An Interview with

ROBERT BRADEN

OH 456

Conducted by Jeffrey R. Yost

on

19 November 2014

Marina Del Rey, California

Charles Babbage Institute
Center for the History of Information Technology
University of Minnesota, Minneapolis

Robert Braden Interview

19 November 2014

Oral History 456

Abstract

This interview with internet pioneer Robert ("Bob") Braden briefly covers his education in physics before concentrating on his long and distinguished career in computer science. He served at the computation centers of Carnegie Institute of Technology and Stanford University before joining UCLA as Manager of Programming (1968-1974) and later serving (1975-1986) as Project Director for Software Research. In 1986 he became a Supervising Computer Scientists and Project Leader at the Computer Networks Division of USC's Information Sciences Institute, where he continues to serve. From its inception (in 1981) he was a member of Internet Activities Board (IAB) and served as IAB Executive Director for a half decade. He was long-term member and Chair of the End-to-End Task/Force Research Group (1984-2005). In these roles he made fundamental contributions to internet protocol standards. Braden is an ISI Fellow and an ACM Fellow.

Yost: My name is Jeffrey Yost, from the Charles Babbage Institute at the University of Minnesota, and I'm here this morning with Robert Braden, Bob Braden, at USC's Information Sciences Institute in Marina Del Rey. This interview is part of the Charles Babbage Institute oral history series. It's November 19, 2014.
Bob, I'd like to begin with some basic biographical questions. Can you tell me when and where you were born?

Braden: Camden, New Jersey, in January 1934.

Yost: And did you grow up there as well?

Braden: I grew up mostly in Princeton, New Jersey, until I went away to Cornell University.

Yost: And in your pre-college days, can you discuss yourself as a student and what you were interested in?

Braden: I was a good student. My father was an electrical engineer at RCA Laboratories in Princeton, a research engineer. So I was exposed to a lot of technical stuff. He helped me build one-tube, two-tube, and four-tube radios, and things like that. I played a lot with electric circuits… […particularly relay circuits. I wanted to understand any complex electro-mechanical system. Reading my father's Bell System Technical Journal, I learned

about automatic telephone switching, from Strowger switches to the Number 5 Crossbar system. I also learned about relay-based computing devices at Bell Labs and elsewhere. Using relays I built binary counters and ultimately a 4-bit arithmetic unit that could add, subtract, and multiply 4 bit numbers. I studied the circuitry of railroad interlocking plants, simple in concept but sometimes wonderfully complex.]

When I was about 14, I became seriously interested in computing, in computers. Actually, I trace the beginning of that interest to a particular issue of the Proceedings of the IRE, the Institute of Radio Engineers [which later became the IEEE]. My father subscribed to the Proceedings. I was leafing through an issue in August 1947 and happened upon an article about the circuitry of the ENIAC computer, and I found that very fascinating. From it, I learned about flip-flops and ring counters and the basic design of (vacuum tube) computers.

After my junior year in high school, I got a summer job at IBM in New York City, working as a clerical assistant on their SSEC (Selective Sequence Electronic Calculator), a large, one-of-a-kind computer that IBM had installed in their "World Headquarters" on Madison Avenue. The SSEC was a public show place, built into the walls of an 1800 square foot room with plate glass windows onto 57th and Madison Ave. (But when the machine was down, IBM drew the curtains!) There was lots of shiny chrome, and pretty young women to show visitors around. In operation, the SSEC put on quite a show, with many dancing lights, spinning tapes, and clicking relays. [The SSEC appeared in the film *Walk East on Beacon Street*, in fact.] Despite its showplace aspect, the SSEC was a

unique computational facility. It was called a calculator, not a computer, because it did not have a stored program. [In popular discourse, this was neither a calculator nor a computer, but a "giant brain", although IBM avoided that metaphor.] However, the SSEC did have conditional branches, subroutines, and looping, under control of loops of 80 column punched paper tape. One of my regular tasks was running the punched-card-to-punched-paper-tape machine to prepare new programs. The SSEC technology was a mix of IBM relays that provided a memory of 150 20-digit words, and vacuum tubes that implemented the ALU. The SSEC was used to solve some 30 problems over its 4 year lifetime. I learned how to program the SSEC in general, although of course a 17 year old high school student was not allowed to touch the machine.

My last task that summer was to program an IBM 604 (essentially, an all-electronic version of the 602A electro-mechanical punched card calculator that I later encountered at Cornell) to run a test case for a problem being programmed for the SSEC. I did this work for a terribly nice young mathematician on the SSEC staff. His name was John Backus. On the last day of my summer job, I had not completed his calculation, due I think to my own errors. John persuaded me to come into work on my own time to finish it up, which I did. I have always been grateful to John for his wise advice and support.

Yost: And what year was that, do you recall?

Braden: That must've been summer 1951.

Yost: That would've been the year before the defense calculator 701, right?

Braden: That's correct. In fact, two summers later, I worked in the same location, programming an IBM 701. They had replaced the one-of-a-kind SSEC; tore out the machine, and installed a 701 in the same room. I programmed it for the summer.

[IBM was insistent about strict technical accuracy of all publications, but they slipped up in calling the 701 a calculator, it was a true stored program computer. I thought of the 701 as an engineered and productized version of Von Neumann's IAS (Institute for Advanced Study) computer. The 701 evolved successively into the 704, the 709, the 7090, and the 7094.]

Yost: That was an active scientific computing bureau, wasn't it, officially called the IBM Technical Computing Bureau?

Braden: Oh… yes. I wrote library programs for transposition and multiplication of large matrices, which was pretty daunting on the 701, with only 4,000 words of memory. My programs had to use magnetic tapes a lot! Programming the 701 was tedious, with no index registers or floating point, and only very crude programming tools -- a very primitive assembly language. [Since the 701 had no index registers, all address modification had to be done by performing arithmetic on instructions.] Since the electrostatic memory of the 701 was very flakey, with a mean time to dropping bits from

a few hours down to a few minutes, self-checking, check point and restart, were necessary for every 701 program.

Yost: Was the programming work you did for clients of IBM?

Braden: Yes. The matrix programs were for a Canadian aircraft company, AV Roe, [They had a tape containing 173-by-something matrices, representing stress measurements on an airplane wing, as I recall.]

Yost: So it was for several summers, prior to attending Cornell, that you did this work for IBM?

Braden: No. During my first summer at IBM I was still in high school, while I was in college during the second IBM summer. While at Cornell I had other computer-related summer jobs at RCA Laboratories, at Princeton University, and at IBM research. I also had miscellaneous part time jobs related to computing during the school year.
I might mention that the second summer I worked at IBM, programming the 701, John Backus had been "sent upstairs" and was rumored to be leading a team of programmers who were writing a program to write programs, which they called FORTRAN. I found it very difficult to understand how a program could write programs. It seemed totally magical to me at the time.

Yost: Do you recall some of the other computer scientists at the IBM headquarters in this specific bureau, the IBM Technical Computing Bureau?

Braden: A few names come to mind – Harlan Herrick, chess champion of IBM World Headquarters, famous for writing a 100 line assembly language program that ran the first time with no errors, and a member of Backus' FORTRAN team. Peter Sheridan, also on the FORTRAN team. I had become friends with John Backus during my SSEC summer, and with my boss Frank Beckman on the 701. I developed a strong connection to Rex Seeber, who gave me my summer job on the SSEC. Rex was in charge of SSEC operation and had been one of the SSEC designers. He later contributed to early experiments on cache memory at IBM.

John von Neumann would come in once a month. I have been told that the SSEC programmers, who were then called mathematicians — and they were mathematicians — would parade in front of him to describe what they were doing and how they were doing it. He would just sit there and listen and nod his head all day. At the end of the day, he'd go home, get his paycheck, and IBM felt very well compensated that a man of John von Neumann's stature thought they were doing the right thing.

Yost: Was Cuthbert Hurd centrally involved?

Braden: He was. I didn't know him, he was at a higher level; he was head of Applied Science, as I remember.

Yost: Right.

Braden: I guess the 701 and the SSEC were in the department or division of Applied Science.

Yost: Can you tell me how you came to decide upon Cornell for college?

Braden: Yes. My father had a college friend who had gone to Cornell as a professor and had started a school of Engineering Physics. Since I was generally interested in both engineering and physics, it seemed like the right place for me. [Laughs.] It's a little hard to believe now, but I basically applied only to Cornell and was admitted there. Life was much easier in those days.

