Doug Engelbart, a researcher at Stanford University, demonstrated a collaborative workspace called NLS (oN Line System) in the 1960s. Doug's vision was for people to use hypertext as a tool for group work. In order to help himself steer his computer's cursor across the screen and select hypertext links with ease, Doug invented a wooden block with sensors and a ball underneath, and called it a *mouse*. In a now-famous video, which I didn't see until 1994, Doug demonstrated using electronic mail and hypertext links with great agility with his homemade mouse in his right hand and a five-key piano-chord keyboard in his left hand. The idea was that a person could interface with the machine in a very close, natural way. Unfortunately, just like Bush and Nelson, Doug was too far ahead of his time. The personal computer revolution, which would make Engelbart's "mouse" as familiar as the pencil, would not come along for another fifteen years. With that revolution, the idea of hypertext would percolate into software design.

Of course, the next great development in the quest for global connectivity was the Internet, a general communications infrastructure that links computers together, on top of which the Web rides. The advances by Donald Davis, by Paul Barran, and by Vint Cerf, Bob Kahn, and colleagues had already happened in the 1970s, but were only just becoming pervasive.

I happened to come along with time, and the right interest and inclination, after hypertext and the Internet had come of age. The task left to me was to marry them together.

CHAPTER 2

Tangles, Links, and Webs

The research center for particle physics known as CERN straddles the French-Swiss border near the city of Geneva. Nestled under the limestone escarpments of the Jura mountains, ten minutes from the ski slopes, with Lac Leman below and Mont Blanc above, it offered unique research opportunities, and the area offered a very pleasant place to live.

Engineers and scientists arrived at CERN from all over the world to investigate the most fundamental properties of matter. Using enormous machines, they would accelerate tiny nuclear particles through a series of tubes that, though only a few inches wide, ran for several kilometers within a mammoth circular underground tunnel. Researchers would rev up the particles to extremely high energies, then allow them to collide. For an unimaginably brief instant, new particles might be made, then

lost. The trick was to record the high-energy debris from the cataclysm as it careered into one of two detectors inside the tunnel, each the size of a house, jammed full of electronics.

Research on this scale was so expensive that it had to involve collaborations among many nations. Visiting scientists would run their experiments at CERN, then go back to their home institutions to study their data. Though it was a central facility, CERN was really an extended community of people who had relatively little common authority. The scientists brought a wide variety of computers, software, and procedures with them, and although they came from different cultures and spoke different languages, they managed to find a way to work together because of their shared interest in particle physics and their desire to see a huge project succeed. It was a tremendously creative environment.

In 1980, CERN was in the process of replacing the control system for two of its particle accelerators. The work was getting behind, and CERN needed help. I had, by chance, been consulting elsewhere in Switzerland when my friend and colleague Kevin Rogers called from England to suggest we apply.

Upon our arrival to be interviewed, Kevin and I were given a tour, and soon found ourselves on a catwalk, looking out and over what looked like a huge, chaotic factory floor. This vast experimental hall was filled with smaller experiments, obscured by the concrete-block walls between them, hastily built to cut down radiation. Continuing along the catwalk, we came to the control room. Inside were racks and racks of computing hardware, with no lighting except for the glow of the many indicator lamps and dials. It was an electronic engineer's paradise, with columns of oscilloscopes and power supplies and sequencing equipment, most of it built specially for or by CERN.

At this time, a computer was still a sort of shrine to which scientists and engineers made pilgrimage. Most people at CERN did not have computer terminals in their offices; they had to come to a central facility, such as the terminal room next to the

control room, to actually program a computer system. Kevin and I would soon join a team of people who would ultimately bring about the demise of that control room. Alas, the racks of glowing electronics would be slowly dismantled and replaced by a boring oval of computer consoles, run by much more powerful software.

The big challenge for contract programmers was to try to understand the systems, both human and computer, that ran this fantastic playground. Much of the crucial information existed only in people's heads. We learned the most in conversations at coffee at tables strategically placed at the intersection of two corridors. I would be introduced to people plucked out of the flow of unknown faces, and I would have to remember who they were and which piece of equipment or software they had designed. The weblike structure of CERN made the job even harder. Of the ten thousand people in the CERN phone book, only five thousand or so were at CERN at any given time, and only three thousand or so were actually salaried staff. Many of the others had a desk, and visited from their home institutions only every now and again.

