AMAZING
G

Kenneth P. Birman

Amazing Progress

The Gathering Dusk

A strange dynamic has emerged to interweave modern society, politics and technology in an ambitious attempt to "restructure" the American electric power industry. One sees a powerful societal demand for change driven by apparently irresistible forces – including some environmentally urgent considerations, such as the need to reduce emissions of greenhouse gases and other pollutants and to exploit natural energy sources like wind and solar power. Look behind the scenes and you'll find some less morally commendable motivations for change. No matter how restructuring plays out, savvy investors stand to make killings and the people who end up in control of the distribution of electrical power will also hold vast political power. And there's a complex technical angle: a whole series of technically audacious steps will be required to actually rebuild the current electric power grid into a restructured one and to control it in a way that guarantees reliable, safe delivery of power. Some of these technical steps are very challenging, reaching well beyond anything we can confidently say that the industry "knows how to do", and yet are fundamental to restructuring (for example, electric power is supposed to become a commodity like pork bellies or corn futures, bought and sold on an open market, but unlike pork bellies, the physical delivery of electric power is not "discrete" – all the electricity mixes together in the grid. How can we make sure that each producer produces the amount of power for which it has contracted – burns the right amount of coal or uranium and cranks out the right amount of electricity? And how should we configure the power grid, in real-time, to deliver the electricity?).

Taken together, these changes will completely transform the industry and the infrastructure on which it depends. But restructuring is off to a rocky start. The winter of 2000/2001, for example, ushered in a range of problems in California, which was at that time the furthest along in implementing restructuring: electric power price "spikes" and near blackouts, plagued the state, and there were all sorts of problems with power quality (one winces to think of the havoc this must cause California's high-tech industries and factories; Intel has threatened to stop constructing VLSI fabrication plants in the state unless the problem is resolved). Elsewhere, there were major delays in deregulation in the eastern corridor, although perhaps this was a happy situation given the picture in California.

In sum: an unfolding drama of massive proportions, occurring in what must have been one of the most stable and sleepy backwaters one could imagine. Revolutionary change in the electric power industry? Is this some sort of joke?

Historically the United States, and other countries, have treated electric power as a *natural monopoly*. The term refers to an industry within which the emergence of a single dominant player is a consequence of some physical characteristic of the product itself. Traditionally, this was the case the case for electric power because of the tremendous amount of infrastructure required: each utility owns both the generators and the wiring needed to provide power within some region, making its own investment decisions and setting its own pricing. Under such conditions, there is little to prevent prices from rising without limit, hence there are regulatory bodies in each state that review all price changes requested by the industry, limit their return on investment and control major investment decisions, while also protecting them against outside competition. With some exceptions, this has been a comfortable arrangement. As is well known to any Monopoly player, owning the utilities is more or less a guarantee of steady income.

Yet the evolution of the electric power industry has not been totally uneventful. Over the past few decades, the industry has come under many forms of pressure, stemming from the desire to increase power delivery capabilities while also cutting back on the construction of expensive new plants. These conflicting stresses resulted in some dramatic power outages. For example, I was in New York City during the summer of 1977, when the power failed twice over a short period of time. It was a particularly hot, humid summer, and during July we began to experience power flickers and brownouts. These culminated in blackouts – first a "minor" one, and then about a week later, a "major" one. The first time power was lost, I went home from work (at the time I was a student at

Columbia University, but was also employed part time as a technician in the Cardiology unit at Columbia Presbyterian Hospital, where I wrote software to analyze electrocardiograms). If I recall correctly, the event occurred around lunch-time, and it wasn't terribly exciting. I ended up spending the afternoon and evening with my friends, Max and Rose, and the power was back by 10pm. An uneventful day.

The blackout coincided with a period of social unrest and the newspapers were surprised at the mildness of the event. The morning headlines proclaimed with evident relief that the outage had produced little or no looting, and Mayor Abe Beame was quoted thanking the police for maintaining law and order. Later, we learned that a squirrel had gotten into a switching station, causing a short circuit (poor thing), and that the power grid was under such stress at the time that it destabilized, causing a general blackout. I wouldn't even remember this first event had it not repeated a week later, with rather different effect.

The temperature was, if anything, even hotter – it was one of those New York summers where sweating doesn't even cool you off. I was spending as much time as possible working, because the hospital machine-room was air-conditioned. And so, in the early evening, I was hacking away when the computer room was suddenly jarred by a strange, tortured sound. Now, you need to appreciate that in those days, computer rooms were extremely noisy places, with computers in all directions, loud air conditioners, big computer disks that could store hundreds of megabytes of data in a unit that looked and often sounded much like a small washing machine, uninterruptible power supplies and the like. While the result was hardly musical, it did have a sort of rhythm to it, and software people became accustomed to the usual sounds of these rooms. I remember that I used to go home after hacking late into the night and as I tried to get to sleep, I would hear the machine room humming like a distant jazz band.

Under these conditions, you can imagine the riveting impact of a really unusual sound. What I heard was a sort of resonant grinding noise throughout the room, the sound you might imagine hearing if a ball bearing broke inside the motor of a washing machine, except that it was coming from all around me and seemed to twist and reverberate. It seemed obvious that something was terribly wrong. In great anxiety (my backups weren't very current and I was terrified that days of work would be lost) I dashed from console to console powering down all the equipment. Moments later, the lights went out. Later, I learned that during an upstate thunderstorm, several major New York City power cables coming down from Canada were struck by lightning, protective relays tripped, and the overstressed system experienced a rapid, cascading failure.

The corridors of the hospital research wing were pitch black, except for flashing warning lights and one could hear alarms sounding from refrigerators and other power-sensitive equipment up and down the hallway. There seemed to be no point in staying at the hospital – they had emergency power in the intensive care units and surgical units, but nowhere else – so I asked the doctor for whom I worked if I could get a ride downtown with him. The highway was deserted and the scene had an eerie feel to it, as if the world had changed in some fundamental way. As we drove we listened to a local news station warning that the area north of Columbia University was the scene of looting, with mobs in the street. My friend dropped me off at the exit of the parkway, and I found myself walking a few blocks in a post-apocalyptic vision of New York.

The city itself was completely dark, although New Jersey, visible across the Hudson, still had power. Even the traffic lights were dark, although perhaps this didn't have a big impact on the drivers who were out that night. New Yorkers don't pay much attention to the traffic lights even under the best conditions. Yet the contrast between the dark streets around me and the bright skyline across the river was more than a little disconcerting. One doesn't often think about it, but New York is not a place that has much darkness at night – indeed, if you find yourself in a dark place in the city at night, you are running a big risk. I remember looking up at the silhouettes of the buildings around Columbia, black against a night sky for the first time in my experience, and thinking that I wouldn't see anything like that again, anytime soon.

When I reached Clairmont Avenue, where I had an apartment, I turned uptown. But if I had been uncomfortable previously, now I had a concrete reason to feel terrified. In the dark gangs of kids were pouring down the street, some carrying crowbars or chains, laughing loudly – and smashing things as they teemed down the street. The windshields of many of the cars were shattered, and I found myself clinging to the shadows against the walls that lined the street. If I had hoped to go unnoticed, I was naive. Some kid saw me and called out – "Hey you!" "Me?" "No need to hide like a fucking rabbit. We're not after your sorry white ass tonight!"

Cheered by this considerate remark, I made it to my building, where a band of residents were guarding the entrance. Upstairs, my parents' dog (visiting for a few days) was beside herself in terror. I took her downstairs with me, but this didn't help at all; she was even more frightened in the street. We ended up back at Max and Rose's place, sipping wine by candlelight, while the dog whimpered under a couch. The next morning, power returned and the headlines told stories from what *Time* labeled a "Night of Terror." As it turned out, the blackout had triggered a wave of rioting and looting, concentrated a few blocks north of my apartment.

Events such as blackouts confront us with the shallowness of civilized behavior. Turn off the lights and, it would seem, we tip the balance between humanity and something feral. During that second blackout, 125th street in New York was looted and gutted – storefront after storefront, blackened and shattered, like a scene from Kosovo. It took years before the area recovered. With the power out, the very foundations of modern society are shaken.

In the wake of these dramatic failures, and dozens of less dramatic ones, the main theme for the electric power industry became reliability and safety. A tremendous effort was invested to understand the sorts of design decisions that place the power grid at risk, and to improve its robustness. The telephone industry, which also experienced some large-scale failures during the same period, underwent similar self-scrutiny and improvement. Entering the 1980's, the United States had an extremely robust infrastructure in these respects.

But new pressures were now emerging. A first wave of major issues arose as nuclear power generation became so expensive (because of heightened safety concerns) that the financial solvency of some utilities was called into question. Power plants are normally financed using long-term bonds, which are repaid using the income from sale of power. If the costs of plant construction and operation soar because of unanticipated factors, or licensing is greatly delayed, a regional utility might easily face a choice between bankruptcy and untenable pricing. At the same time, inter-regional bulk power contracts enabled power pools with excess capacity to export power into regions with less generating ability and growing load. The grid became increasingly interconnected, and new kinds of stability issues were encountered. In one period during 1996, for example, two failures of a power line called the "inter-tie" triggered major power failures – the first impacting 6 states, and the other 14 states and millions of customers. The phenomenon is not confined to the United States: In New Zealand, the city of Auckland lost power for nearly two weeks during February of 1998, when a long-distance power line failed.

Meanwhile, other forces were conspiring to drastically change the industry. Small, specialized power generating equipment (using wind or solar power, or gas fuel cells) became increasingly efficient and emerged as a desirable source of energy in many parts of the country. Growing anxiety about greenhouse gases and global warming led to international agreements to reduce greenhouse gas emissions, which are byproducts of coal or oil fueled power generation. Acid rain in the Northeastern United States was definitively traced to dirty power plants in the Ohio Valley, placing the latter under steadily increasing pressure to install costly scrubbers.

The power industry also faced mounting political pressure. Dr. Massoud Amin is Manager of Mathematics and Information Science at the Electric Power Research Institute (EPRI) in Palo Alto, California, where he leads a national program of research in technologies needed to support the electric power grid. Massoud describes the situation as follows. Starting in 1978, the United States government began to deregulate a variety of industries that either had a monopoly structure (like the telephone industry), or were heavily regulated. This process began with the airline industry and then extended to encompass railroads, trucking, shipping, telecommunications, natural gas and banking. Massoud sees the process as one with deep roots: as early as 1776, Adam Smith, wrote that "Market competition is the only form of organization, which can afford a large measure of freedom to the individual. By pursuing his own interest, he frequently promotes that of society more effectively than when he really intends to promote it." More recently, Professor Alfred Kahn, a fellow Cornellian, guided the airline deregulation process during a term as head of the U.S. Civil Aeronautics Board, and wrote that "Deregulation is an admission that no one is smart enough to create systems that can substitute for markets."

Massoud explains that throughout most of the history of electric power, the institutions that furnished it have tended to be vertically integrated monopolies, each within its own geographic area. They have taken the form of government departments, quasi-government corporations or privately owned companies subjected to detailed government regulation in exchange for their monopoly status. Selling or borrowing electric power among these entities has been carried out through bilateral agreements between two utilities (most often neighbors). Such agreements have been used both for economy and for emergency back up. The gradual growth of these agreements has had the effect that larger areas made up of many independent organizations have become physically connected for their own mutual support. Thus, as the power industry evolved, it became organized into large regional power pools, structured as monopolies.

With the breakup of the AT&T telephone monopoly, there was a strong sense that monopoly pricing was indeed disadvantageous to consumers, and that deregulation would promote innovation while also creating significant investment opportunities. The breakup of the AT&T monopoly has been hugely beneficial. Telephone charges dropped dramatically, there has been a proliferation of new telephone companies (both small and large), and all sorts of new services have been introduced under pressure of intense competition. One could argue that this has been a catalyst for the Internet explosion: prior to the AT&T breakup it would be hard to imagine a scenario in which a single market might be served by literally dozens of Internet service providers, as is now the case in many parts of the country.

