Oral History of Randy Katz

Interviewed by:
Dag Spicer

Recorded July 14, 2022

CHM Reference number: 2022.0092

Spicer: Okay. And Randy Katz, you're here today at the Computer History Museum, and are you okay with me recording this conversation?

Katz: Absolutely. Do I have to click on something? Yeah.

Spicer: Oh, yeah, I guess so.

Katz: Yeah. Good.

Spicer: Yeah. Okay. Great. Well, it's July 14th, which is, in fact, Bastille Day in France. <laughs> Not that that matters. But anyway, welcome, Randy, and thank you so much for talking to us today. You've done a lot of great things up at Berkeley and elsewhere and we want to hear about those things. Usually the way to start this interview process is to go back to the beginning <laughs> to the primordial prehistoric era, if you like, which is kind of your family history. Like how did your family get here? What did your parents do? And we'll start with that and then move on.

Katz: Okay. So happy to talk about that. I would say I had a very unchallenging, unsurprising childhood. My parents were working people in New York City. My father, by the time I arrived in the family, was a taxi driver. He owned his own taxi cab.

So a small businessman, working pretty much six days a week. He had-- he, and really both my father and my mother's family, had sort of prior to the 1950s, been working in the garment industry in New York City. So my dad was a so-called cutter. I remember being told that he used to operate these machines which were cutting out patterns for clothing, pretty much women's <laughs> foundations, as they called it back then, but, you know, all-- pretty much the entire family worked in, both on my mother and father's side, that's pretty much what they did. My dad fought in World War II. He was in the Pacific.

When I was born the family was living in public housing. A few years before that, you know, the 1950 census just came out, so I looked them up and not really-- this was not something that was unknown in the family, but in 1950 census my older sister, my mother and my father were living with my grandmother, which was not surprising. You know, there just wasn't enough housing, right, at that time, and they moved into public housing, and we didn't move into our own home, owned own home until the 1960s. So working people. Not the-- no one in the family had really gone to university or college, so both my sister and I were first-generation and our cousins, first cousins, are also first-generation in the family to attend university.

You know, I've often tried to think about what was my first exposure to computers? And because it's not something you could imagine in that family environment <laughs> really being a topic of conversation, and, you know, I really, in retrospect, think that it was watching television programs like "Star Trek" that really first introduced me to that, because I was 11 years old, 1966, when "Star Trek" first came on TV, and I remember fighting to stay up to watch it, because I think it was on Friday night at 10:00 P.M. or some late time slot <laughs> for an 11-year-old. But I know a lot of people sort of came away from

programs like that with a desire to become a scientist or an astronaut or something like that. Somehow the image of intelligent computing in that had a profound effect on me, and sort of moving forward from there I can date my interest in computers, without having the least ability to access one or program one, to that time frame of the late 1960s.

Spicer: That's amazing. You know, I've heard that over and over again, and a variant on that is people say, "I wanted to be Spock," <laughs> … the rational, scientific… you know. That's fascinating that you bring that up. So well, tell us a bit about your schooling now. How did you-- did you enjoy it? Did you have any activities or clubs in school that you joined, that kind of thing?

Katz: Yeah. Love to. Went to public schools in New York City. The kind that you, you know, it's a 10-minute walk from where you live.

Spicer: Nice.

Katz: Pretty cool. So again, I have deep memories of even my elementary school, because when they built schools in those days they really built them to last, so that particular school, which was Public School 114-- I think it's closed now, no longer open to students-- but it had this staircase, you know, a grand entrance to the school, with an iron staircase-- you know, here it is 50 years later, <laughs> 55 years later, and so, so clear in my mind, this sort of incredible staircase surrounded by marble with these, you know, prehistoric, Precambian, you know, fossils embedded in the marble, and I still remember--

Spicer: Oh, wow.

Katz: --that to this day that, you know, when they built these things they built them as palaces kind of. You know, a school is a palace. But really my interest in computers grew in junior high school, which had, you know, I think in good-- you know, I was going to these schools, again, in the time frame of-- junior high school was late 1960s and high school was early '70s, and we benefitted from having teachers, I hate to say it, but they were male teachers, for science and math classes, that were probably trying to avoid the draft to get into the Vietnam War or to avoid being called into the Vietnam War, but they actually were really committed science and math teachers and they-- in these schools they taught those classes very well. It wasn't until high school in 1970 that I laid my hands on my first computer, so at this time I'm about 15 years old and the high school I went to, Canarsie High School, it's in the eastern part of Brooklyn, was actually very well equipped for its time and place with computers. In fact, the first computer that I programmed was an Olivetti programmable calculator, Olivetti 101 I think it was called, and the first--

Spicer: Yeah, the Programma. Programma 101.

Katz: Programma, which you probably have in the collection somewhere, and--

Spicer: We do. It's on display.

Katz: You know, it was kind of, in retrospect, a really fascinating machine, right, for its time, again, and that a high school, you know, sort of a public high school, had access to something like that.

And the very first program that I programmed, because the lab was open to students and while there wasn't a formal class in computer programming there was kind of a club. You mentioned, you know, "What clubs were you in?" I was on the math-- in high school on the math team, and on the science magazine thing. I was in those sort of things, so I had access to the computer lab, and the first program I wrote was, of course, generating Fibonacci numbers, you know, very simple, incremental loop type program. We also had a computer in high school called the Digiac 3080, which was-- and there was a textbook on--

Spicer: Wow.

Katz: --on how to, you know, what was essentially machine structures textbook, which I have actually in my office at Berkeley somewhere. I stole the book and kept it.

<laughter>

And it was written by I think a professor at Queens College or, you know, one of the local colleges in New York City, and it was a real programmable computer. You could program it, you could write your programs as, you know, programming the bits and the instructions on paper tape or by switches. You could enter it, and you could really write much more substantial programs on it. It was a, I think, a 36-bit architecture. It had what you would expect in terms of, I don't know, 16 or 32 instructions that had all of the arithmetic conditional branching. I mean, it was a real thing, and we programmed it in ones and zeros, but, you know, machine language, and could do some really interesting things with it, and you could do this with your own hands, and that was in the computer lab, and then the third ability was… there was a remote job entry station for an IBM mainframe so you could program it in PL/1, for example and, you know, other languages of the time.

Spicer: Okay. So a couple of questions. [What?] was the mainframe off campus for the RJE (Remote Job Entry)?

Katz: The mainframe was off campus. Yeah, it was run by the board of education and-- but the funny thing is some of my colleagues, you know, colleague students, somehow talked their way into machine operator jobs, you know, at night or on the weekend or something for that computer, so we got to actually go, you know….

Because you had friends who were working there and you could kind of go <laughs> and knock on the door and they'd let you in and play around with this computer that was probably running payroll for the board of education. Who knows what was running on it at the time, but that was pretty--

Spicer: Wow. You mentioned machine language, which if you remember John Backus talking about his motivation you know, pre-FORTRAN, he described writing in machine language as hand-to-hand combat with the machine. Was just a wonderful metaphor.

Katz: Yeah, no. I mean, really, to write a, you know, anything other than the most trivial program, the way that you would normally do it is by actually putting the program in the memory by the switches and sort of load switch and put the result in a register or something that you could then read as lights on and off on the-- on whatever was the, you know, sort of keyboard-sized panel associated with computer.

But yeah, I think looking back on it and with more of an appreciation for the history of where computing was in circa 1970, it's incredible that our high school had access to, you know, like the sort of computer that a kid would have, you know, a home computer maybe 10 years later or 15 years later. We had it in 1970, which is, in the world of IBM mainframes and all this other stuff, we had a hands-on computer as kids, which is pretty cool.

Spicer: Yeah, and one question I want to go back to. You mentioned briefly the name of the machine, which I have never heard of. It's quite unusual. Can you say that again?

Katz: Yeah. It was called the Digiac 3080, and they're actually, if you do a Goo-- D-I-G-I-A-C 3080, the numerals, and you can find the instruction set online. I have a link to it, actually--

Spicer: Interesting.

Katz: --of notes I had put together for us today, <laughs> so you could do a Google search and you'll find, you know, essentially the machine language card for--

Spicer: Now, how big is this? It wasn't-- was it a, you know, like there was something called the Geniac, for example, which was a bunch of Masonite disks, you know -- it teaches you principles of computing but it's not a computer. W as this thing a proper computer with--

Katz: Oh, yeah.

Spicer: Yeah.

Katz: It was a real computer. I mean, it had, you know, very complete instruction set-- <laughs>

Spicer: Okay. How big--

Katz: --associated with it. It was--

Spicer: How big was it? I'm curious.

Katz: It was the size of a, you know, sort of desk.

Spicer: Really?

Katz: The kind of, you know, work desk you would have.

Spicer: Very cool.

Katz: It had, as I recall, 4,096 words [of memory].

Spicer: Wow.

Katz: Thirty-six-bit words. The instructions and data were-- you know, I can't quite remember why it was 32 plus 4, but maybe it had parity or something like that, so it had a core memory. It did not have any sort of magnetic storage with it, so you could, as I mentioned, programming it by inputting via the switches, or we did have a way of both programming it by preparing a paper tape, put the program on it, and it could also output paper tape. So there was a paper tape reader/writer that was associated with it, but the sort of display for it was kind of, you could kind of imagine what 36 switches and lights and an accumulator type thing, which was also another row of lights. I think it was several rows of lights. It was probably, you know, I don't know, 15 inches by 15 inches type display sitting on top of what amounted to a desk and underneath the desk was the electronics for this computer.

Spicer: I think I see something on eBay. Did it have a bunch of patch cables on the front?

Katz: That, you know, that's possible that it did, but I'm not sure. You know, that's not something that's-- there may have been some capability to do that but that is not what strikes in my memory on this.

Katz: You found the digiac3080.wordpress.com?

Spicer: Yeah. I'm really impressed with the size of the word length.

Katz: So it actually… that's exactly it. We did have a typewriter gadget. You see that in that picture on that website. It has the panel, and behind it is a card reader, but I don't remember preparing cards for this machine, although we did have card punches for the IBM remote job entry. But now that I see the picture of it, I think we did have a typewriter set up for some kind of input and output in that panel that you see also in the blowup picture to the right there.

Spicer: Yes. Oh, that's great.

Katz: This thing is about, you know, it's basically the instruction set on it.

Spicer: I wanted to ask you about this science magazine that you mentioned that you were working on in high school. Tell us about that.

Katz: It was basically a science club sort of thing and it was a student-run journal, and a group of us were writing articles for it that would be kind of Popular Science. It wasn't intended as high school student research results. It was more like writing about scientific breakthroughs of the day, so we were like science reporters. By the way, one of the things I'm very proud of is one of my classmates, her name is Helene Elliott, was the editor of our school newspaper in high school and she's still a reporter for the Los Angeles Times in sports, and, you know, again, when you kind of think about, this is a little sideways to computer science, but it's related to the generation that I represent that, you know, a kid from Brooklyn could go to Northwestern, get a degree in journalism, be a woman and get a job in journalism in sports in, you know, when you graduating from college, which is, you know, like 1977. That those careers would then open and you could advance to cover sports for the LA Times, which is, you know, a pretty big gig for a reporter, I would think.

Spicer: Yeah, that's great.

Katz: You know, these are the kids I went to school with and I'm just very proud to have been involved with them and to be of that generation of students that emerged from the education system with those sort of opportunities in front of us.

Spicer: That was spectacular, like your generation and the one just maybe a bit older too. Like I'm thinking of there were these amazing schools in New York City and The Bronx High School of Science too, and like--

Katz: Yeah. I do want to, you know, there was, of course, The Bronx High School of-- there were three magnet schools in New York City, Bronx High School of Science, Stuyvesant, and Brooklyn Tech, and I grew up in the, as I mentioned, you know, about as far east in Brooklyn as you can get, part of Brooklyn called Canarsie. And, you know, the advantage of being four blocks from a high school compared to getting on the subway and even going into, let's say, Stuyvesant, let's say, was, you know, 40 minutes on the subway each way. I mean, the-- of course they were tremendously competitive schools to get into and fantastic teachers and the products <laughs> that came out of those schools, you know, the sort of number of Nobel Laureates and all these-- obviously it attracted tremendous talent and enhanced that talent. You know, I think I did pretty good going by--

Spicer: <laughs>

Katz: --going to a public high school, so...

Spicer: Yes, you did.

Katz: So I think there is something to be said about just the quality of education that existed in that particular time frame and, you know, when I reflect on it I think part of it was the sort of way in which the city of New York had invested [in education]. You know, the high school I went to probably had been built and opened in the late 1960s, so, you know, it wasn't very old when I went there, which may explain why it had such a good computer lab, because it was a new school, well financed. Very different world in

public education these days in many cities, but part of that also was the quality of the teachers, because as I mentioned, they were <laughs> dodging the draft. So, you know, there was a kind of particularly STEM-oriented teachers who were maybe attracted to a career in education that, you know, in a different time-- because I know some of those teachers, you know, after the Vietnam War, they went and they went into finance and they became stockbrokers or--

Spicer: Okay. <laughs>

Katz: You know, they followed different kinds of careers, but also a lot of them did stay in education for their careers. I've been in touch with a couple of my high school teachers even to this day and they stuck with it long after I graduated.

Spicer: Right.

Katz: But maybe we should talk about college a little bit.

Spicer: Yes.

Katz: So--

Spicer: I was just going to mention one thing before college, which is that there was a period in American history, mainly post-Sputnik, I think, but maybe earlier, post-World War II, when there were a lot of PhDs teaching high school, and I'm not a hundred percent sure on the reason. The reason you told me about the draft, that's amazing. I didn't know that, and that accounts for the later cohort, but there was one right after World War II and around in the '50s when it was not unusual to have high school teachers with doctorates in chemistry or physics teaching those subjects in high school, so that's great.

Katz: Yeah. You know, again, as a, you know, sophomore, junior, senior in high school, 1970 to '73, my math and science teachers, I'm not sure a single one of them was over 30, and so they were young, recent graduates. They had been intro-- you know, think of their own schooling was not that much in the past compared to when they were teaching students like me. You know, I didn't have 60-year-old science teachers. I mean, you know, it is kind of interesting to reflect on what the demographics and dynamics were of that time, so I definitely feel like I benefited from it but I also realize, you know, the world is quite different for 15, 16, 17, 18-year-olds today <laughs> than it was back then.

Spicer: Yes. Yeah. All right. Let's go on to college, as you suggest. So yeah.

Katz: Okay. So college, you know, I actually, I didn't mention this, but I was the valedictorian of my junior high school.

Spicer: Oh. Hey, great.