To house contractors who suddenly arrived in a panic to help advance some project or other, management had erected portable cabins on the top of a grassy hill on the grounds. Groups of us would discuss our ideas at lunch overlooking the Swiss vineyards, or as we walked down the long flight of concrete steps from the hill to the experiment hall and terminal room to do the programming. I filled in the odd moments when I wasn't officially working on the Proton Synchrotron Booster by tinkering with my play program, the one I called Enquire. Once I had a rough version, I began to use it to keep track of who had written which program, which program ran on which machine, who was part of which project. Informal discussions at CERN would invariably be accompanied by diagrams of circles and arrows scribbled on napkins and envelopes, because it was a natural way to show relationships between people and equipment. I wrote a four-page manual

for Enquire that talked about circles and arrows, and how useful it was to use their equivalent in a computer program.

In Enquire, I could type in a page of information about a person, a device, or a program. Each page was a "node" in the program, a little like an index card. The only way to create a new node was to make a link from an old node. The links from and to a node would show up as a numbered list at the bottom of each page, much like the list of references at the end of an academic paper. The only way of finding information was browsing from the start page.

I liked Enquire and made good use of it because it stored information without using structures like matrices or trees. The human mind uses these organizing structures all the time, but can also break out of them and make intuitive leaps across the boundaries—those coveted random associations. Once I discovered such connections, Enquire could at least store them. As I expanded Enquire, I kept a vigilant focus on maintaining the connections I was making. The program was such that I could enter a new piece of knowledge only if I linked it to an existing one. For every link, I had to describe what the relationship was. For example, if a page about Joe was linked to a page about a program, I had to state whether Joe made the program, used it, or whatever. Once told that Joe used a program, Enquire would also know, when displaying information about the program, that it was used by Joe. The links worked both ways.

Enquire ran on the group's software development computer. It did not run across a network, and certainly not the Internet, which would not be used at CERN for years to come. Enquire had two types of links: an "internal" link from one page (node) to another in a file, and an "external" link that could jump between files. The distinction was critical. An internal link would appear on both nodes. An external link went in only one direction. This was important because, if many people who were making such a link to one page could impose a return link, that

one page would have thousands of links on it that the page's owner might not want to bother to store. Furthermore, if an external link went in both directions, then changing both files would involve storing the same information in two places, which is almost always asking for trouble: the files would inevitably get out of step.

Eventually, I compiled a database of people and a database of software modules, but then my consulting time was up. When I left CERN, I didn't take the Enquire source code with me. I had written it in the programming language Pascal, which was common, but it ran on the proprietary Norsk Data SINTRAN-III operating system, which was pretty obscure. I gave the eight-inch floppy disk to a systems manager, and explained that it was a program for keeping track of information. I said he was welcome to use it if he wanted. The program was later given to a student, who said he liked the way it was written—written as a Pascal program should be written. The few people who saw it thought it was a nice idea, but no one used it. Eventually, the disk was lost, and with it, the original Enquire.

When I left CERN I rejoined a former colleague, John Poole. Two years earlier, Kevin and I had been working with John, trying to upgrade the then-boring dot matrix printers with the then-revolutionary microprocessor so they could print fancy graphics. The three of us would sit in the front room of John's house, his golden Labrador nestled under one of the desks, and try to perfect the design. We had succeeded in just a few months, but John hadn't had the money to go on paying us a salary, and wouldn't until he'd sold the product. That's when we had started looking for contract work and ended up at CERN.

After I had been at CERN for six months, John called. "Why don't you come back?" he said. "I've sold the product, we've got a contract. Now we need some software support for it." John had incorporated as Image Computer Systems, and Kevin and I returned to help.

We rewrote all the motor controls to optimize the movement of the print head so it was fast. It could also print Arabic, draw three-dimensional pictures, and give the effect of preprinted stationery while using less expensive paper. We wrote our own markup language in which documents were prepared, and the printer could also handle input codes of much more expensive typesetting machines. We could change not only fonts but almost any aspect of the printer's behavior.

