friend of mine in a group that was running one had briefly showed me a bit about it, and it had favorably impressed me (as it did so many others). Thus, my first step was to read all the available Unix documentation from cover to cover. That, plus a bit of experimental detective work, left me in good enough shape to work out how to write simple stand-alone diagnostic programs and load them from disk.

Making the LNI Work

About that time, the first prototype ring interface arrived. It had never even been plugged in; the group at UCI had designed it, and some combination of MIT and UCI had had the prototype produced by a commercial shop (I don't know the details), and it was then shipped to us.

The LNI Version 1 (V1) was a 5-1/4-inch high unit for the standard 19-inch wide rack used to hold larger PDP-11 systems. (There was a V2 design later on—more on that shortly.) The V1 consisted of what was effectively a single large circuit board, using wire-wrap, a now obsolete technology in which each chip socket had tall pins for each contact, and clever devices took a wire, stripped an inch or so of the insulation, and wrapped the resulting bare end around the pin. Unibus cables connected it to the rest of the computer (originally PDP-11s and later on a VAX-11/780 as well).

The LNI had been designed at UCI by Mike Lyle and Paul Mockapetris (of later DNS fame), who were graduate students at the time. Paul did the portion of the hardware that stored a table of eight dynamically loaded names, and Mike did the rest of the hardware (the ring and bus interfaces).

The digital hardware design that UCI produced was not what it could have been—to our cost. For example, the V1 LNI used binary counters for state machines. It took the encoded output from the counter chip and ran it into an *n*-bit binary decoder chip so that, as the state counted up, one output after another became live in a sequence that was, in theory, always ordered. However, the changes in the outputs of the counter sometimes had some timing variation between them as it counted up, and as a result, as it counted from N to $N + 1$, the outputs could briefly show some value other than N or $N + 1$. The decoder dutifully processed the (brief) alternative value, and so every so often one got glitches on the output lines, glitches that drove us half crazy trying to find and fix.

The LNI was a pretty complicated beast.

Also, the LNI was a pretty complicated beast. The people at UCI had decided that for their application (a distributed computing system that supported mobile processes and so forth) they needed complex naming capabilities. For this, the LNI had the ability to have "mask" bits in both the names (a.k.a. addresses) stored in the LNI and in the message header. Think of them as multicast on steroids.[3] A LNI subsystem called the *name table* held eight dynamically loaded 32-bit names (including masks) that applied to that interface.

The LNI had three major subsystems: the Unibus interface, the name table, and the ring interface. Although each of them had only a limited connection to the other two, unfortunately there was no way to fully debug one independently of the others. Thus, it was always an "exciting" challenge to figure out where exactly the latest bug was. A lot of the LNI's hardware footprint was devoted to the name table, although it turned out that that part of the LNI actually worked fairly well. The problems we had to find and fix turned out to be mostly in the interface to the Unibus and in the hardware to drive the ring.

Ken Pogran, a staff member in CSR, was tasked with debugging the LNI. An MIT graduate, while at school he could not decide between hardware and software, so although he was doing programming just before the LNI work, he was ideal for the LNI and enthusiastic about the chance to do some hardware work. Soon after the start, we hired a technician to help him, Joe Ricchio. Ken set to debugging the LNI, and it quickly became clear that we were in for a long slog.

I don't recall exactly the first problem we found; my vague memory is that the first diagnostic I wrote simply tried to read a register in the LNI and retry it if the attempt to read it failed. That kind of short, closed loop you could watch on an oscilloscope, but within a few days, it became clear to Ken that an oscilloscope was not going to cut it as we

got further into debugging more complex functions. At that point, they hired one of the first logic analyzers to help, which was a wise move—I don't think we'd have ever gotten the LNI fully working without it. We first rented it, but we eventually wound up buying it—we used it for so long it was cheaper that way!

The Unibus had several operating modes. In the simplest, the CPU could read or write from something attached to the bus (which could be either memory, or a device). A more complex operation was direct memory access (DMA), in which a device asked the CPU to let the device use the bus, and the device then did a read or write cycle to computer memory itself. Finally, devices could use the bus to request an interrupt on the CPU.

The LNI had to do all three of these, and we spent some time getting that all to work. The DMA had to work well because the LNI didn't have packet buffers; the packet had to be stored in computer memory as it arrived.

We then moved on to trying to get the ring interface part to work, and that turned out to be fairly complicated and bug-ridden too.