The success of the government in breaking up monopolies stimulated interest in extending this approach to the electric power industry. Could the lack of competition be driving prices up, giving dirty, old power plants a free ride, and forcing the public to underwrite poor environmental practices or inept financial planning on the part of regional power companies? Moreover, as the era of leveraged buyouts began to wane, there was growing attention to the immense financial value locked up in traditional, mainstream businesses, and the investment community looked at the power industry and saw gold. Faced with these pressures, Congress acted in 1992, passing a series of legislative measures called the Natural Energy Policy Act. This legislation permits individual states to take actions that will restructure the American power industry into a lean, mean, competitive market. The long-term impact will be international: restructuring in the United States will include the possibility of importing power from Canada and Central or South America, and is triggering a rethinking of power industry structure in many other parts of the world along lines similar to what is being done here. In the short term, change is more advanced in some parts of the country than in others, with California and the Northeast taking the leads.

The broad outlines of the future power industry are defined by this legislation, although it will take at least a decade before all the details are filled in. In the new world, electric power generation will be handed by companies that compete for business. These companies will typically come into existence when an existing monopoly sells off its generators, or investors move into a region and build new generation capacity in the area (new styles of generation are also expected, involving small companies that produce power only under certain conditions, such as when the price is very high, when the weather cooperates, or when a heat wave triggers unusually high load). Power delivery is also expected to involve multiple players: as I write this book, I can only obtain gas and electric here in Ithaca from New York State Electric and Gas, but by the time it goes to press there should be at least two (and perhaps many) companies competing to sell energy in various packages within this region. Finally, much as in the telephone industry, there is a question of ownership and access to transmission lines. The thinking is that long-distance power lines would be owned by regional non-profit companies called "Independent Service Operators", or ISOs, while local lines would be owned by local power companies under some arrangement that guarantees competitive access.

After restructuring, the electric power industry will include a type of commodities trading exchange, for buying and selling electric power, modeled upon the New York Mercantile Exchange (NYMEX), where contracts on commodities can be purchased and sold. NYMEX had begun to offer contracts on natural gas in 1990, which has traditionally been supplied side-by-side with electric power, so the extension of this model to electric power seems natural. The expectation is that electric power markets would operate a bit like an auction, with generators offering various packagings of electric power, and large consumers purchasing them either for resale to individuals, or direct use (in the case of small cities, factories, or similar entities). Such a market will have at least two modes of operation: a 24-hour market, where power can be purchased a day ahead of time, and a 1-hour "spot" market where power can be purchased at the last minute. It is intended that these markets support long-distance sales: if the high capacity transmission lines are in place to carry the power, one can easily imagine a Canadian hydroelectric power generator competing to sell power in Long Island, where local costs became very high when the Shoreham reactor project was cancelled.

However, as Massoud points out, it is not clear that electricity meets all the necessary criteria for commodity trading. The original assumptions of NYMEX and its traders were based on the model of natural gas, which, unlike electricity, can be stored economically. Once a unit of electricity is produced it must be consumed almost immediately; however, whereas a true commodity can be stored for some length of time and consumed when and how desired. Electricity storage devices are capable of handling only a small percentage of an area's electricity requirements. Storage limitations and capacity constraints on inter-regional transfer prevent all available suppliers across the continent from head-to-head competition. An alternative, and more entrepreneurial, view is that furnishing electricity is a service to the end user. Electric service may be segmented into more specific markets such as heating, cooling, lighting, building security, etc., or combined with other consumer services such as telephone, cable TV, Internet connections, etc. Both views may be reconcilable by separating the product, handled by generation and transmission companies, from the service, performed by distribution companies. Deregulation in the United States currently assumes that these two "ends" of the equation (power generation, and retail delivery) are the ones that will become competitive, with long-distance power transport being treated much like highways: a form of government-provided infrastructure.

Restructuring emerges from societal pressures, and is as much a cultural phenomenon as a technical one. The changes that the electric power industry is now experiencing may well be inevitable consequences of the evolution of the industry over the preceding hundred years, and the worldwide environmental trends. Yet whatever the fundamental reasons driving these changes, the technical side of the equation is seriously lagging.

I became interested in the technology controlling the electric power grid when I participated in a 1995 DARPA summer study study, investigating the survivability of the nation's critical infrastructure. DARPA is the United States government military research organization (Defense Advanced Research Projects Agency) and at the time, the program managers organized annual summer study groups to look at questions of potential importance to the military, using the findings to define new research programs. Our summer study was focused on the security and reliability of the computer networks and software that support various forms of nationally critical infrastructure. I'll say more about this broad question later, but as part of our review of the situation we touched upon the challenges confronting the electric power industry. The topic caught my attention, and I eventually became a member of a research consortium funded by the government and the Electric Power Research Institute to study new options for control of the restructured electric power grid. Both groups started by educating themselves, asking experts from the industry to come and talk to us about the trends and the challenges associated with restructuring. Like any student, I listened and took notes and asked questions and, slowly, learned about the situation.

The restructuring of the power grid poses a remarkable number of technical challenges. For example, nobody knows how to build an electric power auction system that would on the one hand encourage competition, but on the other ensure that the supply and pricing of power for the consumer will be fair, stable and cost effective. There is a proposal on the table that the ISO should function as a sort of clearinghouse for power contracts, regulating the auction and also rejecting contracts that might expose the power grid to instability in the event of a problem like a line failing or being hit by lightening. But nobody knows precisely how the ISO would do this.

Here's another mystery. Demand for power fluctuates during the day, and from season to season, depending on how many people are using air conditioners, whether or not the local steel foundry is running its furnace, etc. Electricity producers currently handle such problems by "load-following", which involves producing more power if the load rises and cutting back if the load falls. It turns out that these conditions are very easy to detect, because when the supplied power is less than it should be, the line frequency (normally 60Hz in the United States, and 50Hz in Europe) drops slightly, perhaps to 59.9 Hz. These small changes are measurable, and generators respond by cranking up their output just a little. If too much power is produced the line frequency will rise, leading generators that follow the load to cut back. For example, a coal burning plant might change the amount of coal it is burning each minute, a nuclear plant might adjust the positions of its damping rods, or a dam might allow a bit more or a bit less water to pass. In the limit, if the load rises so much that all the generating capacity has been absorbed, the operators who run the grid as a whole can shed load by causing local power outages or brownouts.

The electric power grid has some properties that make this sort of load-following easy. An important one is that electricity flows in at all the generators and comes out everywhere. There isn't any dedicated connection between generator A and consumer B: the electricity all mixes together inside the grid, and we all end up sharing it. A second characteristic of the grid is that, within any region, the line frequency is guaranteed to be the same at every place it is measured (this has to do with the physics of electricity, and you can just take it for granted). So, if my generator notices that the line frequency is a bit low, I know that your generator just noticed the same thing. We can react in a coordinated way by just agreeing ahead of time on who will produce more power, or cut back, and when. And we don't have to discuss the matter to behave in a coordinated manner: because we base our actions on the same line frequency, our actions are automatically coordinated.

But restructuring dramatically changes the load-following problem. Consider a Canadian hydroelectric producer that negotiates a contract to sell power to Corning Glass in New York. Such a contract requires that when Corning increases its use of power, Canada steps in with the extra juice, and the New York power pool can safely ignore the whole business. The technical issue is to figure our how to make this work. It isn't hard to see that the old scheme no longer makes sense: when Corning's giant furnace goes online or offline, its impact on the power pool will be visible to every generator monitoring the line frequency. Yet, only the Canadian producer should respond to Corning Glass' power needs; otherwise, Canada would be getting paid for power actually produced by other generators. With thousands of load-following contracts in place, we get a very complicated picture, especially when you consider that Corning might not actually consume the power for which it contracts (and, if something breaks, it may not want to do so). In this case the Canadian producer would not want to generate that power, since generating power is costly – it burns coal, pollutes, and so forth.

So our goal is to find a way to match the amount of power produced to the amount being consumed. One way to solve this problem is to propose that some sort of centralized monitoring system gather information from every source of load and every producer, and also maintain a copy of every contract. Periodically, it would compute the right level of production for each generator, and then send some sort of electronic message to the generators telling each one what to do. But such a solution would raise a number of problems: we wouldn't want the entire national grid to be dependent upon a single control system, because if it was incapacitated (perhaps, by a fire), the whole country might lose power. Yet if there are multiple such centers, how should they coordinate their actions? This is an example of a problem that nobody knows how to solve reliably and securely. Load-following is just one of many such challenges; they also arise in the overall area of protection, handling failures, managing the on-line auctions and controlling the ISO, regulating voltage, and so forth.

Another approach might be to have Corning Glass telephone the Canadian producer and to tell it, perhaps once every few seconds, how much power it will need for the next few minutes. (Actually, it would also need to tell a number of other things about the current status of the contract, because this information is also used to control the protective relays that are used to avoid overloading lines in the event of a problem). These days, we solve such problems using computers and instead of making a bunch of telephone calls, they talk to one-another over a computer network like the Internet, but perhaps isolated from the public Internet that we all use for email and Web access. Here, there are a number of other challenges: we would want our solution to be very reliable, so that even if some part of the network or some computer crashes, the power grid will be correctly administered – otherwise, a computer crash could potentially trigger a regional blackout or some other serious problem. We'll need to know that our solution is secure – otherwise, hackers or terrorists might gain access to the electric power control network and disrupt it. We need a solution that "scales", a technical term that means it "continues to work well even if the number of computers using it becomes very large." As you may have noticed, the Web doesn't scale all that well: sometimes it gives very poor responsiveness when you try to go to a popular Web site. Well, we certainly can't control the electric power grid in such a haphazard way!

Solving these problems will require new approaches to computer communications. Indeed, it would seem that restructuring demands a new kind of computer network – a network designed to look much like the Internet, but dedicated for use by the power system, and with built-in mechanisms to provide the security and reliability features needed by the application. On this network, load and production capabilities could be published and in our load-following scenario, the Canadian producer would track Corning's need for power by watching the numbers. But while this is easy to say, actually doing it is much more complicated. Building network software to be reliable and secure (even when things go wrong) is a very hard problem. The industry will need to find solutions to dozens of problems like this one as the restructuring process continues. While it might be easy for a legislator in Albany or Washington to say "the grid will implement such and such a competitive sales model", it is quite another matter to figure out *how*!

So here's the technical dilemma: the industry is confronted with a legal mandate whereby the electric power system must evolve to support a competitive market, offer load-following contracts over long distances, and so forth. Unlike the current grid, which lacks any sort of computer network, doing so will require that the grid have a fairly sophisticated network, over which we would run software to solve the kinds of problems just listed. We know a lot about building secure, reliable software, but very little about mapping this broad knowledge to the specific needs of the electric power industry. Experts view the problems to be overcome as very difficult ones that may take years of research and experimentation to solve. But restructuring isn't even waiting for the technical issues to be identified. On the contrary, the industry is legally compelled to restructure without delay, because its reputation for foot-dragging led Congress to intervene and set the timetable. Now we face the perplexing problem that Congress doesn't want to hear about problems, because it doesn't trust the industry, which is perceived as having a vested interest in preserving its monopoly status.

In effect, it would seem that Congress, trusting technology and competition and distrusting the industry, is performing a massive experiment upon the United States: we're tossing out everything that was previously known about electric power provision and replacing it with something radically new, a complete mystery to those of us (including me) who are supposed to develop it, and doing so on a rapid schedule created by lawmakers with no real technical basis for any of the deadlines. The picture is a snapshot of the times in which we live:

- *An industry slow to evolve.* The impetus for restructuring emerged as much from a failure of the industry to change as from anything else. In this particular case, it seems sensible to speculate that the industry was slow to change because it was profitable and complacent. Not surprisingly, this reduces the credibility of industry representatives who argue against change. In effect, the lack of evolutionary change reinforces a perception that the industry is populated by extremely backwards, unsophisticated relics of a past civilization. But meanwhile, technology has advanced at Internet speed, which seems to be not much slower than the speed of light! Technology is revolutionizing just about everything else. It seems obvious that if we just take some of that technology and apply it to the power industry, there should be a tremendous potential for all sorts of improvements. Moreover, given the success of deregulation in so many other industries, it seems natural to extend the process into the electric power industry.

