Oral History of LaRoy Tymes

Interviewed by:
Luanne Johnson and Ann Hardy

Recorded: June 11, 2004
Cameron Park, California

CHM Reference number: X3800.2007

Table of Contents

LaRoy Tymes

Conducted by the Information Technology Corporate Histories Project

LaRoy Tymes, 06.11.2004

Abstract: *LaRoy Tymes describes his career as a computer operator and programmer at Berkeley Lawrence Laboratory in Livermore and then at Tymshare. When he arrived at Tymshare, the company was running separate discrete data centers for their time-sharing customers and LaRoy designed and developed the supervisor for Tymnet, Tymshare's proprietary network. He also designed a specialized mini-computer, known as the Tymnet engine, to support the Tymnet supervisor. He talks briefly about his career after Tymshare and describes the parallel development of Tymnet and ARPANET.*

Background and Initial Programming Experience

Ann Hardy: I would just like to start with a little background. One of the things that I think is really important for people to realize is what it took to make things happen in those days. So where did you come from and what was your background?

LaRoy Tymes: Okay, we'll paint things in real quick strokes. At 19, I flunked out of the University of Michigan. Psychologically, I was not prepared to leave the community I grew up in, a Dutch Christian Reformed community in Holland, Michigan. And I didn't really know much about the outside world.

Hardy: Did you grow up in Holland?

Tymes: Well, I grew up in Wyoming, which is a community between Grand Rapids and Holland. Most of my relatives were in Holland.

Everybody I knew was Dutch. If you weren't Dutch, you were probably going to hell.

Luanne Johnson: So you went to University of Michigan, but didn't…

Tymes: I wasn't prepared for the adjustment at all. I was having a lot of personal problems. My mother had died; I wasn't getting along with my father, and one thing or another. So I lived in Michigan for about a year afterwards and had a job as a tab operator. Tab operator meaning punch cards, patch panels. At that time I taught myself to program an IBM 1401.

That was the most common computer of the day. I think for awhile there were more 1401's than all the rest of machines combined.

Johnson: I first learned to program on a 1401. I'm very fond of that machine.

Tymes: I can still almost remember the instructions. Incidentally packing up my office the other day, I found an old 940 manual in good condition and a BRS manual. Also in good condition.

Hardy: Did you? You want to give it away?

Tymes: Well, no, but…

Johnson: Well, at least we know you have it. That's the important thing. What we may ask as you to do at some point is to put a letter with your personal effects file that whenever these things get disposed of, whether by you or by somebody else, that those things won't get thrown away. That's our concern.

Okay, so you were a tab operator and you taught yourself probably Autocoder or SPS.

Career at Berkeley Lawrence Laboratory

Tymes: Assembly code, on the 1401. Then, I didn't have enough money for a used car, so I bought a new car which I could get on credit, and started driving. I didn't know where I was

going. I went to Texas and then wound up going west to Pasadena for awhile and then I came up to Berkeley. In the summer of 1963, I was living in a place right on the edge of the campus when I heard about a job for a computer operator at Berkeley Lawrence Laboratory.

I went up there and they said I would probably be better off in Livermore. I'd never heard of Livermore. So I went up to Livermore and was astonished at the transition from Berkeley at 60 degrees to the Livermore Valley at 100 degrees. I went down this country road called Highway 50 -- it was a two-lane road at that time. They talked to me but didn't seem terribly interested and said, "Don't call us, we'll call you." I went back to Berkeley and forgot about it. One of the people I was living with was an electrician making what I thought was fabulous amounts of money. California had just passed the law saying that the electricians' union had to accept people who were not relatives of the people already in the union because, until that point, you had to have a father or an uncle or somebody like that in the union to get in. Now they had to have a certain quota of people who were not related to people already in the union. So, I was going to be an electrician.

And just as I was about to move out, I think it was maybe the third day or so prior to my moving out, I got this letter in the mail from Livermore inviting me out to Livermore to be a computer operator. Had it come three days later I would never had seen it,

They offered me the incredible sum of $425 dollars a month, which was more money than I could imagine for doing something which I didn't really regard as work. So I accepted the job.

I didn't have a security clearance at the time, so I started out in what they called the cooler, which is just a building in a non-secured area. I taught myself FORTRAN II while I was sitting there. I also read the complete works of Sherlock Holmes.

From there I went to an unsecured water pool reactor; it's called a water pool reactor because the core is underwater. You can actually look at it. So I got to play with the research reactor for about six weeks. And then my clearance came through and I was admitted to the Computations Division and was absolutely astonished because I had never seen so much computer equipment. Rows of tape drives and huge, physically huge, computers. I remember the 7030 Stretch which had the first Winchester disk drive. The platters were eight feet in diameter or something like that. And when the heads moved in and out the floor shook.

The console of that machine had something like 10,000 incandescent lights. The CPU drew about 42 kilowatts, and that's just the CPU. And it had a staff of about 8 full-time IBM engineers to take care of it. That was their entire job.

So I set myself a goal, just to amuse myself, of learning to code in Assembly every machine I could find, which I completed except for the Lark. And that was a decimal machine.

I never programmed that one, but I programmed everything else. I just gave myself little exercises. One of them was maybe a little more ambitious than the others. Control Data 3600s had tape units that had pneumatic tape drives, very high performance. They could read tapes at 200 inches per second and start and stop quickly. I found that if I gave a write command to the tape unit and then aborted the write command before it got past the interrecord gap, I could start and stop the tape as often as maybe 300 times a second, which is in the acoustic range. So you could make a lot of noise with those things.

Hardy: He got them to play music.

Tymes: So I wrote the Stars and Stripes Forever. I used the console speaker for the piccolo. It had a drum printer, which I used for the percussion. And I had a row of tape units all synchronized so that they were all pulsing together for the trombones.

Johnson: Did you record any of that?

Hardy: It was great!

Tymes: We put the doors to the tape unit half way down so that they would radiate the sound really good. And you could feel it in your chest. It would go bom, bom, ba bam bom. The whole building would shake when that program ran.

Johnson: I just love these stories. Programming was so much fun in those days.

Tymes: In fact, I wrote music programs for a lot of different computers. I failed utterly on a Control Data 6600, because it would not radiate very well. In fact, that was a deliberate part of its design because, for example, the 7094s had very long cables, and it was common for an operator, if you had a program running, to have two hours with nothing to do. So you would be in another building doing some other chore, but you needed to keep track of what your machine is doing so you set an AM radio next to the intercom. The AM radio would pick up the transmission of the program, which you got to recognize. Every program sounded a particular way and after practice you knew what program it was. And you could monitor your machine on the intercom by listening to this AM radio.

That made the security people unhappy, because it occurred to the security people that a car driving by outside the perimeter of the lab could also pick up these AM transmissions. So, I don't know if they did it deliberately or not, but, for whatever reason, the Control Data 6600 did not radiate worth a darn.