Yost: What year was it that you started?

Braden: 1952. EP was a five-year program, so I graduated in 1957.

Yost: I'm not familiar with what year Cornell started their computing facility, do you recall?

Braden: I was there at the beginning. When I arrived on campus, Cornell had an electro-mechanical calculator called an IBM 602A. This machine would read data cards, do arithmetic operations on the data and punch results. It was "programmed" by plug boards.

[It actually resided in the university accounting office, but I was allowed to use it evenings and weekends. In retrospect, I was astonished at their trust in me, giving me solitary access to that office.] I used the 602-A to do calculations for a physics professor, Lyman Parratt. The plug boards were quite complicated, calculating equations with square roots and so on. I had to program the machine to compute square roots using Newton's iteration.

My second year (1953), Cornell got an IBM CPC (Card-Programmed Calculator), and I did some programming for it In 1956, I think, Cornell finally acquired an IBM 650, a real stored-program computer. They hired a business school type as computer center director. I installed the IBM SOAP assembler on the machine for him. And then for my undergraduate thesis, I did a calculation for a physics professor that won me a footnote in a book on quantum physics. [Laughs.]

Yost: Can you tell me more about that? What it was specifically?

Braden: The book was *The Quantum Mechanics of One- and Two-Electron Atoms* by Bethe and Salpeter. It was Salpeter I worked with, I never actually met Hans Bethe, although I audited his first-year graduate course in quantum mechanics. There was a particular calculation that went into some — I don't remember what — that was suspect, and Salpeter wanted it repeated to a higher accuracy. I don't remember much about it, except it was a fairly complicated integral equation. Anyway, I programmed it, gave him his results, he was happy, and I got a footnote [(footnote 2, p. 354, "Quantum Mechanics

of One- and Two-Electron Atoms", Hans A. Bethe and Edwin E. Salpeter, Academic Press, New York, 1957)].

Yost: Can you talk about any faculty members that were particularly influential to you in your undergraduate years?

Braden: Oh my. Yes. I've heard it said that when you go through this exercise you're often surprised at the result. I had that feeling. I took an elective course in Russian literature from a little-known professor named Vladimir Nabokov, before *Lolita* was published. That was interesting. I took a lot of math courses...(pause) I am trying to sort it out; there were so many. One course that influenced my world view was a very popular course on comparative government taught by Prof. Einaudi, who was a brother of the then-premier of Italy. It was an overflow course; 200 students. I also took courses from a math professor, Mark Kac, who laughingly described himself as a "simple Pole", and who had an international reputation in mathematical statistics. I studied logic with Max Black, another leading figure. I studied electricity and magnetism under Dale Corson, who later became university president. It's so interesting; there were so many. The required Engineering Physics curriculum was very heavy. We basically took all the required courses the math majors took (up to, but not including, Analysis), and the basic physics courses that the physicists s took, and basic engineering courses like machine shop, drafting, mechanics, strength of materials, and AC circuits,. [We also took courses designed specifically for EPs, like quantum chemistry and solid state physics. And then we also had electives. It did not help my social life, but EPs were generally known by the

campus community as exceptional. An EP could get admitted to almost any university course without the normal prerequisites. My EP class of 1957 began with some 25 students and I think 8 graduated.]

So, it was a very rigorous, stimulating, intellectually satisfying, five years. Cornell had a fine faculty, and they put much energy into teaching and took an interest in their students.

Yost: And did you work summers while you were there?

Braden: The second summer was the one I mentioned, programming the 701. The first summer I worked at RCA Laboratories where my father worked, but I worked for an engineer named Jan Rajchman, a brilliant man. He was developing one of the first large-scale core memories, competing with MIT. I worked in his group as a lab technician, soldering and testing, the things that lab techs do. And then for IBM on the 701; then I worked for Princeton University Forrestal Laboratories as operator of their CPC. My last year, I worked at IBM Research in Poughkeepsie, in a group doing research on magnetic materials.

Yost: You graduated with a bachelor's in Engineering Physics in 1957. Had you made the decision that you would immediately go on to graduate school?

Braden: Yes.

Yost: Can you tell me how you came to decide on Stanford and a master's in physics?

Braden: Well, I was aiming for a Ph.D. in Physics and I didn't make it. A significant factor was that at some point, I decided that computing was more fun than physics.

Yost: A number of physicists and physicists in training decided that.

Braden: I decided that before I finished. My thesis was not going well and my advisor had not gotten tenure at Stanford, and moved over to Berkeley. It was awkward, so I easily slid back into computing. How did I choose graduate school? With a ragweed pollen count table. I had severe asthma at that point, and the ragweed pollen count in New Jersey where I grew up is 25 and the ragweed pollen count in Michigan where I thought I might go was 120. The pollen count in the Bay area was 0.01. So that narrowed it down to Stanford and Berkeley. I applied to both and was admitted to both. People said, "you don't want to go to Berkeley because you'll just be one of many, many students; go to Stanford." So I did; which is probably a mistake; I would probably have been better off at Berkeley. Stanford physics was very much concentrated in one area, fundamental particle physics, at that point. It turned out that I did not have much interest in particle physics, and I would have had a much broader choice of fields at Berkeley. [I have made several bad career choices. At one point I interviewed with Corbató at Project MAC at MIT. I should have moved there and later pursued a PhD in Computer Science, but I chose to stay in Palo Alto in the physics department.]

But anyway, my life did get interesting then, as far as computing goes. Stanford had just gotten a Burroughs 220 computer, which was a very interesting machine. It was a tube

machine when IBM was starting to build transistorized machines, and it was a decimal machine when machines were becoming binary. [The 220 had 10 decimal digits using binary coded decimal (1248 encoding), one index register, and built-in floating point.] Its special appeal was its software. It came with a compiler for a dialect of ALGOL-58, called BALGOL, Burroughs ALGOL. The BALGOL compiler was altogether a remarkable piece of compiler construction, far ahead of its time. The comments in the assembly language listing of the BALGOL compiler were written by a graduate student at Cal Tech hired for the summer. His name was Donald Knuth. Now, I talked to Knuth about the BALGOL compiler a few years ago. In fact, you'll find in Knuth's oral history his statement that BALGOL was far ahead of its time and that to read the listing of the compiler was to get a complete course in Computer Science, as it was known then. It used only dynamic allocation of data elements of list-structured memory, allocated from a pool that we call a heap today. There were no fixed-size tables. It used a fairly modern, although not the most modern, parsing technique. It ran very fast at full card reader speed, so that people didn't bother making binary decks, they just recompiled. It was ideal for a university environment. The compilation speed was despite the slow speed of the machine. The B220 needed several hundred microseconds to add two numbers.

Anyway, as was usual in those days — I guess probably it still is — there was a coterie of undergraduate programming geniuses who sort of gathered around the computer center, or lived there. Some of whom finished their degrees and some didn't. I started hanging out with them. One of the leading students who did finish, Larry Breed, went on the write the APL\360 system for Ken Iverson at IBM. Larry was brilliant programmer. Another

very talented undergraduate programmer on the B220 was Roger Moore, who also contributed to the APL system and later on wrote several ALGOL compilers himself. At first I used the 220 for my physics thesis, and then I started studying the BALGOL compiler, reading the code, learning compiler techniques, learning computer science, really.

Yost: What was your master's thesis on?

Braden: There was no thesis required. A Master's in Physics was just a step towards a PhD. [But as far as I can recall, my thesis topic was repeating to higher accuracy a calculation of the hyperfine structure of the ground state of helium. Basically an eigenvalue problem to calculate the atomic orbitals. Pretty mundane stuff.]

Yost: Okay. So you had started work on your [physics] Ph.D.

Braden: Yes. I passed my qualifier and had done my coursework, but I got sucked into… so, what happened was that Stanford quickly outgrew the B220, so they went to IBM and bought a 7044, which eventually became a 7094. The head of the computer center at Stanford was a numerical analyst named George Forsythe. I came to know him quite well, talked with him often. He basically brought computer culture to Stanford. He taught frequent non-credit courses to faculty and students. I think Stanford was one of the first universities to form a Computer Science department, and I think it was because Forsythe had spread computer culture around the campus.

Yost: Do you recall the topic of some of these courses?