- *Pollution and greenhouse gases: an unacceptable status quo.* Even if Congress were to accept that we face serious challenges in evolving the grid in this manner, there are even more serious worldwide challenges associated with the status quo. Global warming is a potential worldwide catastrophe, and averting the worst consequences of this trend simply demands that the power industry reform itself. The United States is by far the world's largest consumer of energy, on a per-capita basis, and one of the world's most serious polluters. We need to change, because the status quo will ultimately be disastrous.

- *Financial incentives.* Restructuring presents important financial opportunities to new players in the emerging market, for the consumer (in the form of lower energy prices), for existing utility owners, and for potential new players. Consider a utility that carries some huge financial problem, like a nuclear power plant that needs to be decommissioned and has been losing money for a decade. The utility can spin off its profitable generators into new companies, freeing them from this financial burden, while leaving the nuclear plant behind in a shell corporation, which theoretically could use the cash windfall to dismantle the nuclear plant. Or, perhaps, it would simply pay out the money as dividends to stock holders and end up going bankrupt. Senior managers would have all sorts of freshly-created jobs to chose among, like running those healthy spun-off companies, and perhaps even raising money down on Wall Street to build new power plants. A big opportunity emerges, which is to build specialized generators designed to produce and sell power to the grid only at times when rates are high. The appeal of this sort of thing is, as one might imagine, considerable. Of course, the bankruptcy scenario raises legal issues concerning responsibility for these "stranded costs" and it seems likely that lawyers will have a lot of work to do in the ensuing struggle over who pays the bills. But when you look at the bottom line, everyone involved in actually making these decisions stands to make a killing.

As I write this chapter, restructuring has encountered a few setbacks: as mentioned earlier, California, in the winter of 2000-2001, suffered a series of near blackouts due to load surges and lack of generating capacity, and prices spiked more than once to record levels. The industry cites these events as proof of the wisdom of restructuring. Their reasoning is that such things merely highlight the legitimacy of the new markets and business opportunities that motivated restructuring in the first place. With so much money to be made, investors are certain to rush in and provide the necessary capacity in cost-effective ways. Yet such arguments have a curious ring – they sound like a form of circular reasoning – and for the moment, California consumers seem to be paying more, not less, for their power. Worse still, it wasn't long ago that Californians could feel confident that electricity would keep flowing under all conditions short of a massive earthquake. Flickering lights are suddenly the norm, much better than in some kind of third world country, but worse than prior to restructuring. Clearly, we've hit some bumps in the road.

So, what will happen next? I'm no better at rubbing crystal balls than anyone else, but I do work in the field of computer networking, and as far as I can tell, the technical problems appear to be solvable. A serious question concerns the "best" way of solving them, and it is easy to believe that some major outages and new forms of power supply instability (such as staggering price spikes) awaits us. Yet it also seems plausible that the industry can in effect "hack together" solutions to the various problems that are faced. That is, the engineers who actually operate the power system on a day to day basis are likely to solve the problems they encounter in an unstructured, ad-hoc manner under pressure of time (and politics), because the inertia built into the industry has made it implausible that more time would yield a better result.

Such an approach has pros and cons. Studying a problem forever is a good way to appear to be doing something while basically maintaining the status quo, whereas smart people can often build a good solution (hacked together or not) simply by diving in and doing it. Yet when we talk about massive software systems, the experience with hacked together solutions is really not all that good.

Later in this book we'll look at some examples of the failures, but for lots of reasons, software systems that are extremely complex and that solve very demanding problems tend not to work well by accident. The designers need to know why the system will work well, and why the system will continue to work well even if something breaks, something comes under heavy load, or something happens in the environment around the system, like a big fire or a major storm, to say nothing of earthquakes or other massive disruptions. Inattention to such matters invites serious setbacks: more of the kinds of disruption we've seen in California, perhaps including some major blackouts or other large-scale instabilities in the grid, triggered by inadequacies in the software. Hacked-together solutions are also insecure, with their complexity leaving all sorts of openings that invite future terrorists to move in from the comfort of a workstation on the Internet in some remote place like Afghanistan or Iraq. With more time, and with systematic attention from an invigorated research community (a point worth making, because research on electric power systems hasn't been a hot item these last years), the problem might well be tractable. But if we move too quickly, we won't have time to figure out how to solve the problems in the right way. A close friend and colleague of mine, Robbert van Renesse once observed, "there is a great difference between a computer system that works, and one that works *well*." I think this sums it up pretty well.

Yet electric power restructuring creates a competitive situation, with many players, and even if the sophistication of the initial products and proposals is lacking, one can imagine that market forces will quickly reshape the industry into a lean, mean, competitive one. What remains is the question of fixing the blame: if an Auckland situation were to arise here in the United States, leaving New York without power for a month or two, how many people would die and who would be to blame? But deferring such worries for the future, it seems plausible that competition will have a positive impact by helping weed out the poor technologies and fostering investment in the most effective ones.

What I find astonishing, and probably without precedent in the history of technology, is the degree to which the United States seems willing to wager its economic future on a bet (perhaps not even a bad bet) that technologies needed to control and operate the restructured power grid will be developed as needed.

There is no particularly good technical reason for believing this, yet the issue is completely ignored by the media and the political community. The developers of the putative new system are viewed as inept hayseeds if they complain that they don't know how to solve the problems being posed, so they keep their voices down and point vaguely to other technology areas (such as stock markets) to buttress their hypothesis that the problem can be solved. Perhaps so, but perhaps not, and even if it can be solved, perhaps this hurried ad-hoc process will deny us the best possible solution.

Also curious is the sense of being on a one-way street here. When I describe this problem to some of my colleagues in computer science, the reaction is that the industry must be crazy, and that restructuring will simply fail. But once a utility is broken into multiple companies, there is no simple way to reassemble the pieces, and it seems unlikely that the past as we knew it will ever again be the way that we operate the electric power grid. Instead, we've bravely launched ourselves into a new world, with our best guess at how it might work, a few good ideas for how to go about building the thing, and the conviction that if we just roll up our sleeves, the problems will yield to good-old American know-how.

Note: As I wrote this chapter, California Governor Gray Davis announced an ambitious proposal to re-regulate the California power industry. The feasibility of reversing course is debatable, as we've seen. More to the point, however, is that we're focusing here on the *technical* challenges of solving problems like restructuring the electric power grid, not the *political* ones. Even if California significantly changes its plans for carrying out deregulation and restructuring, most of the questions raised in this chapter will still arise, because they reflect the technical side of operating power systems differently than in the past. No matter how power is bought and sold, and no matter who sets the rates and pays for "stranded costs," the system still has to be operated safely and reliably, and as the industry advances, the questions we've posed here will still need to be answered!

A Technology Revolution

he restructuring of the electric power grid is just the tip of the iceberg. We find ourselves in the midst of an unprecedented revolution in the roles of technologies, and especially of computing and communications technologies. Explosive growth of the Internet and the emergence of web-browsers and web-commerce has, seemingly overnight, permeated the commercial world. Companies are doing business over the network: computers talking to computers with barely a human in the loop. Web sites like "Napster" and "Gnutella" are transforming the music industry (publishing isn't far behind, with Amazon and Barnes and Noble battling to sell all the world's books through Web browsers, and eBooks starting to show up on the shelves), medical computing systems are poised to revolutionize everything from the local primary care physician's office to the regional network of hospitals and insurers, and the list goes on. We can file taxes on the Web and our refunds show up by electronic funds transfer, we buy things using credit cards. Many people never see more than small amounts of cash; for them, money has been replaced by e-money, stored by computers and transferred with the ease of e-mail. Computing and computer networks are transforming our generation, much as electricity, the telephone and automobiles transformed earlier ones.

But the ambiguous record of technology should give one pause. Most things that the scientific and engineering communities develop bring a mixture of benefits and problems, with the benefits most evident at the beginning, and the problems more clear only after a transition to large-scale acceptance. Some tradeoffs are basically tolerable: cars pollute but not many people trade their cars for bicycles. Others less so: it was only after DDT became popular in the 1950's that scientists understood its tendency to accumulate in the food chain, reaching toxic levels in many animals with an especially big impact on birds, but with serious implications for humans too: DDT accumulates in mothers' milk. Ultimately, there was no alternative but to impose tough environmental laws. Farmers turned to other ways to fight bugs, and songbirds are slowly returning. Breast-feeding is safe again. But the episode did real damage. Will this wave of computing be like electricity or telephones, or less fondly remembered?

Much has been written about what might be called "technology-in-the-small", by which I mean the kinds of computing systems most of us use in our day-to-day work and lives. This would include the computers that sit on our desks, telephone systems, fancy computer-controlled televisions and even microwave ovens. New devices are pretty amazing, and they can be a lot of fun to play with, once you have them set up properly (often the most difficult step in the whole process). But computing in the small, and the trends associated with new technologies for direct use, are pretty remote from the topics on my mind, and we won't say much about them here. Instead, I want to focus on "technology-in-the-large:" big projects that use computers and networks to do ambitious things that impact lots of us all at the same time. More specifically, I'm interested in the kinds of projects that use technology for what our society might view as critical purposes: running the electric power grid, or managing a medical computing system, for example.

The list of critical uses of computers and networks gets pretty long if you care to make lists; off the top of my mind I would include things like running the military (a very complex function that involves all sorts of critical sub-activities), controlling the emergency part of the telephone system (the 911 system), disaster coordination systems used when fighting big forest fires or responding to other emergencies, the computing support for the international banking and financial system, air traffic control systems (there are a few of these handling different aspects of the problem, but we'll lump them together for now), the computer systems that clear social security checks and handle such government functions as Medicaid and Welfare, and the systems that actually operate massive life-critical equipment like big airplanes or spacecraft. Depending on what you do in life, you might want to add some additional items to this list: perhaps, weather prediction, or the systems that run the stock market, or some form of eCommerce. And of course each of these is really more of a category than a single thing: the international banking system is composed of multiple subsystems that each handles some part of the job. But we won't need to get into that degree of detail here.

The trends are well established now: more and more of these big societal activities are moving to networks and becoming highly automated. This is partly a good thing: I'm personally rather fond of web-based interfaces to bank accounts. But these trends also pose some surprising risks: what if we build a new generation of critical care systems for hospitals, and they don't work very well? For example, they might crash at inconvenient times, or have serious security flaws. Obviously, while we put up with some degree of unreliability in our desktop computer systems – if the thing freezes up, you can always reboot it – the situation is more complicated for a medical critical care system. The trends, though, have so much momentum behind them now that we might find that we can't easily back out of the decisions being taken now. Thus if the world commits itself to a new generation of, say, electric power systems that depend on a specific style of computerized control, either we need to find ways to build those controls, or we may find ourselves shivering through a few tough winters!

While one expects and tolerates some flaws in the little computer systems that we use in the small, it may seem counterintuitive to talk about flaws in large computer systems of the sort I've listed here. And, indeed, some kinds of large systems work much better than the small systems of which they are composed. But this isn't always the case; many kinds of complex systems are more like your car – lots of things can break and for quite a few of them, you may face an expensive repair before you can next drive the vehicle. The corresponding phenomenon for a large societally critical function would be a period when, say, the United States Social Security Administration Web site is down and people can't apply for benefits, or a period when the electric power grid is down because the computers that are supposed to control it have somehow gotten "snagged" over a failure, or a problem of some other sort.

As I write this book in early 2001, such problems still seem fairly remote: most of the most important government uses of computers and networks remain speculative and futuristic. But the decision to move in this direction – to build such systems and to become dependent upon them – has in many cases already been made. Who makes such decisions? This is an interesting questions; with technology, we'll see that the buck doesn't really stop anywhere. The trends that may determine important aspects of our future, such as the feasibility of keeping secrets or the ability of the military to actually control our armed forces, are somehow self-perpetuating, driven by the same cycle that brings us endless new releases of computer systems and software to run them. Why this is happening, and how engineers charged with actually building such systems deal with the expectations imposed upon them, are the kinds of questions that fascinate me, and I want to look at some of them here.