The business went well, but the technology we were working with was limited to what we could put into printers. I felt I needed a change from living in Britain, and I remembered that CERN had a fellowship program. In the spring of 1983 I decided to apply, arriving eventually in September 1984. As a gift upon my departure from Image, John gave me a Compaq personal computer. It was touted as one of the first "portable" computers, but it looked more like a sewing machine, more "luggable" than portable. With my new PC, and the freshness that comes with change, I wrote in my spare time another play program, called Tangle. I wanted to continue to explore the ideas about connections that were evolving in my head.

In an extreme view, the world can be seen as only connections, nothing else. We think of a dictionary as the repository of meaning, but it defines words only in terms of other words. I liked the idea that a piece of information is really defined only by what it's related to, and how it's related. There really is little else to meaning. The structure is everything. There are billions of neurons in our brains, but what are neurons? Just cells. The brain has no knowledge until connections are made between neurons. All that we know, all that we are, comes from the way our neurons are connected.

Computers store information as sequences of characters, so meaning for them is certainly in the connections among characters. In Tangle, if a certain sequence of characters recurred, it

would create a node that represented the sequence. Whenever the same sequence occurred again, instead of repeating it, Tangle just put a reference to the original node. As more phrases were stored as nodes, and more pointers pointed to them, a series of connections formed.

The philosophy was: What matters is in the connections. It isn't the letters, it's the way they're strung together into words. It isn't the words, it's the way they're strung together into phrases. It isn't the phrases, it's the way they're strung together into a document. I imagined putting in an encyclopedia this way, then asking Tangle a question. The question would be broken down into nodes, which would then refer to wherever the same nodes appeared in the encyclopedia. The resulting tangle would contain all the relevant answers.

I tested Tangle by putting in the phrase "How much wood would a woodchuck chuck?" The machine thought for a bit and encoded my phrase in what was a very complex, tangled data structure. But when I asked it to regurgitate what it had encoded, it would follow through all the nodes and output again, "How much wood would a woodchuck chuck?" I was feeling pretty confident, so I tried it on "How much wood would a woodchuck chuck if a woodchuck could chuck wood?" It thought for a while, encoded it, and when I asked it to decode, it replied: "How much wood would a woodchuck chuck if a woodchuck chuck wood chuck chuck chuck wood wood chuck chuck chuck . . ." and it went on forever. The mess it had made was so horrendously difficult to debug that I never touched it again. That was the end of Tangle—but not the end of my desire to represent the connective aspect of information.

I had always stayed on the boundary of hardware and software, which was an important and exciting place to be, especially as software more and more took over hardware functions. When I applied for my fellowship to CERN, I specified that I wanted a job that would allow me to work on both, and suggested three places there where I could do that. I ended up being hired to

work with "data acquisition and control," the group responsible for capturing and processing the results of experiments. Peggie Rimmer, who hired me, would also teach me, as it turned out, a lot about writing standards, which was to come in useful later on. I was in a position to see more of CERN this time, to appreciate more of its complexity. Although attached to a central computing division, my group worked with the individual experiment groups, each of which was a diverse mixture of scientists from all over the world.

By 1984, CERN had grown. A new accelerator, the Large Electron Positron accelerator, was being built. Its tunnel, twenty-seven kilometers in circumference, ran from a hundred meters under CERN to, at its farthest point, three hundred meters beneath the foothills of the Jura mountains, dwarfing other accelerators. The computing diversity had increased too. A newer generation of computers, operating systems, and programming languages was being used, as were a variety of networking protocols to link the many computers that sustained the big experiments. Machines from IBM, Digital Equipment Corp. (DEC), Control Data—we had them all, as well as the new choice of PC or Mac in personal computers and different word processors.

People brought their machines and customs with them, and everyone else just had to do their best to accommodate them. Then teams went back home and, scattered as they were across time zones and languages, still had to collaborate. In all this connected diversity, CERN was a microcosm of the rest of the world, though several years ahead in time.

I wrote a general "remote procedure call" (RPC) program to facilitate communication between all the computers and networks. With RPC, a programmer could write a program on one sort of computer but let it call procedures on other computers, even if they ran on different operating systems or computer languages. The RPC tools would work over whatever network or cable there happened to be available in a given case.