Braden: Programming, you know, how to write BALGOL for the 220. I helped teach these BALGOL courses. And so, when Forsythe discovered that Stanford had contracted for an IBM 7044, he was, let's say, distraught, because the IBM system had relatively primitive software. Their Fortran compilers were slow compared to BALGOL, even though the IBM hardware was much faster. And Fortran had nowhere near the sophistication of BALGOL as a language. There was no way that Stanford could run the number of jobs a day on the IBM system that they were currently running on the much slower Burroughs machine. So, Forsythe said to me one day, "Gee, it would be nice if we had a BALGOL compiler to run on the 7094!" Larry Breed, Roger Moore, and I accepted that challenge, and we ported the Burroughs compiler onto the IBM mainframe. It took us six months. Today, porting is trivial, but in those days it was a black art. There was no literature and very few working examples, so we had to sort of feel our way, but we succeeded. At the end of some 20 stages, we ended up with a BALGOL language compiler for an IBM system, written in its own language; written in BALGOL -- SUBALGOL, actually, Stanford University BALGOL -- which could compile itself at full tape speed, which is 100 times faster than the IBM system could do. [Laughs.] It was a very successful effort.

That got me really deeply into compiler construction and programming languages; beginning computer science.

[A more complete description of my experience in computing at Stanford during this period will be found in the Annals of the History of Computing (Braden, R., "Burroughs Algol at Stanford University, 1960-1963." Anecdote, *IEEE Annals of the History of Computing*, v 35, no 4, Oct-Dec 2013.) It includes a brief description of our process for porting Burroughs BALGOL into a self-compiling SUBALGOL on the IBM system.]

Yost: Did BALGOL have an influence at any other universities?

Braden: Yes it did. The 220 was quite popular in places that were fairly heavy into academic computing, like Cornell and Case and Stanford Research Institute and… gee, I can't remember the names now but there were maybe 20 and laboratories that had 220s, and I assume they used BALGOL; I know that Cornell. I don't know that they used it as effectively as Forsythe did.

Yost: You mentioned the decision to go from the Burroughs 220 to IBM systems. Do you know who made that decision?

Braden: No. Actually, it was probably made on the golf course, in those days. [Laughs.] When big machines were sold, it was intensely political.

The group at Burroughs, the small group of four or five very, very exceptional programmers who wrote the BALGOL compiler, went on to develop the Burroughs

B5000 hardware. In a sense, they built BALGOL (ALGOL-60, actually) into the hardware; it's a stack machine.

Yost: That was very successful in influential scientific communities.

Braden: It was. Anyway, now I lost the thread. Let me say that once the [IBM SUBALGOL] compiler was essentially finished, I needed another job. I had given up on my physics thesis, at this point. I went and talked to George Forsythe again. My wife had finished her Ph.D. at Stanford in Psychology and she had been offered a teaching job at the University of Pittsburgh. Forsythe said, "Well, there'll be a place for you at Carnegie Tech." So he dialed professor Alan Perlis at Carnegie Tech, and said something like, "Al, I've got a promising young man here, you may to be able to find a place for him." So I got a position [at Carnegie Tech], with a weird job title [(Assistant Professor, Computation Center).] It was not tenure track, since the Computation Center was not an academic department. But it seemed that whatever Perlis wanted from the administration at Carnegie Tech, he could get. So in 1962, I left Stanford and moved to Pittsburgh, to teach and do research programming at Carnegie Tech, now Carnegie Mellon, of course. An important part of my job was teaching the introductory programming course, S205, with over a hundred students. Perlis was a great teacher and he was fond of teaching S205, but he decided he had to get someone else to do it, so he hired me to teach it. [I recall one of his wonderful exam questions: design a formatting language for printing diagrams of motels. Someday I will write a short reminiscence on Perlis.]

I also taught a few other courses on compilers and operating systems. At that time my knowledge of operating systems was primitive but then operating systems were primitive, too, at that point. Perlis' people wrote their own time-sharing system, and I wrote a text editor for it.

I learned a lot from Perlis. He mentored me, really, and I sometimes say that most of the computer science I know today I learned from Perlis in the 2-1/2 years I spent at Carnegie Tech. Allen Newell was another person I worked with; and he, of course, is famous as a leading AI — no, not AI, cognitive science — person.

Yost: Did you have any interaction with Herbert Simon?

Braden: Yes. Well, I think he shook my hand once. [Laughs.] Herb was certainly around and a presence, an august presence, but I didn't have much direct contact with him. There are some names I recall. Art Evans. I think that three graduate students -- Jerry Feldman, Ralph London, and Dave Parnas -- each served as my graduate student TA [for S205]. [Laughs.] Jerry Feldman's thesis was *A Formal Semantics for Computer-Oriented Languages*. I believe that his work introduced semantic formalism into the code generation phase of a compiler-compiler. Ralph London got on in AI; in fact, he did AI research here at ISI for a while. Parnas is well l known for his pioneering work on software engineering.

Pittsburgh was a pretty grotty part of the world in 1962, as the English would say, but I had a great and exciting job at Carnegie. After 2-1/2 years there, my wife and I said, "Look, we came back [to the East coast] to prove to ourselves that we could still stand the

eastern climate. We have proved it. Why do we keep torturing ourselves"? Leaving Carnegie was a bad decision from a career perspective, but we missed California acutely. So I looked for a job in the Bay Area [and] made another one of my bad career decisions. I was offered a job at Stanford University, building and directing the SLAC (Stanford Linear Accelerator Center) facility of the Stanford Computation Center. The other job I was offered was at SRI, working for a wild-eyed researcher named Doug Engelbart. He was doing weird things. I thought, "No, I don't think I want to work for this guy", and I chose to go back to Stanford. Probably I would have been better off going with Engelbart. Several years later he gave his now-famous demonstration of his NLS system at a joint computer conference.

Yost: Before we get into some more questions with Stanford, I understand the center at Carnegie Institute, in 1961 obtained a Bendix G-20, is that right?

Braden: Yes, that's correct.

Yost: Can you talk a bit about that machine?

Braden: A really interesting machine. It had an interesting instruction set, unlike any that I've seen. A generalized addressing scheme could use multi-word instructions to do complex address computation with indirect addressing. Another unique feature of the G-20 is that it had no integer representation, only floating point. I wasn't doing hardware myself, I was a software person. Jessie Quatse was the hardware guy. He extended the

machine, modified the hardware, I don't remember, but I think he put in virtual memory. They were fearless at Carnegie Tech, in those days. I just used this big box full of transistors.

Yost: Was Perlis essentially in charge of the computer center, in addition to being a faculty member?

Braden: Yes. Later on, while I was there, they hired a computer center director but until then, Perlis was running it. Einar Stefferud was the computer center director. He spun off a career in consulting and business computation.

Yost: In 1965, you become the Associate Director of the Computation Center for the SLAC facility?

Braden: Yes. At SLAC I was mentored by Bill Miller, in the development of management skills, which I sorely needed. Bill Miller was a physicist who had become a serious Computer Scientist who ran the computation group at SLAC. We went through a computer selection for SLAC, and it came down to a choice between a CDC 6600 -- a Seymour Cray creation -- and an IBM 360 Model 91, a Gene Amdahl creation. The 91 was the top of the line of the 360 series. Both were supercomputers. Miller and I basically made the decision to go with the IBM 360/91 system. Well, the physicists really wanted the CDC machine. It was a FORTRAN oriented system. They did not want to learn PL/I or JCL, [laughs] and most of the other particle physics laboratories had CDC 6600s, then

7600s. It took some diplomacy to calm them down. Bill Miller very cleverly managed this.

Yost: What were the important aspects or what was your major rationale for going with the IBM 360/91?

Braden: Somewhere in my books there is a report, if I could find it, to answer that. One aspect of the selection process was visiting the factories, and it was night vs. day. The 6600s were built in a big room, and there were two or three machines in various stages of completion scattered around this room, littered with electronic test equipment and so on. And then IBM had a large building with individual compartments for each machine, and there'd be a sign overhead with the customer name. So a customer could visit there and see ***his*** machine being built and tested. And, of course, IBM took us through their semiconductor fab lines; IBM was the leading, bleeding edge in integrated circuit development. And in fact, they developed a special high speed, high density version of their Solid Logic Technology (SLT), called ASLT, Advanced Solid Logic Technology, for the 91. There's a long story behind that, but their degree of sophistication was amazing, just amazing. And IBM had well-established capability for maintaining their systems.