In this book, my interest isn't primarily technical, yet because my expertise is very much grounded in technology, I want to try to avoid drawing conclusions about subjects on which I find myself on uncertain terrain. For example, there has been much speculation about the impact of technology on social patterns of behavior and even some suggestions that technology can play a major role in triggering episodes of violence by young people. To me, this is plausible – anyone who has actually played with the modern generation of extremely realistic video games will realize that these have a remarkable impact on the psyche of the player: the droning music (if you care to call it music), the constant edginess of the scenarios presented, the graphic scenes of body parts flying and blood spurting. I'm told that designers compete for the greatest accuracy in their depictions of spilled entrails. How could such technologies *not* impact their users –mostly, impressionable boys at precisely the age when alienation and restlessness drive so many into rebellious behavior?

Yet to speculate in this way is risky: lacking factual evidence pointing to a causal relationship, one is all too likely to cite subjective information and from it, to reach unwarranted conclusions. In the world of scicence and technology where I live, explanations need to rest on a hard core of facts and conclusions, and we need ways to verify things experimentally. A premise should be, if not indisputable, at least reasonably credible!

There is another equally invidious trap, which is to portray technologies and progress in negative terms. It seems obvious that the overall picture for technology is a complex one: most things we do are beneficial to at least some people, and quite frankly, very few technologies have much potential to hurt anyone. Of course, it is hard to predict the impact of a technology, and there is no doubt that many of the new technologies have an unparalleled potential to disrupt. But is this a bad thing? Traditionally, progress has been a *good* thing, especially here in the United States, where the entire economy seems to be built on a kind of continuous ferment that endlessly introduces new kinds of products and new commercial efficiencies. I'm not fond of chain restaurants and tasteless tomatoes, but after all, nobody forces me to eat in McDonald's, and as for the tomatoes, it would seem that I'm not alone in my reaction to the insipid supermarket varieties; tomatoes with real taste have reappeared in the stores.

But neither is technology necessarily good, particularly where it impacts very large numbers of people or where it plays an explicit role in public sector activities like electric power generation, medical care, air traffic control or government services. When these kinds of activities are developed, they have consequences that effect more or less everyone, and this can amplify technical deficiencies in a remarkable manner. In what follows, my real interest is in this amplification effect and its implications. The basic question is this: is it appropriate for government to use and think about technology in the same terms that we as consumers do? And if not, what are the implications of the differences?

While the digital tide continues to rise, I would argue that it is only just beginning to lap at our feet. To what degree do we really *depend* upon technology in a day-to-day sense? Certainly, one can enumerate all sorts of uses of technology in the world around us; technologies pervade just about everything. And yet they are mostly unobtrusive, hidden around us, but without rising to the threshold of massive, pervasive controlling influence. If you think back to the Y2K debacle, when pundits breathlessly predicted the end of the world as the clock relentlessly counted down, it strikes me that the lesson was as much one of our surprising independence from technology as not. Anyhow, while it is easy for a newspaper reporter to ask that we "suppose that all the computers were to fail simultaneously," such a scenario defies credulity. When the Y2K event finally did occur, quite a few computers had minor problems, but the average computer rode the thing out with little, if any, signs of stress.

In a classic essay he titled "The Tragedy of the Commons", Garrett Hardin wrote in 1968 about the perils of a self-interested form of capitalism where each of us labors purely on the basis of unenlightened self-interest, seeking only to maximize our own rewards. This form of society can actually thrive, Hardin argues, when resources are unlimited. But inevitably, natural resources reach their limits and the time comes when my flock of sheep competes with your flock of sheep for grass and water. Suddenly, the ecology of our little village collapses – the "commons" are denuded – and the very social mechanisms that previously drove the village through a long cycle of growth and prosperity emerge as its downfall. The essay was emblematic of a trend well established in the 1960's: one of government intervention to sagely guide trends that, left unregulated, were seen as damaging. Yet since the 1960's we've also learned a great deal about the fallibility of government as a guiding force. The problem with Hardin's perspective, fundamentally, is that even if guidance is needed, at the time it is needed one can rarely agree on the nature of the problem, the appropriate form of intervention, or even the likely consequences of intervening.

One can easily argue that with technology today, we are witnessing the emergence of a form of digital commons. The essential point is that we live in a world surrounded by intangible information resources: a sort of foam of digital bits that plays varied roles in our lives, doing everything from delivering email to the computer or carrying telephone calls to keeping planes flying and running insulin pumps for diabetics. Below some critical density, the argument would have to run, use of technology should ideally follow the path that maximizes self-interest: I select the stereo that best matches my life-style, or buy the computer system that I find easiest to use, and this creates a global competitive pressure that constantly improves products, hopefully in ways that please me so much that I eventually give in and upgrade these systems. But as we cross some magic threshold, the formula shifts: by all picking Microsoft Windows for our computers, we accidentally promote an unimaginable economy of scale and market power for Microsoft, and suddenly, Windows is all that anyone can find, for any purpose at all – even critical ones, like running the computers in a cardiac intensive care unit. I don't know about you, but waking up from a heart attack to see a blurry Microsoft Logo on the heart monitor at my bedside might be enough to trigger a second attack!

So here we see a phenomenon akin to the one Hardin discusses: while the self-interested use of technology did great things for us during the past few decades, perhaps we stand at the threshold of a new situation in which technology-run-rampant poses a kind of generalized problem that impacts everyone. For example, suppose that it takes a few years for California to get the power grid back to normal. Meanwhile, from time to time consumers will pay whopping bills, and their electronic devices (including big ones like refrigerators and television sets) will suffer damage from voltage fluctuations and the like. Add all of this up and we're paying a hefty surcharge to obtain the eventual benefits of restructuring, although in the same period of time some of the companies and investors involved may do very well indeed. So here we have a series of events very close to the kinds of things Harkin describes, with computer network technology down at the core, running the show – computers and networks that the power utility companies purchase from the same places you and I buy them from, programmed using the same tools people use to build new software to run on Windows, and inheriting the same limitations.

On the other hand, is it really Microsoft's fault if their products appeal to lots of customers? Moreover, the world has lots of critical computing applications; shouldn't this represent an exciting market for some future entrepreneur, who realizes the futility of competing directly with Microsoft but spies an opportunity here to sneak in through the back door, offering a much better computer system to the world's power systems and hospitals and banks, and then as the scale of his own market increases, perhaps even daring to offer it as a better computer system for everyone else? Why shouldn't market forces tend to solve such problems? Can we understand enough by studying the trends to distinguish between these two plausible world-views and perhaps to reach an opinion about which one is more credible? More to the point, are there ways that our society can gently steer through the digital shoals, reaping the benefits of technological progress while avoiding the reefs?

In what follows, I want to approach this broad issue by sharing some observations about the current trends and the ways that engineers have responded to them. My focus is on "big" rather than "small" technologies – systems that do things like supporting air traffic control, as distinct from the latest and best web-enabled telephone. And in fact my focus goes beyond this to look mostly at activities with broad societal consequences or that represent explicit government initiatives, rather than things we do as individuals. I do this because such projects quickly reveal a disturbing pattern: most, if not all, seem to be embarking with great enthusiasm on trajectories that lead perhaps not to disaster, but at best to compromises and easily foreseeable problems. We all depend upon electric power and air traffic control and medical care, and we share an interest in seeing these kinds of societal activities operate smoothly, reliably and efficiently. So it falls on us all, I think, to try and understand what drives these trends and to try and shift them to a firmer footing. Otherwise, we'll all need to put up with the mess government tends to make of such things when it moves forward enthusiastically but improvidently.

Your Flight May Be Slightly Delayed…

first became interested in the trends that drive technology (and our uses of it) when working on a completely different subject: I was curious about what is being called a "crisis" in computer reliability and security. As a computer science researcher, most of my career has been concerned with techniques for building very reliable computer networks, which continue to work securely and correctly even if something in the network breaks or comes under attack. This is the sort of computer system one wants in settings like hospitals or banks.

As one might imagine, not much of what we play with in research settings ever makes it into real-world systems. In fact, much of what we do academically is completely impractical. Just the same, working in this area of research does tend to heighten one's consciousness about the limitations of the computers and software available for home or office use. It is hard *not* to feel a bit concerned when one considers the tremendous variety of critical applications being moved to networks built using mass-market products. As Leslie Lamport, an early researcher in the area commented, "A computer network is a system within which you can be prevented from doing your own work by the failure of a computer you've never heard of, which provides a critical service you weren't even aware you needed!" One wouldn't expect to see an air traffic control project running on an unmodified network of this sort, yet this is precisely the trend. Moreover, the trend extends well beyond air traffic control, to include gamut of computing applications for which failures or insecurity could be extremely dangerous. They permeate our environment; unremarked, yet omnipresent.

During the 1990's, the U.S. government became increasingly worried about the vulnerability of the critical computer-based infrastructure in this country. As mentioned earlier, I was invited to participate in a study of the problem (although the study itself has been long-since forgotten). We investigated, were duely alarmed, and reported our findings to Dr. Anita Jones, then Deputy Secretary of Defense for Research and Engineering. Our findings helped set a research agenda for the area, but I was left with the sense that we had only scraped the surface, and I set out to research the topic for a book. The question led me far from the original topic.

More than anything else, government interest in the subject was triggered by a crisis that arose from an attempt by the Federal Aviation Agency to develop a new generation of air traffic control software for the United States. This project failed, very expensively (more than six *billion* dollars were lost), and catapulted the issue of life- and safety-critical uses of technology dramatically into the public eye.

The FAA's plans to upgrade American air traffic control systems date back at least to the early 1970's, but the question came to prominence in August of 1981, when President Ronald Reagan fired 11,350 striking air traffic controllers (almost 70% of the workforce). Administration officials asserted that with improved automation, the need for human involvement in the air traffic control process would surely be reduced, and that the skies would be safer than ever, with fewer controllers working far more efficiently. Indeed, the system was confidently predicted capable of handling steadily increasing numbers of flights, and decreasing spacing between them, all with fewer people in the loop.

IBM's Federal Systems Division (later spun off to Loral) won what was then the largest commercial contract in history, and set out to build a modernized air traffic control system using state of the art computers and networking technologies. However, the people in charge of the project misestimated the complexity of the task and the time needed to complete it. Within a year the effort was already missing major deadlines, and ultimately had to be scaled back dramatically, delivering on almost none of its goals, and leaving American air traffic controllers in a terrible fix.

At first glance, the FAA's fiasco would seem to be managerial – poorly informed managers devised unrealistic plans for the project and when these eventually failed, the mounting delays and cost overruns jeopardized the project. This is more or less the official position of the US government on the situation. By fixing the blame in this manner, the government implicitly suggests there was nothing seriously wrong with the technical scope or premises of the undertaking; that it would have succeeded if it had merely been managed better.

One can find some support for this view by comparing the American project with one in France, which has been quite successful. On the other hand, the efforts differed in many ways. The French project started after the American one, but from the outset was much less ambitious. Whereas the American effort undertook a top-to-bottom redesign of the entire air traffic control system, the French decided to limit their effort to a small corner of their existing system, and worked just to replace the controller consoles with little groups of computer workstations, designed so a team of controllers can jointly manage a sector of the sky. Other parts of the system, such as the radar and the computer that manages flight plans, were left untouched. As I write these pages, a new project to upgrade a second major part of the system is only just getting underway.

The contrast between the French and American experiences suggests that the American project suffered from a sin of hubris: the FAA undertook an overly ambitious project, and stumbled both by mismanaging the effort and by overreaching. Following this chain of reasoning, one now finds two flaws in the American project. The government can be seen to be at fault, for having set such ambitious goals in the first place. At the same time, we can criticize the technical leaders of the development team, who should have been in a position to recognize unrealistic goals and to challenge them at the time the effort was initially bid by IBM and its commercial partners.