Katz: And the salutatorian of my high school, so I had a good set of choices of where to go to college. I decided to go to Cornell primarily through the influence of my physics teacher, John Augenstein. He was a Cornell grad and he took a group of us to visit the campus somewhere in the fall of the-- you know, like the year you're applying to colleges. So I was-- senior year he took five or six of us, put us in a van and drove us up to Ithaca, and that was really the only non-New York City-- I also applied to Columbia. I'd gotten into Columbia. But that was the only non-New York City college that I actually visited, so that was the one <laughs> I decided to go to.

Spicer: Ah, lovely.

Katz: And by happenstance, of course, it had, you know, an outstanding computer science department. So I went to college with the idea that I probably would be pre-med and I took a computer science class from David Gries, which was the canonical Introduction to Computing with PLC, you know, kind of a student-friendly version of PL/1. I mentioned that we actually had been able to program in PL/1 in high school, which is, you know, kind of unusual. Probably no one listening to this who was, you know, born later <laughs> knows what PL/1 was, right, but it was IBM's procedural language, not unlike an ALGOL or a Pascal type language, all the usual sort of linguistic features, but maybe not quite as pointer and type, safe, as, you know, later languages would be, but a powerful language, and, you know, introduction to computer-- to compilers, linkers and all this.

Anyway, at Cornell David taught the sort of Introduction to Computer Science course where you learn data structures, and there were a lot of things that I had never been exposed to as a programmer in high school, and I remembered the day I decided, "Screw this pre-med stuff. I'm going to be a computer scientist," was the lecture on inserting and deleting items from a linked list. It is so clear in my memory that as David Gries went through and described the algorithm for inserting and deleting in this complex structure, and I have to say, I left the lecture not understanding a thing. How did that work? I don't get it. What was going on in there? So I went to my dorm room and I'm sitting there with four colored pens sort of working through with little block diagrams creating the block, creating the pointer to it, putting the pointer in the predecessor, you know, breaking links, making links for inserting and deleting, and the beauty of an algorithm, that "this is the code and this is how it executes" is kind of like the experience of learning, in some sense, you know, it struck me that it was the same experience as learning to read.

When I was first exposed to reading <laughs> in elementary school, you know, I kind of like, "How does this work?" and the teacher says, "Just think about the words on the page and your mind will speak them to you." So I started kind of mouthing the words. You know, sort of in your brain you're-- know the words, but I'm mouthing the words and the teacher came over and said, "Don't mouth them. You'll understand the words without that," and, you know, that thing <snaps> clicks and you the-- sort of the image of the word. The same thing happened with, "Here's the code, and the code executes, and this is what the code does. It creates these pointers. It sort of inserts them in something. It sort of takes it out and copies it over," and that sort of executing and simulating the static code in your mind was akin to looking at a printed page and the words are echoed in your brain, and it was just such a beautiful moment for me that introduced me to, "This is the science behind, this is the--" you know, "This is an incredible intellectual activity. This isn't just like a fun hobby thing. This is beauty. This is a conceptual beauty," and so from

that point on I decided, “Wait a second. No more biology.” <laughs> “I’m taking computer science classes.”

And so I was fortunate that Cornell had a program called Cornell College Scholars, and it allowed me to create a computer science major before there was a computer science major at the time. The department at, you know, so we’re talking now 1973 when I took this class. I think the department at Cornell had only been established in maybe ’65, ’66. You know, the faculty there were, you know, had been on the faculty for less than seven years, you know, seven or eight years. This is very early in the whole thing, right.

Spicer: Yes.

Katz: You know, most of the people who were teaching me were either-- had degrees in math or maybe in a scientific discipline. Maybe just a very small number of them actually had a PhD in computer science. So I feel like I was taught computer science maybe not by the first generation of computer scientists but by the first generation of academic computer scientists, people who had gotten their degrees in computer science. So I feel, you know, I’m 67 years old, that my college generation was maybe the second generation of those who had been formally trained in computer science even though, of course, computing goes back to the ‘50s, and there could’ve been some earlier departments of computer science but the-- offhand I don’t remember which was the first one, Purdue or whatever, but sort of there were a bunch that were established in the time frame of 1965, ’66, and I benefited. You know, it was kind of like being in the right place at the right time from that.

Spicer: You know, I’m thinking also at Cornell had a very effective, although, you know, I’m thinking of Frank Rosenblatt, for example. I don’t know if that name means-- but yeah, he was there really early doing that kind of early--

Katz: The courses I took was from, you know, I’m blanking on his name, but there was a department at Cornell called Theoretical and Applied Mechanics in the College of Engineering. You know, it was an applied physics kind of department and one of the courses I took was from an elderly professor who had worked with Rosenblatt, and so I actually took a course somewhere around 1975 or ‘6 which was an introduction to perceptrons. [Professor David Block, Fall 1975, T&AM H601 “Introduction to Bionics and Robots”. According to https://ecommons.cornell.edu/bitstream/handle/1813/18056/Block_Henry_David_1978.pdf, “He collaborated with Frank Rosenblatt, the builder of the “Perceptron,” a self-organizing learning machine which provides a deliberately simplified model of the human brain;”]

Spicer: Oh.

Katz: And that kind of thing. So, you know, he was still teaching that, those concepts. Where, you know, primarily the mathematics behind that sort of approach… essentially it was neural networks really. I mean, I guess they--

Spicer: Yes.

Katz: --they've been reborn in that formulation now, but it was very, you know, sort of a really interesting class from one of the more elderly faculty at the time.

Spicer: Yes, and you mentioned computer science as a discipline and how it came, you know, I don't know if I have the name right, but I think Richard Wexelblat may be considered one of the earliest PhDs. I don't know. Andy van Dam.

Katz: I think that's him, yeah.

Spicer: Andy van Dam is pretty early too, I think.

Katz: Mm-hm.

Spicer: But anyway--

Katz: Yeah, and at Berkeley, which we'll get to. You know, Jim Gray was the first--

Spicer: Oh, right.

Katz: --computer science PhD also in that sort of time frame of kind of the middle 1966-67, you know, around those kinds of time frame. You know, the other thing that was a tremendous advantage at Berkeley was Richard Conway, who was a-- I'm sorry, at Cornell. Richard Conway, who actually was a professor of industrial engineering, had really in a very early way committed a lot of his time and energy to teaching people how to program and develop that entire, you know, PLC system, and also there was a FORTRAN system which, WATFIV, which I think came from one of the Canadian universities, Waterloo.

Spicer: Waterloo, yeah.

Katz: And they sort of created--

Spicer: I used that.

Katz: They created an environment that allowed students to submit batch programs in-- they created a very nice environment for students to be able to kind of run a lot of computer programs <laughs> inexpensively, and they built around the PLC compiler auto correcting your syntactic mistakes and providing a lot of really great support for beginning programmers. So it was, again, the right place at the right time, I think, for my generation of sort of early computer science, you know, formally trained computer science students.

A couple of other things about my Cornell experience. I was lucky to get a job as a computer consultant at our social sciences computer center at Cornell. It was in a building called Uris Hall, Old Rusty, and it had terminals, and it was connected to the time-sharing option of IBM. So I actually had the ability to kind of do work with VM/370 virtual machine system while as an undergraduate. They had a very nice

interactive editing job submission, results, time-sharing system that, you know, I'd never used it in my classes. It was actually set up for the social scientists to be able to use statistical software and stuff like that, which I would, you know, help them with, but it was very cool to see the dawn of interactive… sort of broader IBM supported interactive computing again in that time frame about mid-1970s.

Spicer: Exactly. I wanted to follow two threads. One is your personal educational experience through the '60s and '70s, but then also, what's happening in computer science? I know this is a giant question, so don't feel like you have to answer it comprehensively, but are there any grand brushstrokes you can say about computer science in the '60s or '70s? How was it changing as a discipline? That would be useful to know.

Katz: Yeah, and it's something I really do want to capture when I-- when we transition to discuss grad school.

Spicer: Oh, okay. Perfect.

Katz: You know, so maybe we should do that, so--

Spicer: Yeah.

Katz: Because the contrast is really interesting. I graduated from Cornell in three years to save some money, and also because, again, the high school I went to I had a lot of so-called advanced placement credits. So I had a year of calculus, I had a year of chemistry, I had, you know, I had a bunch of, you know, the advanced placement you kind of credit it when you go to college, and that meant that if I wanted to I could graduate three years instead of four, so I decided to do that, and I had also the benefit, although was a sobering benefit, of taking some graduate courses in theoretical computer science at Cornell, because there weren't a lot of undergraduate classes and I sort of ran out of them even in three years, so I took theoretical computer science courses from Juris Hartmanis and John Hopcroft.

Spicer: Oh. Wow.

Katz: Who, of course, all those guys wrote the textbooks that we were using, and also would go on to win Turing Awards, so what a tremendous experience, <laughs> right, as a--

Spicer: Incredible.

Katz: As a junior, senior, to be able to take those classes, although I did not do as well in those classes as maybe <laughs> some of the students that I was taking them with, like people like Dexter Kozen and, you know, other people who became leading lights of theoretical computer science with their PhDs from Cornell. Anyway, [I] applied to multiple graduate schools with a little bit of a handicap because I graduated in three years. So I didn't have quite the same record maybe as if I <laughs> had stuck around for four years. I was little naïve on that, but I did get into Berkeley, and several friends from Cornell, Mike Sipser, who's on the faculty at MIT now, and a couple of other friends, had gone on to Berkeley for grad

school in computer science, and I decided to follow them even though I actually didn't get financial support, so it meant, you know, taking out loans to go to graduate school in computer science, which is sort of mindbogglingly-- how could that have ever been the case?

<laughter>

Katz: Fortunately for me, as soon as I got to Berkeley I was able to hook up with Larry Rowe and teach as a teaching-- getting support as a teaching assistant actually for a course in machine structures, which turned out to be the course of my career I most often taught.

Spicer: Ah.

Katz: And it was the very first course I was a teaching assistant for at Berkeley. But here's what I wanted to, you know, to come back to your question, Dag. So 1976, I had been exposed to the sort of one-off computers like the Digiac that we talked about.

But most of computing that I'd been exposed to at Cornell, I mean, at Cornell and even at high school, was IBM. The world was dominated by a small number of these large computer companies, and really, IBM was dominant, and it's really interesting in retrospect as we talk about graduate school, that in them days, in order to sell you a computer IBM more or less gave you the software, and my sense was that software was free. You, you know, in order to make the computer useful, IBM had compilers and an operating system and this other stuff, but it really was about selling you the hardware, and so you had these tightly integrated stacks. There wasn't a lot of third-party software that was out there. If you were an airline, you didn't go and buy the reservation system. You're American Airlines. You hire a bunch of programmers to use the tools from IBM. You create the thing. So there isn't a big, third-party software industry in 1970.

Spicer: Right.

Katz: I graduate in '76, and, you know, the world is going to-- it's like a nuclear bomb got dropped on it in just a few years, because there isn't a big third-party software industry. You have an industry that's dominated by a large, you know, sort of a small number of very large corporations that are soup to nuts fully integrated proprietary software, oh, you know, and hardware, and the world is about to change dramatically. The microprocessor comes along. You know, Intel, 1972. It takes-- MOS technology. You know, new memory technologies are coming down the road. The first microprocessors are emerging in the late '70s. Now you could, you know, in the latter part of my graduate career, Larry Rowe had a great relationship with Apple Computer. We had Apple IIs to play around with, you know, not really to do serious work with. They're emerging by the end of the 1970s. The IBM PC something like 1982. You would know-- probably know the dates better than I do.

Spicer: '81, yeah.

Katz: But between 1976 and 1980, the microprocessor, the emergence of personal computers in the background, minicomputers, the PDP-11, the VAX, are also happening and are transforming the nature of scientific or engineering computing, but they would get transformed again by early '80s with workstations that are fueled by the microprocessors from Intel and Motorola. So it's a very short cycle and quickening pace of evolution and revolution, and while it's fun to map the generation of computers from IBM to DEC to Sun Microsystems between 1976 and 1982, the world is entirely different in terms of the impact that microprocessors have had on it. But also the world of software is starting to open up because of now, you know, IBM opened the architecture of the IBM PC, meaning they no longer had that tightly integrated stack of software, and now there are third parties who are developing the operating system, CP/M or DOS, and the application suites, spreadsheets and so on, and there's like this flowering of an industry that hardly existed pre-microprocessor.

When I went to Berkeley, I spent the summer of 1976 working at an office of the administration of the city of New York developing basically applications for a New York City government office, and databases. I got exposed to databases for the first time. Data collections, how they used it, how they collected it, how they analyzed it and processed it. So when I got to Berkeley the first thing I did was to walk into Mike Stonebraker's office and say, "Professor Stonebreaker, I'm really interested in databases. I've had this eye-opening experience of the summer job I just left. Can I work with your group?" and this is the time when they're doing the early work on the Ingres project, one of the first relational database systems.

I keep talking about being at the right place at the right time. So I was able to essentially join this project like a year or two into a five-year project, and-- but the thing from a historical perspective that I find extremely interesting about this is, first of all, they're developing a major piece of application-oriented software, a database management system, on top of a minicomputer architecture. So it's now migrating from-- you could only do this sort of thing on a mainframe -- to now a smaller kind of computer that well-financed universities can get their hands on. You don't have to be a Fortune 50 company to have the computing power to do this sort of thing. On top of Unix, which is sort of first-generation of open-source software that many, many people are using.

So there's community development around this, and that was a big aspect of Ingres -- the way in which the software was being distributed to other people who wanted to get their hands on database technology, which, you know, sort of further evolved into the entrepreneurial efforts of Mike Stonebraker and Gene Wong and Larry Rowe to commercialize Ingres as a university software spinoff based on the source code developed at Berkeley. But of course to be a real product it must be much further enhanced and redeveloped and all those things in an effort to pioneer a new third-party software industry segment -- the relational database -- and because this is the age of the beginnings of Oracle and Ingres, if you're a corporation and you're not going to develop your own software for data management, you're now have a mindset of, "Okay. I'll enter a marketplace and buy it from a company."

This is, to me, the beginnings-- now, of course there's the beginnings of the spreadsheet and the word processor and all of those as elements of desktop software and an industry around that. But, you know, there's a whole other software segment that fueled corporate America spending a lot of money for software not for their PC desktops but for the enterprise computing systems that opened up in the early

‘80s, and I think that Ingres may well have been-- was certainly a pioneer in that space, if not, you know, it certainly was among some of the very first companies that was a university spinoff feeding that marketplace.

So I find because today in sort of 2022, software is just such a dominant piece of the computing puzzle driving the field forward. Not to minimize architecture, but in some sense, the history of computing in the 1980s was-- I used to make a joke when I was teaching courses about this stuff, that it was like computer zoology. You know, you talk about the machines, the IBM Stretch and how that led to the 370 and the sort of antecedents of the DEC architecture and where the microprocessor and the 8080 and the 8086 and the 186 and the 286. You know, it's kind of like these family trees, but there was this whole other thing happening of the development of software as an industry, which I find extremely important and interesting, that really got it started in that 1980s, and I think I had the benefit of doing my thesis research in the context of the Ingres project at Berkeley, which was really one of the early examples of that, and also, of course, university entrepreneurship.