And I think that we applied some pseudo computer science — it was really urban myths — about how much memory it takes to support a multi-programmed CPU of given power. There did not seem to be enough memory on the 6600, whereas the IBM system could have two million bytes. I forgot how much memory was on the 6600. Anyway, we

thought there were architectural issues with the 6600. That's all I can recall. But I do recall that Bill and I did our homework in writing a justification to the AEC. We also programmed benchmark code segments for both machines.

Yost: One thing I neglected to ask you about the computer center at Carnegie was, had they launched a time-sharing operation while you were there?

Braden: Yes, absolutely. They wrote a time-sharing monitor for the Bendix G-20. It was done by an extremely bright undergraduate, who I don't think ever finished school. Carl. Lefkowitz did the whole system himself; all the interrupt level stuff. It was not a general time sharing system, however, it was a CRJE system using teletypes as remote terminals. I wrote the online text editor and user file system.

Yost: Was there any consideration with regard to computer security and privacy at that time with multiple users?

Braden: No. It was a different world then. Who would want in?

Yost: In 1967, you were promoted to senior research staff. Did that significantly change the types of responsibilities?

Braden: What actually happened was they decided I was not cut out to be a manager. [Laughs.] And at the suggestions of some employees, they decided yes, maybe I'd be

happier doing research. I then moved from the Stanford Computation Center to work directly for Bill Miller at SLAC. My research mostly involved trying to make a silk purse out of a sow's ear; trying to adapt the IBM TSO system to be a real time-sharing system. We were starting to play with scheduling algorithms and trying to understand what the possibilities were, operating within the existing OS/MVT framework. But I didn't stay there long. Within a year, I accidentally met Bill Kehl at a Westwood restaurant. He had been the director of the University of Pittsburgh computer center when I was at Carnegie Tech. So I knew him, and when I met him again in 1968, he had just become director of the UCLA computer center, the so-called Campus Computing Network (CCN). UCLA was slated to get a 360 Model 91, which was now familiar territory for me, and Kehl hired me as Manager of Programming. In a sense, that is both the beginning and the end of my life's story. Soon after I became Manager of Programming, ARPA came to UCLA and said, "We would like you to connect your 360 Model 91 to the ARPANET and sell service on the ARPANET. However, ARPA will not pay for the software to hook to the ARPAnet." As Manager of Programming, I got the task to figure out, "What is an ARPANET? What is an IMP? What is a back end?" [Laughs.]

Yost: When you joined UCLA, had you had any awareness of the ARPANET project?

Braden: No, not at all. It was all new to me. But that being, in some sense… I told you this was the end of the first half of my life and career, and the beginning of the second half, my work on networking and the Internet.

I encountered a familiar feature at UCLA, a core of bright undergraduates who lived at the computer center [laughs], and loved to reprogram the IBM system in the middle of the night. They'd leave system shards on the floor for me to pick up in the morning. Oh, that was a challenge. But anyway, the original purpose of the ARPANET was to do resource sharing and we were to be a resource to be shared. Now, we were a unique resource in several ways. The IBM system is a batch processing system, so we could supply batch remote job entry service over the ARPANET with the IBM mainframe. Almost all the other ARPAnet sites were computer science research outfits, usually university departments, who almost always had DEC 10, TOPS 10 systems; and used ASCII. Of course, IBM used EBCDIC character encoding, so there was a fundamental communication barrier. The IBM operating system wasn't programmed, wasn't designed, to be easily extended by customers. As a result our relationship with IBM was dicey at times, there was a little bad blood. For example, the first thing we had to do was to "invent" interprocess communication for OS/MVT. IPC is a fundamental function in an operating system, but it did not exist in the IBM system, so we designed one, the Exchange, and added it to the kernel as a Supervisor call. I say we; we didn't invent IPC, of course, I was aware of the concept from the Project Mac work. There was a beautiful paper in ACM *Communications* describing the Multics semantics that I was familiar with. I taught from it once.

Yost: So when you arrived at UCLA, was the interface processor kind of your central task from the start or did that come slightly later?

Braden: I'm not sure what you're asking. I was manager of programming, responsible for both system programming and user services. Connecting to the ARPANET was in a sense a hobby; ARPA did not pay for it. They said, "You make your money from selling time and we expect you to fund the software development yourselves." At that point we needed ARPA's money to survive financially, so we did the software.

Yost: Did you have any interaction with BBN and did you learn anything from that?

Braden: We had a great deal of interaction with BBN.

Yost: Can you talk about that?

Braden: ISI, BBN, SRI, Utah, MIT, --all the early research sites formed the networkworking group that met several times a year. All of the people involved in the ARPANET program would get together, and I attended those meetings. One of the first things that I had to do was to design a protocol for remote job entry over the ARPANET, because such a protocol did not exist. I documented it in RFC 88, I think it was; one of the earliest RFCs, defining the first version of the protocol.

Yost: Do you recall some of the major issues and challenges that were discussed in those early meetings among the ARPANET working group?

Braden: The first task was to develop a file transfer protocol, FTP. Again, I was the odd man out because the IBM had a truly baroque file system, and we wanted to let customers have access to some of the features of it, whereas all the other systems were UNIX systems with a very simple file structure. So if you look at the FTP spec there's some weirdness, complexity that's there because I asked for it.

Another thing was that the first research program that ARPA wanted to send to our 360/91 was a seismic processing job. The US had seismic listening stations somewhere in the world that collected data on tape. They wanted to ship the tapes to us at UCLA, do processing on the 360/91, and then FTP the processed data in real time over the ARPANET to a "basement" in the Washington D.C. area. This would require some 10 hours of ARPANET data transfer daily, so we were very concerned about efficient transmission of data over the network. I specified a data compression feature in FTP which is there because of this project. Alex McKenzie at BBN designed a clever checkpoint/restart mechanism that was also adopted into the FTP spec.

Anyway, file transfer was alas the first thing that network working group tackled. No, the first was to design and implement NCP (Network Control Protocol), which gave you end-to-end transport service. You needed this transport protocol with a windowing scheme. I was not principally involved with NCP. Jon Postel and Steve Crocker and other people who had been at UCLA for a while in the computer science department were the principals behind development of NCP, which eventually was replaced by TCP/IP. So the first task of the network working group was to define NCP and the handshake necessary

to create a connection, the initial connection protocol or ICP. Then they worked on FTP, and later they worked on TELNET, and finally, e-mail. E-mail was originally just what MIT folks call a "frob" on the side of FTP. But then later, Jon Postel redesigned it as a separate SMTP protocol.

Yost: As Manager of Programming, how many people were in your group?

Braden: Twenty, I think.

Yost: Quite a sizeable group to manage.

Braden: Yes. And I was naïve. [Laughs.]

Yost: Can you talk about your management style and how you went about bringing on the right types of people for the tasks?

Braden: I started with a group of undergraduates, several of whom were really very talented. Steve Wolfe, who wrote our NCP, unfortunately, died of a heart attack. The very talented young man who wrote the inter-process communication mechanism, Stu Feigin, went on to become one of the key technical people at Amdahl. Now he's at Oracle. Anyway, so I started there, and I advertised, actually, and interviewed people. Some worked out; although some bright people came and decided we were too

production-oriented and not theoretical enough and left; some came and stayed. I always looked for people from good universities; a prejudice, I guess.

But we did some great work. Unfortunately, I didn't do any publishing in those days. We had this supercomputer, a 360/91, which is capable of 10 million operations a second but it had the ponderous IBM operating system, which meant that transition from job to job took 30 seconds. One semester I was teaching an undergraduate course in systems programming, and I challenged the class to write a fast job monitor. I said that when the last card of the job went into the card reader, I wanted the output printer to start moving. I wanted it to look instantaneous because, dammit, with a machine that powerful, there's no reason why we couldn't do that. Well they did it! It was called Quickrun, and it was the mainstay of batch computing at UCLA for many years. I don't know if it still is. As the final "g-whiz" touch (which I was not responsible for) — one of the guys, the head of operations, came up with the idea of a self-service printer for Quickrun. He put the printer up against the wall, put a slot in the wall so the printer paper came out from the slot. So the user gets in a queue, walks to the self-service 2501 card reader, puts his deck in, takes his deck out, walks across the room to the printer output, tears off his output, and then walks away. Just about that fast. It was terrifically successful.

Yost: You characterized the MVT OS as immense and amorphous.