But the evidence is ambiguous. The engineers associated with the project complain that they could have delivered the FAA's desired system, had the requirements not evolved more or less continuously during the lifetime of the effort. In their eyes, the project simply didn't hold still long enough to permit them to deliver a working system. For their part, officials associated with the FAA point out that from the very beginning the project anticipated the need for some number of iterations, because until air traffic controllers were given access to a prototype, some of the practical issues associated with using the new system couldn't be evaluated (the French system went through at least two iterations of this nature, factored directly into the project schedule). Like any complex computing project, early prototypes of the American system suffered from serious design problems. Some give and take is important; otherwise, the engineers might deliver a solution that "works" in a legalistic sense, yet is difficult or even dangerous to use. Practical considerations obvious to controllers who need to use the technology day to day may not be quite so evident to the designers, most of whom have no real experience of air traffic control.

The picture gets even muddier if you ask technical questions. Very briefly, the computing problem at the core of the American project is as follows (I say "American" because the French system isn't identical and this particular problem doesn't arise in the French system). The software inside the AAS can be imagined as a row of desks at which technicians sit, each charged with doing a complicated calculation. The desk on the far left of the room is next to the window, and the technician sitting there has the job of looking out at the window and jotting down the position of all the planes he can see in the sky (this is what the software inside a radar does). Every few seconds, he passes the current list of positions to the table on his right, where a technician matches up these coordinates with aircraft tracks (trajectories) already known. The next table takes a list of trajectories and matches these with aircraft identification information from the little radio transponders associated with most large planes. Yet another technician looks up each plane in a big filing cabinet, makes a copy of its flight plan and any changes or recent instructions, and adds it to the accumulating folder about this aircraft track. Another technician extrapolates, asking where planes will be in the future if they follow these paths without changing course. Finally, at the far right-hand side of the room a technician takes all of these kinds of information and shows them to the air traffic controller. Perhaps that last technician also warns the controller about any important actions – this plane and that one, for example, will come too close if they continue on these trajectories for three more minutes. From left to right, we want the whole chain of events to take no more than a fraction of a second, since a controller's decisions must be based on the current situation, not the situation from some time in the past.

Of course, what I am describing as a job done by people is really done by computers, and I've taken liberties, but the basic idea is right. Now, to make this process reliable, we need to worry about many issues, but a dominant one is to somehow deal with crashes or other kinds of failures. For simplicity, we can imagine a crash as being a situation in which some technician leans back in his chair just a bit too far, the chair tips over, and the poor fellow finds himself lying on the floor, perhaps unconscious. Naturally, this wouldn't be a planned event, like taking a coffee break. Moreover, assume that technicians never look left or right – nobody can actually see that this fellow has "crashed[1]". Of course, they might still infer that there is a problem somewhere in the pipeline by noticing that outgoing paperwork is piling up, or that incoming paperwork has suddenly stopped. On the other hand, it isn't obvious that just because paper is piling up, some technician is on the floor. The problem is that technicians work at variable speed, so fits and starts in the pipeline are completely normal. How would one distinguish the two cases? Are papers piling up because the next fellow just crashed to the floor, or because he happens to have fallen a bit behind on his paperwork? Is the problem with the next guy in line, or somewhere further

[1] When I teach a class about fault-tolerant computing students are often surprised to realize that in computer networks it is genuinely hard – one could say "impossible" without really exaggerating – to reliably detect failures. Obviously, if a computer crashes, restarts, and then tells the other computers in the network "I failed but I'm ok now", we can trust the resulting failure detection. But if we need to react to a failure while the computer in question is still down, we always run some risk of being fooled by some other kind of problem – something hangs temporarily, or the network temporarily gets disconnected – that can mimic the symptoms of a failure, and yet the affected computer hasn't really crashed. In a technical sense we can try and finesse this by saying that if a computer isn't responsive enough we'll treat it as faulty no matter what the reason. But now your computer might decide that mine has failed, while mine is convinced that yours has failed. If you and I are air traffic controllers, this might lead us to both try and control the same sector of the airspace! So, people who work on topics like fault-tolerant computing for air-traffic control must be very careful to set practical goals, specify their assumptions in great detail, and to prove the adequacy of their methods. This isn't a kind of problem for which a solution can be hacked out in a late-night session some evening, unlike many other kinds of problems that really *can* be solved in a caffeine-induced haze.

downstream, or upstream? One can even imagine that sometimes, a fallen technician leaps back to his feet and resumes work, while in other situations, there might be a long delay. In effect, technicians are unable to detect one-another's failures.

To make the pipeline reliable, we'll need to somehow replace these errant technicians, but without certainty that a crash has occurred. We can simplify the job by making some assumptions – a typical one being that at most one such crash will happen at a time. With this in mind, one might double up the technicians: create a dual pipeline, with a second person doing the identical tasks in each stage. Now, without getting technical, let me just point out why this might be a very hard thing to do in practice. If the technicians are really identically duplicated, one might worry that they would lean back under identical conditions, falling to the floor simultaneously, and denying us the increased reliability we hoped to obtain. But if the technicians are not identically duplicated, perhaps they won't do the identical things, leading to chaos: not all the decisions in a computing system are cut and dried ones, and when something slightly ambiguous happens (is this trace from one plane, or two, one right behind the other? Is this a flock of birds, or a plane?) they might reach different conclusions. Timing problems could occur: what if the two halves of the pipeline get wildly out of sync?

Solutions to problems such as these normally take the form of rules. We could introduce some rules by which pairs of technicians periodically exchange status reports to ensure that they never become uncoordinated. Perhaps, before taking certain actions, they wait for one-another's confirmation (although one has to be careful with rules like this, since, after all, you want the whole scheme to keep working even if someone drops to the floor in the middle of such a dialog). But now we risk getting too technical. So I hope you'll trust me when I say that problems like this can be solved, under various assumptions and with lots of limitations, but that it gets fairly complicated and sometimes, the solutions have the effect of slowing things down. This last aspect is worrisome, of course, because our basic goal is to maintain the snappy response times that air traffic controllers require for safe management of the airspace.

Now, if we return to the actual AAS project, it turns out that from a technical perspective, while *most* engineers associated with the project argue that it could have succeeded, *others* dispute them. For example, Dr. Philip Thambidurai was hired in 1989 to head a team to evaluate the reliability of the system. Thambidurai quickly discovered that the project was working with a computer network built from off-the-shelf components (little different from the computer networks used in most offices or homes). Although reliability was a prominent goal (indeed, *the* prominent goal), no engineering effort had actually been invested on overcoming the kinds of failures commonly seen in these complex systems! Instead, the designers pointed him to some relatively academic research on the subject, which addressed more or less the problem just described (and in fact did it more or less by duplicating each stage of the pipeline), but had never been used in a real setting. Thambidurai posed literally dozens of questions about precisely how this work could be applied to the emerging computer system design – but nobody had answers. He wrote memos to the project management team, but they, too, went unanswered. IBM's workstations and computing software work pretty well, as such things go, but standard IBM networks are still far from the reliability needed in an air traffic control system.

Air traffic control systems normally require 24-hour operations, and must guarantee that controllers will experience no more than 10 seconds of system downtime in 10 years! To gain some sense of how stringent such a requirement is, consider that the network used in my department at Cornell is down a day or two each month. All sorts of things can break and bring the network down, and the software itself sometimes has problems too (in fact, the software seems to fail much more often than the hardware). Now, I can always get a cup of coffee if my computer isn't responsive, or try rebooting the thing, but an air traffic controller needs continuous computing support to do his or her job safely! This is particularly important if, as is still planned, the load per controller rises substantially to take advantage of the productivity benefits of the computing system. So, if we want to use standard computers and software to support a critical application such as an air traffic control system, we need a way to automate the handling of what might be called "routine failures". These include things like wires being kicked out of sockets, computers crashing, or software that wedges up.

To achieve very high levels of reliability, the FAA needs to keep the system as a whole running even if some small parts of it goes down. Thambidurai became convinced that the system they were building was quite likely to suffer from the same sorts of mundane outages that plague my computer network at work. In fact, the problem was even more severe than in a normal office network, because an office computer user doesn't really notice brief outages of 15 or 30 seconds at a time, and these are rather common. But in an air traffic control setting, outages of more than a second or two at a time are dangerously disruptive. This was the core of the problem: Thambidurai discovered that the research papers ostensibly explaining how the system could be made reliable were actually incapable of meeting the necessary timing constraints, and that if they were pushed to guarantee fast responses, the reliability would suffer – a fairly deep tradeoff. Worse still, as noted earlier, the research in question was very far from a recipe for taking an existing system and making it ultra-reliable. Yet, the designers apparently expected that if they simply built their system in the "usual" way, the technique recommended in the research paper mentioned earlier could be waved like a magic wand, transforming their unreliable system into a super-reliable one. This was extremely unrealistic, yet Thambidurai's criticisms fell on deaf ears.

Philip Thambidurai -- and many other critics – are convinced that had the project advanced just a bit further, it would have been stymied by the unreliability of modern computers and software. But the issue remains academic, since the system was scaled back so drastically. Indeed, the bigger problem right now is that lacking this upgrade, the controllers are incredibly overloaded and the equipment supporting them is older and creakier than ever.

If we go around the loop yet another time, we arrive at a new set of culprits. Perhaps President Reagan should be blamed. After all, he fired the air traffic controllers and by deciding not to rehire them, set the timetable by which the new system would be needed. Yet the President was just echoing the prevailing sentiment within an administration that also gave us the Strategic Defense Initiative – a sweeping plan to build an umbrella capable of protecting the country against a rainstorm of nuclear missiles. By that metric, the FAA project probably looked dull and short-sighted. Such thinking, in turn, reflects a broader presumption about what technology should cost and what can be done with it.

This kind of reasoning leads to some unexpected conclusions. For example, the underlying pressure on air traffic control arises from air travel: rising levels of traffic are compelling closer spacing of planes and this puts increased load on the controllers. Growth in air traffic presumes steady improvement in the technology of air traffic control. But air traffic is driven by the economy. Does a healthy economy then require that the American air traffic control system evolve to become more and more automated?

Or perhaps the fault lies in the choice of standard, off-the-shelf technologies. Yet there really wasn't any other option. A few decades ago, projects such as this built their own, special-purpose computing systems. But that era has ended. Today, one really can't buy anything *except* computers and networks designed for home and office use. Only mass-produced, mass-market technologies are cost-effective enough to justify the enormous investments required, for example to build semiconductor fabrication lines. Similarly, while it is common to hear criticism of the major "operating systems" as unreliable and bug-ridden, the reality is that there are very few other options, and even fewer likely to be supported for twenty-five or even fifty years. So, if the FAA was freed to build special purpose, ultra-reliable computers and to put special-purpose operating systems on them, it is doubtful that even the government could afford the expense, or that the decision would look particularly prescient a few years down the road.

Making matters worse, there hasn't been much of a market for ultra-reliable computer networks, and the major vendors aren't very enthusiastic about adding expensive features for use by a very small, esoteric community. This is one reason that the successes of the reliability research community have only rarely made the transition into products. Up to the present, such products haven't brought a very high premium in the market, and they are much more costly to develop. Nobody wants to make a big commercial bet on them.

We arrive at a peculiar conundrum. In effect, something is going wrong but it isn't anybody's fault. For some reason, the economic system simply requires certain types of growth, such as growth in air transport, and we expect the government to ensure that the necessary supporting infrastructure will be available: air traffic control, electric power, telecommunications. The government, in turn, presumes that the technology needed can be procured more or less on schedule. The stock market pressures big technology vendors to efficiently respond to their markets, for example by not adding lots of bells and whistles that don't translate to revenue growth. And the project engineers, although perhaps uncomfortable with the overall picture, just try to do their best.

In believing that technology could improve productivity, President Reagan was hardly espousing a radical new philosophy: up to the present, technologists *have* generally been able to respond to these kinds of needs. Air traffic control is certainly a sensitive problem with special safety implications, but at the same time, it has a great deal in common with more conventional problems, of the sort that arise all the time in large banks, military command and control systems, or other big computing settings.

Yet the project failed dramatically. If we discount the incompetence theory, we arrive at a perplexing thought: perhaps society is encountering some sort of fundamental law concerning technology in a capitalist economy: "technology shall get us so far, but no further". If so, we're facing some pretty serious problems ahead, because quite a bit of social planning is based on the sensible assumption that if things have been advancing in such-and-such a manner for a while, absent some significant looming obstacle, they will continue to do so for a while longer! Here, the suggestion would be that perhaps larger scale is its own obstacle, that mere success can emerge as the roadblock to advancing in our use of a technology. In contrast, most forecasts for the future assume that technology can expand without real limits.