Spicer: Right. So can you just tell us a bit about your role on that project? Although you did a bit already, but maybe dig in a bit there, and then what came after?

Katz: Yeah. So I worked with Professor Larry Rowe on my master’s, and we were working on software that would make it easier to develop database style software, and I enjoyed working with Larry but, you know, I was sitting with the Ingres group and I can’t claim to have sort of developed any of the software code that was going on, you know, getting integrated into the Ingres system. This was stuff that was sort of built on top of it, but I was sitting in the same office with this army of undergraduates who were developing the code. Mike Stonebraker, you know, always believed in the power of a chief programmer, team led type thing. So Bob Epstein, who went on to found Sybase, and before that a company called Britton Lee, was the--

Spicer: Oh, yeah.

Katz: --graduate student at the time who was the chief programmer of the Ingres project and was running this, you know, five or six undergraduates who were doing a lot of …

Spicer: Oh. Wow.

Katz: …software development, including a name you’ll know, Eric Allman, who, of course, is a pillar in the story of electronic mail in the internet, involved in of all this. So I was working on some application and programming language tools to support database development and then my actual PhD advisor was Eugene Wong, and he came back I think from a sabbatical-- I’ve forgotten where he’d spent his sabbatical-- and was, I think, interested in recruiting some graduate students. So he spent a lot of time talking with me about ideas and sort of seduced me away from Larry, and my PhD under Gene’s supervision was about integrating together the two primary commercial database systems at the time, which were essentially graph-oriented, so-called COADASYL databases, and the table databases of

relational. So I developed a set of techniques for mapping between them structurally and also the software that was accessing them for my thesis.

Now I think being at the right time and the right place, it's very interesting to go back and look at sort of-- and this is something I'd love to sit down and write up because it's so interesting to me. Berkeley, between 1976 and 1980. My cohort as graduate students included Bill Joy, who had joined the department in 1975, Eric Schmidt, who was class of 197-- you know, also became a graduate student 1976. We had people like Eric Allman and others who were—Kurt Shoens. People who were going to make tremendous contributions to development of Berkeley Unix. Olzalp Babaoglu, who's a professor in Italy now. You know, there were people who-- the faculty were Domenico Ferrari, Bob Fabry. You know, Ken Thompson had spent a sabbatical maybe year before I joined graduate school where he introduced Unix to the Computer Science Department. Bell Labs' Unix. And he had offered, like a seminar class of, "Okay, kids. Let's sit down and read the code." So there were all these people who were, you know, read in on the internals of the AT&T Unix, and they went and started modifying it, so this is-- this, of course, is-- Bill Joy is perhaps the best known for developing new editors and going in and sort of hacking on the Unix code, and eventually, by the time frame of the latter part of the 1970s, the DARPA contract to implement TCP/IP in the Unix stack for execution on VAXes, instead of BBNs, IMPs and TIPs, -- really established a tremendous reputation for Berkeley in operating system and software development. But just a tremendous, incredible set of people who were working on enhancing the software and becoming famous because they distributed it. Again, it was an early example of open source. Some people say, well, the Berkeley copyright notice, you could use it for any purpose as long as you attribute that the code is based on code that was developed at Berkeley, as opposed to the new license, which is, you know, if you make a change you have to give it away for free. It had the basis of allowing commercialization on the Berkeley code base, so is it really open source? But they distributed the software with the comments in it to anyone willing to pay a tape charge.

Spicer: Wow.

Katz: You know, sort of pre-widespread distribution of software over the internet, and so, it was just a tremendous privilege to be at Berkeley in the midst of all this. While working with the Ingres group you have Eric Schmidt working on local area networking, building -- people like Kurt Shoens and Eric Allman -- building the first sort of Unix-based, internet-based mail applications. So, Dag, one of the things that frustrates me so much is I'm sure that somewhere in the time frame of 1977 I sent an email message to someone who wasn't at Berkeley, but I can not to this day remember who that was, or, you know, from the viewpoint of 2022, the ability to send electronic mail, not to someone in the office next door but someone on the other end of the country, was such a momentous red letter day in the history of computing and my own personal experience of computing, yet I cannot remember doing that. I mean, it's amazing to me the, you know, the technical capability that if you were at sort of a major university in the late 1970s of what you could do. I mean, I'm sure I took it for granted. You know, it's probably some old chum from Cornell who was at MIT or Stanford or someplace like that, you know, reaching out and saying, "Hello, look. We don't have to talk to each other on the phone. We can kind of send an email to each other," and how that experience of computing is almost universal today. You know, think to

yourself. Do you remember the first time you sent an email to somebody? You know, for people of a-- you may remember-- <laughs> you know, some people--

Spicer: Not-- yeah.

Katz: I don't.

Spicer: I don't remember the specific email but it was in grad school. They gave us an account. This would've been in the '90s, so it's nothing compared to yours, but a friend of mine recently sent me an email that I had sent in like in the late '80s, I should say, and it was really, "Wow. This is really a blast from the past," so I know what you mean. The other thing you can do is, I mean, I don't know if you posted to any newsgroups at the time?

Katz: Yeah.

Spicer: That might--

Katz: Yeah, I know.

Spicer: Those are still--

Katz: Yeah.

Spicer: Yeah.

Katz: The bulletin board systems and things like that. Yeah, it's really interesting, so, you know, it's so important to do things like we're doing now to capture some of the stuff because--

Spicer: Oh, yeah.

Katz: --at the time, there were things going on that were really momentous that we may have taken for granted and never really thought of it as exactly how revolutionary the capability actually was, you know, so...

To me that, sending that first electronic mail, you know, sort of falls into that category. Berkeley got onto the ARPANET a little late, you know, sort of in the 1977 kind of time frame through the LBNL [Lawrence Berkeley National Laboratory] connection, in part because of the growing visibility of the Berkeley Unix activity. You know, brought us back into good graces with ARPA, or DARPA, as it was, you know, DARPA or ARPA. Whatever it was called in the 1977. That kind of got us in that, that brotherhood or, you know, sort of companionship of the MITs and the Stanfords and the Carnegies of the world.

Spicer: Right. One thing I wanted to ask is we have three projects that you sent me that you thought were really kind of big moments in your career. Are we at the first one yet? Which I think--.

Katz: --a vary-- I'll be very brief. It's, again, the rise of local area networking. You know, there was so much that was happening in the context of the 1970s of microprocessor, MOS, you know, semiconductor technology, local area networks, Berkeley Unix, open sourcing, network stack in the software that was widely distributed essentially for free to industry as well as academia, you know, really fueled this incredible growth of the computer industry into the 1980s, I believe. The way in which Berkeley Unix was the foundation of a lot of the early workstation company software stacks. Of course, Bill Joy, Eric Schmidt at Sun Microsystems in the early '80s and, you know, just the whole evolution of that is just so interesting and important. I mentioned the database as a category of software big companies would be willing to pay big bucks for so that you could have software companies supplying that demand, as opposed to, you know, it's a few programmers working at IBM working on this sort of stuff. So it's just really fascinating to me, you know, that dynamic of how the industry was propelled forward by a set of fortuitous kind of technological innovations of microprocessors, semiconductors, open-source software, database package software and so on that really came out of, you know, I was like right there … all of that was happening in the '70s.

Spicer: Yes.

Katz: But yeah. I'm happy to sort of move on to my own research projects now.

Spicer: Okay. Great. So let me just pull up my email. The SPUR memory system. I think-- was that the earliest one?

Katz: Yes.

Spicer: SPUR, Snoop and RAID.

Katz: Yes.

Katz: It's actually in the-- in the order that these things happened, SPUR, RAID and Snoop. <laughs>

Spicer: Oh, okay. Yeah. Yeah.

Katz: Were the three research papers that I sent you. So going through my stuff...

Spicer: Yes.

Katz: I don't know. Can-- how-- I have to--

Spicer: Oh.

Katz: --move you a little bit here.

Spicer: Nice. Yeah, I can see those. Nice.

Katz: Okay. Sorry.

Spicer: Is that your own custom VLSI or...?

Katz: Yeah. Yeah, so these are the SPUR chips that...

Spicer: Oh, right. SPUR.

Katz: And, you know, done by MOSIS.

Spicer: Oh.

Katz: There is the-- this one here that I'm pointing to is the cache controller memory management chip. There's a floating-point coprocessor and the processor chip.

Spicer: Wow.

Katz: So in terms of my history, I started my work life at-- I went to work for Bolt, Beranek and Newman in Cambridge. I have to be honest with you, Dag, I didn't think I was cut out to be an academic, that I would be a successful professor. <laughs> So I went to, you know, what was a very sexy industry company at the time, BBN, and BBN, which, of course, was the architects and implementers of the first stage of the ARPANET. Was a very exciting company to go work for, but it turned out I got assigned to a project of building a commercial electronic mail system, and that's because they were trying to figure out, "Okay. We're doing good work working for the government on contracts, research contracts and developing contracts from the government. Can we go into make more money from, you know, providing this commercial software?" So it made sense for BBN to do this sort of thing, but I'll tell you, you know, developing a mail system in what was called RAT4. <laughs> I kept asking myself, "I got a PhD for this?"

Spicer: Wait a minute. I'm sorry. The audio cut out. Did you say it was in FORTRAN, the email assistant?

Katz: It was in rational FORTRAN. RAT4.

Spicer: RAT4. Oh, my gosh, and even--

Katz: Which was-- I liked the FORTRAN. That makes a little bit more structured.

Spicer: Okay. It just seems like FORTRAN would-- I'm not a, you know, programmer by any means, but FORTRAN seems kind of an odd match for the problem.

Katz: I, you know, think of all the data structures you have to build as array instead of you don't have pointers available to you. So I railed against a lot of the engineering decisions that were being made and made myself very unpopular very quickly.

Spicer: Oh, no. <laughs>

Katz: I have to say that my boss was Dave Walden, who sort of is really one of the heroes of the early ARPANET, but, you know, I think I maintained a good relationship with him over my career but I felt, you know, it was not a good match. So I went to work for a company called Computer Corporation of America in Cambridge and they developed an early database technology, again, for the government. Basically sensor databases. They built a kind of quasi-relational database system that was used as part of a nuclear testing sensor monitoring network.

So [it was an] important societal application of the time, and it allowed me to kind of do research and publish papers and things like that, but I knew I would go for an academic career and I applied for and got a job at the University of Wisconsin-Madison, which had a very strong database group led by Dave DeWitt, and [I] enjoyed launching my academic career there for a couple years. I actually, while there, transitioned into VLSI. Wisconsin sent me to Stanford for the summer to learn VLSI design. I had the chance to meet Lynn Conway and sort of get into that community. I had been exposed to that at my last year in graduate school at Berkeley. A lot of that kind of work was going on. Dave Patterson and Carlo Sequin were-- the CAD work of John Osterhout. A lot of that work was starting, you know, just as I was leaving Berkeley or just soon after I left Berkeley. I was interested in getting back into it and Wisconsin was gracious enough to send me to a training course at Stanford to do that so I could teach these courses at Wisconsin, and I became so successful at that that I got a job offer to return to Berkeley, which I took. <laughs> I'm sorry, Wisconsin.

So my first major project at Berkeley was a collaboration with Dave Patterson. He had just finished two, of course, the RISC projects had been completed a few years before. He had worked with colleagues on a project called SOAR, which was Smalltalk On A RISC. A different, more interpretive type-safe language developed at Xerox PARC called Smalltalk. So he built an architecture chip around supporting that, and he was ready for another project, and at the time, mid-1980s, the opportunity was, you know, through Dave's leadership we received funding from DARPA to develop sort of an alternative to the MIT LISP machine that would be a, you know, possible alternative platform for the kind of PROLOG inference-oriented computing that was becoming quite the rage of the fifth-generation Japanese project and all of that.

So Dave led development of the CPU architecture. It's in the line of evolution from RISC through SOAR to SPUR. It has many of the features of the RISC architecture, the simple instructions, the delayed branches. It had a little bit more instructions for type safety, but generally a very simple instruction set. Register window, concept, that kind of thing. Dave also worked with graduate students on the floating-point coprocessor. This, of course, is in the context of Professor Kahan at Berkeley developing the IEEE floating-point standard. So it was a risky approach for implementing a floating-point standard, which, of course, was of tremendous interest in the mid-1980s time frame. Those are really-- Dave was leading those, but the part that I carved out for myself and my students was the memory management piece. So we knew that there would be a third chip component to the chips that we were developing, a memory management chip, and also this was meant as a desktop type multiprocessor, so you would build kind of a multiprocessor system around the chipset. So there were two essential elements to what this memory

management unit would have to do. The first was to maintain coherent caches, and the second was to implement the memory translation from virtual to physical addresses, and so, you know, an MMU that did the latter function, was not so unusual in the time frame of 1985, but the support for coherent caches as part of a memory system, of a multiprocessor memory system, I think was one of the areas where we were really innovative.

Spicer: Great. And were there workstations or computers built from these chip sets?

Katz: Yes, yes, we did. We built a prototype so that we could run actual benchmarks on it. We didn't go into widescale production. This was not like a Berkeley answer to Sun workstations.

Spicer: Right. <laughs>

Katz: But we built a real hardware setup so that we could run real software on it. John Ousterhout was one of the faculty involved in the project. He ran a group of students develop an operating system called Sprite, which could run on single processor workstations but also had features that allowed it to be booted and to run successfully on our multiprocessor setup. So that was kind of a feature of the kinds of projects that-- for me, this was the first one of these but for Dave Patterson it was like the fourth one, of putting together university research projects which involve new architectural work of VLSI implementation.

In his earlier projects, John had supported Dave in developing new CAD software, the MAGIC system, helped develop the chips. In the SPUR project, this was the first one where John kind of reentered his life as a developer of operating systems. So Ousterhout and his students developed Sprite as a system software that would run on the Sprite multiprocessor workstation. There were the early ideas that would lead to a log-structured file system and other things of that nature that were coming out of the Sprite code base. So rather than work on CAD, John was working on operating systems for the architecture and hardware that we were developing as part of the SPUR system, and one of the features of the memory system was that I developed-- so the students who were involved in this were Susan Eggers, who became a professor at Washington. She did mainly work on the cache coherence protocol and analysis of that protocol, and David Wood, who became a professor at the University of Wisconsin, who worked on the implementation of the memory translation system, and Mark Hill also was involved in the project, more from the cache analysis, you know, how to size the caches in the right way to get the biggest bang for the buck, and the where this comes back to, you know, the collaboration with operating system, is in the memory management system we actually built a memory management system without a traditional cache of page table entries called a translation lookaside buffer.