Johnson: Nothing this sophisticated, but one of my clients was a construction company out in Danville. This was the kind of place where the president had a big office with a big lounge and a television set. Of course, back in those days we were always working in the middle of the night to get a dedicated machine to do the debugging.

What we would do is turn on the intercom on the phone next to the console typewriter so that every time it typed a message the sound got broadcast throughout the building. There would be a short message – ta-da-da – at the end of every job step. But if there was an error message, then we would hear a much longer message being typed -- da-da-da-da-da. We'd go back and sit in back in the president's office and watch TV and we could hear the program running through the job steps. But if something came up that was da-da-da-da-da we knew we had to run back to the computer room and fix the problem. Nothing as sophisticated as the AM radio, but I got to see a lot of late night movies while I was debugging.

So when you were at Livermore as a computer operator, you taught yourself to program.

Tymes: Yeah. And also while I was there, I ported the Stars and Stripes program over to the Control Data 3800 and 3400 with mixed results. The problem on the 3800 was that it had a cache, which made instruction timing difficult to predict. I couldn't control the timing of my inner loop, so the piccolo on the consol speaker didn't sound too good.

Johnson: When was this... '63, '64, '65?

Tymes: '65. On the 3400, I immediately burned out the tape controller. And I thought I was in a lot of trouble, because those tape controllers cost far more than I could make in a year. I could never pay for it. And Control Data was actually grateful, because I pointed out a design flaw that they didn't realize was there. So rather than get mad at me, they redesigned the tape controller so it could play my program.

Johnson: Did they distribute your program to other Control Data installations?

Tymes: Oh, yeah, it was a standard part of the Control Data maintenance tape. So it got played all over the world. That was my introduction into real-time programming. You might not think of that as real-time programming, but that's exactly what it was.

Then I got the idea of going back to school and, since the lab was run by the University of California, they strongly encouraged people to go back to school. Cal State at Hayward was just getting started and they were looking for students and were willing to overlook my poor academic record. And I started there and did much better.

Hardy: When did you start at Cal State?

Tymes: I think I started there in '65. I don't remember what month. Then it was going so well, I decided I wanted to finish and get my degree. I told them I was going to quit and go back to school full time. And they said, "Don't quit, we need operators on weekends." On weekends there's nothing to do, because you're monitoring maybe three computers, but they would run programs that ran for hours. So for hours all you had to do was just make sure that everything was okay. So I would sit there all alone doing my homework for an eight-hour shift.

Johnson: What were you majoring in?

Tymes: Math. So that was a perfect job for somebody working their way through school. Then when I graduated they wanted me to be programmer, so I became a programmer with them. I think I got $600 dollars a month.

Hardy: When did you graduate?

Tymes: I got my bachelor's degree in math in '66, I believe. There was a guy who was getting drafted and they needed a replacement for him and he was responsible for supporting the machine shops and the Metrology Division. They had numerically-controlled machine tools that were considered pretty much science fiction at that time.

Not only was he leaving, but he was kind of overwhelmed by the job because he was not much of a programmer. He was thrust into that position against his better judgment. And there were a lot of programs that were running on old obsolete machines that were going away and they needed to be rewritten. It had evolved helter-skelter over a period of many years, so there was no pattern or logic to any of it. Even though they all served a common purpose.

So one of the major assignments I took on was to come up with a logical scheme for doing all

this stuff and have one set of programs for everybody to use. And that's when I had my first date with politics.

Because we went around to the all users of this stuff and asked them: If you could have whatever you want, what would it be? But I want one common set of rules for everybody to follow, one format. And that provoked enormous rage on the part of some people. They wanted things their way with no changes.

And they made all kinds of incredible threats, but I knew that the threats were meaningless. Because since they were in other divisions, all they could was complain to their bosses, who would complain to their bosses, and so on to the top of the hierarchy. And then the complaints would trickle back down, going through the hands of many people who just didn't care. And my boss, Bud Wirsching's, only concern was that as long as I took care of everything and he didn't have to get involved, I was doing a good job. It would be literally months at a time without my even seeing him, even though we were only a few hundred yards apart. So I knew there was absolutely nothing these people could do about it and, since I was the only one supplying this stuff, I knew that I could write the software pretty much any way I wanted to.

I came up with a first set of standards, and documented the whole thing, and said this is how all these shapes are going to be defined, and I wrote some FORTRAN II programs that supplied all this stuff.

Johnson: You said how these shapes would be defined?

Tymes: Yeah. For the machine tools and the drafting department and the inspection department. So everybody for the first time was forced to run on the same railroad tracks. As a part of that, I got to visit other parts of the Atomic Energy Commission empire, especially Oakridge, Tennessee, which was also a real revelation to me. I got to see some of their big magnetic separators, and I never really got involved directly in weapons, but I got to at least approach it. I was involved for the experimental devices. I was involved with every aspect of them except testing.

Hardy: So it was about that time, in '66, when you first talked to Tymshare.

Tymes: When I first talked to Tymshare was in '66. But I was more interested in what I was doing at the lab at that time and I didn't have much feeling that Tymshare was going to go anywhere.

Johnson: Why did you talk to Tymshare? Did they come find you?

Tymes: Yeah. I think they originally approached me to write a FORTRAN compiler.

Hardy: This is because Norm [Hardy] and I, at Tymshare, had worked with LaRoy so we knew he could do all this. We were old friends.

Tymes: Right, I met both Norm and Ann at the lab. And another friend we had in common is Barbara Schell, who was responsible for the Control Data version of the FORTRAN compiler on a 6600…

Johnson: She was working for Control Data at that time?

Hardy: No, she was working for Livermore.

Tymes: We were at Livermore and she was sort of a department of one, and there was another group of people headed by the …

Johnson: So, did Livermore have a unique version of the FORTRAN compiler?

Tymes: Well, no, Control Data supplied a FORTRAN compiler…

Hardy: For the 6600.

Tymes: …that Barbara Shell was maintaining and making available to the lab.

Hardy: Little things change in the computer at each installation and there were some things we had to fix.

Johnson: In other words, the compiler had to be tweaked for that installation, so she had the responsibility for that.

Tymes: But there was another group headed by Hans Bruijnes. His group was responsible for writing languages for the lab. They had their own version of the FORTRAN compiler. They

regarded the compiler from Control Data to be competitive to them and they didn't want other users to use it. They wanted everybody to use their compiler.

So it was intensely political. There was a time when Hans Bruijnes, especially after The Stars and Strips Forever, wanted me to join his group. But I didn't care to work in that kind of stuff. I really wasn't interested in what they were doing. It seemed to me what they were doing was totally unnecessary.

In fact, one of the last things I did before I left the lab…there was a common set of library routines on all the other computers. So if you wanted to use a particular printer or a plotter or anything that was common to all those different machines, you had to use this library. And the library that Hans Bruijnes supplied for the 6600 didn't work very well, but at least it sort of worked. And this library did not exist at all for the package that Barbara Schell was making.

So, in the last two months that I was at the lab, I developed a library that was totally compatible with the other machines, so that you could move from a Stretch or a 7094 to the 6600 without changing a single card in your program.