Here, let me again make reference to the idea of a digital commons. Suppose that for a wide array of technologies, there are basically two states – two modes of operation. The most familiar state arises when the technology is used sparingly. Here, individual decision making is adequate to yield the best global outcome – we rely on the grassroots phenomenon of individual self-interest to achieve our collective objective. But the second state, less frequently observed but perhaps soon to become more prevalent, is one in which the density and critical role of technology has caused it to play an increasingly visible and vital social function, where a confluence of events and trends creates an urgent social interest in the way that technology is used, and in its capabilities and limits. As we make the transition from the first state to the second, it is natural that there should be some degree of social dislocation: after all, in the past, airlines increased the density of flights without constraint, and air traffic control agencies deployed air traffic control technologies without much trouble. But just as water, chilled beyond a certain point, suddenly freezes and exhibits very different properties, as we reach and then pass this critical density of technology and need, suddenly the properties of the air traffic control system become radically changed.

One might go so far as to venture a basic observation about the digital commons: *technology on the digital commons is characterized by complex interdependencies, such that cause and effect can no longer be treated in isolation for the various processes involved.* Outside the digital commons, we succeed in making technology decisions without much concern for the settings in which the technology will be used, and social decisions without much attention to technology. But as we step onto the commons, this feature of our digital childhood is left behind. Mature use of technology, it would seem, demands a greater degree of control, and of self-control. A complex social process is needed: one as capable of imposing limits on the numbers of airlines and the density of flights as it is of specifying requirements for future air traffic control technologies.

In this view, the problem is intrinsic, like the risk of pollution associated with using fossil fuels to power vehicles. But I could be wrong. Perhaps it is just going to take us much longer to develop these kinds of systems than we ever imagined possible. Such an outcome would be almost as troubling, however, because our system seems to understand "faster", whereas "slow down, dangerous curve ahead" is quite another matter.

Can we learn from the French? They were more modest in their goals, placed engineers at the top of the government side of the effort (American government hires bureaucrats, but the French have technocrats), and tried to limit the functionality of their system at each step in order to contain the scope of the project. One can and should wonder what it is about French culture that made it possible for them to approach the problem this way: the opposite of the American "style." Yet we should also keep in mind that the French have not been leaders in the Internet revolution. Is it possible that the cultural attributes needed to revolutionize computing are somehow at odds with those needed to succeed in building systems like the one for air traffic control?

If the issue was limited to air traffic control, I suppose we could just resign ourselves to flying less. But air traffic control is just one example among many. Earlier we talked about power systems. To give yet another scenario, consider the challenges of extending medical care over computer networks, so that patients who might traditionally have needed hospitalization (such as elderly diabetic patients) can have a more normal lifestyle in their homes, without the high cost of in-home nursing. Doing so implicitly assumes a networking technology capable of providing services of a very critical nature, although here the focus is more on security against intrusion rather than continuous availability. For example, if a patient were to receive the wrong dose of insulin, or the doctor doesn't see the correct blood sugar measurements, the result can easily be injury or death, but it is hard to imagine a situation where a few seconds of downtime might be dangerous. The designers of such a system need to convince themselves (and the Food and Drug Administration) that their system presents medical workers with accurate information and that the drugs administered will match the doctor's or nurse's orders. All of this has to occur over a network, reliably, securely and with the degree of privacy legally required for sensitive medical records.

Were I writing a crime novel, I imagine how the first scene might play out: " Jill was surprised at how easy it was to press the enter key. She didn't think of herself as being very good with computers, but once past her initial hesitation, this turned out to be child's play. Eight hundred miles away, in Florida, Aunt Sadie– rich, dull Aunt Sadie – suddenly felt dizzy. It didn't take long. And by the time her body was found by the cleaning lady the next morning, Jill was contemplating a lovely Vuitton handbag. She did feel some misgivings: was the bag perhaps just a bit fancy for a funeral?."

Why exactly do we believe that these kinds of computer systems are even feasible? A characteristic of the era in which we live is the presumption that anything within the common experience of technology can be adapted beneficially to more or less any other setting. Yet the actual track record of the industry is mixed. The air traffic control project was a very high profile undertaking and we all depend on travel, but the same pattern of overreaching and then failing can be seen throughout the computing industry. All the amazing successes notwithstanding, one can easily enumerate long lists of failed technical projects. A 1995 study published by the Standish Group suggests that not more than 20% *succeed.* They found that about 50% of big projects limp to completion over budget, lacking critical functions, and late; the remaining 30% are so impaired as to be abandoned outright. Many systems are basically scraped without ever being used. Moreover, failures can be identified within every kind of company and every style of computing – the pattern is ubiquitous. It even extends to the new wave of Internet companies with the zillion-dollar market valuations: the stock market may (briefly) smile upon them, but I would be surprised if even 20% have a technology that works well, and of these, the percentage that could honestly be considered a "breakthrough" is miniscule.

Interestingly, failures fall into broad patterns, a point that Jerry Saltzer is fond of making. Saltzer, an MIT faculty member widely credited for developing many key features of the Internet, has made a study of the ways that complex systems fail. Many unsuccessful systems suffer from what we call "second system effect." This term refers to a situation where a technology project is based on a prior successful project, but so many new features are added that the new system is fundamentally changed from its predecessor. Such situations seem to invite the developer to underestimate the difficulty of the undertaking: they can easily fool themselves into thinking that because the prior system works well, anything based upon it will too. But new features can represent such drastic changes from the original design that in the end, the project is little different from one that started from scratch with no history to learn from at all. A second pervasive phenomenon can be termed the "bad news diode"; this captures the idea that when management is overly optimistic, the developers very often fail to pass "bad news" back up to their supervisors; instead, management bases its thinking on unrealistic assumptions, while the developers labor gamely although fully aware that the project is doomed. Many systems suffer from what might be called an "excess of ideas"; this is a bit like having too many cooks in the kitchen. Although spices are good, more are not always better! Very often, such systems include bad ideas, not just good ones. Books have been written about the "mythical man-month", which captures the observation that if a project is falling behind, adding people often makes the problem worse, not better. The problem here is that the incremental return associated with additional workers is often smaller than the overhead associated with coordinating a larger effort. In reality, very successful technology projects often revolve around a core group of just a few very talented individuals, who need to be left alone to do their work. And then there is a recent trend that might be called the "magic bullet" phenomenon; basically, this involves the emergence of a new technology perceived as revolutionary. As I write this book, the current magic bullet is clearly the Internet and the Web, but there have been many generations of magic bullets. The problem, of course, is that in the early days, we rarely understand the *limitations* of the latest shiny new idea. Why do these patterns persist? As we will see, deeper forces often push a project forward even if it was unrealistic from the outset; many of these common failings of technical projects are simply the outward manifestation of an underlying societal pattern that deserves attention.

Without belaboring the point, one is driven to conclude that a whole series of critical projects are now positioning themselves for major problems of the same sort that the FAA project encountered. These run a big risk of either not succeeding at all, or of delivering something substantially weaker than the application requires. As we look to the future, we can find many other examples where technology clashes with a critical societal expectation relating to safety, privacy, security or reliability, and where the same confluence of factors seem to be driving a technical process in advance of the underlying capabilities.

Suppose that a technology is deficient and that the deficiency is harmful. How is the engineer different from a resident of some huge anonymous city who turns and walks away from a person lying injured on the sidewalk; probably an alcoholic in a drunken stupor, but perhaps a victim of an accident? What should we do when we see an accident ready to happen? Where does obligation impose itself? Keep in mind that technology isn't being led down a path of willful abuse, willful intrusions into privacy and willful twisting of the facts. Yet, within a society, millions of small actions can have a profound impact. And technology has a uniquely amplifying effect. Small decisions literally have a global impact. How often do engineers think in these terms?

Engineering, as a profession, has always struck me as being perplexingly indifferent to ethical tradeoffs. Those who raise questions about the appropriateness of a technology tend to be ignored as whining pessimists, or branded (derisively) as neo-Luddites. Obviously, there are important examples of situations where society has asserted the right and need to "steer" technology development, such as pesticides, drugs, and equipment used to treat medical patients. But the prevailing view seems to be that unless some past abuse had dreadful consequences and compelled a response, a *laissez-faire* approach to regulation and supervision is quite adequate and appropriate. The technologies we've been discussing fall into this category.

What sorts of questions could have been posed about the air traffic control project? Early in the project, the project leaders faced a series of decisions that involved tradeoffs between functionality of the system, safety and reliability of the technology, and the costs and complexity of developing it. By failing to frame these questions as tradeoffs, they failed to strike a sensible balance – or, perhaps, deluded themselves into believing that the technical effort would succeed despite possible delays or budget overruns. Managers and engineers face such questions all the time, but rarely acknowledge the complexity of the forces that guide their decision-making. Very often, complex projects have, at the core, a leader who weighs ambition and the chance for glory against the risk of failure and decides to take that risk. But before we place all the blame on individuals, we should also try to understand the role played by society itself, since the kinds of technical trends of concern in this book are rooted in the demand by society for new kinds of systems. If the societal push is somehow unrealistic in the first place, small wonder that the resulting technology is inadequate.

It is interesting that society is relatively quick to frame ethical questions about medical problems, which often trigger important public debates and foster strong opinions, and yet noticeably reluctant to do so in engineering contexts. Indeed, questions about ethics in engineering are often greeted with hostility. One might speculate that the difference is rooted in the tendency to pose medical questions in terms involving individuals, personalizing them. Issues that impact society as a whole lack a similar immediacy: the individual is unable to imagine him or herself in a specific role and is unlikely to see the scenario as "injust" or to form an emotional attachment to any of the players. This could explain why technology critics are often perceived as having a private agenda: their passion for the issue is otherwise inexplicable.

Whatever the reason, we rarely address ethical issues in engineering; instead, the world operates close to the other extreme. Most technology is developed and deployed purely in response to market considerations, and with total indifference to consequences that cannot be proved without a shadow of a doubt. Short of invoking some sort of repressive government oversight system, would it not be possible to inject some small amount of feedback into the system, to achieve a very modest form of control over the processes we've unleashed, and which are so clearly remolding and redirecting the world around us, and us with it?

Majority Rule

There is a very curious shared element to the scenarios we're reviewed so far, and to some of the ones we'll be looking at later in this book: it seems as if groups of individuals are somehow involved in a large-scale collusion to rip everyone else off, or at least foist off some sort of inadequate technology on the rest of us, often for personal gain. Yet conspiracy theories are always dubious: is anyone really ready to believe that the national electric power industry is organized enough to do such a thing? Of course not! So we need to try and understand why these large projects seem to emerge from nowhere and to structure themselves in potentially harmful ways. And why are all those engineers just sitting on the sidelines, hacking out defective electric power grid control systems and insecure hospital systems? Why haven't there been armies of whistle-blowers warning us about impending disaster?

I want to suggest that there may be aspects of modern American society and culture that tacitly encourage these apparently conflicting trends. On the other hand, this is the kind of shaky ground that makes me as a scientist nervous: how could one demonstrate that our society is somehow causing us to develop risky medical systems on the one hand, discouraging the engineering community from seeing the issue on the other, and basically causing the rest of us to applaud the progress and complain that the only issue is that things aren't moving fast enough? To my taste, this isn't a scientifically well-posed hypothesis, of the type one could experimentally confirm or refute. Nonetheless, it seems likely that our problem lurks in precisely such a dynamic.