Braden: Ponderous is what I said, I think. But yes, it was immense, but I'm not sure if amorphous is quite right. This is before virtual memory, so user jobs and subsystems like Quickrun and the NCP ran in fixed memory partitions. We did large-scale multi-

programming within the four million bytes of memory. My boss Bill Kehl was a gambler and at one point — we had a two-million byte machine like everybody else — he decided to buy an additional two million bytes. He persuaded the university, the board of regents, to spring for the other two million bytes, so we had the largest machine outside of the US government. There was another four-million byte machine, actually, somewhere in a basement in Washington D.C., whose name tag was "empty." [Laughs.] But other than that, we were the largest machine around and one of the fastest. That was … oh yes, memory. OS/MVT had the concept of a process, which IBM called a task, so that was okay. In memory, we would have a batch processing partition; we would have the NCP partition for the subsystem that ran the ARPANET, 100K bytes; another subsystem that ran remote job entry over phone lines, that was another 120K bytes; you added up all that. UCLA class registration was automated to run in real time on our 91. This shared some of the code from the ARPANET NCP, actually, so there was a partition for that. I'm trying to figure out what else, but there were other things in memory. We didn't have the ability to map physical memory arbitrarily into virtual address spaces. Eventually the 360/91 was replaced with an IBM 3033 that had virtual memory, but that was much later. I told you my programmers were changing things in the middle of the night; they actually looked at the IBM OS code and discovered a lot of it was very primitive. It looked like it had been written by a man whose only knowledge of programming came from a two-week FORTRAN course. [Laughs.] So really, you know, branch here and go to here, branch here,… So my programmers sometimes took a factor of five out of the OS code, cleaned up the code. They took out a factor of two, certainly. Altogether it was a

big hairy system and it tended to crash a lot, which got me in trouble a lot, and we went through hell in keeping the system up and providing all these services.

Yost: Can we talk a little bit about the debugging process and the software you developed?

Braden: We didn't have any on-line access. We had a UCLA-developed console remote job entry system called URSA. We'd sit at a console and compose a program, then submit it for batch. That was a CRJE system that occupied another partition. I'm sorry, what was the question?

Yost: Debugging.

Braden: So, debugging OS code generally involved reading memory dumps, core dumps; four million bytes of core dump is pretty awful. My programmers worked many hours poring through dumps. The MVT system was not robust, internally. IBM did much better with the MVS system. With these memory dumps, subsystems — particularly the ARPANET subsystem — had built-in some simple break point mechanisms.. But debugging was primitive; life was tough.

Yost: Some individuals you acknowledged as providing helpful ideas or ideas you drew from, I'd like to get your comments on some of these individuals. You mentioned the symbolic tag matching paradigm of the Exchange that was inspired by Dave Walden.

Braden: Yes.

Yost: Can you discuss that a little bit?

Braden: Yes, Walden wrote an early RFC, RFC 62, on inter-process communication. I was quite dazzled to read it; it was very cleverly written. It introduced the concept of rendezvous. When two processes want to communicate, there has to be some place for a rendezvous between the two requests, to pair them up and make a connection. He made abstract arguments for why the rendezvous should be at the initiating end or the responding end. I don't remember the details today, but I was deeply impressed by it and it certainly provided intellectual stimulus.

Yost: By the way, I'm in close contact with Dave. He's on the Computer Society history committee with me, so I'll say hi to him for you.

Braden: Thank you. You probably know that Dave was editor for the *Anecdotes* and the story I told you earlier about the Burroughs 220 at Stanford was all in the anecdote that I wrote.

Yost: Yes. What about contributions from Steve Wolfe and Stu Feigin?

Braden: Well, like me at that point, they didn't know anything about IMPs or packets, but they caught on very fast and [were] very smart programmers. There's folklore that a really good programmer is five times more productive than a poor programmer. And these guys were in the upper, upper edge.

Yost: Two other individuals, both from Rand, Eric Harslem and John Heffner.

Braden: Yes. We worked with them. I mentioned that the first major computational job that ARPA wanted us to do was with seismic data, but that didn't last long. The next ARAP job was to do climate dynamics calculations. Rand had a climate dynamics group and the 360/91 was ideal for climate modeling, because that sort of differential equation computation requires large memory for efficiency, doing mesh calculations. Four million bytes on the machine was important, and so Bob Kahn himself came to UCLA and we negotiated a deal where we would provide service to Rand. ARPA would give money to Rand and Rand would buy time from us. So, although Rand was just up the street from UCLA — we could've used a taxi instead of a TIP, as we used to say — we had to make the ARPA code actually work, really work in production. I don't remember exactly when; it must've been the early 1970s; so we spent a lot of time on the phone with Eric and John. 'I sent a byte,' or 'I sent a packet.' 'I didn't get it.' 'I'll send another packet.' 'OK, I got that one, but the Check Sum's wrong.' [Laughs.]

Yost: We haven't talked much about the interaction between the Campus Computing Network group and the computer science department. Could you talk about that and when you started interacting with Leonard Kleinrock and others?

Braden: [There were substantial economies of scale in big mainframes, so many universities bought the most machine they could afford. But a large mainframe is a single point of failure; when the system inevitably goes down, many users are affected. So CCN was not very popular with the campus users. The Computer Science department was quick to find fault with us, but otherwise they ignored us. I used to say that CS represented the divine side of computing while CCN was the secular side.]

Once we determined that we needed to connect to the ARPANET, we needed to interact with Kleinrock's group, and particularly Jon Postel, and Charley Kline, and who else? Steve Crocker. I think Vint Cerf was still there; not sure. But Steve Crocker was the leader of Kleinrock's research group. Steve Crocker is fond of saying that the graduate students felt like children who expected the parents to come along and tell them what to do, but the parents never came. [Laughs.] So the grad students just went on solving the problems. We had to interact with computer science department just to connect our computer, the 91. We had to have a host interface built to connect to the IBM I/O channel. This was a point of friction with IBM, who strongly disliked having a foreign device on their channel. Fortunately, the computer science department was in an adjacent building to CCN, which was in Math Sciences at this point. I guess maybe because of proximity, I got to talk to Postel quite a lot, and I'd always see him at working group

meetings, and we would read each other's RFCs. RFCs quickly became an important communication tool, thanks in large part to the dedication and editorial skill of Jon Postel, the RFC Editor until his death in 1998.

Yost: And RFCs, of course, were much than important research notes; they're more like a publication, in many ways.

Braden: Well they've become that. Originally, they were just notes. Some of them were, 'The next meeting will be on such-and-such date.' Or, 'last month we sent 10,000 packets.' Or, at the other extreme, 'here's a whole new way that you could think about online data management,' or 'here's a new protocol.'

Yost: I guess there was quite a range.

Braden: A wide range. It was much later that the IETF adopted the RFC series as their publication channel for Internet standards documents. In fact, the RFC series still has an independent existence, so they still publish some non-IETF [Internet Engineering Task Force] documents. That history starts getting into people, and politics, and so on.

In 1972, which is I think the third or fourth year of the ARPANET research program, ARPA put on a demonstration of ARPANET in a hotel in Washington, D.C. We were there from UCLA/CCN, and we demonstrated ARPANET remote job entry on CCN's Model 91. Actually, someone in Washington opened a TELNET (remote terminal)

connection into a TOPS-20 system at BBN across the ARPANET, and then the TOPS-20 system used my NETRJS remote job entry protocol across the ARPANET to submit a batch job at UCLA. After the job ran, the results were returned from UCLA to a printer in the demo site in Washington D.C. We were very nervous, but it all worked.

Yost: That was a really important event to . . .

Braden: That was a really important event.

Yost: . . . spreading the news about this important network.

Braden: Right. And in 1975, I got my first direct funding from DARPA — it was still ARPA, I think, at that point — to work on what was called the National Software Works, which was a library system and data management system, using time-shared computers and the 91. It had some important intellectual spinoffs but was never successful because ARPA didn't put big enough computers into the role. But shortly after that, Vint Cerf invited me to join the Internet working group, this is about 1977 or 1978, I don't remember exactly.

Yost: 1978, I think.

Braden: Okay. My memory that dates back that far is pretty hazy, I'm afraid. So I started going to meetings defining TCP/IP; actually, I was in the subgroup doing TCP. The

ARPANET code for the 360 had been written by people who worked for me; I didn't cut any code. But I did write the TCP/IP code myself. [It was a very subtle programming task].

Yost: That was one of six prototype implementations?