And not only that, but they ran a lot better than any of the library from Hans's group.

Johnson: That's pretty amazing to go from an IBM Stretch to a CDC machine without changing anything.

Tymes: So, as a result of that, about a year later, after I'd left the lab, I happened to meet Hans at a tradeshow, and I said "Hi " and tried to be friendly and he looked off in the distance and tried to pretend I wasn't there. You could see the corners of his mouth were drawn down and he stomped off. He took it as a personal attack which was not my intention at all.

Tymshare

Hardy: Anyway, so what happened is you wrote these libraries for Barb and then how did you get to Tymshare? It was '68.

Tymes: Well, I left the lab shortly after that and got my master's degree also at Cal State Hayward. And then in February of '68, I got my master's degree, and I wasn't really looking for a job but the Tymshare offer was still there, they still wanted me to write a FORTRAN compiler. So I went to Tymshare and I never did work on a FORTRAN compiler.

I looked at the company and, of course, coming from the lab I knew what big machines could do. I was very impressed with how powerful these big machines were. And it seemed to me that a company like Tymshare needed to have a small number of very large, powerful machines with big disk capacity and, in order to reach their customer base, they would need a network, a communications network, so they could reach out over a large area.

Their business model was to build a large number of very small computer centers and scatter them all over the place, which I thought was not economically viable.

Hardy: LaRoy turned out to be right.

Johnson: Yes. So they didn't have a network. These small centers were connected though, weren't they?

Tymes: No, they were not.

Hardy: No, they were not. They were all standalone centers.

Johnson: So you came in with the idea that what was needed was a network as the backbone to all this.

Tymes: A typical computer center – they only had three at that time -- had a pair of SDS 940's.

Johnson: So they must have had discrete customers for each of the centers.

Development of Specialized Terminals

Tymes: Right. Each one served a very local base and they all had to dial in. And if you lived a hundred miles away that meant calling long distance to dial into your computer center. So I couldn't convince anybody that any of my ideas were viable, and I couldn't do anything about the computer or its storage. That left the third area, the communications part, and the way I dealt with that that was to break it down into pieces. The first piece was the terminals. These 940's had what we called CTE, Customer Terminal Equipment, gear which is pretty expensive.

I thought I could replace that with a minicomputer that could do all the stuff the CTE did at a fraction of the cost, and be programmable besides to accommodate different terminal types and

different baud rates. Depending on the terminal that called in, you would identify your baud rates. You could have different baud rates with the CTE gear too, but each port had to be dedicated to a particular baud rate, which meant two things. One is that you had to have different phone numbers for different baud rates, and secondly, of course, it wasn't being used efficiently, because if you didn't have any 300 baud customers at the moment, it couldn't be used something else. So it didn't make very efficient use of resources if you had many different baud rates to support.

I was so sure that it could be done, I proposed this to Tom O'Rourke, the president of the company at the time. I told him I could do it even though I'd never done it, and for some reason he believed me, and committed to buying a 940 without CTE gear. Had I not been right, there would have been a major problem, because there would have been no way to use that 940.

But, fortunately, everything worked. I remember Tom O'Rourke asked me how I was going to determine the baud rate of the terminal when the user dialed in. And I hadn't given it any thought. So, off the top of my head I said, "Well, I'll just look at the length of the start baud and analyze the character as a pattern. And from the pattern, sampling it many times a second, I'll determine what the first character is."

He said that been tried at GE a couple of years earlier and they decided it was not feasible, so why did I think I could do it. And it hadn't occurred to me that it was a problem. In fact the truth was, it hadn't even occurred to me how to do it.

Until he asked the question. I didn't tell him that. But two weeks later I had it working. And in fact I was able to determine the terminal type by analyzing the pattern of the first character typed.

Hardy: When did you actually get the go-ahead from Tom do that first mini, do you remember? Was it shortly after you came?

Tymes: Shortly after I came, yeah.

Hardy: It was spring of '68.

Tymes: And the very first minicomputer was an SPC12.

Johnson: SPC12, what the heck is an SPC12?

Tymes: It is a 12-bit, very anemic, special purpose processor. But Larry Goshorn, President of General Automation, that Norm Hardy and I both talked to, knew something about real-time programming and he introduced to me the idea of using Boolean equations to sample many ports at the same time.

As an example: the stuff I call “hang and answer logic”. There were two signals that you could read and one you could write. The two that you could read were data-set ready, which told you whether somebody called in or not; carrier detect which told you whether there was a tone coming from a modem or just a coupler on the other end. And data-terminal-ready was the signal that you could control to answer or hang up the phone.

So those are three bits. And you could implement timing considerations and all the other kinds of things that you’d have to go through. For example, once the phone answered, you’d wait a certain amount of time for the carrier to come up, and then you expected the data to arrive shortly thereafter. And the advantage of doing it all in Boolean equations is that if, as in this case, the word size is 12 bits you could handle 12 ports at a time.

Another advantage was that since there were no conditional tests or jumping or anything like that, it did the same thing every time and took the same amount of time to execute every time, which was very important. So you didn’t have to worry about worst-case conditions or debugging special situations. Everything was handled the same way at the same time.

And then to carry it a step further, we didn’t have UARTs – Universal Asynchronous Receiver Transmitters -- in those days, so turning all those bauds into characters was done in software. I developed a method of sampling the data lines, 1200 times a second, storing it in the ring buffer, and then going through the ring buffer after the fact to pull the characters out of it. So it was a very efficient way of implementing UARTs.

Johnson: And what would this have been written in, this software?

Tymes: Oh, everything was Assembler. 100% Assembler for everything.

Larry Goshorn showed me that technique and then Norm remembered that, much earlier, the IBM 704, had something similar for driving the printers. That machine I think had a 36-bit word and you output a 72-bit array so many times per second and that would control which hammers fired in the printer. If the bit was set for that particular thing, the hammer would fire. So you had this big matrix with one bits corresponding to when the hammer was to fire and zero bits everywhere else. And you just output this array whenever the wheels were in the proper

position for what it was you were trying to print. But that was the only example he could think of where people would use arrays, bit arrays, to treat large number of ports.

So Larry Goshorn showed me how you could combine this with Boolean equations to process many things simultaneously and the amount of compute time was the same always, so you didn't have to worry about momentary overloads where you couldn't keep up. In fact, the total amount of compute time for heavy traffic was less than using interrupt routines. With an interrupt routine, there's a certain amount of overhead going in and out of the routines. And you could easily construct cases where, if everybody typed a character simultaneously, you couldn't handle the interrupt routines fast enough to be ready for the next character.

Hardy: Where was Tymshare when you came in 68?

Tymes: They were in Palo Alto on East Meadow Drive.

Hardy: The computer center was at East Meadow, but that was all that was really ever on East Meadow. They had just the office on Distel when they started the company.

Tymes: Okay. Why do I remember East Meadow Drive then?