I began to re-create Enquire on the Compaq. I wrote the program so that it would run on both the luggable Compaq and the VAX minicomputer made by DEC that I was using at CERN. I didn't do such a good job the second time around, though: I just programmed in the internal links, and never got around to writing the code for the external links. This meant that each web was limited to the notes that would fit in one file: no link could connect those closed worlds. The debilitating nature of this restriction was an important lesson.

It was clear to me that there was a need for something like Enquire at CERN. In addition to keeping track of relationships between all the people, experiments, and machines, I wanted to access different kinds of information, such as a researcher's technical papers, the manuals for different software modules, minutes of meetings, hastily scribbled notes, and so on. Furthermore, I found myself answering the same questions asked frequently of me by different people. It would be so much easier if everyone could just read my database.

What I was looking for fell under the general category of *documentation systems*—software that allows documents to be stored and later retrieved. This was a dubious arena, however. I had seen numerous developers arrive at CERN to tout systems that "helped" people organize information. They'd say, "To use this system all you have to do is divide all your documents into four categories" or "You just have to save your data as a WordWonderful document" or whatever. I saw one protagonist after the next shot down in flames by indignant researchers because the developers were forcing them to reorganize their work to fit the system. I would have to create a system with common rules that would be acceptable to everyone. This meant as close as possible to no rules at all.

This notion seemed impossible until I realized that the diversity of different computer systems and networks could be a rich resource—something to be represented, not a problem to be eradicated. The model I chose for my minimalist system was hypertext.

My vision was to somehow combine Enquire's external links with hypertext and the interconnection schemes I had developed for RPC. An Enquire program capable of external hypertext links was the difference between imprisonment and freedom, dark and light. New webs could be made to bind different computers together, and all new systems would be able to break out and reference others. Plus, anyone browsing could instantly add a new node connected by a new link.

The system had to have one other fundamental property: It had to be completely decentralized. That would be the only way a new person somewhere could start to use it without asking for access from anyone else. And that would be the only way the system could scale, so that as more people used it, it wouldn't get bogged down. This was good Internet-style engineering, but most systems still depended on some central node to which everything had to be connected—and whose capacity eventually limited the growth of the system as a whole. I wanted the act of adding a new link to be trivial; if it was, then a web of links could spread evenly across the globe.

So long as I didn't introduce some central link database, everything would scale nicely. There would be no special nodes, no special links. Any node would be able to link to any other node. This would give the system the flexibility that was needed, and be the key to a universal system. The abstract document space it implied could contain every single item of information accessible over networks—and all the structure and linkages between them.

Hypertext would be most powerful if it could conceivably point to absolutely anything. Every node, document—whatever it was called—would be fundamentally equivalent in some way. Each would have an address by which it could be referenced. They would all exist together in the same space—the information space.

• • •

By late 1988 I was plotting to somehow get a hypertext system going. I talked to my boss, Mike Sendall. He said it sounded like a reasonable idea, but that I should write up a proposal. A proposal? I had no idea what went into a "proposal" at CERN. I thought, however, that I'd never get the go-ahead to develop a hypertext documentation system unless it was approved as a formal project. I thought hard about how to get the excitement of this idea into a form that would convince people at CERN.

Although Enquire provided a way to link documents and databases, and hypertext provided a common format in which to display them, there was still the problem of getting different computers with different operating systems to communicate with each other. Ben Segal, one of my mentors in the RPC project, had worked in the States and had seen the Internet. He had since become a lone evangelist for using it at CERN. He went around pointing out how Unix and the Internet were binding universities and labs together all over America, but he met a lot of resistance. The Internet was nearly invisible in Europe because people there were pursuing a separate set of network protocols being designed and promoted by the International Standards Organization (ISO). Whether because of the "not invented here" feeling, or for honest technical reasons, the Europeans were trying to design their own international network by committee.

I was intrigued with the Internet, though. The Internet is a very general communications infrastructure that links computers together. Before the Internet, computers were connected using dedicated cables from one to another. A software program on one computer would communicate over the cable with a software program on another computer, and send information such as a file or a program. This was originally done so that the very expensive early computers in a lab or company could be used from different sites. Clearly, though, one computer could not be linked to more than a few others, because it would need tens or hundreds of cables running from it.