In addition to the normal processing of tokens, an LNI had to be capable of initializing the ring—that is, introducing a token into a quiescent ring. When sending a packet, the sending station drained the packet off the ring; all the other stations (including any destinations) merely watched the packet as it went by. To prevent spurious tokens seeming to appear in data inside packets, the LNI had to look at the data in the packet it was sending and "bit-stuff"[4] to convert any output data bit patterns that looked like a token into an alternative form. This was a process that the receiving station would of course have to undo.

The V1 LNI made heavy use of then-new programmable logic arrays (PLAs), which we wound up reprogramming a lot as we got the LNI working. The logistics of that were challenging because we didn't have a PLA programmer for a while, and we had to depend on the kindness of the salesmen to use theirs!

Still, getting all that to work turned out not to be the only problems. Others issues involved more purely electrical problems. The link from one LNI to the next was a single twisted pair, with the data and clock combined into a single self-timing signal. A big problem (one we didn't really understand at the time) turned out to be that the coding scheme used to combine the data and clock wasn't guaranteed to have a "0" average voltage, which produced a varying DC offset (via the capacitance of the cable) and made the signal harder to discern.

However, the biggest problem I remember had to do with clocking. There was no master clock for the whole ring; each LNI had its own. In fact, there was no master station, which would have been a single point of failure—all the LNIs were identical. To deal with the slight variation between the clocks on different interfaces, the LNI used a system where each bit time was divided up into six slices (the LNI had a 6-MHz master clock that was divided down to 1 MHz to provide a clock for the ring), and an LNI that discovered that it was skewing from its neighbor could add or subtract a slice to a particular bit on the input side. The output side was always sent at that particular LNI's native clock speed. I seem to recall that we had some grief getting that all working properly because the rest of the circuitry had to stall or skip one high-speed clock cycle while the ring interface portion kept running.

Eventually we did get the V1 LNI operating properly, although it took a lot of work. The prototype unit had been wired with blue insulated wire. When Ken made fixes, he used red wire. By the time he was done, the first prototype had an incredible amount of red wire in the parts of the board that interfaced with the bus and the network. Only the part where the name table was stored was mostly still blue.

About the time we were finally getting it running, we had a visit from a VIP (someone from DARPA, I assume), and I was tasked with writing a demo. All we had at that point in the way of software was small diagnostic programs to do things like send a packet and so forth. I quickly whipped up a simple demo that involved the user typing on the keyboard of a terminal, sending the character around the ring in a packet, and then displaying it. (I don't remember now whether this demo used two computers or one. It may have been two terminals attached to a single computer.) Obviously, this would have been easy to fake, but as I recall, it seemed to suffice.

The V2 LNI

We produced eight V1 LNIs and put them into service to provide data transmission to a number of the time-sharing machines at LCS that were not fortunate enough to be on the Arpanet.[5] However, it was clear that mass production of network interfaces was not an appropriate use of LCS's resources. We then

teamed up with a company called Proteon to design and produce the V2 LNI, a follow-on 10-MBit/second commercial product. (Proteon's boss, Howard Salwen, had shared an office in graduate school with LCS Director Michael Dertouzos.)

At that point, Proteon was mostly a specialty analog communication shop, building things that could be described as "*N* GHz uplinks for NASA" in small quantities. Involvement with the V2 LNI changed their future completely.

The V2 LNI was not an evolution of the V1, but a wholly new design. It came as a set of two smaller boards. We split it into two boards because it would be necessary it into plug into machines other than just machines with a Unibus, so there was a host specific board (HSB) and the ring interface card (CTL). The V2 LNI used a totally different system for the clock, a sophisticated purely analog system where the clocks in all the machines in the rings independently adjusted their clock speeds until there were an integral number of bit times around the ring.

This too took forever to make work properly. As we deployed a larger number of stations on a single ring, the whole system became unstable and would not lock up, so it had to be modified. In a later follow-on 100-Mbit/second product from Proteon that became the basis for the open Fiber Distributed Data Interface (FDDI) ring standard,[6] they went back to the original V1 LNI approach of using digital means to adjust bit times, with independent, free-running clocks in each interface.

In addition to the different ring clocking mechanism and a higher speed (10 Mbits/second), we deleted the entire complex name table system. We felt that that function, if needed, would be more economically implemented in a software layer. The programming spec for the interface was considerably redone based on our experience with the V1—an exercise that ended up in a memorable blow-up between Jerry and I when I refused to follow his instructions on one particular detail!

The detailed design of the interface to the host computer was also entirely different; the problems involved in interfacing between an asynchronous bus (the Unibus) and a system with a fixed clock (the ring) led me to indicate to the engineer at Proteon who was designing the host interface card, Henry Arbour, that we should use a design technique that MIT was then pushing in its introductory digital design course, 6.032, which used entirely unclocked logic.