Hardy: Well, because you probably came over to the computer center to work. Verne and I worked over in that computer center and you probably did too.

Tymes: When I first heard of Distel, it had a bathroom that didn't have a roof. Barbara Mennell complained bitterly about using the bathroom when it was raining.

Hardy: Right. It was awful. Distel was one of the shabbiest buildings ever built.

Johnson: Why were they in this cheap…?

Hardy: Cheap rent. They had no money, so cheap rent made sense. It was pretty new so it looked OK until you had to go to the bathroom.

Tymes: Who was the guy right next to me who was writing SUPER BASIC? At Distel. I don't remember. Anyway he had this idea for writing SUPER BASIC and that was supposed to be a great thing.

Hardy: Was that Arden?

Tymes: No, it wasn't Arden. The name Lewis comes to mind.

Hardy: Dan Lewis.

Tymes: Arden Scott, what did he do?

Hardy: Well, he did the business apps. He did things like RETRIEVE and did something that was almost like the first spreadsheet way back then, for which he doesn't get any credit, but that was the kind of stuff Arden worked on. LaRoy and several of the rest of us were kind of buried in what we today call the operating system or the middleware, because you had to get something out there. And so we tried to make things so people could write their own apps. Arden came along and put apps on our computer so that people could not only could use their own apps, but could do their presentations using some of the stuff he wrote.

Johnson: From the background work that I've done, there was a shift to a focus on the business apps beginning in the early '70s.

Hardy: Well, that's when the market crashed. And all the engineering work went away. Lockheed got unfunded so the only thing to do was switch over to business apps. That's when Arden really came through with business apps. But, you know he had to have all this infrastructure, which was what the rest of us were working on. And SUPER BASIC, crazy as it sounds, really helped a lot, because it was incredibly easy to use. So people could get on the computer and write little programs in SUPER BASIC with, in fact, no training.

Johnson: Like customers?.

Tymes: Well, engineers who had little programming background could use it. And it was a big deal.

Johnson: Up to that point then, most of the customers were really just using the time-sharing facilities writing their own applications using SUPER BASIC and running on top of the middleware software that you guys were doing.

Hardy: Yes, that's right.

Johnson: Anyway, so where are we? We're now in the early '70s right?

Tymes: No, we're not that far. We're in 1968.

Johnson: Okay, let's go back to 1968.

Hardy: Yeah, we got the SPC12 up and running and the CTE equipment out the door, because it was terrible. It was a terrible weak spot in the 940.

Tymes: Major reliability problems.

Hardy: Huge reliability problem. Not to mention all the extra costs for having all these ports that weren't being used most the time, and load balancing and all of that stuff.

Tymes: We had piles of cables on the floor. We spilled peanut butter and jam and there were ants everywhere. I remember the ants; it was just awful. There were piles of cables and modems in great disarray.

The SPC12 era was only a few months. And it was replaced by the Varian 620i, a 16-bit minicomputer. The first ones had 4K of 16-bit words. And I had the idea that for some remote offices, rather than build a new computer center if we only had a few customers and the economic base couldn't justify it, we could put telephone lines out there with 2400 bit modems at either end, synchronous modems, the Bell 201s, and with the 620i, I could time division multiplex up to 21 teletypes over this thing. I couldn't get anyone to believe in Tymnet but I could get them to believe in these little baby steps.

Hardy: Do you remember when the Varian actually went in and the SPC12 went out?

Tymes: Towards the end of 1968.

Hardy: Like October, November.

Tymes: Yeah, somewhere around there. So I got that to work, but it could only serve 110 baud devices which wasn't a problem because all of our customers had Model 33 teletypes at that point. And I could time division multiplex 21 customers on a 2400 bit per second line. And still give them good response.

The next step in the evolution of Tymnet was the 8K 620i, and that was in 1969. 8K of 16 bit words. And with that…I don't know if you've seen some of the earliest papers on Tymnet, but I describe virtual circuits ..

I brought up the concept of a virtual circuit where the circuit was assembled to a fixed port on a 620i, which corresponded to a port on a particular SDS 940. And now the character processing was a little more sophisticated. There were things like character gobblers and stuff like that.

The 940 system was very much developed around full duplex teletypes, which means that we really took full advantage of the full duplex, in particular the ability to type ahead. So if it was printing at you, you didn't have to wait until it was done printing before you typed what was coming.

A little side bar here, the IBM 2741, which I implemented much later, locked the keyboard when it was printing and you had to wait until it was done before it would unlock the keyboard and you could start typing. And I asked an IBM engineer, "Why on earth do you lock the keyboard? What purpose does it serve?" And he was baffled by the question. It was such a stupid question, obviously how else can you control the customer?

So their paradigm was that the purpose of the computer was to control the customer. And my paradigm was the computer was a tool to be used by the customer. Totally different point of view. I was equally baffled by his attitude. We came from different planets, I think.

But, anyway, in order to implement this echo properly, I could no longer echo without the time division multiplexing. I couldn't echo from the 940 anymore, because if you were typing in a normal way, the echo would come…I think it's been determined that there's an abrupt threshold at 70 milliseconds. If you delayed more than 70 milliseconds, it felt like the mechanism was in molasses or something. And your typing error rate went way up.

Hardy: There was a study which established that it was around 70 milliseconds.

Tymes: Yeah, I measured it. And then I thought, well this is just me and I tried it on other people and they all had about the same numbers. So it was something about the way human beings reacted to the acoustic feedback.

When the acoustic feedback isn't there, the phenomenon isn't nearly so pronounced. But on a mechanical thing like a teletype, it was very pronounced and the number was fairly constant for everybody that I ran a test on. The typing error rate went way up.

And it felt terrible. There was almost a physical sensation. So I had to do the echoing from the Varian 620i most of the time, but when they were typing ahead I couldn't echo because otherwise the character would be echoed in the stuff that was typed, so then I would have to defer that, send it into the 940, and then get back into the immediate echo mode and the mechanism for doing that is explained in one of my papers.

Hardy: And so you got remote echo working in late '69?

Development of the TYMNET Supervisor

Tymes: Late '69. Or '70. I'm not sure. But up to this point I was able to sell each little brick to management at Tymshare, in particular Tom O'Rourke, on the basis of an immediate need. If they could see the need for that then it would save them money and allow them to do stuff they couldn't do before. The next step was the really grand step, which was that the supervisor dynamically built circuits, and for that we took a leap of faith because they didn't see any point in it.

Hardy: Okay, so there was never a supervisor with fixed circuits.

Tymes: No. The circuits were all assembled there.

Hardy: But once the supervisor came up, it immediately went to variable circuits, it didn't ever come up with fixed circuit plans.

Tymes: So, then there was a period of over a year…in fact a lot of it was done in Santa Cruz in 1971 when I was living there with David Gardner.