Instead, the cache entries kind of lived in the same cache as instructions and data and we organized the system so, you know, leveraging-- rather than having very complex software for managing page tables, we built a system so 99-plus percentage of the time if you access a page table entry because you're missing in the page table, you will find it in the cache, and the cache was accessible because of the chip set in a single-processor cycle. So it's very fast. If you miss on a page table entry -- so it's the first time you're accessing a data structure or you're jumping to a part of the code you've never accessed before -- there's a very long and complex process to kind of walk a tree to find the right thing and you needed

support from the operating system to do that. So it was really in the philosophy of RISC to do the frequent things in hardware and simplify the hardware as a result of that, and when you need to do something complicated, do it in software. That was part of that entire philosophy behind the way in which we built the main-- the virtual memory system for SPUR. So it would've been impossible to do it without a close implementation of the page table management with the team that was building the operating system.

Spicer: Right. How long a project was this for the team members? Was it a one-year, a two-year? More?

Katz: These are generally-- so that's a great question, Dag. So generally, five-year projects.

And there's an entire history-- and again, this is something that's on my to-do list of writing down what I, you know, basically it's the Patterson model <laughs> of experimental systems. You know, in the first year or two, you're doing the design of the system. This is a personal approach for these experimental projects to evaluate existing systems to understand their bottlenecks. So there's a very big performance evaluation piece. Run benchmarks on existing hardware. Figure out how you could improve their performance or postulate how you could do it, and then figure out, you know, maybe you have a brainstorm. Maybe it leads you to an approach for doing things differently, like I just described the putting the page table entries in the cache, mapping it into the address space. So you come up with that idea and then you design a system to support that and that design, because the goal is to actually implement hardware, is going to take a year to design it, validate it, use the CAD software, submit it. Maybe it's another year to build it, test it, prototype it out. Maybe you discover from some of that, you know, sort of implementation you could've done it in a better way once you're starting to actually evaluate on the real stuff, and then you're kind of like in year four. You've got that second-generation system, you know, sort of enhanced and refined, working better. Now the PhD students can actually write up their PhDs and get them done in the fifth year while you're overlapping your thoughts on what the next project is going to be. So generally, these are, you know, the implementation phase might be a couple of years, but the evaluation and design phase leads to a couple years on the front end of that and the write it up and graduate is maybe a year on the end of that. If you follow kind of my five-year model.

Spicer: Yes. Exactly.

Katz: And so we did that for, you know, we did that for the SPUR project, we did that for the RAID project, we did that for a set of projects, the BARWAN project, which we'll talk about, you know, so that is a very common model that led to the great success that Berkeley had over the 1980s, '90s, 2000s, in terms of building these kinds of projects in a university environment.

Spicer: As far as the division of labor goes, like would one person be assigned the CAD, the hardware stuff, and someone-- you know, like, is there a division of labor or is everyone kind of doing everything on these projects?

Katz: You know, one of the things that also is quite unusual about SPUR, and we had the great benefit of being able to duplicate this kind of idea going forward, is conceive of a project ambitious enough that you

can't do it yourself. So it's not just design a processor chip. Dave Patterson could do that without any help. It's about, you know, building a workstation. So okay, Randy, I need your help because we're going to do a full hardware design on this thing all the way towards not just testing chips but a real workstation. So Randy and Dave are brought on board. You know, Dave brings Randy on board to help with the hardware. He brings John on board for the software, because if we're going to build a real workstation we need an operating system on this thing and we need a platform for developing applications. Because it was intended to build a LISP language system on top of it, we also brought in Paul Hilfinger and some other faculty [Richard Fateman] who were experts on LISP. We also brought in some faculty who-- Dave Hodges, who was a circuit designer, because we knew we could push on some of the underlying circuit design. There were some interesting things related to the timing architecture and so on in order to build things fast or to communicate chip to chip in a fast way. So we brought Dave Hodges in, who was a semiconductor circuit researcher, so that he and his students could contribute to the project in that way. So it wasn't kind of like a bunch of people, you know, choose what you want to work on. It was intentional in terms of bringing to bear interdisciplinary, you know, across disciplinary expertise, because of the ambition of the project. That we're going to build a chip set. It was going to talk to multiple chips. It would have to be fast and power efficient. It would be a RISC architecture. It would have a full memory system. We would run on it a real operating system. We would run on it a real language system. You know, all of those things together said, "Invite this person, invite this person, invite this faculty member," who would then bring in their students, and again, you know, you can't undertake these kind of projects without the level of research funding that an agency like DARPA could bring to bear, you know, so that our ambitions could be met with the resources to really tackle that ambition.

Spicer: Yeah. Now, that raises an interesting question. How do you match the funding to the project? And like do you-- is there enough flexibility there you can still kind of do what you want ...?

Katz: Yeah. So, you know, at the end of the day this was probably five or six faculty, each of whom had anywhere let's say order of four students working on this project with them. So we're talking about the level of funding for a team of couple dozen people. We hired also a full-time engineer to help with the day-to-day engineering of printed circuit boards and sort of-- who is still working at Berkeley, by the way. The guy is fantastic. Ken Lutz. And so how do we fund this thing? So at the time we had, in addition to the DARPA funding, I had NSF support through the Presidential Young Investigator program, so I had some unrestricted funding from NSF. A couple of other people working on it, John, for example, had some funding, that allow you to work on things that you don't have to submit a proposal for.

Which is kind of nice.

Spicer: Yeah.

Katz: And Dave Hodges, who was much more senior than any of us. You know, maybe had been a professor for 10, 15 years before we ever started. Even Dave. He introduced us to a model of how to have projects like this funded from industry. So industry really likes our students and, you know, at some level, understands that there's a competition for talent. That there's a set of things that a university can explore that are hard for a company to explore and certainly over my career became more and more

difficult. IBM Research and Bell Labs research in 1980s were pretty open and could do what they wanted. By the 1990s and beyond, these research labs were more, you know, had to show value for commercial products that were coming out.

But even in the time frame of the SPUR project, which was, you know, let's say started in that design and thinking phase in 1984 and went all the way to 1989, these industry connections were kind of interested in working with the university to explore some things that maybe were off their main path. So we had support from companies like Intel, and even though it was a startup company-- well, it was probably public by this time, Sun, Digital. So we mined the primary sort of computer companies of the time to form a consortium to support us, and the way that they looked at it it was a relatively modest investment on their part. You know, a fraction of an engineer, in order to be able to have a seat at the table to see what an interesting research group at a university is doing. To learn through the model that we had, which were, you know, twice yearly research retreats where we met with our industrial sponsors, invited our ARPA people, DARPA people, to come too, and sometimes they came. You know, to review where we are and what our current ideas are, and it was a two-way street because the industry people would say, you know, "We know how to do that. Don't work on that."

Spicer: <laughs>

Katz: "That other thing, well, we don't know. You know, that's something we haven't really spent time on. That's a good thing for you guys to work on."

Spicer: Wow.

Katz: So it wasn't-- it was a very dynamic kind of thing where it wasn't, "Okay, Berkeley, this is a contract to work on. Thanks." We went to them and said, "We're doing SPUR. We're going to build… you know, this is the set of things we're working on. This should be interesting to you. Are you willing to join a consortium to watch what we're doing, give us some feedback, get to know our students, have our students do internships with you?" You know, kind of learn from what we're doing, but also give us feedback, and the cost is half the cost of an engineer. Maybe a third of the cost of, a quarter of a cost of an engineer, and surprisingly a lot of companies said "Yes," and also, at the time, there was a program that the state of California had called the University of California MICRO (Microelectronics Innovation and Computer Research Opportunities) Program, which also we got some state matching of industrial funds.

So I guess the end of the story of your very insightful question is we had core funding from DARPA. It was essential to have commitment of a significant amount of funding to launch a project like this but it wasn't sufficient. Wasn't, you know, it was necessary but not sufficient. We also worked with industry to provide flexible funding, and from that we were able to get some matching funding from the state of California, which had kind of an industrial policy. A little bit of a, "If university works with industry we'll give you some-- we'll match some money so that you can do even more," and so we put together a consortium and a portfolio of funding sources that made it possible to tackle these ambitious projects, and SPUR was the first one based on the success really that Dave had with the RISC projects and the interest that those generated, and from that, once you start the ball rolling, the next project, the next project, the next project,

once you have a tradition of success in doing projects like this, it's not that hard to get companies to sign up. So we'll be able to talk about that.

Spicer: You got a track record. You know, one thing you said that really struck me, which was the companies telling you what not to study. That is so important to, you know, especially to a grad student, <laughs> right, who's about to spend--

Katz: This is--

Spicer: --the next five years on a project, and yeah.

Katz: To be honest with you, Dag, that was kind of like the superpower of this approach. It's, again, I want to emphasize, it's not a contract, "Berkeley, analyze system X and produce a report." It's kind of-- industry has a seat at the table and we want to guide you towards the real nugget problems that we don't know how to solve, or that we think are interesting but in our need to get products out the door we don't have the time to explore those things, and that is gold for graduate students working in this kind of environment, which is at the-- this is not theoretical research. This is computer engineering research. This is industrially relevant research. If you want to be relevant, I mean, this is, again, the Dave Patterson mantra of, you know, people want to kick the tires. You have-- and today in software you can do this in a way that was harder in 1985 to influence the software field but, you know, the way Mike Stonebraker and Larry Rowe and Gene Wong did it was to start a company. You know, now you can do things by open sourcing your software and getting people to use it, like, you know, what Databricks and, you know, the Berkeley A.I. software has done more recently. In hardware, we were really building chip sets and putting them at least in prototypes so people could see software running on them and that would be the way to open their eyes and get them to believe the simulation and the analytical results that we had sort of developed as part of our research papers.

Spicer: Yeah. I like that you also mentioned some of these companies probably feel a little frustrated with the emphasis on this having to be product-oriented research. There can't be any, you know, more foundational research, for example-- foundational with a purpose, not completely theoretical. But so maybe this gives them a chance to kind of get around the corporate structure of not having a lab of their own -- does that make sense?

Katz: Yeah, and-- absolutely. And again, you know, in the time frame of 1985, IBM Research Lab and Bell Labs were still very powerful, independent research laboratories. Xerox PARC. We worked with Xerox on this because of Dave's Smalltalk connection and consulting connections with Xerox. You know, the research labs, industrial research labs, were still very powerful entities, but 10 years later it was totally different. But still, you know, the idea of meeting together in a consortium base is precompetitive. You know, the companies also learned a lot from the other companies by carefully observing the sort of questions and feedback that they were sort of asking and giving, and on the neutral ground of the university project, companies could, you know, engineers at companies can talk to engineers at other companies in a way that, you know, maybe even at a, you know, in an industrial setting would've been much more difficult.

Spicer: Oh, yeah.

Katz: So they were-- and this became a much bigger deal in our RAID project, which we'll talk about next, than maybe in the architecture space, because in 1985-ish, RISC architectures were still very much viewed by the Intels of the world as an oddity, right. They were more interested in, you know, "Is there something interesting going here in the memory system or the--," you know, "--John's operating system that we could learn from?" They were still I think a little poo-pooing RISC as a concept in 1985. But they were, you know, they were interested in how you better support languages. That had been a thread of development at Intel with some of their architectures, you know, sort of in the preceding years, so there was interest but they didn't view it as, you know, "This is something we absolutely need to be involved in, otherwise we'll lose our competitive edge," until some of our later projects.

Spicer: I might have this wrong, but was it AMD who did a RISC implementation of the x86 architecture that ultimately got Intel to take RISC--

Katz: No. I don't want to lie by saying I think that's so, because I don't quite remember.

Spicer: Yeah. I don't remember either, but it's-- anyway, we'll cut that out. <laughs>

Katz: It is very possible. You know, some of Dave's students, they did, so there was a startup company, I've forgotten what it was called, where I think that that was kind of what was the essence of the company in the time frame of that, you know, sort of 1980 [Actually, late 1990s according to https://medium.com/@madmedic11671/the-transmeta-story-f0d8169e24b1. The company was called Transmeta, and Dave's RISC students David Ditzel was one the founders; Also NexGen, where Robert Yung, who was I think Al Despain's student, also worked on a RISC-based x86 implementation]. We were building a x86 compatible architecture and, you know--

Spicer: That wasn't Transmeta, was it? No.

Katz: That's it.

Spicer: Oh, was it? Okay.

Katz: It was-- I believe it was Transmeta.

Spicer: Right.

Katz: You know, so some of Dave's students were involved in that.

Spicer: Right. Linus Torvalds worked for them for a little while too, I think.

Katz: That's-- and I believe that's true too. Yeah.

Spicer: Yeah, yeah. Anyway, sorry. Let's move on. Do the--

Katz: Let's move on to RAID. RAID.

Spicer: Yeah.

Katz: So I think, you know, if SPUR, you know, one of the big contributions from SPUR, I know we talked a lot about the memory management system. It was virtually addressed caches and it had a lot of interesting features, but I'm not sure that it particularly had long-legged influence on industry. The cache, the sort of coherent cache algorithm as implemented in SPUR, at least it introduced the term snooping caches.

Spicer: Oh.

Katz: The idea was you would build a cache controller that would watch the bus traffic and use information from, you know, the interaction between other processors and memory because it was a shared bus memory system, in order to ensure that the caches were coherent. We had something called the Berkeley ownership protocol. Other researchers developed variations on their own, but pretty much every multiprocessor, you know, in your laptop, it's probably a 32-core processor if you bought it recently, uses some variation of this kind of technology. So that did have sort of a big influence at the end of the day in terms of actually being in real hardware even in 2022. So let's move on to RAID. So I mentioned that there was a lot <laughs> of exciting things happening in the computer industry at this time, so I and many people like myself became, you know, really excited with the introduction of the Macintosh. I wasn't interested in the Apple II and I never really got into the IBM PC, but the Mac, you know, that integrated system, was so compelling, and the integration of the software for word processing and spreadsheets and drawing and all that other stuff. I needed to get a Mac.

Spicer: <laughs>

Katz: So I bought, you know, I bought a Mac-- I think my first Mac was the Mac 512, and very soon bought a Mac Plus, and, you know, when that came out, and I can't quite remember which version of the Macintosh had it, but somewhere around 1984 or '85, the Mac of the time, I think it was the Mac Plus, had the SCSI, Small Computer System Interface connection on it, where you could buy a 10-megatbyte disk drive for the price of the computer, by the way. It was about $3,000.