Braden: Right, exactly. That was fun for me. You know, I had to write in IBM assembly language, because that was all that was available for the IBM system. (IBM had a PL/I-based system programming language that they were using internally, but they refused to release it to customers, which was a great disappointment.) I would show up at working group meetings with my binder of IBM assembly listings, which was two inches thick. And the others would come with their TCP written in C on three sides of paper. [Laughs.] They kidded me a lot about that. But we had designed the original NCP code in modular fashion, and the TCP/IP code just slid right in, to replace the NCP. And so, on January 1, 1983, the big Red Flag day, the cut-over day when ARPANET officially changed to TCP/IP, the 360/91 was up and running with the new Internet protocols.

Yost: Moving back, for a moment, can I get your comments on Net CRT and that character display protocol?

Braden: It was my effort to expose more of the IBM facilities to the network. No one was interested in it, so it was just an exercise. IBM had a line of displays called 2260s, which we used for our URSA CRJE system at UCLA, and I basically defined a virtual

2260. It was more of an intellectual exercise than anything else because, as I say, no one was interested in it. Most of the time-sharers used teletypes, or I don't remember what kind of terminals. By then, there were glass CRTs available; CRT terminals.

Yost: Can you tell me how NetRJS evolved in the 1970s?

Braden: Again, we modeled the protocol on the local phone line RJE system. The computer center had a dozen customers around the basin who had card reader/printer terminals. They'd read in their card decks and send them over phone lines to the 91, which would machine, and process them and get them back. I said let's mirror that structure. We didn't have a permanent monopoly on selling batch service over the network, over the ARPANET. University of California at Santa Barbara had an IBM 360/75 that they somewhat aggressively marketed. They wrote their own NCP, got all that going. And later on, University of California San Diego (UCSD) did the same thing with a Burroughs B6700. UCSD came up with a different scheme for remote job entry, and eventually we had a meeting to try and thrash out the difference, try to come up with a common scheme. I've pretty much forgotten the details. This is deep in the arcane issues of *Protocol Design*.

For remote job entry over phone lines, CCN used a subsystem called RJS, which two of our undergraduates had written at UCLA. RJS was much better than IBM's equivalent. My NETRJS remote job entry protocol for the ARPANET was simply a network mapping of RJS. The NETRJS server ran on the 360/91, and the remote user had a

NETRJS client. The scheme that UCSD preferred was that the remote user interacts with the server through a control channel, and then the server reaches back to the client and does a file transfer of the job input and output. So it's a question of who's master and who's slave, in the process of transferring data — I think — but it's been a long time since I've thought about that. None of this was intellectually very interesting. We had a service to sell, we had to do it. One interesting thing, though, about reaching back — talking about the climate dynamics problem and competition with UCSB. Because our 360/91 system had so much memory, we were able to do large-scale multiprogramming. With a large number of concurrent processes, we were able to keep the CPU busy most of the time. The issue that I faced as manager of programming was how to charge for a multi-programming environment to give the users the right incentives. I came up with a scheme which treated CPU, memory, and IO as separate cost centers. There was a formula into which you'd put the CPU time, the I/O op count, and the memory requirement to compute the total cost of the job. The memory cost was the product of CPU time and partition size. If you ran a job that used a significant part of the four million bytes, the cost of computation became quite high. Well, ARPA was not happy about that, because the RAND climate dynamics people really wanted to use the four million bytes. We ran their job in the middle of the night; we took down everything else we could and gave them as much as we could of the four million bytes. But it was too expensive, and again, Bob Kahn showed up and said we've got to fix this. So we negotiated with Bob Kahn a change in our pricing formula, putting a knee in the curve. The cost of computation increased linearly with resource use up to a certain point, then it went, and then it went way, way down. Most people on campus weren't aware of why

that peculiar knee was there, but it was to make it feasible to run the ARPA programs. And in particular the RAND climate modeling programs funded by ARPA.

Yost: From its inception in 1981, you were a member of -- I think it was originally called -- the Internet Configuration Control Board that became the Internet Activities Board. Can you talk about that group and what you see as the most important accomplishment, as well as the greatest challenges in the early years?

Braden: Well, that's a big story which has been recounted recently by a number of people. Not all of those accounts agree and not all of them agree with my memory. It was originally created by Vint, as I remember. The ICCB was created by Vint, who was the ARPA program manager and tended to be very hands-on. If he was interested in a problem and got involved, he wanted to make all decisions about what apparatus was going to be used, and how programming is going to be done, what's protocol going to look like, and so on. He truly couldn't do all that, so he created this body of 12 guys — 12 people that were all guys — mostly ARPANET researchers who then became part of the Internet working group, to give him advice. That was the Internet Configuration Control Board. I think that Vint invited me so I could create an IBM version of TCP/IP. To establish credibility for the new protocols, he needed to show that they were not just computer science research toys, but could be implemented for a leading mainframe. In any case,. I was immensely flattered. Many of the ICCB members were very bright, dauntingly bright. I was certainly not an intellectual leader, but I think I made some contributions. It was a fun time, because we were cooperating with the Europeans, so we

had ICCB meetings in Oslo, Norway, in Oberpfaffenhofen, Germany; and in Malvern, England, collaborating with the NATO military communication groups there. I don't remember the exact succession. I'm not sure, but Barry Leiner was there for a while. Over the years, the ICCB, which included most of the major players in the early Internet experiment, became more and more influential. Barry Leiner became the ARPA program manager for the Internet program. [He] told us one day at an ICCB meeting that he was going to reorganize the ICCB into task forces and that everyone in the room had better chair a task force, or you're out. [Laughs.] [At the same time, the ICCB became the IAB]. I didn't know anything about security, so I couldn't do that; I wasn't interested in Internet inter-operability particularly; I wasn't a real packet herder so I wasn't really a big gateway person and was not interested in a gateway task force; and so on. But the one area that I had worked in, over the years, was host software for the network, so I said, "I'll chair an end-to-end service task force." [Later the "service" was dropped.] I convened the first E2E Task Force meeting in 1984, and I headed that group for 13 years.

There were 11 or 12 task forces. After a couple of years, the Engineering Task Force and the Gateway Services Task Force were merged to become the IETF --- the Internet Engineering Task Force. The IETF grew very rapidly under the leadership of Phillip Gross, seriously distorting the IAB organization. It became clear that we had to reorganize again. At an IAB meeting held at the University of Delaware, Phill Gross and I had dinner together at the Crab Pot Restaurant in Newark, Delaware to talk about this problem. We suggested that there should be just two task forces, the IETF and an umbrella research task force, the Internet Research Task Force or IRTF. The other

original task forces would be renamed research groups and folded into the new IRTF. I think we drew organization boxes on a napkin. Phillip and I called this little-remembered event the Crab Pot Compromise. In any case, we took this proposal back to the IAB and they accepted it. This reorganization worked, and within the task forces that became research groups, nothing else changed, really.

I had a wonderful time running the End-to-End research group. I wanted to form an opportunity for intellectual discussion and debate among the best minds interested in the end-to-end issues and the Internet architecture. I chose the membership carefully, but limited its size to a number that could fit comfortably in a room and ensure intensive discussions. There was an inner circle of key people, it varied but usually was about 10 people. I was fortunate to attract Dave Clark of MIT, known informally as THE Internet Architect, to attend all the meetings. Other "permanent" members included Van Jacobson (LBL) and Craig Partridge (BBN). We met twice a year for 1 or 2 days. I would invite a few interesting (and sometimes controversial) guests to each meeting. I succeeded in creating meetings that were rich in ideas and information about technical and architectural issues in the Internet protocols. The members looked forward to the meetings. Generally, members would volunteer to give dense presentations on their new ideas and report progress on old ideas, then I would use these presentations to trigger discussion and perhaps debate. I crafted the agendas very carefully, with many slots for general discussion.

The E2E research group had no formal authority, but the individual members had significant influence through their regular jobs, and as a result the E2E group made a difference. For example, the E2E group was the driver behind the DARTnet testbed in which multicast and video teleconferencing were developed. About the time that the E2E task force began, Van Jacobson was in the process of working out his congestion control algorithm, and at each meeting, he would get up and describe what he was doing and why, what he learned, and what the next step was, and so on. A third example was a strong statement in RFC 2309 to the Internet community that active queue management would become vital to Internet performance. Today, some 17 years later, the predicted slowdown has now become apparent, and the IETF is now working on defining a standard for active queue management. The authors of RFC 2309 is in fact the set of full members of the E2E RG in 1998.