I was putting in lots of hours. I didn't come into work all that often, because I was sitting at home coding and figuring things out. Because I had to write, not only the code for the Varian 620i's, to accept the commands for changing the circuits and build these tables and stuff, but I also had to write the supervisor program itself, which ran on the 940. And the supervisor program would get the user's name and the user's password and possibly supplemental information, like if he did not want the default computer, which computer did he want. We had a Master User

Directory, called the MUD, which had an entry for every potential user, with his name and password and account information, what access privileges he had, all that kind of stuff.

Hardy: All encrypted, by the way, which is far better than they're doing on a lot of computers today. It was one way encryption so if you stole a file, you couldn't get it back. I wish we had a few companies who understood these things today.

Tymes: Yeah, Verne Van Vlear, and what's his first name, Blood.

Hardy: Guy Blood.

Tymes: Guy Blood. They got involved in that, in the MUD. I think you were involved in modifying the 940 operating system.

Hardy: I changed the 940 monitor to work with the supervisor.

Tymes: We had some peculiar requirements. In order to keep up with real time, I couldn't go through the file mechanism. We reserved part of the disk and when I made a hash of the user name, I knew the physical location, a physical sector on the hard drive. So I could go right to it without tumbling through a file directory. And that was essential because otherwise the disk would not have been fast enough to keep up with real time. The disks were physically huge, but not very fast.

Hardy: Very slow.

Tymes: So I was bypassing the normal file mechanism of the operating system. I don't know whether it was you or somebody had to put in a special system call in order for me to do that. I didn't do much maintenance. I think Verne did that.

Hardy: I added the BRS. Verne did the maintenance.

Tymes: Verne did the MUD creation and maintenance. Because another problem that we had was the 940s were extremely unreliable and so were telephone lines. So if you have one single point controlling the net, what are you going to do when that goes down? And the answer was we had multiple supervisors at different geographic locations, and when the Supervisor in Active Mode, SAM, was running, it would send sleeping pills to the others to keep them suppressed.

When it died, the others would wake up and contend with one another for control of the network. One of them would prevail and immediately take over.

Hardy: Not only was it unreliable, but the clock in the 940 was such that you had to take it down everyday, or the clock would go down. So, it was designed to be unreliable.

Tymes: Just in case it did stay up, stay on its feet that long, you had to shoot the thing. Yeah, I forgot that part. So, anyway, a corollary of that was that all the MUD files on the different supervisors had to be synchronized, so if a person changed their password, it got changed on all of them. And I didn't get involved in that. That was entirely Verne Van Vlear and Guy Blood. I was the user of the MUD, not the maintainer of it.

So, anyway, that meant that all the stuff, the supervisor and the 620i's, all had to be brought up together. Everything had to work.

Hardy: We had to do this in one day.

Tymes: One day in November of 1971, we brought it all up and from November of 1971 until about a year ago, it ran continuously without a single system failure. It never went down.

Hardy: No, never went down.

Johnson: In other words, there were components that might have gone down, but because of this built-in redundancy…

Hardy: The network never went down.

Tymes: Telephones lines would go out, nodes would fail. But as a system, it was only booted once and ran for over 30 years. 32 years.

Johnson: Wow.

Hardy: And can you imagine Microsoft contemplating a system that was reliable after 30 years? Thirty minutes, maybe?

Johnson: Wow, that's impressive.

Hardy: It was very impressive.

Tymes: There weren't any test runs, there weren't any prototype runs at it.

Hardy: Yeah, it was just all or nothing.

Tymes: Today we're going to do it. We did it, it ran.

Hardy: I can remember, by that time we were over on Bubb Road, and I can remember the one of the salesman coming into my office at the end of the day and saying, "I've never had so many drinks in one day in my life." Total panic, on the part of the sales force. **Total!**

Tymes: It was going to work or it wasn't. And it worked.

Johnson: Congratulations.

Hardy: And LaRoy did all the coding. You know, he really did **everything**!

Tymes: Yeah, I did 100% of the supervisor and this was all Assembler.

Hardy: And, first shot, it worked. I don't think there's very many people who have any kind of a record close to that.

Johnson: That's amazing.

Tymes: There was one other thing…I had a problem because, up to that time the login for the 940 was rather peculiar in that you had to type your username, which should echo, your password, which wasn't suppose to echo, and various account information, but the number you dialed determined which computer you were going to go to. And the sales force…there were three sales regions which were all very paranoid about one another.

Hardy: Competitive … Crazy competitive!

Tymes: Yeah, that was one of the weaknesses of Tymshare.

Hardy: Absolutely one of the biggest weaknesses.

Tymes: For instance, on a particular node, one sales group would want all the ports on that node for its district to shut out the other two districts. So a customer on the other two districts couldn't get in. And with the supervisor, the port was not committed until the user logged in. So this ability of marketing to shut out the other two divisions was lost, which they weren't happy about.

Another thing was that we had to agree on the format of the login stream, and I innocently asked, "Do you have any ideas?" That produced an extremely emotional meeting with all kinds of stress, and "You have to do it my way or I'll kill you" and all that. So I thought well…

Hardy: This was passionate at times.

Tymes: So I thought, well, from the user's point of view, he doesn't really care. He's going to type one stream that echoes and one stream that doesn't. He doesn't give a darn about what all these fields mean.

So I came up with a format where he could type his name and password. And if he wanted something special, like if he wanted to go to some huge computer that he normally didn't access, he would have to tell me the name of that computer or the number or something like that in addition to his name, so there was this optional subfield for special conditions. After the exasperation of listening to all these marketing types telling me how it had to be done, I just came up with this real simple scheme and did it. And they said customers will never agree to it, we'll lose all of our customers. And after it was up and running, I never heard a single complaint.

The only peculiar thing was, because I wanted to be general about terminal types, they did have to type a character to identify what kind of terminal they had, but they had already had to do that before the supervisor.

And for years, because our initial customers all used teletypes that would print out at a 110 baud ASCII, they got a message: Please type the letter D. And if you had a 300 baud CRT terminal that would come out with a particular kind of garbage, and customers got trained to where they wanted to see that garbage. They didn't want to see the garbage message changed.

In February of 1972, we had our first non-Tymshare customer and that was the National Library of Medicine. In Bethesda, Maryland.

Johnson: That was that poison antidote database.

Tymes: And I had the problem of how do I interface to an IBM 360, I'd never done that before.

Hardy: Then you had to go to different machines, right. You had to go to their machines.

Tymes: Yeah. The IBM 360 thought that everybody used an IBM 2741, obviously. So I had to convert everything to that and, furthermore, since some terminals were faster than others, I wanted to run at 300 baud, which their equipment would not support. So when I tried to get information about it, they would tell me, "Oh, this is the most fantastic equipment in the world, it will run it at all these different speeds, and it will handle all the stuff." Will it run at 300 baud? Every time I asked that I got more hype.

The answer turned out to be that there was one guy who made it his entire living modifying the cards to get this gear to run at 300 baud. There was a multilayered board and he knew where to drill into the board down to a certain depth to cut a trace inside the board to make some patches to make the thing work at 300 baud. And he made a good living going around the country doing that. Because IBM would not do it, and a lot of their customers required it. Talk about a niche business.