Spicer: Yeah. <laughs>

Katz: Oh, my God. You don't need floppy disks anymore. This is like a real-- finally. It's a real computer once you could buy magnetic storage for it. So this is the time frame of the mid-1980s. You could do that. It was an expensive-- you know, we're talking about a $6,000 computer in 1985 money. You could do that with the Macintosh and it's something that even professors wanted to do and needed to do. If I roll the timeline back a little bit, in-- I mentioned that I left Wisconsin to go and take a faculty position at Berkeley. That was in the fall of 1983, and it turned out that Dave DeWitt, who was a professor at

Wisconsin, also planned and did do a sabbatical at Berkeley from fall '83, spring of 1984. So he roped me into conversations of Dave DeWitt and Mike Stonebraker on what was considered at the time a very hot topic in the database community, which was so-called main memory databases, and you'll see what this has to do with <laughs> RAID in a second.

So a main memory database, was the technology trend of semiconductor memories, declining cost per bit of memories, led to the ability to build bigger and bigger memories. You know, you could see that on the horizon it'd be possible to attach to a computer, 32-bit architectures were coming online. It's amazing from the viewpoint of today that, circa 1980 a 16-bit address architecture was huge, but by the '80s their 32-bit architectures, 4 gigabytes, you could have a gigabyte of memory on the machine. It was believable that in the not too distant future-- you may not have been able to afford it quite yet, but you could see where things were going, so it was conceivable to put the database, you know, interesting databases directly in memory. How does that change the way you think about databases?

There was a brainstorming seminar that we had at Berkeley in fall of 1983, and one of the parts that I latched on to was, well, semiconductor memory, you trip over the power cord, you lose <laughs> the data that's recorded in memory. You still have to make this stuff persistent. You still have to at the end of the day write it to something which is magnetic. So you need to write it to disk. What is the fundamental performance limit of the number of transactions you can execute per second? Because Dave DeWitt was a big, you know, this was Jim Gray's transaction processing benchmark, so Dave was also very interested in those benchmarks and, you know, you want how many debit, credit transactions can you do per second in a main memory database? You still have to make those transactions persistent. You still have to write things to disk. The disk becomes the bottleneck. How can you, in a very clever fashion, reduce that bottleneck? Well, at the end of the day if you have a transaction log, if you keep track of what you changed in memory, now it's not all of the data but you can kind of organize it as a log and sequentially write that log out every once in a while and the number of those log writes are actually the bottleneck.

Spicer: I see.

Katz: So we wrote a paper based on this. It was really, you know, interesting work, very exciting work, and I kind of left that to work on the SPUR project, and I mentioned in the mid-1980s, you know, now these sort of shoebox size disks are becoming widely available. So in the time frame of 1986, we're thinking about what comes after SPUR? So 1984 is the initial ideas, 1985, '86 is the intensive design and the building of the thing. Late 1986 we're starting to think about actually what the PhD projects are going to be on top of this thing, '87, '88, '89, the students are actually, you know, build the hardware and are evaluating it so that they can write up their dissertations and get jobs. So in 1986, we're thinking, "What comes after SPUR?" You know, the initial ideas associated with this, and, you know, I'm working with Dave and we're brainstorming and we say, "You know, one thing that hasn't really been looked at is I/O. People worry so much about, first of all, process of performance that was related to RISC, and the memory system bottleneck, and that, you know, was kind of-- we had taken a crack at that with the SPUR project. Why don't we start thinking about I/O?" And, you know, I kind of, in the context of what to work on next I brought up, "You know, you can buy a disk drive for like $3,000 now," and at the time an IBM

3380 was the mainframe disk drive of the era. This was the hundred thousand dollar washing machine sized appliance, right?

Spicer: Oh, my gosh, yeah.

Katz: And IBM, of course, would charge you-- not that we knew the real pricing of these things, but we guesstimated a hundred thousand-- you know, somewhere in the order of a hundreds of thousands of dollars. So you could replace that thing with 20 or 30 of this-- this guy had a couple years before or year before bought for my Macintosh. What is that? How does that change the application?

Spicer: Ah, yeah.

Katz: So now you have the ability of what you've just spent a-- let's say you spend a hundred thousand dollars, you buy an IBM mainframe disk, or you spend a hundred thousand dollars and you buy 20 five-and-a-quarter-inch essentially PC disks, which were really, you know, it turned out, the market for that was mainly engineering workstations, so the Suns and Mentor Graphics and companies of that era were the market for those five-and-a-quarter-inch disks, but for Randy I was able to get it for my Macintosh. So if you stop and think about it for a second, gee, that IBM disk can do something like 50 I/O's per second because of the mechanical nature of how the actuators work and so forth, and now I've replaced it with 20 of these little disks, which, you know, in theory, can work independently of each other, so you could do 20 times as many I/O's per second as one gigantic disk, and, you know, we did the calculations and you could get comparable capacity for about the same price.

So my God, if you sort of thought about replacing disks with what amount to arrays of disks, what a great idea? And, you know, who, you know, rained on our parade, but somebody said, "Well, wait a--" you know, we're broad in terms of presenting this idea to anyone who'll listen to us, right?

Spicer: Mm, yeah.

Katz: At some point in the brainstorming, somebody, whether it's Randy or Dave or one of the grad students-- you know, maybe was Garth Gibson. You know, disks break, and now you got 20 disks that are more likely to break. You know, the probability that one of them is broken at any time is 20 times what one disk might be, and so that's where the "R" in RAID comes from, is, "Oops. When you replace sort of the system of one level of complexity or componentry with a system of a much larger number of cheaper components, you have to consider the reliability of this system constructed from the cheaper components." It's kind of like, you know, the old joke-- I'm not sure who gets the attribution for it at the beginning-- but it's the astronaut is sitting on top of the rocket that's going to send him into space and he jokes about the number of parts in that rocket, all of which have to work, and of course, they were built by the lowest bidder.

Spicer: That's right.

<laughter>

Katz: You know, so just stop and think about that for a second while you're sitting on top of this explosive device.

Spicer: Yes.

Katz: Same thing is true about storage systems in terms of we're proposing replacing a highly engineered IBM single disk drive storage system. There's many of them in a real system, but, you know, replacing the building block with 20 little disk drives, what is the reliability implications of that? And let me tell you, between Garth Gibson, who was the sort of first graduate student on the program, on the project, and Dave and I, we did a deep dive on reliability engineering. You know, we had to learn the mathematics behind how to do that, and, you know, what the calculations were and all of those things, and of course, the strategy for doing it is to introduce some level of redundancy. It's the only way to improve the reliability. So how did we go, you know, from that realization to how this is going to affect the kind of system we wanted to build? So Dave had just come back, you know, somewhere in that 1985 time frame. I forgot whether he had done an industrial leave or sabbatical but he went to work for-- he went and spent time at Thinking Machines in Cambridge.

Spicer: Whoo. Wow.

Katz: And, you know, that's another lesson. Lot of academics, when they take a sabbatical or take a break, they go to another university. Dave pioneered the strategy of go to industry. He had been at, you know, to-- the thing that led to the RISC work was a sabbatical at DEC. He had gone to Thinking Machines, and Thinking Machines, at the time he was there, had really a sort of bit-parallel model. It was more like a SI/SD type architecture that they had in the first generation of a Connection machine.

Spicer: And they--

Katz: And they [Thinking Machines] evolved towards their second generation [aka Thinking Machines 2], which was more [based on standard] microprocessor nodes. So they were building a storage system that was, you know, think of it as 32 drives with, you know, it almost looked like a set of memory chips. One bit of each word was put on each one of those 32 drives, so it was kind of, you know, across a row, and how did you introduce reliability into memory systems at the time? You added an error correct, a detecting and correcting code, some kind of Reed-Solomon type coding, which was another 7 or 8 bits for 32 that would tell you the position and what the bit should be.

So you needed a fairly complex coding schemeto do that but, for people who were willing to pay for it, that was the sort of standard way that memory was protected by redundancy. Now, a common way that computing systems had introduced redundancy into them, you know, up until this time, was to do what was called mirror disks. So keep two copies of the disk. If one fails, you got another copy, and of course, when you stop and think about something like that, you have the challenge of every time I write I have to also update the other one and make sure the update took place and deal with all, "What happens if this one worked but this one didn't work?" You know, so a lot of complexity but you can solve those, you know, and this was part of the secret of Tandem in its early days as a reliable distributed system. They

kind of solved the mirror disk kind of approach and the overall approach to redundancy of hardware and software systems to ensure a nonstop type system. But the problem with that approach is for certain applications, airline reservations, banking, real-time banking and so on, you're willing to pay whatever it costs to do that, it's not particularly economic. So we had the model of duplicate the storage and always write twice, or the approach that Dave had been exposed to, which is treat the disk drives like they were semiconductor memories, bit interleaving with an error correcting code, and then, you know, it just kind of came to us of if you're going to go down that road of error correcting codes, hold on a second. There's a lot simpler thing that memory designers have done in the past, which is to put parity on memory, so at least you know one of the bits is messed up because it doesn't match the parity. So for the cost of one additional bit position and putting, let's say, even parities so that all the ones across those 32-bit positions plus the parity position are always an even amount at once, if you ever calculate across and see an odd one, you know that one of the memory chips has been blasted by a cosmic ray and a one became a zero or something. So this was a common technique that was used. It had not been, to our knowledge, used by disk drives before so you could do that, and wait a second, because again, I think I should credit Garth Gibson on this because he was doing a deeper dive on the reliability research and calculations. There's, you know, parity is an error detecting code. It will tell you, at the cost of one additional bit on a 32-bit word, that a bit is wrong, but it won't tell you which bit is wrong. So it's not a correcting code, it's a detecting code. If you want detection and correction, that's way more expensive. That's what the Thinking Machines guys do. But when you're working with disk drives you have another feature. When a disk drive fails, it has its own error correction mojo on the disk drive because of the inherent complexity of reading magnetic fields off of a spinning disk, when in this array structure the disk drive doesn't respond, it's because the disk drive has a hard crash, or it really has tried and it can't figure out how to respond, and we're thinking about SCSI, Small Computer System Interface disks, which talk a high-level protocol. They tell you when they have an error. Memory chips don't work like that. You read them, you get a one or you get a zero. You don't know if it was the right one, and so you could use additional information. You know the disk drive has failed and now you can make use of so-called erasure codes. You know the position and you know the bit, and with parity you could tell what the bit is that should've done there, and that's the magic that led to first the RAID level four, which was parity protected, you know, storage array, and a little further on we figured you could do more simultaneous writes if you interloop the parity, and, you know, the rest is history.

Spicer: Yeah, that's really cool because there was-- I don't know if it was the PC but there was an IBM design where they decided to forgo adding a parity chip because the loss of reliability of adding that extra chip actually outweighed the benefit that they would achieve -- I mean, when you say you had to master all these reliability calculations, it must've been very difficult to process these variables.

Katz: Well, also, if you're IBM you know what your failure rates actually are for your units in the field, and that's not something that was generally published, although there was a point in time later, you know, not when we were at the early stages of RAID, but later Jim Gray published a paper about, you know, when he had left IBM and was now working at Tandem. He published a very influential paper which described why distributed systems fail or something like that, which had data about the nature of failures.

Spicer: Oh, yeah. Saw that.

Katz: But we didn't know how frequently disk drives fail. We, you know, you could kind of guess or you could sort of extrapolate from manufacturing what they claim their mean time to failure is and stuff like that, but a lot of it is guesstimation. But I'm not surprised. You know, there had been a project at IBM at the Palo Alto science lab which really went ahead and had pioneered some of the sort of parity protection ideas in a storage system that we weren't aware of because it was-- it hadn't been published in the literature. It was more like technical disclosure bulletins which academics typically don't read, and not to take anything away from the people who did this early work, it was a fellow named [Norman Ken] Ouchi at IBM who was mostly associated with it, and basically the work they did is in the generational shift from removable storage, from the 2311 to the fixed media IBM 3000 series disk drives. [See his Oral History at https://www.computerhistory.org/collections/catalog/102740119.]

Spicer: Eleven? Yeah.

Katz: 2311 storage devices that IBM had to-- the 3000 Series, 3300 Series, where it would be a fixed magnetic media that you would not be replaceable. IBM was concerned, "What about disk errors?" So is there a way to build more reliability in beyond the on-track or on-cylinder coding that they already had? So they looked at sort of parallel parity ideas in the context of that and it turns out that the 3380s were more reliable even though, you know, they couldn't be replaced, and that they wound up being more reliable because it was in a sealed environment and they didn't need to achieve their engineering design goals, the kind of cross-device parity support to achieve their reliability. So it never got implemented in real systems, but some of the ideas were buried in technical disclosure bulletins from IBM.

Spicer: Yes. Oh, that's really interesting, and the notion of ACID when it comes to databases, you know, like-- I forget what they stand for, but they're the four cardinal principals.

Katz: Atomicity, concurrency… I'm forgetting two. <laughs> [Atomicity, Consistency, Isolation, Durability]

Spicer: I had no idea RAID was actually inspired by databases, by work on databases, so that's very interesting, and would this… how did the RAID technology interact with the database software for things like rolling a transaction back, for example, or was it pretty easy to, I mean, was there somewhere an operating, a higher level of software, a sort of hierarchical storage management? I don't know if that's what you would call it, but some kind of storage management layer would have to be created; is that correct?

Katz: Yeah. That's very interesting that you ask that question. So in the time frame of 1986, when we're thinking about doing our next project, we actually collaborated with Mike Stonebraker on a proposal that we submitted to the National Science Foundation for a project called XPRS X-P-R-S, and the theme of the proposal was actually, "Could we use SPUR as a database machine?" and the reason this was of interest to a database person like Mike Stonebraker was there was a lot of, you know, after main memory databases there were lot of discussions in the time frame of the 1980s about hardware support for databases, database machines, and there was a bit of a controversy that was going on in the database community about so-called shared-nothing architectures, which was think of a processor and a disk on a local area network talking to another processor and a disk and building a distributed database system

using that architecture. So it's mainly communication and messaging and you have to deal with consistent updates and all the ACID things that you're talking about in a distributed system environment compared to the opportunity that an architecture like SPUR provided, which was a multiprocessor and a coherent sort of view of memory so that if a transaction running on processor A updates something in memory and processor B updates it, we can ensure that there's no lost updates. That, you know, there is a logic behind how it happened because we have this abstraction of a coherent and consistent memory. So we wrote a proposal about that to work with Mike on shared-nothing versus a traditional multiprocessor would be a shared-everything. Memory is shared, disk drives are shared, and that this would also launch us into thinking about architectural support for I/O, but I have to be honest with you. By 1987, we got so in love with the multiple disks, the array of disks, that the sort of main memory piece that had been the heart of the original NSF proposal, I mean, Mike Stonebraker used it to build, you know, I'm not-- he would know better than I, but this was the era of building Postgres and taking the Ingres kind of approach to a distributed system. So, you know, he used his funding from the proposal to sort of move his database research into the Postgres that was distributed database world while we used it to explore many disk drive type storage systems and we really never circled back to really make the database run on top of the storage array when we built it, and we went ahead and got additional significant funding from DARPA for RAID to-- as part of the High Performance Computing and Communications program. So we wound up working very closely with NASA Ames to use RAID technology in more of a scientific computing environment. So I'm not sure what Mike Stonebraker would <laughs> tell you about. Did we abandon him, or, you know, we opportuni-- we sort of followed our noses in terms of where we saw the research opportunities, and the funding kind of followed it but it went on a slightly different path than what our original ideas had been on that.