There weren't really any efforts in the standards world to pick one standard (be it bus or ring), so everyone did their own thing.

We did all that, and the V2 LNI Unibus host interface card worked well from the start. The prototype had only about two dozen fix wires on it, unlike the V1 LNI. With small changes, the V2 LNI Unibus host interface card was modified to produce host interface cards for the Q-Bus (a lower-cost bus from DEC for the LSI-11 family) and the Multibus.

The V2 LNI, and its successor, the 100-Mbit Proteon ring, went on to become successful products for Proteon, although the full story of Proteon is left for another day.

Legacy of the V1 LNI

The V1 LNI did contribute one thing that remains with us today: its network wiring pattern, originally called the *star-shaped ring*. The main downside of the ring seemed to be the repeater aspect (because a failing node could take down the whole ring), but MIT did a lot to address that aspect.

Obviously, with a hard-wired ring, where each machine is an active repeater, if one machine is powered off, the ring ceases to work. Putting a relay at each machine to pass the signals through if the machine is powered down is one approach to deal with this, but it has potential issues. First, if several machines in a row are powered off, because of the increased wire length, it's unclear if the signal will make it from the last machine prior to the powered off group all the way through to the next functioning machine. Second, if an interface suffers a hardware failure in the repeating circuitry, that can also take out the entire ring.

With input from Jerry Saltzer,[7] MIT developed the star-shaped ring concept, which includes a central wiring point and a wire that runs out to each interface and then back to the wiring point. For the V1 LNI, the device at the center was a passive box, and if

The evolution of Ethernet has led to networking systems that in many ways closely resemble the star-shaped ring networks of the past.

you wanted to power down a machine, you had to trundle over to the wiring point, unplug the line to that machine, and plug in a jumper instead. For the V2 LNI (and other later rings), there was a relay on each port at the central box that normally bypassed the cable run out to each node, and the interface included a line from that computer to energize the relay and cut that node into the ring.

IBM turned out a ring design (done by IBM Zurich, completely independent of the MIT work, which reached many of the same design conclusions) that was eventually standardized as IEEE standard 802.5. It originally operated at 4 Mbits/second and was later upgraded to 16 Mbits/second. It too used the star-shaped ring wiring scheme that had proved to work quite well.

There weren't really any efforts in the standards world to pick one standard (be it bus or ring), so everyone did their own thing, letting the market decide which technology would win out. The whole process extended over a very long time.

The initial Xerox Experimental Ethernet preceded the MIT work, and the MIT V1 LNI preceded the DIX Ethernet, which was roughly contemporaneous with the V2 LNI. (I remember Jerry getting information through the grapevine, before the DIX specification became public, that the DIX Ethernet was going to operate at 10 Mbits/second, so he insisted we boost the speed of the V2 LNI to match). The IBM ring LAN came along some time later, shortly before Proteon developed the 100 Mbit/second ring.

Rings and Busses

Rings eventually faded away, which could lead one to think that bus systems "won." In fact, the typical Ethernet of today is fundamentally different from the original Ethernet. Ironically, the evolution of Ethernet has led to networking systems that in many ways closely resemble the star-shaped ring networks of the past. The hub-and-spoke wiring scheme for LANs that rings pioneered has proven its utility and is now ubiquitous.

Ethernet had originally used the Xerox PARC technique of deploying a single wire (the bus), running the length of the network and having individual stations tap into that wire (originally, via actually drilling a hole in the coaxial cable used back then). That large bus has been replaced by a system of point-point links that run from host interfaces to active repeaters/hubs/switches, which are similarly interconnected among themselves by point-point links. This produces a star-shaped wiring scheme (without the relays), which consists (at the electrical level) exclusively of point-to-point links.

Why did this happen? A number of factors led to the systems we see today.

The ring people had initially seen two advantages to token rings over CSMA-CD busses.[8] First, the analog electrical environment of rings was much simpler—one transmitter, a wire, one receiver. Second, theoretical studies indicated the token access control would behave better than CSMA-CD at very high loadings. (This latter turned out to be a nonissue because LANs were never loaded that highly for long periods of time.)