The solution was to communicate indirectly over a network. The Internet is a network of networks. Its essence, though, is a set of standardized *protocols*—conventions by which computers send data to each other. The data are transmitted over various carriers, such as telephone lines, cable TV wires, and satellite channels. The data can be text, an e-mail message, a sound, an image, a software program—whatever. When a computer is ready to send its data, it uses special software to break the data into packets that will conform to two Internet protocols that govern how the packets will be shipped: IP (Internet Protocol) and TCP (Transmission Control Protocol). The software labels each packet with a unique number. It sends the packets out over the phone or cable wire, and the receiving computer uses its own Internet software to put them back together according to the labels.

The Internet was up and running by the 1970s, but transferring information was too much of a hassle for a noncomputer expert. One would run one program to connect to another computer, and then in conversation (in a different language) with the other computer, run a different program to access the information. Even when data had been transferred back to one's own computer, decoding it might be impossible.

Then electronic mail was invented. E-mail allowed messages to be sent from one person to another, but it did not form a space in which information could permanently exist and be referred to. Messages were transient. (When the World Wide Web arrived, riding on top of the Internet, it would give information a place to persist.)

CERN's lateness in adopting the Internet was surprising, because the laboratory had been very much on the leading edge of networking and telecommunications. It had developed CERNnet, its own home-brewed network, for lack of commercial networks. It had its own e-mail systems. And it was at the forefront of gatewaying between different proprietary mail and file systems.

I was interested in the Internet because it could perhaps provide a bridge between different computer operating systems and networks. CERN was a technological melting pot. Many physicists were used to Digital's VAX/VMS operating system and the DECnet communications protocols. Others preferred the growing rival operating system, Unix, which used Internet protocols. Every time a new experiment got started there would be battles over whether to use VAX/VMS and DECnet, or Unix and TCP/IP. I was beginning to favor TCP/IP myself, because TCP was starting to become available for the VMS, too. It didn't initially come from Digital, but from Wollongong University in Australia.

Using TCP/IP would mean that the Unix world, which already used TCP/IP, would be satisfied, and those in the VAX world could get into the Unix world, too. Finally, there was a way for both contenders to communicate with each other, by picking up a piece of TCP/IP software from Wollongong. I became so convinced about TCP/IP's significance that I added code to the RPC system so that it could communicate using TCP/IP, and created an addressing system for it that identified each remote service in the RPC system. That's when the Internet came into my life.

For the proposal, I also had to think out what was needed to scale up Enquire into a global system. I would have to sell this project as a documentation system—a perceived need at CERN—and not as a hypertext system, which just sounded too precious. But if this system was going to go up as a way of accessing information across a network, it would be in competition with other documentation systems at CERN. Having seen prior systems shot down, I knew the key would be to emphasize that it would let each person retain his own organizational style and software on his computer.

The system needed a simple way for people to represent links in their documents, and to navigate across links. There was a model in online "help" programs: If there was an instruction or

tool on the screen that a user didn't understand, he just clicked on it and more information would appear. This approach was called *hot buttons*, a derivative of Ted Nelson's hypertext that had subsequently been used by Apple Computer's "Hypercard" and later in some way by many point-and-click help systems. I decided that on my system, if someone wanted to put a hypertext link into a piece of text, the words noting the link would be highlighted in some way on the screen. If a viewer clicked on a highlighted word, the system would take him to that link.

The pieces were starting to fall into place. TCP/IP would be the network protocol of choice. For "marketing" purposes, I would propose the system as one that would work over DECnet, with the added benefit that someone could communicate over the Internet, too. That left one hole: For people to communicate and share documents, they had to have a simple but common addressing scheme so they'd know how to address their files and others would know how to request files. I adapted the simple RPC addressing scheme.

In presenting my argument to an experiment group, I would note that they typically have different kinds of documented information—a "help" program, a telephone book, a conference information system, a remote library system—and they would be looking for ways to create a consistent master system. They would have three choices: (1) design yet another documentation scheme that is supposedly better than all the ones that have been attempted before it; (2) use one of the existing schemes and make do with its limitations; or (3) realize that all these remote systems have something in common. I would tell them, "We can create a common base for communication while allowing each system to maintain its individuality. That's what this proposal is about, and global hypertext is what will allow you to do it. All you have to do is make up an address for each document or screen in your system and the rest is easy."