Spicer: Right. Can you speak a bit about RAID once it was released in the wild? I mean, it's had an incredible influence on computing.

Katz: So even from the very beginning. So in the time frame of the summer of 1987, so we had this proposal in '86 and had been brainstorming in '86 an into the spring of '87, and, you know, I went back and I have-- I got to show this to you.

So I was able to put together some of the timelines of when we had specific conversations about specific things related to RAID, but in the time frame of the spring-- and it's interesting when I go back and look at things, we make so much progress as faculty researchers in the summertime and things are so slow <laughs> during the academic year.

<laughter>

Katz: That, you know, conversations happen, but real work happens in the summer.

Spicer: Interesting.

Katz: So in the fall-- the fall of '86 and spring of '87, we had some, you know, we're collecting ideas, we're figuring stuff out, we have, you know, little meetings once a week, but it's not until the summer of

’87 that we really do a deep dive on all these parity tricks and do the analysis that says, “This is potentially a better way to build a storage system,” and in the fall of ’87 we write the paper, which, again, part of the Dave Patterson model is don’t hide your research. Get feedback early. Send it to everybody you know in industry. Send it to Jim Gray. Send it to these guys at Control Data. Send it to these guys at Thinking Machines that Dave knows and get their feedback on this. Jim Gray thinks it’s a stupid idea. You should do mirrors. You know, the guys at Thinking Machines think it’s a stupid idea, but you get feedback, right?

Spicer: Mm-hm.

Katz: But it turns out that some of the people we sent this sort of October tech report to are storage people at the disk drive companies and things like that, and they start phoning us, and even before the end of the year they want to find out more about this project that we’re doing and what we’re learning about arrays because they had all been seeing the trend line from, you know, IBM mainframes are 14-- big, 14-inch disks. Disk drives on our, you know, minicomputers from DEC are 8-inch. Now we have on our Macintoshes and workstations five and a quarter. IBM announced a three and a half inch. You know, even during the lifetime of that project they were-- the 1-inch, you know, proposed disk drives were being talked about, or 1.3-inch, something like that. So they see arrays coming and they want to understand what we’re talking about. So building that industrial consortium that I talked about before for SPUR was easy-peasy for RAID, because it was kind of these guys are really doing the sort of analysis that we don’t have time to do ourselves that we’re kind of think, you know, “This is a business opportunity for us, so we really need to understand what these guys are doing.” And so that consortium was IBM and Control Data and Sun and Storage Tech and, you know, Seagate and, you know, just a bunch of companies like that.

Spicer: Were any of them thinking along those lines or was this completely new to them?

Katz: No, it was not new. So, you know, to give industry its credit, they’re very much aware of the introduction of small form factor disk drives as products coming out of, you know, Seagate is interested in, “We know how to make these smaller things. We know we’re going to go to PCs. Is there a bigger market?” you know, and they’re kind of-- some people are starting to talk about building for whether it’s workstations or media stations or minicomputers or main-- you know, mainframes it’s little harder to get in there, and even Storage Tech is kind of like, you know, “We need a competitive edge against IBM in the mainframe.” Storage market is, “Do these guys ha--” you know, we’re kind of thinking about these sort of ideas. They’re pretty way out for us, but these guys seem like they’ve really been thinking about it. We need to talk to them,” and so yes, industry was aware of this. One of my favorite stories is we learn from David Hiltz, who’s one of the cofounders of NetApp, that really, they read the RAID paper while they were, you know, it’s basically a files-- you know, sort of a network-detached file server or block server for workstations. At the first-- the first, you know, major piece of hardware they should implement is a RAID controller, and in fact, it was the first board that they implemented as--

Spicer: --remember those IBM microdrives? The little--

Katz: Yes.

Spicer: Did anyone ever make a little ray-to-ray off... That would've been cute.

Katz: No. We knew about it and we had really a strong collaboration with a team at IBM. You know, I get from the history of the industry, interesting things are happening here, so IBM Research is moving from San Jose to Almaden at the time, it turned out that the IBM Storage Group had been at San Jose. It would be very soon sold to Hitachi. Some of the researchers, fellow named Mike Mitoma went to-- went from Research to go work for the storage part of IBM, which was also at the San Jose site. So he knew about our work from the research side and came and talked to us about the possibility of a strong collaboration between his now storage group. You know, we would be like the advanced research piece of this effort that was underway at IBM to sort of prototype an array, and the technology that they made available to us that went into our second rate controller which is sitting at the Computer History Museum, was early access to those IBM 3 1/2-inch disk drives.

Spicer: I see.

Katz: So that the RAID II prototype was built out of 144 3.5-inch disk drives. No one had built a system with more than a dozen, you know, like a dozen disk drives hanging off of a controller. We built one with two, you know, 12 dozen, <laughs> and we could've never done it without getting that technology from IBM. So maybe somebody also did something like that with the microdrives. We had moved on when they became more widely available.

But I feel like we kind of prototyped that concept of building a large-scale disk-based storage system under the control of a single controller in that time frame of 1988, with the help of IBM. It was crucial. You know, they were part of our consortium. We had separate, you know, one-on-one meetings with them. They were even gracious enough to kind of share with us some of the challenges that they were seeing in their design and they wanted to learn from ours, but we told them that we wouldn't, you know, our model was not to work exclusively with any [one] company. I think of that has … you talked about the dissemination of the concept. So this goes back even to the Berkeley tradition with the SPICE circuit simulation software developed at Berkeley by our EE colleagues, that there was this strong ethos that put its way through Berkeley Unix and Ingres into RISC and RAID, of, "Work broadly with industry." Don't, you know, the model is not to keep your intellectual property close and get it out through intellectual property by forming a company, although Ingres did that. You know, that we want to distribute our software, distribute our ideas, make our ideas widely available, and there are lots of different ways of doing that. Distributing software is one. This consortium model is a different one. Because you could read our papers, but the papers are often a year behind what your current thinking is and the ability of industry, you know, both industry and the university team, I cannot minimize the value of this, you've kind of-- you've hit the nail on the head of that ability of industry-- this is what we're working on-- and industry telling you, you know, "That's a problem we know how to solve." <laughs>

Spicer: Yeah. Exactly.

Katz: "That other thing is not. Work on that." It's so powerful to maintain relevance and to maintain impact of the work you're doing when your research is not theoretical or you can make a claim, you know,

I'm working on quantum computing. It's 20 years from practice. We're working on stuff that's just a few years ahead of industry, which can be dangerous as a researcher. You could be scooped by industry, but it also has the benefit of you can really influence industry if you get it right when you're successful in doing that, I think... You know, some of the work on the coherent caches had an impact on industry for my work and RAID is the-- it's a stellar example of how the work that we're doing at the university can lead by just a little bit what industry is doing but then industry, you know, dives in and can really benefit from it. But no single company-- you know, they're going to compete on who can do it best, not on the simple idea.

Spicer: Now, Garth Gibson won an ACM Distinguished Dissertation Award for his work on that…

Katz: Yes.

Spicer: Just thought I'd mention that. What I'm curious about, just one last question about RAID, was could you explain RAID's place in the world today? Like do all data centers use RAID now or do high-performance computing centers? You know, what's the-- where does RAID appear in today's landscape?

Katz: So it's interesting because the concept has gone through some cycles. So first of all, let me say that RAID, while we concentrate on the parity idea and it sounds like, you know... Implementing the software for the controller, which has to, first of all, do the parity calculations, but also ensure that the disk drives get updated in a consistent fashion. It figures out how to schedule and run the rebuilding that's necessary a disk drive really does fail and you have to reconstitute what its value was on a new hot spare, there's a lot of complexity. John Ousterhout, again, was a collaborator on RAID and really built the log-structured file system that… God. What was his student's-- his student, the faculty member at Stanford. There was another ACM Dissertation Award for the log-structured file system. Mendel Rosenblum. I haven't completely lost my memory. Mendel Rosenblum won it for developing the log-structured file system which was intimately integrated with the RAID system because writing things as a log, you know, writing them sequentially, eliminates a lot of the update overhead and makes the performance of the disk array so much better. You know, the inherent "I got to update the data and update that parity." Now I just log write everything. It helps a lot of things when you can do that, and so having the hardware and architecture and software system integrated to come up with the best possible solution is important. This means that in industry in adopting these things, developing the software for the controller was no mean feat.

So a lot of these products out of companies like IBM and StorageTek took a very long time before they got it right. Sun was, you know, defunct now, but their RAID controller had a lot of reliability issues, not in the basic hardware but in the software for managing the RAID. It's hard to get this stuff right. So, you know, I think there is a lesson in all of that, but again, you know, now RAID, software RAID, is, you know, kind of a standard feature of Linux systems. You don't need any special hardware, you could organize your disk drives as a virtual array, and all the software is there in Linux to do that in probably in just about every operating system. There was a point in time when Google data centers used the approach of, "Let's take that mirroring idea and take it to the extreme. Let's have seven, eight, nine copies of files."

Spicer: <laughs>

Katz: You know, geographically distributed everywhere, so who needs RAID?

Spicer: Wow.

Katz: Because if this one disk fails there's six or seven copies somewhere else in the cloud. You know, as this amount of storage has grown and that kind of strategy even for Google is, you know, needs to be moderated. I know that for a fact it may be more sophisticated coding than just the standard parity, but the storage systems that Facebook implemented, because one of my colleagues in EE has collaborated with them on subsequent coding schemes, they da basically error correcting and detecting coding on the copies of our data, our Facebook data, for example. I think there's been some published papers about this as well. So RAID is definitely out there for mainframes and it's-- on the low end, you know, running Linux on your desktop so you could turn on a software RAID if you're worried about any loss of data on your local site. It's been-- at one point it sort of lost some interest in cloud storage but has been reintroduced. As we get to the point where the amount of storage and the amount of data moving around the cloud is getting you into those number of bits accessed that start to exceed the inherent error correction capabilities of a single disk drive. You know, it's one error in how many gazillion billion but the number of bits <laughs> floating around in the cloud, you're going to get a hard failure at some point, and, you know-- sorry. A soft failure where the disk drive exceeds its ability to tell you that the data it's delivering is an error. If the disk drive fails that's one thing. It's a-- what you're really worried about is Randy's bank account at bit [position three] got flipped and instead of having a hundred thousand dollars in his IRA, it says he has zero dollars in his IRA, or something like, right.

Spicer: <laughs> Yeah.

Katz: And so, you know, RAID has become more prevalent again in, you know, for large-scale storage systems.

Spicer: Right. Wow, wonderful.

Katz: For many decades, yeah.

Again, I have to say, as you would imagine, it's a lot more sophisticated and lots more bells and whistles, a lot more real-world considerations than what we had to deal with in the 1980s, as you could imagine.

Spicer: By the way, I'm curious if there were any standards or qualifications that people had to meet? Perhaps [meeting] performance standards in order to call themselves RAID? Did it have to be compliant to some standard or was it still, is it still kind of the Wild West, compnies have different-- I know there's different levels of RAID, but their implementation is probably left up to the vendor, right, other...?

Katz: Yeah. So, you know, it is very interesting as a university researcher to observe this. Self-organizing industrial consortia, I think it was called-- I don't know if it still exists-- but I believe at one point in time it

was called the RAID Forum. So industry groups sort of form in order to standardize on the terminology so that customers can tell, you know--

Spicer: Got it.

Katz: --you know, what's marketing and what's reality kind of, at least in terms of what's claimed in terms of the naming of the [RAID] levels and things like that, how to benchmark them, how to compare their performance.

Spicer: Exactly.

Katz: All of these things kind of developed completely independent of our group at Berkeley. I think Garth Gibson, when he was at Carnegie, may have been involved them, but no, at Berkeley we moved on to the next project, so, you know… that's kind of an interesting aspect about, again, I'll call it the Dave Patterson approach to these system projects… we never did the same thing again and again maybe with a few bells and whistles. You know, it's kind of RISC architecture and then memory architecture and then storage architecture and hopefully we have time to talk about my BARWAN project and the Snoop TCP protocol. So Randy moves into wireless networking. You know, it's constantly evolving. It's not like trying to own the world of RAID or the world of RISC.

Spicer: Got you.

Katz: Even though the engagement is very important, you know, so the RISC-V work that Dave is doing now, Dave and Krste Asanovic, who left the university to start a company to commercialize that, you know, they're very actively involved with the industrial consortia that's standardizing the RISC-V architecture, so being engaged with industry is a good thing but, you know, your research evolves, goes into different areas, and it shouldn't be owned by a university. I think that was the problem with Berkeley Unix, you know, touching on broader computer history, is that the group developing it stayed at Berkeley throughout the 1980s. I think it opened a window of opportunity for Linux and Torvalds, you know, Linus, to do what he did, and in the race to, you know, whose operating system would come to dominate, by holding it so tightly in a I guess you would call it the cathedral model versus the bazaar model of, you know, Berkeley Unix being the cathedral with the close tight developer group and Linux being the bazaar. Linux ultimately had a much broader use, so a university shouldn't be in the business of owning the thing, and so I'm glad that industry really developed its own approach for managing RAID.

And RAID is, you know, sort of in the-- you know, these projects overlap. So as I mentioned, say 1986, we start having the, you know, what are we going to do next? 1987. It's going to be some combination of SPUR as a database machine, but in 1987 it really became arrays. You know, sort of 1986 it was going to be can we apply architecture to do something good for databases? By 1987 it's we're off on this we're going to do something-- we're going to improve I/O systems by making use of large numbers of arrays but also make them reliable, and then we get the ability to build our very first prototype was probably 1988. That was also when the paper appeared that was very influential, and we built it out of a Sun workstation

with 32 five-and-a-quarter-inch drives in it, and everything in software, and by 1989, 1990, we're building hardware which is FrankenRAID, of which we--

Spicer: Yes.

Katz: You have no idea how proud I am that our prototype is amongst the collection of the Computer History Museum. I'm just so incredibly proud that it's there and people can see it. But that was, you know, 144, three-and-a-quarter-inch disk drives. Sorry, 3 1/2 inch. It was five-and-a-quarter inch, with a custom controller board that went into the back of a, you know, a high-performance back link for supercomputers. It was actually--

Spicer: Mm, HIPPI?

Katz: Forgot the name of it. HIPPI [High Performance Parallel Interface]. HIPPI connections, but it was designed-- it's kind of an early direct-attached storage device to a supercomputer through the HIPPI channels. And we developed a hardware network.

Spicer: Was it connected to a supercomputer at Berkeley? Why did you have the HIPPI on it?