People would argue fiercely in E2E meetings, and then go back to their regular jobs and exert a major influence on the developing Internet technology. I would also invite some people with differing viewpoints, like Raj Jain of DEC, Sandy Fraser of Bell Labs, and Dave Cheriton of Stanford, to name a few.

Yost: So these companies obviously had significant financial interest. Can you talk about how that was handled with them?

Braden: Financial interest really didn't become important until 1991 when the Internet was opened to the public. During the 1980s, NSFNET was really only universities. It was

only when there was mass audience that financial issues became important. All sorts of ugly things had to be dealt with when that happened.

Yost: What were the years that you were executive director of the board?

Braden: I was executive director of the IAB, but I don't remember what years that was. It must've been the late 1980s but I honestly don't remember. A lot of the IAB minutes of that period, I produced. There was so much going on then, it's hard to [remember]. And then there's a whole story — which has recently been regurgitated, actually — in which the IETF ate its parent, disbanding the IAB. The IETF was noisier and had much vendor backing, so it told the IAB to go away and took over the function of standards approval that the IAB had kept for themselves. That was the famous Kobe meeting. I was the IAB Executive Director then, so I guess I was still Executive Director until the IAB was disbanded; 1985 to 1991, something like that. The IETF re-established a body they called the IAB, but it had no authority over standards.

Yost: Can you talk about OSI a bit?

Braden: Oh, yes. There were a number of critical periods where the Internet could easily have failed. Fortunately for us, the telco's thought we were dumb and that packets -- statistical multiplexing -- would never work. Congestion would kill you, which turns out to be true, of course, and we [Here "we" means the Internet research community and in particular Van Jacobson] figured out how to avoid congestion. Anyway, the telcos

convinced themselves that data traffic was trivial compared to voice traffic, and that this would always be so. So fortunately, the Telcos just snickered. The OSI folks, however, were deadly serious about being the Chosen Ones. They belonged to a recognized standards body, and they believed they should define the protocols. We regarded them as a threat, certainly. In 1989, the IAB included a person who was strongly associated with the OSI world, and he persuaded the rest of us that OSI was inevitable. It appeared that the political pressures to adopt an international standard would beat us in the end; we would be unable to resist. This was a bad judgment, but we were convinced of it. And we convinced ourselves that the OSI protocols, which by this time had morphed to look very much like TCP/IP, with CLNP playing the role of IP amdTP4 being TCP, could be a step forward. And in fact, in some ways TP4/CLNP was better. For example, TP4 had selective acknowledgements, which we didn't get in TCP until some years later. So we convinced ourselves that OSI was really just a re-engineered, more modern version of TCP/IP. And since the world was going that way, the IETF should look into this, should consider adopting the best parts of OSI. I don't think we actually said we should go this way, but we should look at what the advantages and cost would be to go in that direction. The IETF was a roomful of a thousand true believers in TCP/IP, and they went crazy. [Laughs.] They just went crazy, and there were some very unfortunate things said. Hotheads said things they shouldn't have. But it resulted in, as I said, the IAB being disbanded. And curiously, that killed OSI; they accidently killed OSI without meaning to. The ferocity of the anti-OSI sentiment generated in the IETF killed OSI as a viable option. Once the IAB was disbanded, the IETF had full control of the Internet protocol standards.

I should say that a very important factor in the success of TCP/IP in the beginning was the fact that it was included in Berkeley UNIX. One part of ARPA was doing Internet research, while another part was supporting operating system research. ARPA wanted a standard platform for their computer science research, and they gave Berkeley money to develop the BSD Unix system. The two areas in ARPA got together and told Berkeley, "You must include TCP/IP in the BSD kernel." Berkeley did not like that at all, but they came along kicking and screaming. In places where the Internet protocols didn't fit the operating system, they changed the protocol rather than the operating system [laughs], resulting in a mess that fell to me to straighten out some years later, in the Host Requirement's RFCs. I had some monumental debates with Mike Karels of Berkeley. In the end we were able to agree on common ground. But anyway, ARPA's computer science research programs were given BSD Unix as a research platform, and it had TCP/IP in it, so naturally they used TCP/IP. And so that was a big start towards broader acceptance of TCP/IP. And then vendors like Sun Microsystems had this free, open source Unix with a working TCP/IP code in BSD. Many system vendors used this code, about a dozen machine venders, probably. So BSD Unix was another important factor in the ultimate success of TCP/IP.

Yost: In 1986, after nearly two decades, you left UCLA and came to USC's ISI. Can you tell me about that career transition?

Braden: First of all, in 1981-82, I took a personal sabbatical. I went on leave from UCLA — Bill Kehl was kind in letting me do this — and spent two years working at University College London, in their Internet research program. I worked for Peter Kirstein, head of computer science at University College and one of the initial designers of the Internet. He was the only non-American to get DARPA money, I have heard. As a result, UCL had an ARPANET IMP with a TCP/IP implementation, communicating with the US over an experimental packet satellite. Peter had a very close relationship with the US. In fact, he got his PhD in microwave physics at Stanford while I was still a physics grad student, and by some weird trick of history, my wife typed his thesis. [Laughs.] I worked at UCL for two years, and that's where I learned the C language, LSI/11 programming, and UNIX. I wrote a terminal protocol translator between the Telnet protocol on the Internet side, and the X-25 network that the English academic community had adopted. This Terminal Gateway was successful.

[Nights and weekends I spent in Kirstein's basement laboratory on Gower Street, logged into UCLA/CCN over the Internet and maintaining the 360/91 TCP/IP code. Occasionally the Telnet connection would hang; I generally took that opportunity to go to dinner. When I returned I could just continue as if there had been no outage; no state was lost. That was a convincing demonstration of the robustness of TCP, and of the evil of a TCP keep-alive. As a result, the section of the Host Requirements RFC 1122 on TCP keep-alives comes as close as I could get to banning them.]

Do you want to take a break?

[BREAK IN INTERVIEW]

Braden: We have 35 minutes.

Yost: Can you tell me when did the DARTnet testbed start, and describe that project, and what was accomplished with it?

Braden: DARTnet (DARpa Testbed network) was created and used during the 1990s, roughly. Its users were the usual suspects: DARPA-funded Internet researchers, generally on either the E2E research group or the IAB. We needed a place to run wide-area experiments on Internet protocols without breaking the Internet. At that time, government agencies like ARPA and NSF who supported research were obsessed with higher speed as the only important research direction. They thought that all the research nuggets were buried in gigabits, but my friends and I in the E2E research group didn't believe that. We thought that lots of important research needed to be done where much slower speeds were adequate or even superior. Experience proved us right; a great deal of important research was done on DARTnet. Examples were congestion control, multicasting, video teleconferencing, packet scheduling for real time traffic, flow synchronization, network time keeping, and security. With Dave Clark leading the charge, we eventually persuaded DARPA to fund a set of dedicated T1 circuits, connecting about 15 research sites and with two cross-country circuits. The routers were Sun Sparcstations running a modified Sun OS, thanks to a generous donation from Sun. All members of the DARTnet community had the common root password and could reboot the routers with their own kernels. They could get exclusive access to routers,

scheduled in three hour blocks. The E2E RG actively monitored DARTnet progress, provided a discussion venue, and nurtured collaborations among researchers.

NSFNET was another one of those risks, 'just pull the fat out of the fire just in time.' NSF customarily doesn't support facilities like the Internet, they fund a researcher part time during the year and full time during the summer, and a graduate student, and that's it. Somehow, Steve Wolff persuaded them to fund a set of regional networks, subnets, around particular universities and colleges, and to hook them all together with a backbone to form NSF Net. That was quite critical because it allowed the Internet to grow, getting a whole new population of potential users, basically the academic community of the country. It also had the unintended consequence; [laughs] that NSFNET managed to break the Internet. It just broke it. We did not yet understand congestion and the fuzzballs were overloaded, and the Internet went into what's called congestion collapse, a situation where every host is trying to send data and sending as fast as it can, timing out and then retransmitting, not making any progress. By then, all the hosts on the Internet are doing that simultaneously, so the Internet's full of packets, none of which are useful because all of them are discarded. That's a stable situation, once you get there. Fortunately, Van [Jacobson] came by out of the woodwork from physics land, where he'd been designing accelerator control systems, and he figured out slow start and other schemes for congestion control and avoidance. But we actually experienced congestion collapse on NSFNET, so it became a very large research experiment even though we didn't figure it was an experiment.

Yost: Do you recall when that was?