In March 1989 I took the leap to write a proposal. I wanted to explain that generality was the essence of a web of information. On the other hand, I felt I had to make the system seem to be something that could happen only at CERN. I was excited about escaping from the straitjacket of hierarchical documentation systems, but I didn't want the people responsible for any hierarchical system to throw rocks at me. I had to show how this system could integrate very disparate things, so I provided an example of an Internet newsgroup message, and a page from my old Enquire program.

I was brash enough to look forward to having a web of data that could be processed by machine. I said:

> An intriguing possibility, given a large hypertext database with typed links, is that it allows some degree of automatic analysis. [. . .] Imagine making a large three-dimensional model, with people represented by little spheres, and strings between people who have something in common at work.
>
> Now imagine picking up the structure and shaking it, until you make some sense of the tangle: Perhaps you see tightly knit groups in some places, and in some places weak areas of communication spanned by only a few people. Perhaps a linked information system will allow us to see the real structure of the organization in which we work.

Little did I know that Ph.D. theses would later be done on such topics.

For all the decisions about which technical points to include in the proposal or exclude, and which social advantages of the system to emphasize, I was rather light on the project management details:

> I imagine that two people for six to twelve months would be sufficient for this phase of the project. A second phase would

almost certainly involve some programming in order to set up a real system at CERN on many machines. An important part of this, discussed below, is the integration of a hypertext system with existing data, so as to provide a universal system, and to achieve critical usefulness at an early stage.

By the end of March 1989 I had given the proposal to Mike Sendall; to his boss, David Williams; and to a few others. I gave it to people at a central committee that oversaw the coordination of computers at CERN. But there was no forum from which I could command a response. Nothing happened.

While I waited for some kind of feedback, I tested the idea in conversation, and reactions varied. CERN people moved through a number of overlapping loyalties, perhaps one to CERN, one to an experiment, to an idea, to a way of doing things, to their original institute . . . not to mention the set of Macintosh users or IBM/PC users. Another reason for the lackluster response was that CERN was a physics lab. There were committees to decide on appropriate experiments, because that was the stock-in-trade, but information technology was very much a means to an end, with less structure to address it. The situation was worse for very general ideas such as global hypertext. Even the RPC project, also an exercise in generality, had little formal support from within CERN, but it had enough support among different groups that I could keep it going.

In the meantime, I got more involved with the Internet, and read up on hypertext. That's when I became more convinced than ever that I was on the right track. By early 1990 I still had received no reactions to the proposal. I decided to try to spark some interest by sending it around again. I reformatted it and put a new date on it: May 1990. I gave it to David Williams again, and again it got shelved.

During this time I was talking to Mike Sendall about buying a new kind of personal computer called the NeXT. NeXT Inc. had

recently been started by Steve Jobs, who had founded Apple Computer and brought the first intuitive point-and-click, folders interface to personal computers. Ben Segal, our Unix and Internet evangelist, had mentioned that the NeXT machine had a lot of intriguing features that might help us. I asked Mike to let me buy one (bringing Ben with me for weight), and he agreed. He also said, "Once you get the machine, why not try programming your hypertext thing on it?" I thought I saw a twinkle in his eye.

By buying a NeXT, we could justify my working on my long-delayed hypertext project as an experiment in using the NeXT operating system and development environment. I immediately began to think of a name for my nascent project. I was looking for words that would suggest its new kind of structure. Mesh, or Information Mesh, was one idea (used in the diagram in the proposal), but it sounded a little too much like *mess*. I thought of Mine of Information, or MOI, but *moi* in French means "me," and that was too egocentric. An alternative was The Information Mine, but that acronym, TIM, was even more egocentric! Besides, the idea of a mine wasn't quite right, because it didn't encompass the idea of something global, or of hypertext, and it represented only getting information out—not putting it in.

I was also looking for a characteristic acronym. I decided that I would start every program involved in this system with "HT," for hypertext. Then another name came up as a simple way of representing global hypertext. This name was used in mathematics as one way to denote a collection of nodes and links in which any node can be linked to any other. The name reflected the distributed nature of the people and computers that the system could link. It offered the promise of a potentially global system.

Friends at CERN gave me a hard time, saying it would never take off—especially since it yielded an acronym that was nine syllables long when spoken. Nonetheless, I decided to forge ahead. I would call my system the "World Wide Web."