But he did quite well at it. So, he modified a bunch of boards so that I could talk to them at 300 baud, which was sufficient to satisfy all the different terminals that we were using. I wouldn't say it was an effortless bring up, but again…when the day came to bring it up, we brought it up, it worked, and it never went down.

Hardy: And they were ecstatic.

Tymes: Bill Combs was the sales/marketing guy. He was our first Tymnet marketing guy. Everybody expected it was going to take months to get this thing working and it just came up and worked. Art Caisse was not involved yet, he came by shortly thereafter.

Hardy: I don't when Art Caisse came.

Tymes: He wasn't there for National Library of Medicine, but I think he arrived shortly thereafter. And then he became responsible for interfacing the non-Tymshare computers. And he was living at the time with Ida Cole, who was director of software for the Apple II and later became some high mucky-muck at Microsoft, but she couldn't get along with Bill Gates.

Hardy: Do you know how to get ahold of Verne Van Vlear?

Tymes: He keeps very much to himself. The last time I got ahold of him was very indirect. He is so paranoid, he doesn't want anybody to know his phone number. I got in touch with him through Susan Sharon because I was an expert witness in a lawsuit. She had his number. It was a lawsuit between a couple of companies about databases. They claimed they had invented all this stuff and I was telling them about this MUD file and I claimed I was doing this as early as 1971. And Verne supplied something or other. It is amazing what things get patented because people don't know what came before.

Hardy: Oh, I know.

Tymes: But I was using commercially all of their concepts in Tymnet more than 20 years before they filed their patent.

Hardy: That is so prevalent today, filing patents on stuff we did at Tymshare, 20, 30 years ago, earlier. And these guys all think they're so smart that they just thought up this great new thing. No, it's been around for 30 years.

Tymes: I have a confession to make, I made the same error. I was very proud of myself for designing this routing algorithm that computes the very best route through the network. I would do things like every line that would have a cost to it, which would vary depending on the kind of terminal you were using; and it was one set of parameters if you wanted to pass this spot to another active terminal; there was another set of parameters if you were moving a file, because all you wanted was a bulk transfer from one computer to another. So the definition of best had a lot of qualifiers depending on what you were doing. But whatever your definition of best was, I could come up with a numerical solution to get the optimum route through the network. And I was kind of proud myself for having figured all that out. And then much later I was reading a description of my algorithm, except they call it Dijkstra's algorithm.

Hardy: Oh no.

Tymes: And I found out this mathematician not only stole my idea and published it and got it named after him, but the guy had the gall to do it 17 years before I even thought of it. That was really nervy of him. Yes, I had to admit he published first. So I've been guilty of the same error. But at least I wasn't brazen enough to try to patent it.

Hardy: That's the trouble these days. People are just patenting everything.

Tymes: Yeah, you come up with something and decide, oh, I'll patent it. Ideas like compression of data, or doing things efficiently … You ever take a look at Morse code? The most common letter is E, that's a dot. The second most common letter is T, that's a dash. Z is a really obscure letter, so that's more complicated.

Impact of Tymnet on Tymshare

Hardy: So just talk a little bit about the impact of Tymnet on Tymshare? I know there are so many ways you could approach that question.

Tymes: Well, the first impact was just to save money, and allow them to grow the customer base much more. But I think a second impact was to allow Tymshare to even survive. Because, I believed passionately at the time, and even with 20/20 hindsight I still believe, that their original business model was not viable.

Hardy: Absolutely. I think they came to believe that, too.

Tymes: I don't think they took anywhere near the advantage of Tymnet that they could have. There were lots of opportunities. For instance, Warren Prince had a banker background. And I tried to sell him on the idea of credit card verification.

Hardy: They did do that.

Tymes: They did do that, but it took them a long time. And by the time they did it, we lost a window of opportunity there.

Hardy: They could have grown in a number of areas, that being one, and taken that entire market instead of somebody else. They missed a lot of markets they could have had. The guys at the top were not technical.

When did you leave Tymshare?

Tymes: I left in 1981.

Development of the Tymnet Engine

Hardy: Okay. What did you do at Tymshare after you got the supervisor up? There was another 10 years in there.

Tymes: Yeah. It soon became obvious that the 940 supervisor couldn't be expanded beyond certain limits. It didn't have enough compute power…

Hardy: The 940 was a little lame to say the least.

Tymes: It couldn't access the disk rapidly enough. We had eight pages of 2K 24-bit words, and that couldn't be expanded. We needed a more powerful machine. So we were looking around for a machine to do it on and settled on the Interdata 8/32, which didn't exist yet, but the 7/32 did exist. At that time, I hired Joe Rinde from Livermore. I don't know if it was his first task or not, but he wound up with the task of porting the supervisor to the 8/32. Then we had a problem. When Interdata finally came up with an 8/32, it was far below what we'd been promised.

Hardy: Didn't meet specs.

Tymes: It was a wimpy machine. The instruction set was good, but their implementation was poor.

Johnson: Interdata was the company?

Tymes: Yes. So then I got the idea that, well, okay, I'll just build my own computer, having no hardware background. And I hired a summer student, I can't remember his name, who got nowhere on it. He just set the schedule back by three months, because I thought he was doing something and he wasn't.

So then I took over that myself and designed the CPU and wrote the microcode to emulate…because we had an investment in this 7/32 instruction set. Joe Rinde had re-written

the supervisor for the 8/32. We couldn't go back to the 940 and we didn't have another machine that we could port it to, so we had to support the existing code. We also had some other code -- John Kopf was writing ISIS at that time.

Hardy: That's right. There was a lot of coding going on.

Tymes: We had a lot of commitments to this instruction set and it was all in Assembly. So because we had this big investment, I thought the shortest way to salvage the investment was to build a proper machine that could execute this code with maybe an order of magnitude more performance than Interdata could provide. And to make it even higher performance, a lot of the more critical routines wherever it was really compute intensive, I would do directly in microcode.

So by cheating and putting some of it in microcode instead of Assembly we got even more performance. And so that lead to the Tymnet engine, and then there were three other people…

Hardy: Okay, when did the first Tymnet engine come out?

Tymes: It was 1974. There were three hardware engineers, Barry Burnsides, Ron Graves, and a third guy. Larry Pizella? John Offenbacker knows them all.

Hardy: I don't know John Offenbacker.

Tymes: He's sort of woven in, in unrelated ways. He played a role in Tymnet even though he was never a part of it.

Hardy: Oh, okay.

Leaving Tymshare

Tymes: His first introduction…in fact, Terenia, who's living with him now, was working for Barry Burnsides at Tymshare.

So John Offenbacker, whom I had never met, hand laid out the PC boards. So he knew who I was even though I never heard of him. And he was part of the reason I left Tymshare.

Hardy: Oh really, why?

Tymes: I'm jumping out of chronology here. I had this idea, at that time… Printed circuit boards were made originally by putting tape on Mylar, on light tables and then you photographed that to make the mask to do the photoetching of the PC-boards.