Braden: No I don't. But based upon that — remember, the Internet is an experiment, it's still an experiment — and experiments can fail. [Laughs.] There are a number of interesting failure modes. Of course, the current one is the net neutrality issue; carving out the Internet into fiefdoms owned by big corporations. Kleinrock asked me a question.

Yost: What about the Deter [pause]?

Braden: Let's talk about DARTnet. DARTnet was where multicasting was worked out. ISI and BBN developed packet video and packet voice over DARTnet. We had a working video teleconferencing system between the two sites over DARTnet, which we regularly used for meetings. Van used DARTnet lot for his experiments. It produced a lot of good research.

Yost: And was it the only testbed of its kind?

Braden: I think so. It's certainly the only testbed that this set of exceptional people had access to. Van developed — of course, he developed a lot of things — but in particular, he developed the Mbone, a set of audio video conferencing tools, which were tested on DARTnet. Then the IETF started using it on the Internet for broadcasting their meetings. They gained a lot of valuable experience with it. It could not have happened easily without DARTnet.

Now, of course, network testbeds are not uncommon, but we have what we think is one of the premier testbeds in the country here at ISI, the DETER testbed. It is aimed primarily at security research but is used for other things. We have a 500-plus node PC cluster downstairs, which is used by multiple experimenters at any one time, typically 20 users or more concurrent and independent experiments. A lot of classes use it for classroom exercises, and serious researchers use it. It's funded by DHS. That's actually where most of my effort has gone in the last few years.

Yost: Can you talk specifically about the aspects of the DETER security testbed that you've worked on?

Braden: First of all, DETERlab began as a clone of the testbed architecture called Emulab by its inventor, Lepreau at the University of Utah, in a testbed he called Emulab. We then modified and extended it in some really important ways to create DETERlab. The primary purpose was to provide a safe environment for performing security experiments with risky code, like worms and viruses.

DETERlab is basically a rack of 500-plus high-end PC computers, each with four network interfaces that are wired to ports in a set of very high capacity VLAN switches. An experimenter defines the topology he wants. The testbed control software then allocates a set of nodes and programs the VLAN switch to "plumb" the desired topology. The experimenter now has a set of nodes for his experiment, plumbed according to his

specifications and loaded with the software he's requested. He has root access to all his nodes. Each experiment running simultaneously is isolated from all the others, and we typically have something like 20 experiments executing in DETERlab at the same time. My own interest in DETERlab right now is looking at using the testbed for experiments on cyber physical systems, like the power grid.

Yost: So it's a testbed that is used for both very significant research projects, but also to educate students.

Braden: Yes. We typically have about 20 classes using DETERlab.

Yost: From 1998 to 2010, you took a lead role with the RFC Editor. Can you talk about that position and what your goals were in doing that?

Braden: Immediately after Jon died in 1998, Joyce Reynolds and I took the lead of the RFC Editor, as a gesture of appreciation and respect for Jon Postel. So RFC editing and archiving remained at ISI even though Jon was gone. After Joyce left ISI to manage documentation at Nominum, I carried on alone. Jon had been the RFC Editor for many years, and Joyce had taken an increasing role. At the time of his death, there were maybe 2,000 RFCs. Now it's 7,000.

Sometimes it's difficult to figure out how to edit the RFC documents. There are some professional network people who never took AP English, or some have English only as a second language; their RFCs tend to need considerable editing. On the other hand, people

with writing skills generally hate to have their stuff edited. I've gotten over that, I assure you, but it used to really annoy me, even when I knew the editor was right. We tried to adopt policies consistent with Jon's policies, so for well-written RFCs we had to learn how to do "light" editing. In general, the editors strive for consistency more than anything else, and they're very good at that. I was fortunate to have two really gifted editors who did the actual work, and they are still doing it -- Sandy Ginoza and Alice Hagens. In general, of course, technical editors are generally not subject matter experts, but they can pick up a surprising number of semantic errors just by checking on consistency. For example, a word may appear capitalized in one place and not in other places; ooops, maybe that's just a simple typo, but maybe not. The Internet Society … another whole story, -- but the Internet Society was funding the RFC Editor, and the administrative head of the Internet Society chose to be kind of tough, shall we say, about funding the RFC publication functions at ISI. So I had to write formal and somewhat elaborate proposals and justify how many pages an hour that we could edit. Then ISOC would give us less money than we asked for, and we'd have to get along, and we did. But we would get big backlogs sometimes. That made the IETF unhappy, for good reason. The whole Internet standards process could be held up waiting for an RFC to be published.

One important principle Jon established was that the RFCs never change. That's one of those 99.998 percent truths, but essentially, an RFC once published never changes. So if you want to change one, you have to republish it with a new number. So it is a true archival series. Some of our customers did not understand that.

Yost: Certainly something that's been very useful to historians. Old RFCs going all the way back…

Braden: Yes, I know. Yes, back to RFC-1. I actually have paper copies of the first several hundred RFCs. The earliest RFCs were published only in paper, although very early the ARPAnet had file transfer going, and subsequent RFC distribution used file transfer over the network. Jon produced paper copies locally. They're in our vault downstairs, I guess, we have a complete set of paper copies. This probably doesn't make sense any more.

The lawyers could not keep their hands off, so we had to deal with intellectual property issues. It took endless amounts of my time to get that straightened out.

I should say it's fun to feel you're contributing something. I think that carrying on the RFC series as a contribution to the process, the IETF, and the Internet.

Yost: You chaired the RSVP working group for five years at the IETF, is that correct?

Braden: Actually, Lixia Zhang and I co-chaired the RSVP working group.

Yost: Yes.

Braden: Back in the late 1980s, the Internet research community, and the End-to-End research group in particular, were worried. The congestion control and avoidance mechanisms in TCP back off when the network gets congested; it slows down. The Internet stability depends upon that. If a lot of the traffic is not TCP or TCP-like, but is for example streaming audio and video, it won't back off under congestion and the Internet could have congestion collapse again. We were worried that in the future there'd be a significant amount of video and audio teleconferencing traffic on the Internet, although we weren't there yet. We discussed this problem in the End-to-End research group for some time. In the end, the best idea we could come up with was to develop what we called integrated services (RFC 1644, I think). Part of this proposal was a resource reservation protocol RSVP, which could set up "calls", connections with admission control, for streaming traffic. If there's not enough guaranteed capacity available in the network to carry your flow, RSVP could give you a busy signal. RSVP's fatal flaw was that it could require a lot of state in the routers, and the IETF has a visceral reaction against state-full routers. The E2E group believed it could become necessary, whether or not the IETF approved. And when we consulted people who really built routers, they said, "Ah, this doesn't worry us; memory's cheap, we can do it." Nevertheless, the IETF reacted against RSVP, and it's not much used. Its protocol complexity did get out of control. We kept adding features to RSVP in response to user requests, and RSVP got too complicated. I don't count RSVP as success, although if we hadn't done it, we wouldn't know what not to do. [Laughs] I mean, the RSVP spec serves an important function, just being there as a placeholder for some connection-oriented mechanism for setting up calls. With the rapid growth of VoIP, the Internet may well

come to require significantly more router state, following the path traced by integrated services.

I should make it clear that Lixia Zhang at UCLA was a major contributor to RSVP. She originated the soft state design of RSVP, built around IP multicast. She and I co-chaired the working group, although in practice I mostly ran the working group meetings. I wrote, and she rewrote, the RSVP RFC. Deborah Estrin also made major contributions. Lixia and Deborah were members of the End-to-End task force/research group.

Yost: Okay. Before we conclude, are there any topics that I haven't brought up that you'd like to discuss?

Braden: I'll probably think of them this afternoon. You might mention that one of my current interests is application of DETER to cyber-physical systems, in particular to the electrical grid. The computerization that's happening in the electrical transmission grid revives some very interesting old problems about network protocols. They are planning to stream data on a wide scale, and they plan on using TCP and UDP and so on. And it isn't clear that simply applying the Internet protocols in a straightforward way will solve their problems, because they have unusual requirements for robustness and reliability, and low and constant latency. So the same problems keep coming up.

One of the things that I had started doing but have not — I've gotten diverted by work — is to put online an emulator for the Burroughs 220 that I've written. I have an original assembly listing of the BALGOL compiler, which I typed in, and I'm close to getting the BALGOL compiler working in the emulated environment. I think that'll be interesting to

get that online and have people play with it. It's fun. I spent a lifetime in computers because they are fun.

Yost: Thank you so much for doing this interview. It's very useful.

Braden: Thank you.