Not long after the first World War, Nobel laureate Elias Canetti tackled a similar topic in a book he called *Crowds and Power*. He sought to explain what it was about European culture that sparked the horrors his generation had witnessed. Canetti writes of the differences between large groups of people and individuals, making the point that when people come together in a crowd the joint demographics of the crowd sweep aside individual characteristics, and that the behavior of the crowd is consequently remote from the behavior of the individuals; indeed, he suggests, crowds are easily capable of acts that would scandalize the individuals of which they are composed. He evokes the emotional fever that can sweep a crowd, the loss of individual control, the way that the individual, playing a tiny part, finds that part enormously amplified, for example when by throwing a single rock through a window, the participant in a riot is astonished to see that he has played some role in the utter decimation of a neighborhood. The crowd, Canetti suggests, is neither more nor less than the sum of its participants. Rather, it transforms them, draws upon them, amplifying some behaviors and inhibiting others – taking on a life of its own, uniquely shaped by the culture of the participants and yet unfathomable if one considers those participants outside of the context created by the crowd.

Technology, and attitudes towards technology and science are also a form of crowd phenomenon – a behavior of large numbers, in which our individual comportments vanish into the aggregate, and yet that aggregate might be very different if we, individually, were different. The phenomenon is encountered at many levels in contemporary society. We see, for example, the sense in which cultural demand is compelling the evolution of certain sectors of public enterprise towards greater and greater reliance upon technology. The mass market determines the availability and characteristics of products, and what is the mass market but the behavior of a crowd? Attitudes towards science and technology swing wildly from almost religious reverence to the exaggerated fear and hysteria evident not just in the pronouncements of the back-to-nature and Neo-Luddite extremists but even in the mass hysteria surrounding the Year 2000 phenomenon. The stock market has taken on elements of a mass-market thermometer: any given market sector is valued first by the current societal perception of that sector, with rational business valuations entering the picture only if the public as a whole is relatively disinterested. When a sector is hot, like the Internet sector today, this triggers absurdly high stock prices that only make sense when one realizes that nearly everyone wants to be part of the action. Meanwhile, other sectors that actually have very strong potential languish for lack of mass appeal.

Canetti approached the question through a synthesis of what are fundamentally generalities – he appeals to the common experience, to the likelihood that most of his readers have been part of a crowd, and some of his work is eerily prescient. His interest can apparently be traced to rioting over inflation in Frankfurt in the 1920s and to the 1927 mob-burning of the Austrian Palace of Justice in Vienna. Just a few years after his book *Auto-da-fe* appeared, rampaging crowds tore through streets of Berlin on *Kristallnacht,* when the pathological Nazi hatreds exploded. One finds it hard to imagine how a person of the period could possibly have pretended to be a mere bystander, on the periphery, touched by such a crowd yet not truly a part of it. One would suppose that even the most remote observers should have been horrified. Yet, somehow, the individual managed to disassociate him or herself from the violence of the crowds. Canetti, it would seem, had his finger on the pulse of the Austrian and German society of the period, and tried to understand the roots of mass violence and mass insanity.

There are big differences between the society in which Canetti lived and ours. For example, while large numbers of people may have turned their heads and willfully ignored the gathering storm clouds, great numbers of Germans and Austrians (and others) were complicit in the Holocaust and the events that led up to it. I have a lot of trouble seeing great numbers of engineers as willfully complicit in, say, the erosion of personal privacy that may to be underway now, even if their actions are contributing to that trend.

Yet if one thinks about other aspects of German society under the National Socialists, there are disturbing parallels between what happened then and what is happening now. Then, as perhaps now, society elevated certain principles to unquestioned supremacy, despite their obviously damaging consequences. Our guiding principle seems to be the beneficial role of technology. In the kinds of big projects of interest to us here, the benefit is specifically financial: it permits us to reduce the human role in government and potentially to increase the financial efficiency of providing public services on a large-scale. There are clearly tradeoffs involved: to gain these financial benefits, we seem to be accepting some costs, such as a potential loss of privacy – a phenomenon that we might well look back upon, someday, as being a very high price for the putative benefits of a cheaper, more efficient health-care system or a better way to pay our taxes. Moreover, the current trends disenfranchise substantial sectors of the population, a phenomenon that the beneficiaries of the trends seem to prefer that we ignore. The National Socialists embraced dramatic, sweeping change, massive building and social projects, again finding an echo in the modern embrace of technology. Primo Levi tells us how Hitler's "beautiful words", captivated a society preoccupied by an illusion of moral superiority. Canetti, similarly, tries to capture the sense in which the crowd can be entranced by an idea, elevating it to a level that the individual might never have personally accepted. Are there not elements of a new moral superiority in the language of the technical community, writing of the transforming and revolutionary impact of this or that new technology?

Yet all of this leads back to the question of complicity in evil. The Germans and Austrians of the time surely knew that the new Germany was being inaugurated by an era of unparalleled brutality. Nothing of that sort is happening now. We may come to regret the causal loss of privacy that is being ushered in by the current new wave of technology, but I doubt that even the extremists in modern society secretly welcome this trend.

In March of 2000, an old friend of mine, Bill Joy, made national news by speculating publicly on the question of where all of this might lead. I was a graduate student at Berkeley from 1978 to 1981, a period when Bill basically ruled the place. He was a hacker par-excellence, famous for extending an early version of Unix (a popular system for controlling medium-sized computers) with all sorts of amazing new features. Nearly every day Bill would announce some new version of one of his programs; extremely creative work that took really bold new strides. It came as no surprise to us when he left without his PhD to found Sun Microsystems, a computer company that ultimately dethroned companies like DEC and IBM. Bill was the kind of impatient person who would never wait until tomorrow if he could see the path to some dramatic advance today.

Now a billionaire several times over, Bill found his way onto the front pages by predicting that within another twenty years or so, technologists will have created artificial microscopic life forms, or "gray goo", capable of reproducing and even evolving. He imagines that intelligent replicating organisms might end up competing with us. Bill quotes Ted Kaczynski, the Unabomber, highlighting Kaczynski's concerns that technology will be the agent of some future cataclysm. Interestingly, though, Bill departs from Kaczynski in the near term; whereas the Unabomber believed that disaster has already struck, Bill's concerns are directed primarily to a class of futuristic and rather speculative technologies, based on major kinds of advances that go well beyond anything we really know how to do, or even can see on the horizon. They remind me of a certain type of science fiction.

Bill had much less to say about the current situation; although he frets that modern computing systems really should be more secure and reliable; after all, his own company is a market leader and describes its own products as trend-setters in precisely these respects. Now, Bill is typically very outspoken and quite blunt. I find it hard to imagine that he would hesitate to rock the boat if he saw a real near-term threat, even if his own interests were at stake. But the bottom line is he seems pretty happy with current technology, or at least with the technology from his own company (perhaps less so with that of the competition).

One might expect that the techno-alarmists worried about *near-term* scenarios would mostly be of a sort of left-wing, Neo-Luddite persuasion[2]: roll back the

[2] Concerns that technology threatens society are nothing new. In 1811, Ned Ludd, a "feebleminded lad" working as a weaver's apprentice, damaged two mechanical looms. When Ludd was punished, other weavers resentful of job losses to automation construed Ludd's actions as a blow against the machine, and began an uprising now known as the Luddite movement. The focus was on the damaging impact of automation on English culture and the workplace at that time. But little came of the movement, and 200 years later, very few would call for a return to mechanical weaving. Even at the time, it was recognized that although automation displaced many weavers from the industry, it was also creating considerable wealth, invigorating the economy, and creating many new kinds of jobs.

The Luddite movement presaged the contemporary "Neo-Luddite" movement, which I would characterize as generally antagonistic towards technological advances and modern culture. Adherents range from philosophers calling for the elimination of technology (and a return to simpler days) to activists who practice "monkey wrenching", by sabotaging technology, freeing laboratory and farm animals, disrupting businesses that harvest or develop natural resources, and so forth. It is becoming common for the popular press to attach the "Neo-Luddite" label to anyone who expresses reservations about technology.

machines, trade in our clothing for handsome garments woven out of some sort of hemp, eat brown rice and carob brownies, live in harmony with the wild animals. But in one of those perverse cycles that seem to rule modern politics, the far-left and far-right view current technology trends with equal alarm. Spend time with the most conservative members of the military community, and you'll find them citing many of the same worries to justify increased spending on "cyber defense" and "cyber warfare", terms that refer to new weapons and tools for defending the nation's critical information infrastructure (these people are fond of torturous phrasing and acronyms), and for attacking our enemies of the moment. So far so good, until one realizes that these tools would need to reach into just about everything – to nail a cyber-terrorist, one needs to monitor his actions, so these would also allow the government to monitor *my* actions, and yours. Who are these putative terrorists, anyhow? Is this threat real, or are they simply

---

And so, one reads of "Neo-Luddite scientists" doing little more than urging a "go-slow" approach to biotechnology and genetic engineering, or expressing concerns about the disposal of nuclear wastes – perspectives that are easy to understand and accept. Yet the term is also applied to extremists like Ted Kaczynski, the Unabomber, who initiated a wave of terrorism and murder from his remote cabin the woods of northern Wyoming. His targets were individuals associated with developing and marketing technology. David Gelernter, a Yale researcher whose interests are fairly close to mine, had his hand nearly blown off and lost most of his sight in one eye when one of these went off in his department's mail room. By lumping the former with the latter, it seems to me that the contemporary press reveals a broader societal perspective concerning scientists and engineers. It would seem that we are expected to be resolutely in favor of technology or utterly opposed, so that once an individual criticizes any aspect of technological progress, he or she is revealed as an apostate capable of any heresy. And while this is obviously not the view of very many reporters, I suspect that it is a rather prevalent attitude within the public as a whole.

The roots of neo-Luddism predate the movement itself. The idealization of Man in the state of nature recalls the writings of Rousseau, who in the 1700's speculated about human nature in the absence of government or technology. Galileo was placed on trial by the Church when his scientific findings challenged basic tenants of the religious dogma of the time. The Greek philosophers who invented philosophy and mathematics also reflected at length on questions that relate to science and society. In their debates about the relationship between mankind and government, one can easily find elements of the contemporary dialog about the relationship between humanity, technology, and modern economic systems. But frankly, I have real doubts that it makes sense to look at humanity independent of technologies. We can steer towards a better future, but I find it hard to see how we could go back: paradise was lost long ago, if indeed we ever lived there.

indulging a paranoid fantasy that would reduce the freedoms associated with the Internet without bringing meaningful protections or benefits?

Tibor Janosi, a colleague of mine, grew up under the Eastern European Communist regime, under the watchful eye of a totalitarian dictator. He contributes a rather bleak perspective. Tibor tells me that hope, in oppressive societies, resides in the inefficiencies and incompetence of the government: that, after all, they can't stop you from listening to Radio Free Europe down in the basement late at night, or monitor your conversations with parents and friends. You know which of your nosy neighbors might actually be spying upon you, and you arrange your behavior to respect the utmost normalcy in her eyes. But the same technologies that are granting ubiquitous, anytime anywhere access to information in the West could be applied very differently in today's totalitarian societies: in China, perhaps, or Iraq. Tibor tells me that he shudders to imagine what modern technology could do in such a setting – he imagines his own childhood, but now in a world where microscopic bugging devices might be hidden in just about any setting or material, where tiny video cameras lurk in every corner, and where even the radio has been replaced by Internet radio: information that can be traced to the listener: undeniable proof that he or she was violating the law.

President Clinton, speaking about the rise of networking in China, commented that free, unrestricted access to information would transform that country, propelling it inevitably towards democracy. "Control the Internet?" he asked. "They won't find it easy!" But Tibor worries that the West will present China and similar countries with a gift of an entirely different nature. Could it be that the very technologies that have the most liberating impact in the West (if that is indeed their ultimate impact, something not at all clear today) could represent the ultimate repressive tool in other hands? What terrible irony that would be.