And then came the Gerber photo plotter, which is a two-dimensional XY machine which had something like a 35-millimeter projector, except instead of enlarging an image on a screen, it condensed the image onto film emulsion. And to draw a trace on a board, it would expose a round dot on this film and then go, dit-dit-dit, wherever the trace went.

So, if your board had thousands of holes and traces, it would take hours of zipping around to make your film master. And a multilayer board, a four-layer board, would require one film master for each layer, plus a silkscreen for the solder mask. So it could take quite a few hours to make up the film on this plotter.

I had the idea for using a photo-typesetting machine, which in those days had a peculiar kind of CRT tube. The face plate of the CRT tube was made of millions of glass fibers perpendicular to the fixed plate. The phosphor on the inside of the tube was optically coupled by glass fibers that it was laying on to the outside of the tube.

You could lay this tube on the film emulsion and expose the film directly. It was also used for strip chart recorders. The primary application was phototypesetting and still is. So I thought if you can do phototypesetting … I looked at the characters in National Geographic and Playboy Magazine and I thought those characters are of sufficient quality so if you could do that artwork on a PC board it would be good enough.

So I had the idea that I could take a Gerber file and translate it, and drive a CRT tube to make the image, maybe just a little bitty image, and then pick it up, move it over an inch, make another an image, and step over the film granite bed. And I thought that would be a great way to make a bunch of money. I had a Commodore Pet computer hooked to a big tangle of wires driving this tube that I got from a phototypesetting machine.

I could image an IC pattern in it -- that's all I could do was just image this IC pattern in my apartment. And this guy named John Offenbacker, who heard of me through Terenia, came over and saw me image this little piece of 35 millimeter film. I guess I did a dog- and-pony good enough to convince him to invest in this thing and we formed a company. And that was 1981, that's when I left Tymshare.

We developed a photo plotter. One of the results is the granite in the kitchen because I was

impressed with what you could do with granite. Something you can set a hot pan on without worrying about damaging it.

So he had his fingers in Tymnet even though nobody in Tymnet knew who he was, except Terenia.

Hardy: What was her name?

Tymes: Terenia Kirbechuk.

Hardy: How do you spell her name?

Tymes: T-E-R-E-N-I-A, but don't ask me how to spell her last name. It begins with K-I-R…

Johnson: She was at Tymshare?

Tymes: She worked for Barry Burnsides, yes. Did drafting. Drew up the blue prints for the Tymnet engine.

And then later on we enhanced the Tymnet engine by adding a variety of peripherals to it. But the algorithm defining the best route to the network, I did partly in microcode and partly in Assembly, and after I wasn't involved in that anymore, nobody understood any of that stuff. So as technology changed, they reimplemented the Tymnet engine as newer technology without ever changing any of the microcode. Until 1996, when MCI WorldCom hired me to redo the code. Prior to that the Assembly microcode ran for 22 years without a glitch.

It went through several hardware changes, and nobody would touch the code because nobody understood it. Sometimes the fact that nobody understands the code makes it more stable.

Johnson: You, yourself, wrote it. I mean it was written by one person.

Tymes: Right. All the examples I can think of, of really reliable code, were written by a single individual, not by a group.

Relationship of Tymnet to ARPANET

Hardy: How does Tymnet relate to ARPANET?

Tymes: The ARPANET evolved at about the same time as Tymnet. It's kind of hard to pick the exact date now. For Tymnet you can say February 1968, because that's when I came to Tymshare and worked pretty much continuously on Tymnet until 1981. But the beginnings of the ARPANET are a little fuzzier, because the ideas for it kind of coalesced from many different places. It was funded entirely by DARPA and served originally nothing but colleges and universities until other entities became involved. The differences were that Tymnet had to be profitable right from the very beginning. ARPANET had no constraint; in fact, it consumed very large amounts of money. ARPANET was able to use 50 kilobyte lines, so the data rates were much higher. ARPANET originally did not have any terminal interfaces; it was entirely computer to computer.

In fact, you could say the Internet is sort of that way today, because whether you look at your PC as a computer or terminal is a point of view, it is sort of the same thing. The distinction between terminal and computer nowadays doesn't exist anymore. Whereas in Tymnet days there was a clear distinction.

ARPANET could not have a centrally controlled point, because it was politically not viable. It had to be distributed. Because it was distributed you didn't have any particular place that had all the information to make the optimum use of the net. So the efficiency never approached the efficiency of Tymnet.

It was vulnerable to logjams, thrashing, problems which the Internet is still vulnerable to today. There is no inherent security built in, which continues in the Internet today. Nobody ever successfully hacked into Tymnet in all of its history.

The Internet is awfully easy to hack into. You can inject bogus packets with bogus source addresses at thousands of different points in the structure and do things like denial of service attacks, and all the usual kinds of stuff. In Tymnet you could never do that, because in Tymnet all traffic flows between the two ends of virtual circuits, which were constructed by a central supervisor. And the supervisor itself communicated with each individual node through its own virtual circuits.

Tymnet from the beginning was aimed at interactive use, character strings, and packets were added. The simulation of packets was added later to be compatible with the ARPANET.

Hardy: When was that?

Tymes: That was ’76, ’78. When we had gateways to Telenet. And the ARPANET was the other way around, hence the introduction of TCP.

Tymnet had no use for TCP. TCP is sort of a way to give the illusion of a circuit on top of a packet structure. Tymnet had flow control, node to node, on every section of every circuit, so it was not possible to flood a particular node with excess data. There’s no such mechanism in the Internet. In Tymnet, there’s no concept of a lost packet or retransmissions because of a lost packet. Whereas it’s required in the TCP protocol.

Time delays through Tymnet were rigidly controlled so you had control or guarantee of latency. That was there from the very beginning. And the Internet to this day still can’t give you any guarantees.

A lot of these differences stem from the fact that Tymnet, at any given instant, is controlled from a central point, whereas the Internet is dispersed. There’s no central control. So at the base, the two most fundamental differences are the control and the fact that internet is packet-oriented kind of like the mail system. Versus virtual circuit-oriented, which is more like the telephone system.

Johnson: Great metaphor that makes it easy to understand.

Hardy: And then would you say that the primary reason that Tymnet didn’t become the Internet, instead of ARPANET becoming the Internet, is just politics? Because it was centrally controlled and politically that was just infeasible?

Tymes: Well, there are two things. One is that nobody at Tymshare ever really…

Hardy: Nobody had a vision.

Tymes: Yeah, nobody had a vision. Nobody took it and ran with it. Because the window of opportunity was wide open for a long time. And it was never really exploited. That was problem number one. Problem number two is a problem that you see even in the Internet today, that the one central thing in the Internet is that there has to be assignment of IP addresses, domain names. And that, as you know, has been a real bone of contention. How dare this company, or that company, be the dictator of our fate. So you can imagine what it would be like.