Katz: We connected it-- so, you know, again, an interesting collaboration with Thinking Machines. Thinking Machines again too, which was also funded by DARPA as part of this High Performance Computing and Communications program, actually had a way of connecting the Thinking Machines to HIPPI, and so they gave us the chassis for that. We didn't-- we basically attached a set of Sun workstations through a network, a high-performance computing network that was based on HIPPI as the links and--

Spicer: I see.

Katz: --the back end of that to the storage was through HIPPI. So we actually built a system where there's a set of front-end workstations through a multi-hop network with HIPPI links to our HIPPI-attached storage prototype, which was FrankenRAID, with 144 disk drives in it and our custom disk controller that we built and wrote the software for, and the log-structured file system ran on those workstations with, you know, enhanced-- with Sprite. So we did the-- again, the full, you know, antenna thing, and we-- our benchmark was doing things like video playback. So I remember we had a video demonstration of the system working and they-- we put the final system together, benchmarked it and all these things in the time frame of 1992, so students put together a video, which is out there on YouTube somewhere. Now. YouTube didn't exist in 1992.

Spicer: Right.

Katz: Where you can actually see we're playing back a scene from "Young Sherlock," the film "Young Sherlock Holmes," while one of the graduate students is taking a disk drive and yanking it out of the controller.

Spicer: Oh, that's a good demo.

Katz: There really no interruption in the video playback.

Spicer: That's cool.

Katz: It's literally uninterrupted playback onto a workstation across the network.

Spicer: Yeah. Nice.

Katz: So we really built the real thing and the chassis, which demonstrates the mechanical design of the disk drives, down to the way in which the disk drives are specially packaged to make them easy to remove without destroying the disk drive and easily replaced. Again, that engineer, Ken Lutz sort of oversaw all of the-- I think we may have hired a consulting company to work with us on the mechanical design, but he sort of oversaw worrying about all that stuff, which faculty and graduate students don't know that much about mechanical packaging of disk drives. He oversaw that, along with again, the complete demonstration system that we built.

Spicer: Wow, that's great.

Katz: So we didn't stop at the papers and we didn't stop at the chipsets. We actually built the real system and measured it.

Spicer: Yeah, and then you released it in the wild, and it just took off, and it took on a life of its own.

Katz: Yeah. That I consider if-- some things are home runs and the work on SPUR, I consider that a home run. RAID is the grand slam homerun. It's not just a no hitter. It's the perfect game.

Spicer: Yeah. Oh, yeah. Nice. Yeah, definitely. That's a good one. Well, what's our next project? It's SOAR, I think I meant to say, not SPUR. Right?

Katz: SOAR was Dave Patterson's project…

Spicer: I'm sorry. That's Smalltalk on a RISC. Right. Okay, what was the third project you mentioned

Katz: It's called BARWAN and that's where we did this new TCP work. So let me tell you about that. How we get from RAID to that. So I felt it was time for me, in the timeframe of now we're sort of order 1990, and I'm thinking about what will Randy work on next, and also, I'm feeling it's great working with Dave, but I'd like to work on something on my own, lead something on my own with some collaborators. So a new faculty member, Eric Brewer is joining Berkeley, and the two of us concoct a project called-- we started getting interested in wireless. So what is the hot thing of the moment? 1990 was really wireless LANs were becoming an extremely exciting topic in the timeframe of 1990. This is NCR and Bell Labs are producing this card that you can plug into your IBM laptop that will provide wireless, essentially wireless

Ethernet access, so you can kind of carry your laptop around, and it's connected to the network. Oh my god, this is like really cool. So this is pre-wi-fi and actually wi-fi before it was standardized.

Spicer: Whose network is it connecting to though? What is that? What's going on there?

Katz: It's connecting to the campus network.

Spicer: Are those the little Ricochet modems you're talking about that used to-?

Katz: That came along.

Spicer: Later.

Katz: That later. So this is a PC card that can plug into your laptop which had a PC card slot on it.

Spicer: Oh, yes, right.

Katz: And hanging off of it was an antenna that you could put on the back of your laptop, and it talks to a base station that essentially has the same thing, and inside it is a PC on the one end, and on the other end is an Ethernet connecting to your campus network.

Spicer: Oh, okay, I got it now.

Katz: It's doing, basically the function of a level two bridge in networking parlance. So it's sort of a bridge between the wireless world and the Ethernet world of your campus network. The key thing is, in 1990, you can go and buy this stuff, and of course the base station and the things cost thousands of dollars. So a research group can sort of get a vision into the future by spending money, which is another important concept, right? Give research groups money to buy technology, which is way too expensive to be sort of commercially widespread but seed it with the people who are going to do interesting things with it. So the first interesting thing is, wait a second, this is kind of a weird network. What happens if-- how good does standard networking protocols work across this wireless network? And it turns out the performance is not very good. Sort of like your Ethernet at this point in time is 10 megabits a second. You're getting kilobits over the wireless network. That's horrible. How can we improve that? You also have the problem that it's a laptop. It's not plugged into the Ethernet. It's virtually plugged in over this wi-fi antenna. You can get up and walk around with it. How do you deal with mobility in it?

So there's a set of really interesting things in terms of networking to make work with this world that's now enabled by I can spend a couple of thousand dollars, buy a base station and a plugin for my laptop, and start playing with it. So it becomes a very prevalent sort of project or set of challenges. Not just at Berkeley, but at many places where also jumping on top of the opportunity here and industrial research labs in the timeframe of 1990. In 1992, I got very energized by the political campaign of Bill Clinton and in particular Al Gore. So the fact that someone's running for vice president of the United States who had

been so influential in the legislation that was leading to the development of the internet. Even in 1992, it was early for the internet. But it was technologically savvy and so on.

I decided to take a leave from the university and go work in Washington at DARPA, so I was actually a DARPA program manager from [January] 1993 to [December] 1994, and when I went to Washington my goal was to establish a new research program in mobile computing. Because this was so cool and there's so many people starting to work on this. They need funding to make it happen of a size which was large enough to really have an impact and also build a community and have them work with each other to advance and accelerate mobile computing. There had been work on sort of packet internet in the 1970s, into the 80s, that was very exciting. But it had kind of ended and was no longer being pursued at DARPA by 1993. It was time to revisit it in the context of the new wi-fi and wireless LAN technology that was coming along.

By the way, that was the time I was in Washington, where I connected the internet to the White House, which is an entirely different story that maybe if we have time, we can come back to that, but I'd say we want to use the opportunity of this oral history to talk about the technology.

Spicer: Oh, great. Yeah.

Katz: So I was in Washington, '93, '94. I come back to Berkeley in '95 after my two-year stint in Washington and I need to work on a new research project. So it's going to be in wireless and one of the ideas that I had when I was in Washington was this notion of overlapping wireless networks because you mentioned the Ricochet. So in the interim, by the mid-1990s now, in addition to wi-fi and wireless LANs, which essentially look like Ethernet, you have also packet radio technology, which is out there, and that's exciting because the wi-fi stuff works kind of in your university building. That's kind of the scope of the wireless LAN. But the Metricom Ricochet technology, they were deploying it in the city of Berkeley. They were deploying it in the Bay Area. Now you have wireless connectivity that actually can follow you around in your car while you're driving from home to work. That's a whole set of new technological challenges, and then in addition, this is an era where the cellular technology providers are starting to think about how they integrate data more completely to end devices that are in phones, computers, as phones would very quickly become with the introduction of the iPhone, of course, and whether they saw it or not, now they want to integrate data with the digitized voice packets which are going over the digital cellular networks.

At this time, the cellular radio technology was GSM. So there was an opportunity to work with data over cellular as well and there was satellite and so on. So we formulated a project called the Bay Area Research Access Network or BARWAN. Sorry, Bay Area Research Wireless Access Network, BARWAN, and I worked with Eric Brewer on this, and Eric was my software collaborator because you're now moving between networks, which are let's say megabits per second in the wireless LAN to tens-- a hundred kilobits per second when you're using the Ricochet, to maybe 10 kilobits per second when you're using data over cellular, to maybe a kilobit per second over satellite link and satellite phone, and so how do you build applications that can adapt dynamically to these different variations in the network available bandwidth and possibly the loss rates associated with different kinds of wireless networks?

So he and his students did some great work on building network browsers that could be very smart and adaptive. As your bandwidth changes, they did smart things like if you have a low bandwidth connection, make sure to prioritize the text part of the webpage. If you're on a medium bandwidth kind of connection, bring down the text and a -- transcoded in the network -- black and white version of the images, highly compressed. If you're on a good bandwidth, then just the normal HTML. Just download it and do that in a dynamic fashion that can be aware of what the limitations on the link are. So he contributed a lot. He and the students did tremendous work on developing these things that over time, they wound up becoming part of HTML 3.0, where they worked their way into the standards in terms of prioritizing and alternative representations. But they pioneered a lot of that in software that they developed as part of the project.

Spicer: Was Armando Fox part of that group?

Katz: Armando was part of that group, receiving his PhD as part of that, and also a fellow named Steve Gribble, who was a faculty member at the University of Washington, but is now-- I believe he's a senior engineer at Google these days, but yeah, Yatin Chawathe. I've forgotten his last name but he also was a key engineer at Google now but was a developer as part of that. Again, it's in that style of we have an industrial consortium and so on. But the technology I wanted to talk most about in the context of that project, which had lots of different technological components was this new TCP.

So as you know, the transport control protocol is a very essential component of how the internet works and why the internet doesn't meltdown as it has scaled up. It's a protocol that is kind of an end-to-end adaptive protocol that detects losses of information flow from source to destination and slows the sender down dynamically, is sort of dynamically determining that there's a bottleneck between sender and receiver, and doesn't try and overwhelm that bottleneck, and that's sort of-- there's a lot of variations and evolutions of how it works to do a better job of that. But that's, in nutshell, what it does, which is one of the great magical aspects of why the internet can work for however many billion people use it today across such a diversity of ways in which people interconnect to the internet.

Spicer: Yeah, that's amazing.

Katz: So wireless. Wireless is kind of interesting because first of all, the bandwidth typically between the sender and the base station is all wired and high. But between the base station and the wireless endpoint, it can have problems. First of all, it can be a lot slower than if you have your campus gigabit Ethernet. Now you're talking about a megabit maybe wi-fi network. So there's a big speed mismatch. But also that network, you turn a corner. You're playing with your thing. You're walking down the hall and you turn a corner. All of a sudden you're in a dead spot, and a whole bunch of packets don't make it through, and that behavior has big implications for how TCP works because TCP in the timeframe of the mid 1990s now, had been optimized for when you see packet loss, it's because of congestion, back off, slowdown, you're overdriving the network, when in a wireless link when you see losses, you've just stepped into a dead zone. You might step out of it in a second.

So there are lots of performance implications from that observation and so we tried to solve those with Snoop TCP. Now you know I mentioned snooping caches. So the concept of the snoop which goes back

to that very first project was watch and see what's happening on the bus between the processors and memory and do something intelligent in the way you're managing your cache based on that. The concept here is the same. In the base station, deploy software that's watching how packets are arriving at the base station from the wireless link and from the wired side, and do something intelligent based on what you can infer from that, and so a behavior which is a packet arrives from the internet, and you send it out over the wireless.

Now the way TCP works is when the receiver gets the packet, it sends back an acknowledgment, which is used to pace forward the next chunk of information. So acknowledgement comes. Wait a second. No acknowledgement came back. I guess that packet was really lost but the packet was lost over the wireless network. It isn't necessarily the packet. It isn't necessarily due to congestion. It's due to loss on the wireless network. We want to do something intelligent so that the receiver doesn't decide there's congestion, I should send stuff much more slowly. But we do want the sender to not overdrive the wireless link. So we send forward-- the way TCP works, it gets a chunk of packets at a time. It doesn't send one packet, wait for the acknowledgement, then send the next packet because that would be terrible performance, given the end-to-end delays between a website off in the internet and a laptop sitting at Berkeley sort of wirelessly browsing the web. So you send forward from the base station one TCP segment. We'll call it a packet, a second packet, a third packet. Maybe they all fall on the floor. Let's send a fourth packet. Oh, we got an acknowledgment for the fourth packet. That means that something is reaching, and there's enough information in that acknowledgement that comes back that you could, if you had buffered packet one, two, and three in the base station, you could now send packet one again, packet two again, and packet three again. You're basically retransmitting packets. You're correcting the error on the link by resending information. Now, if there was congestion on the link, and you're resending things, that's a very bad strategy. But if there's loss on the link, this is a way of making the link appear to applications that are running on the laptop that the link was reliable.

So basically, this is what Snoop TCP does. It violates traditional network layering. So it does two things which most network gurus would say is horrible. At the link layer, it allows visibility into transport because the link layer needs to know about the structure of TCP packets and acknowledgments to do what it's going to do, to figure out when to retransmit. So networking people don't like to violate layers. That's one of the-- a key tenet of networking. The second one is, we're maintaining information at the base station, and the base station could fail. That's not a good thing. You shouldn't have state in the network. It should all be on the ends, not in the network. So there was an element of arguing why this actually does violate some networking sort of truths or principles, but it's for a good effect, and it doesn't really hurt anything, and so the student who was-- there were several students involved with this. But the students who really led the argument was Hari Balakrishnan, who is now a faculty member at MIT and also won an ACM Dissertation award for this work. He and other students, Elon Amir, and Venkat Padmanabhan, Elon is a tech company CEO. Venkat is a senior researcher at Microsoft. Basically, they argued, in part, in the paper that I asked you to take a look at that you can implement this in a way which ensures that if the base station fails, you haven't hurt the end-to-end behavior of TCP. You haven't really hurt anything in terms of TCP correct protocol behavior if the link layer is aware of what's going on in the transport layer, and it has the benefit of dramatically improving the throughput, realizable over a lossy wireless link. So much so that in our collaborations with Ericsson, and a student named Reiner Ludwig—[again, I'm

blanking on his last name], Reiner, who I'll fix this when we do the transcript. Reiner Ludwig was kind of seconded to us from Ericsson to complete his PhD in our group from Europe and a lot of those ideas were influential in the way in which Ericsson would implement next generations of link layer retransmission in the cellular system that you and I are using today.

So again, it's sort of a very interesting-- it's part of that evaluation, design, and implementation strategy I talked about earlier in the oral history of evaluate the wireless link and see its bad behavior. When it comes to TCP throughput over the wireless LAN technology while you're actually walking around and causing packets to be lost because of the vagaries of the radio link, TCP, instead of trying to send things again soon, thinks that, oh, my god, the network is overwhelmed, I better start small, and it's painfully slow. I mean, it's hard to remember in the context of 2022, what browsing the web over one of these wireless devices in 1995 would look like. So you know it's a problem. There's a way to fix it. You can maybe do something intelligent in the base station by watching how TCP is behaving and sort of intercept what's going on at the base station level, to not wait for the end sender to adapt, but to fix it locally, to do a local fix in transmitting, and that had a tremendous improvement in terms of wireless performance, and this played well with the work that Eric Brewer and his students like Armando Fox were doing because we had an application that was very visual in terms of if you could keep the TCP behaving while using Armando and Steve's browser, you would visually see a higher quality rendering of the webpage on the wireless device that your eyes were glued to. That was really a great project for that.