Although the latter point was not significant, the former was: point-point links are both easier to work with from an analog point of view and are more welcoming to new transmission techniques, such as optical. In fact, an early MIT ring deployment was an optical link from the LCS building, the famed 545 Technology Square, to the main campus. Getting there involved going under some train tracks, which was difficult. Proteon produced an optical link using a laser, and we set one up in a window on each side. For Ethernet, doing point-point cable runs to a multiport hub, which placed all the transceivers on a short bus inside the box, solved the analog issues with connecting lots of stations to a large bus (noise, reflections, and so on).

Also, even with only point-point cable runs, the CSMA-CD access control method did not scale well in either speed or distance. Ultimately, the real, unavoidable problem with CSMA-CD (one not clear in the early days) was that, as the speed and/or physical size of the network increased, it needed longer packets to make sure it didn't have an "unseen" collision.

In other words, with CSMA-CD, there was a fundamental relationship between speed, network physical size, and minimum packet size: increasing either of the first two required increasing the last. The CSMA-CD access control method could not scale up.

Early Ethernets often used analog or digital repeaters (to reduce analog noise, reflection, and other issues), but the contention zone over which the CSMA-CD mechanism functioned extended out to all the interfaces connected to such repeaters. Use of analog or digital repeaters (any of which could clean up the signal) could reduce the analog issues, but they couldn't get around the CSMA-CD's fundamental problem of its speed/size limitations.

When 100-Mbit/second Ethernet appeared, to accommodate stations operating at different speeds, the connecting nodes became bridges (packet switches) and not simply bit-by-bit repeaters. In other words, the contention zones disappeared because the network became a collection of packet switches.

With all the wires (both switch-switch and switch-interface) being point-point links, and with the links being connected via active circuitry, today's Ethernet is, in electrical terms, far more similar to the early rings than it is to the early Ethernets. The major difference with rings is that there is no circulation (no repetition of data from station to station).

Thus, ironically, one of the major claimed advantages of Ethernet over rings—that its coaxial cable transmission medium was totally passive, with no dependence on the correct operation of active circuitry—disappeared. Not that anyone really cares; in practice, everything works fine.

With today's Ethernet network not really using contention any more as a channel access method, but instead being an interface specification used to connect stations to a packet-switching node, neither the original CSMA-CD busses nor token rings truly live on. Like the screw base for light bulbs (and many other examples), the equipment on either side of the interface has changed beyond recognition, and only the Cheshire Cat's smile of the interface remains.

Acknowledgments

A heartfelt thank you to Jerry Saltzer and Ken Pogran for taking the time to review an early draft of this article. Both caught a number of factual errors and provided helpful editorial comments.

References and Notes

1. A proposal by a particularly clever MIT undergraduate for something called a "contention ring," which still circulated bits around a ring but did not use a circulating token, instead contending for access to the channel á la Ethernet, showed that the circulation is truly the most fundamental thing about ring networks, not the token, which is purely a channel access control mechanism.
2. F.J. Corbato and V.A. Vyssotsky, "Introduction and Overview of the Multics System," *AFIPS Fall Joint Computer Conf.* (FJCC), part 1, 1965, pp. 185–196.
3. D.W. Wall, "Mechanisms for Broadcast and Selective Broadcast," Tech. Report 190, Computer Systems Laboratory, Stanford Univ., 1980.
4. Bit-stuffing refers to a process in which user data that happens to have the same bit pattern as those reserved for control functions on a communication line (such as packet start and end markers) is temporarily replaced, during transmission, with alternative patterns. This process often involves inserting an extra bit in the middle of reserved patterns—hence, the name.
5. F. Heart et al., "The Interface Message Processor for the ARPA Computer Network," *AFIPS Spring Joint Computer Conf.* (SJCC), 1970, pp. 551–567.
6. F.E. Ross, "FDDI – A Tutorial," *Comm. IEEE*, vol. 24, no. 5, 1986, pp. 10–17.
7. J.H. Saltzer and K.T. Pogran, "A Star-Shaped Ring Network with High Maintainability," *Proc. Local Area Communications Network Symp.*, 1979, pp. 179–189.
8. J.H. Saltzer, K.T. Pogran, and D.D. Clark, "Why a Ring?" *Computer Networks*, vol. 7, 1983, pp. 223–231.

J. Noel Chiappa *is an independent researcher working in information systems architecture and software, principally computer networks. He has been a member of groups developing technical standards for the Internet, such as the TCP/IP Working Group and its successors (up to the IETF), since 1977. He served as the IETF Steering Group's area director for Internet services from 1987 to 1992. Previously, he was a member of the research staff at MIT's Laboratory for Computer and worked with a number of companies, including Proteon, to bring networking products based on work done at MIT to the public. Contact him at jnc@alum.mit.edu.*