Oh, a little story on that, we had a supervisor in Paris, France. And that was, I don't remember the date on that, '76 I think, late 70's. But I remember the French people were aghast at our proposal that the names of all the French customers had to be in the MUD that was to be stored in the United States. And they said you wouldn't conceive of putting the names of American customers on a supervisor in France. And Norm Hardy was in the meeting and Norm said, "Why not?" A supervisor is a supervisor; its geographic location really isn't material. If it's controlling the net it does have to have all the information of all the users to function. So to us it was a pure technology problem. But to the French it was matter of pride.

Johnson: National pride.

Tymes: Yeah, they had to have exclusive control over all the French customers. And so before it was installed it was really a bone of contention. After it was up and running, it suddenly ceased to be a consideration.

While I'm on the Paris subject, one of the European salesman said "We have to 1200 baud in Europe or we're dead." And he went to Laszlo Rakoczi, who was my boss at the time, and Laszlo came to me and said, "How fast can we implement 1200 baud for Europe. They have to have it." And I told him, "Why do we need 1200 baud for Europe, they don't have 1200 baud terminals there. And besides I have all these other things that I really have to do, why should I give priority to that?" And he says, "We absolutely have to have this." I said, "Tell you what, I'll implement 1200 baud and it will deploy everywhere, but I want to know six months from now what its usage is and I track every login, so I have a complete record of everything, and we'll see what the usage pattern is. And if it turns out that it is not used in Europe, I don't ever want to hear one of these marketing demands again."

He rather foolishly agreed to that. Six months later we went to Europe and what we found was it was deployed everywhere and nobody had heard of it. In Germany, running faster than 200 baud was illegal, because Germany made all of the modems and acoustic couplers and they couldn't handle 300 baud. So their way of fixing the problem was…make it illegal. Typical German response.

In England, what we were doing was blatantly illegal. Not a problem, every month we bought a pardon from the Queen.

And we also paid a key deposit for every room in England where our equipment operated, so that's how we got around it in England. In France, where they had… let's see, Datapack was Canada, was Transpack French? Anyway, they were building their own glorious network, which

was obviously going to be far superior to Tymnet, but they would deign to allow us to continuing operating until their net was ready. Well, it never did run. They spent enormous sums of money on it and it never became operational.

Hardy: Typical French engineering problem.

Tymes: And to further underscore their distain for us, we couldn't talk to any of the high mucky mucks, we could only talk to one, relatively junior, member of their telephone network. But then I happened to be talking to the French operator and he had a CRT and I said "Do you want to run at 1200 baud?" And he said, "You can't run at 1200 baud." Well, there's this knob in the back of the CRT, just turn it 1200, and then when you want to login or identify, just type the letter A. And so he did that, and he was astonished.

He was hard wired so he didn't have to go through the telephone network. So that operator became our first 1200 baud user in Europe.

So I made Laszlo Rakoczi live up to his agreement.

Hardy: No more demands from Europe.

Tymes: No more demands from marketing, period. I'll listen to marketing, but I won't be dictated to by them.

X25 was another important thing. X25 in a lot of ways was modeled after the scheme that we used to go node to node. In fact, a lot of it was just lifted right out of Tymnet.

Of course, there were committees involved and they always have to make changes. But the original X25 was pretty much lifted straight out of the internode protocol of Tymnet, which was convenient for us because it made it easier for us to add X25 customers who wanted to talk to us at higher data rates.

And somewhere along the line people got the idea that X25 was inherently a low-speed protocol, because it ran on 4800 and 9600 bit per second modems. And I don't know where that idea came from, because speed is dictated by the modem.

Could you run X25 over a DSL? Why not? It's a protocol. It doesn't care what the data rate is, the data rate is dictated by your equipment. So I never understood why…

Hardy: Politics.

Tymes: Yeah. The one thing about X25 is because the Tymnet protocol was designed around the idea that telephone lines were very flaky and you had to have reliable transmissions, so you had to have an efficient way to retransmit as much as required and no more than is required to maintain the integrity. X25 wouldn't do that. And some of the more modern stuff, what's a good example…over a cable modem for instance. The net error rate over a cable modem is much lower than we had in the early Tymnet days. But there's this paranoia about you have to retransmit every single thing and do it as efficiently as possible. Retransmit only the packets that were in error. The need for that is much less now. Although it's coming back, because wireless transmission is somewhat of a similar problem, wireless is inherently unreliable. So some of these old concepts are being rediscovered and reinvented again.

Hardy: Well, maybe, they'll get new patents on all those old ideas again.

Tymes: We kind of wandered off in several directions.

Hardy: Well, very good ones though. I guess one of the things I'm looking for is things that you remember that were specific events that we should put in the timeline for people to discuss. And we've hit a lot of them today, but are there any others that I might have overlooked?

Johnson: Highpoints.

Hardy: Highpoints. Well the NLM was one of them certainly, and the deployment of the first Tymnet engine was one. But are there other particular high points?

Tymes: Did I mention that in 1996, the supervisor was finally converted to a SPARC workstation. And the Tymnet engine finally died.

Hardy: Right, in 1996.

Tymes: The SPARC workstation is actually cheaper.

Hardy: It is. It's mass produced.

Tymes: You can't compete with…If you went to a new planet and had to build one Pentium chip it would cost you billions of dollars. But because this development has already been done over so many years and Pentium chips are made by the millions, you can buy a Pentium chip for $200. Why would you build your own CPU, one off, to try to compete with that? The economics of mass production has just wiped out…the day of somebody sitting down and think I'm going to build a Tymnet engine, those days are gone.

Johnson: Well, it was part of the whole evolutionary process.

Tymes: Kind of like the Wright Brothers building their plane, because they couldn't go buy one.

Johnson: You mentioned Telenet. What was Telenet?

Tymes: Telenet piggybacked on the ARPANET technology with those same nodes and software. And they were going to be a competitor to Tymnet. Vincent Cerf, and I can't remember the other name now.

Johnson: According to one of the Tymshare annual reports, when the FCC decided that Tymnet had to operate as a value-added communications network, there was a reference to Telenet. So Telenet was going to be a commercial enterprise, a competitor to Tymnet, based on the ARPANET protocols. And what ever happened to Telenet then?

Tymes: Well I think they faded off into the sunset.

Johnson: Here it is. Telenet brought a complaint against Tymshare with the FCC requesting regulation of Tymnet, and Tymshare filed response on why Tymnet should not be regulated. It sounded like the outcome of all that was that Tymnet did end up being regulated then.

Tymes: Yeah, there was no question that what we were doing was illegal. But until Telenet came along…

Johnson: Nobody noticed.

Tymes: Nobody really cared. And Telenet made it an issue as a way to limit Tymnet. Which was rather peculiar because Telenet was completely dependent on Tymnet to do all of its

accounting, access control, and to provide universal access. So without a gateway to Tymnet, Telenet could not have done anything.

Johnson: Well, I think this wraps it up. Thanks so much, LaRoy. This has been fascinating.

END OF INTERVIEW