Spicer: That’s amazing. I can't help wondering if the ALOHAnet project in Hawaii which used radio links to interconnect the islands, and I know influenced Ethernet was sort of the…

Katz: Absolutely, and wireless LAN as well.

Spicer: Was there any inheritance from ALOHA that influenced this project?

Katz: During the lifetime of the project, I did have an opportunity to meet with Norm Abramson, who is Mr. ALOHAnet, and we brainstormed on a lot of these kinds of ideas.

Spicer: Oh, great.

Katz: At this point, he had retired from the University of Hawaii, and had moved back to the mainland, and was pursuing some entrepreneurial opportunities related to better ways of doing wireless LANs, and to be honest with you, I don't know how that wound up evolving. But certainly, it was such a privilege to have an opportunity to have interactions with him on you know. He was open to these ideas. Another guy who was-- Jesus, I wish I could remember all the names. [I was thinking of Paul Baran, one of the conceptual founders of packet networks, who was at RAND Corporation in the early sixties.] One of the sort of venture capitalists who played into Metricom. He was very, very influence in both the technology-- Darn it. I'm just forgetting his name. He lived in Atherton, but we had a great interaction with him on a technical basis on this BARWAN project and I'll dig up his name when I get a chance to update the transcript. He was a RAND [the Federally Funded Research and Development Center in Santa Monica] researcher. Then he became a venture capitalist. He invested in Metricom. He was very involved in some

of the early 3Com stuff commercializing ethernet with Metcalfe and so on. Just man, he's such an important person in the field. But I'll get his name. Age and infirmity.

Spicer: I wanted to ask you just as we wrap up, where does this last invention appear? You mentioned Ericsson. Has it been absorbed into the way things are done now by everyone?

Katz: Yeah. So you know how the internet evolves through requests for common RFC [Request for Comments].

Spicer: Right.

Katz: I really credit Hari Balakrishnan for pursuing the, if you would, standardization of TCP through the RFC mechanism. He also was a very keen participant in the Internet Engineering Taskforce, which is kind of the brain trust for evolving the internet protocols and he was the champion for this approach, even though at the surface of it, it sort of violates certain sacred cows the way it behaved. But ultimately, it got to an RFC stage and was generally adopted by the internet community. But I think the place where you would really feel it's industrial impact today is on the reliable link layers of data over cellular with LTE, 5G, whatever comes next. These ideas have found their way into ever increasing intelligence in the base stations.

Spicer: What are your plans for the future, and then also, where do you see computer science going in let's say the next, I don't know, whatever you're comfortable with, 10 years, 50 years?

Katz: Is it okay if I give you just the two minutes on sending an electronic mail to the White House?

Spicer: Oh, of course. Oh, I'm sorry. Yes, of course. Yeah.

Katz: No, but your questions are, of course, my plans and the future of the field. So first of all, you know, I was at Washington. I basically arrived in DC to take the first working day of 1993, whatever it was, January 2nd, and there's such a learning curve in moving from being a DARPA, beneficiary of DARPA programs as a researcher to becoming an effective program manager. I really appreciate the sort of breaking me in that John Toole, who was CEO of the Computer History Museum a few years back-- he was my boss and actually, my boss, boss was Steve Squires, who it's a shame he's passed away because for me, he's one of the individuals who, if you're familiar with Asimov's Foundation [trilogy] he is the Harry Seldin… an exemplar of the sort of people whose careers are in government... behind the scenes, they are advocates, and program builders, and secure the funding that makes people like me have the kind of successful research careers that we ultimately have because we've been given community, and we've been given resources in order to excel, and they're not often recognized. So I'm glad I got the word out, both on John who after-- at some point, Steve was moved into a different position, and John was my boss, but he was the deputy boss when Steve was there.

Spicer: Wonderful. That's great.

Katz: A lot of connections to the Computer History Museum that we don't have time to explore but are interesting. So I arrived in January of 1994. I'm learning how to be a program manager. I'm learning a lot about how to sell a program at DARPA because I wanted to get this mobile computing program going. But there's also a new administration in Washington and I get a phone call. I didn't get it. But Steve got it, or maybe it was the DARPA director, but I think it was Steve got a phone call from Tom Kalil, and Mike Nelson at the White House, who are working with the Vice President on his technology agenda, and the Clinton Administration is shocked by the low level of technology in the White House. It's 1960s gear and we're in 1994, 1993. So okay, DARPA, and they also contacted NSF, and they contacted GSA, and they contacted, you know, “Can you help us?” and this was just before Presidents Day weekend. So I think Steve brought it to the group of program managers and said, “Anyone willing to volunteer their weekend and go into the White House and figure out what's going on, on a long weekend?” and here's the sole guy on a university secondment, “I'll do it. I'll do it. I'll do it. I volunteer. I volunteer.”

Spicer: Oh, yeah, you can see the White House.

Katz: We got to go in, and indeed, it was push- it was sort of push button phones, party lines, an panel switchboard, and it's not hard to imagine.

Spicer: A certain tape recorder, perhaps.

Katz: A certain tape recorder. It’s a centralized email system that ran on a DEC minicomputer called OEM 1. But it turned out that it was kind of interesting that there was a local area network. It was in the vice president's office and the reason for it is the vice president is the only person in the executive branch who has a job also in the legislative branch. They are the head of…

Spicer: The speaker of the house.

Katz: They’re sort of the…

Spicer: Or the Senate. Sorry.

Katz: Speaker of the Senate, sort of the leader of the Senate, and so he was able-- Dan Quayle, who was the vice president for the preceding president who was George Bush, was able to get some funding to have a good computer system in the office of the vice president. Anyway, long story short, what we did was, with DARPA sponsorship, established whitehouse.gov as a website. This is in the days of NCSA Netscape. So it's pre sort of open to the general public. But we established www.whitehouse.gov as the website. We established email accounts for president@whitehouse.gov and vicepresident@whitehouse.gov. We established-- actually, from a technological viewpoint, which is kind of interesting, we knew that this would be the target of the most prolonged and sophisticated hacker attacks once it went live, so we better implement a security architecture around it, and the first open source internet firewall was the project that was developed as part of this bringing the White House online. So there were some interesting technological spinoffs of all of this, and it was just a tremendous opportunity to see the low state of technology, including the technology in the situation room, in the

basement of the White House, or the technology on Air Force One. Didn't get to fly in it, but we got to visit it, and but make some contribution on how to bring it into the 20th century, late 20th century, if not quite the 21st century.

Spicer: That's interesting.

Katz: So let me come back. Oh, sorry, if you have a question.

Spicer: No, I was just interested in who was the main vendor for that setting up the network, and I mean, were there any companies, or was it like a military contractor who set it up?

Katz: Yeah, it was a military contractor. Again, I'm blanking on the name of it, but it was one of DARPA's technology, and Steve Crocker was the principal investigator on it, and it was a Maryland based company that did sort of contract research, development, and implementation for DARPA, so it was very easy to have a little bit more money to buy some of their sort of engineering expertise to set up the server and stuff like that, while also giving them a little bit of a juicy contract on developing this open source firewall, which is both a great research project and a great sort of exemplar for the community that would need security support as more universities, and research labs, and other things sort of connected to the internet as it was clearly evolving towards commercialization in 1993, and I don't know if that software code base, how it influenced the commercial offerings that would eventually come down the road, but I'm sure people read it, and looked at it, and got ideas from it. [The company I was thinking about was called Trusted Information Systems, and Steve Crocker was a Vice President there at the time.]

Spicer: At this time, just and then we'll get on to the final questions, how rare was it to have email? Like was the White House ahead of the trend, or was it keeping up, or behind?

Katz: Yeah. So I can tell you in 1993 the President could not send an email to his Secretary of State, or there was no real way that one department could talk to another department. They communicated by paper memos. Within departments, like the vice president's office had a local area network, so they had PCs and LAN software, and the White House, in part because of presidential records laws, I believe, had invested in the White House Executive Office of the President having kind of a digital email system, but it was not possible to send an email from that system to anybody else. So if you're an assistant to the assistant to the assistant to the assistant of the counselor to the president, you could send an email to one less assistant or anybody else. But you couldn't really speak outside the Executive Office of the President. With the introduction of-- the first step was can you connect the White House to the rest of the world, and it did influence work that the Vice President really championed, which was to get the entire government on the ability to communicate with each other digitally and electronically. It's one of the great things that was part of my experience in Washington was to be part of a taskforce that the Vice President actually led on reinventing government through information technology, and I think we benefit. When I've retired, I really switched over to Medicare, and I use the Medicare website, and chatting with people on Medicare because you can't get through to anybody on the phone, and getting all the forms, and all of the-- receiving all digital billing and all those things.

Spicer: Great. Can't wait.

Katz: The reason we can do it in 2022 was something that started, I think, in 1993 with that taskforce of how the government can work with each other better, communicating with each other, and provide better services to the public through digital means, instead of having to go to the Social Security office, which I have no idea where it is. It's probably not in the best part of town.

Spicer: And that really was Al Gore, wasn't it, mostly, wouldn't you say?

Katz: It started. But of course, it wasn't done overnight. So there were advances in the George W. Bush, and Obama, and Trump, and they kept doing it, and I can tell you that, in my interactions for government services, I'm actually very impressed with the Social Security Administration and Medicare, and I'm sure it's because it's the largest percentage of the American population interacts with those agencies, and sort of works with their government on that, compared, for example, Immigration and Border Control, which are kind of very difficult to deal with in my experience.

Spicer: Yeah, mine too.

Katz: Yeah, I think it's just an example. Anyway, I think there is a tremendous legacy there, and we could do better, but it's advanced a lot. I know we're more or less at time. So let me spend a couple of minutes on what I'm planning to do next. So I feel that you spend five years of your life fooling around as a kid. Then you spend the next 20 years of your life as a student going to school, which is kind of work, or advanced school like I have, and so now you're 25, and you're really ready to start working, and so I've had about a 40-year career, 41 years of working in industry, at University of Wisconsin, Berkeley, a little side activity at DARPA. I closed out my time at Berkeley as the vice chancellor for research, which is a whole other conversation much broader than computers.

Spicer: Amazing.

Katz: But I'm ready to have some goof off time. Again, because there isn't that much time left to my life, and I realize that, and so one of the things I am very excited about is really capturing some of those big trends that I've benefited from and observed in my career, as we have discussed in this oral history, particularly the way in which that tightly coupled stack of IBM hardware and software. In 1970, when I programmed that Olivetti 101, you got the software for free from the computer vendor. You paid an arm and a leg for the computer. All the stuff you plugged into the computer came from that vendor. It cost a fortune. They controlled everything. They controlled the pace of advancement. To the world we live in now, which is this ecosystem of someone comes up with a good idea for a new app or programming language or something, and they can build a community, it seems, overnight, and the world can advance so rapidly because we've broken the stack up into pieces. The processor, maybe it comes from Intel. Maybe it comes from ARM. Maybe it comes from Apple. The sort of system could be a Samsung, or Apple, or some form of Dell, or the number of companies building hardware all over the place. The number of companies providing software are huge, apps, app tools, all these things, and the ones that succeed are the ones that are communities, and in some sense, it's really interesting that software was

essentially free in 1970. You bought the computer. They threw the software in. To Ingres in my career, where you want third party database software, you're willing to spend a lot of money for it because your only choice is, is IBM or DEC going to build a system that meets your needs? Maybe they will. Maybe they won't and it's not good to have only one vendor. So it's good that there's an Ingres and an Oracle, and sort of maybe you could buy the software from DEC, or maybe you could buy software from IBM, so you have a dynamic and a competition, so things advance by better features, better performance, and so on, and that's really kind of stuff that happened in the 80s, and these days, software is free again. When you think about Google search, or some of the applications and so on that we use on cloud, but again, our use of them, we're the product, in some sense, of these things. So it's really interesting and I'd like to wrap my head around describing how the field has evolved. So I have a writing project ….

Spicer: Oh, good.

Katz: But I think at this stage, I will conclude that other than historical research, my research career has ended, and in terms of the field, I think by far, the most exciting thing that I have observed is the way in which artificial intelligence has really embedded itself. Well, and computing has embedded itself in virtually everything that we do in our lives, from the cars we drive, to our processes, our communications, the way we interact with our friends and family, and in a way that again, in 1970 when I started or when I was in college '73 or in grad school in '76, I could never have predicted that billions of people would be using this stuff. I was preparing myself for a select priesthood of users and developers. I had no idea. I would never have guessed that whatever it is, 4 billion people have sent an electronic mail message, or maybe it's more than that, 5 billion have done that by the end of my career. It's just amazing how embedded it is, and now I think we're a lot more sensitive to some of the risks of an all-digital society, both in terms of our dependence on what we know are difficult to make reliable systems, as well as the way in which they can be manipulated and used for not only good things, but also harm, and I think as we go to the next generation of ever intelligent systems, as artificial intelligence, of course, is widely available, but is evolving rapidly, that we'll do a better job as a field to consider those unintended consequences earlier, like the inherent biases in AI algorithms, and so on, and do a better job, both in terms of how we educate our workforce, the people who are the future computer scientists, and the public about what the risks are, and what they should demand from their systems. So while I think there's a lot of great technology, quantum computing, all kinds of things coming down the road, but the way it will immediately touch people is the ever-increasing intelligence and our ever-increasing dependence on these systems. Will they do the right thing? Will they do what we want? Will they do it reliably? These are challenges for the field, I think in the sort of immediate future.

Spicer: Yeah. Oh, that's wonderful. Thank you, Randy.

Katz: Yeah. Thank you. I wanted to also say we've known each other forever, through the early days, and I don't know, maybe even the earlier days in the formation of the Computer History Museum. I just want to underscore how delightful, at various points in times, it's been to collaborate with you on one project or another. I'm so glad that you are the one to collect this oral history from me, and really, one of the heights of my-- peaks of my career is the willingness of the Computer History Museum to accept our donation of

really, the second RAID prototype that we built at Berkeley called FrankenRAID, also known as RAID number two.

Spicer: That is one of my highlights and that was one of the earliest- I started in…

Katz: Earliest acquisitions, maybe.

Spicer: For me, I started in ‘95. So I think that's about the time that it came to the museum, maybe a bit after. So well, Randy, it's been just delightful chatting with you, and thank you so much for spending the time today, and I'm just going to pause or stop the recording here.

END OF THE INTERVIEW