Consider Bill Joy's morbid speculations: *why* does Bill, at the very summit of technical creativity and success, fear that technology might evolve out of control, creating "gray goo" that might become a dreadful menace? I suspect that fundamentally, Bill perceives that we've enthroned technology into such a position of unquestioned dominance that, in ceasing to question, we find ourselves at its mercy. Modern engineering tends to pursue all possible technical steps, then lets market forces select among the resulting capabilities. Perhaps, as Bill suggests, this sort of natural "evolutionary" path has the potential to lead in directions opposed to human aspirations; perhaps technology, limited only by our engineering abilities, might become increasingly hurtful, much as interactive video games, at least within a certain genre, seem preordained to become more and more graphic; more and more explicitly violent. This leads back to such questions as the safety of the future air traffic control system, and the ability of emerging medical record-keeping systems to protect the privacy of sensitive medical data. Quite possibly, even probably, our actions today have the potential to wipe out traditional notions of privacy and some forms of safety that we've come to take for granted. The danger, it would seem, is that in doing the obvious thing by applying technology to important public-sector problems, we are implicitly accepting an erosion of these kinds of social rights. Implicitly, though, in contrast to the situations that people like Canetti and Levi witnessed, where for great sectors of society, the events were deliberate and even welcomed.

As King Lear draws to an end, Lear's son Edgar comments that "The gods are just, and of our pleasant vices, make instruments to plague us." The pleasant vice that seems to plague our own system is in many ways tied to the whole idea of capitalism: ever greater efficiencies rewarded financially by ever improving salaries, productivity and returns on investment. If the system is indeed responsible for all sorts of secondary problems, our eyes are apparently blinkered by the way of life we enjoy. Surely, the most dramatic trend today is the vast surge in the roles and prominence of technology – notably computers, communication, and networking technologies. Perhaps this very technology is leading our entire society down a twisted path into a dark, tangled forest.

Oddly, the demand for technologies that indirectly benefit the public, such as health-care computer networks, is in many ways even stronger than the demand for technologies that we might use directly, like better laptop computers. So, while the portion of the population that might be called computer literate seems to have reached its limits, the community at large still expects that state-of-the-art technology will drive the evolution of the health care system, the air traffic control system, the banking system, and so forth. This public pressure takes the form of demands for lower taxes (forcing greater efficiencies and ultimately, automation), expectations of higher productivity (which determine the ability of companies and municipalities to borrow), and community behavior, for example when people shift bank accounts to banks with larger numbers of ATM machines (for reduced ATM fees and greater convenience).

One could belabor the point by repeating the same kind of analysis in the case of air traffic control, or electric power grid management, or any of a number of societal activities, but the phenomena are very similar in each case. Individually and communally, we experience direct rewards from some form of progress – larger raises, the ability to live our lives longer and more comfortably and under the conditions to which we are accustomed, or perhaps we simply see better profits and hence good performance in our mutual funds, which invest in the industry. All of this creates an irresistible pressure to increase the roles of technology in the associated societal sectors. The technical steps involved, by and large, seem incremental (or at least are so perceived). And so we follow a seemingly preordained path that demands ever-greater use of technology, even if the percentage of the population using laptops has begun to reach its limits.

One might question the incremental nature of some of these developments. The American air traffic control project, for example, had an all-or-nothing character. But the French experience belies any simple characterization of such a project as "intrinsically monolithic." Instead, we should recognize that from the will to the way there are still many choices, some of which reflect cultural styles. To consider the example in question, for example, we need to recognize that the United States is quite different from France in the way that the government approaches technology projects. In effect, we have different "cultures" of and surrounding technology.

In the United States, career government officials are rarely engineers; more often, these are people with a law or business or economics background, who basically are professional managers focused on the application of government regulations to carry out government initiatives and priorities. In my experience, with the exception of people working in the research arms of government organizations (NSF, DARPA, NIH), American government employees have far greater respect for process than content; indeed, they are often caricatured as being fiercely proud of their inability to comprehend scientific or technical issues. I doubt that the problem is quite so simple, but it is certainly true that the upper level officials in most government organizations are more focused on the budget battle and on the legal reporting and oversight process than on the technical decisions being made by IBM or Loral on behalf of the FAA.

In Europe, and especially in France, the situation is entirely different. Unlike the United States, where the "best and the brightest" pursue careers in technology, law and medicine, in France many of the very best students follow a track that combines engineering skills with government, and leads into upper-level government positions. Thus, at the very summit of the French air traffic control effort one finds a small cadre of engineers, indeed, several of the most brilliant of their generation. These are prestigious positions, and even the top engineers within French companies seem awed by the honor of working for and with such people. Yet just as in the United States, these are also the people who fight the budget battles and negotiate the labyrinth of French politics, no less Byzantine than the American one (although less roiled by waves of political appointments after elections, a perennial American phenomenon that greatly disrupts the continuity of major government initiatives and technology projects).

To me, it comes as no surprise that the French air traffic control project was more modest (being more knowledgeable about the problem they faced, the government officials responsible were loath to take on more than they felt they could successfully deliver). The project had far greater continuity and in fact I am not aware of any particularly disruptive management transitions over the course of the effort (I would have noticed, because I visited many times over the years – I consulted with them on the best ways to use some software I had developed for their system). The government expressed a strong interest in technology questions and choices, and pushed back when they felt that a vendor was proposing to do something too complex, or failing to take full advantage of industry standards and trends. Like any project, the French one had its setbacks – for them, the acquisition of Digital Equipment Corporation by Compaq was perhaps the most serious because Digital Equipment was the main supplier of the consoles themselves, leading to worries that the consoles might not be supported for very long. I'm told that this was eventually worked out in a way that had very little impact on the effort. Yet even the likelihood of setbacks had been anticipated and the schedule was unperturbed.

And so I look at this project and tend to see an array of reasons for its success. An important one is the talent of the individuals responsible for the effort within the government. For example, Damien Figarol (who heads the part of the agency that owns the project) was enormously impressive, and he put together a tremendous team of senior technology aides. Figarol managed this group brilliantly – and I would say this even if I had no technical role in the project at all. His group worked in a broader climate conducive to this style of effort, accepting of a more hands-on, incremental approach to the problem, and willing to invest in technology in the long term. In the United States, a person like Figarol wouldn't have lasted long: the prevailing culture of the FAA drives people like him into the private sector. The problem is that here in the United States, government agencies suffer from a tendency to alternate between long periods of inaction and emergencies, when a catastrophe occurs or a Presidential directive shifts the public spotlight. When that does happen, the experts appear out of thin air and, from one day to the next, a massive project is launched, unprecedented in scope and cost, immensely profitable to the companies hired to do the work, and so sweeping that a single big push can completely rewrite the story of air traffic control worldwide, although this particular big push instead collapsed in ignominy. I find it odd, yet American culture apparently demands this sort of lurching around; Congress, renowned for erratic funding of technology and science, apparently prefers it this way.

Stepping back, it is striking that an individual such as Figarol, as well as his role, emerges so naturally out of the broader French culture and approach to technology. This is not the sort of crowd phenomenon that fascinated Canetti, yet it seems just as real, just as much an amplification of the way that French view technology as were the sorts of manifestations of power on which Canetti focused. Similarly, the chronic tendency of American projects to overreach, the occasional dramatic American technical success that propels the whole industry forward, and our inability to respond to a need like the one in air traffic control seem to come as part-and-parcel: all three phenomena emerge out of a broader cultural relationship between American society and technology.

In this connection, it is interesting to look at the American reaction to the Year 2000 problem, which turned out to fit well with the societal predilection for big dramatic problems, and where the underlying sense of impending disaster worked to favor success. Those of us who work in computer science heard about the problem long ago, but nobody paid very much attention until late in the 1990's. The first time I personally realized that it might become a big deal was when Michael Brodie, a senior technologist at GTE Corporation told me about it during a week I spent in Brazil giving a short course. Brodie is a fascinating person – he was once an Episcopalian minister, but he decided to take some time off and somehow ended up at GTE, where he became a senior technology guru. GTE's main line of business is cellular telephony, and it seems consistent with Brodie's personality that he would become fascinated with the complexity of the modern telephone system. The fact is that nobody understands why the worldwide telephone system is so stable – many would say that it is simply a miracle that the thing works at all (consider the vast number of companies and systems that have to cooperate to place a call and bill for it!). The complexity of the system is staggering, yet it works remarkably well. Being drawn to miracles seems to be a prevailing motif of Brodie's career.

When I met him, Brodie was working on untangling some of the complexity of the GTE cellular telephone network. His office is papered with vast diagrams having the appearance and general complexity of an integrated circuit design, purporting to document the interactions between GTE computing systems and the flow of data needed to keep his organization's computing systems running. It was Brodie's job to find some way to simplify the mess.

Brodie himself is fascinated with what he calls the "magic bullet" phenomenon, a manifestation of mass thinking at the corporate level. He suggests that in years of observing technology, he has again and again seen senior management fall prey to a form of utopian thinking, along the following lines. For a period of time, things go well, causing management to become paranoid (Andy Grove, co-founder of Intel, is widely quoted as saying that "only the paranoid survive"). Paranoia, suggests Brodie, often takes the form of suspicion that a company's most dangerous competitor will make a break-through in some technology area that one's own company is struggling with. And so there is a tendency, nearly overwhelming, to react by panic when a major advance is touted in any technology area that seems to bear upon this paranoiac fear. Brodie calls this a "magic bullet" effect – management suddenly summons all the technology people to announce that a magic bullet has been discovered that will transform and revolutionize the company, and all employees will immediately begin to use it. Brodie gives several examples but I won't try to list them here. The Internet is a good case in point, however. Indeed, it may transform companies in a dramatic way, but by the time this has happened, the Web will probably be close to twenty years old. Brodie speaks not of the long-term issue, but of the panic that seems to occur when management first learns of a technology, like the Internet, in its earliest days, perhaps two years after it is invented.

Now, a project that might make good sense as a technology reaches maturity often is doomed to failure if launched prematurely. And so Brodie gives very convincing examples of how management, trying to fire silver bullets at presumed demons and vampires, merely shoots itself in the foot. His point is that a more seasoned management team might gain the benefits of new technology with much less risk of accidental suicide by calming down, realizing that immature technology is nobody's friend, and limiting early forays to simple experiments and proofs of concept.

Brodie was the first to really drive home the significance of the Year 2000 issue for me. Although he was not actually responsible for this problem at GTE, the person who did have that responsibility worked down the hall. Today, long after the event, we all know that it was something of a non-event – the Millennium ended quietly, the lights didn't go out, and the world didn't end. But at a company like GTE, this was the triumphant outcome of a crash 3-year technology project that cost a huge amount of money – I've heard numbers ranging from $400 to $600 billion for the nation as a whole, although such figures are deceptive, since much of this investment had other beneficial consequences. At GTE, the cost was running $50 million per year at the time Brodie told us about it, and was estimated to demand as much as $300 million total.

In the end nothing substantive failed, so one might assume that the Y2K problem was merely a chimera. But the situation actually was pretty dire back in 1997 or 1998. At GTE, they did an experiment to understand the scope of the problem. Company experts isolated their most critical computing systems and set the clock to midnight minus ten minutes, and then let them run. System after system died or malfunctioned when the timer ran down. The Year 2000 problem wasn't a non-problem for GTE. Without any doubt at all, had the company not done anything, GTE would have gone out of business on January 1, 2000. The issue was genuinely a crisis of massive proportions. Moreover, most major companies made similar discoveries when they ran these kinds of tests.

GTE, and other companies, came through unscathed. My guess is that a big part of the problem, worldwide, was eliminated by Sun, Microsoft and other big vendors, which launched emergency efforts to harden their software and operating systems. Obviously, a bug that would shut down Windows on a particular date might genuinely have a very broad, simultaneous impact. Now, even given some assurance that the computer will keep running, a company like GTE also needs to know that its important applications will keep running, and making sure of this, and fixing or replacing the ones that don't, was an immense undertaking. Big companies had no choice: no company wants to risk of going completely out of business through inattention to a looming technical issue. But you can see how the likelihood of a cascading failure drops dramatically once we simply fix the issues that could shut down the computers themselves.

There may actually have been a second technical reason that the Year 2000 problem didn't cause very much disruption. If we step way back and ask what the problem involved, it could be paraphrased as follows. Existing computing systems have become extremely complex and interdependent. We knew that as of, say, January 1998, many programs failed when the dates were set forward and the clock was allowed to switch from 1999 to 2000, and we suspected that our ability to detect these problems was rather limited. Accordingly, many experts speculated that at least some date-rollover problems would slip through the testing, and would then trigger cascading failures, rippling through the network and shutting down huge swaths of our information-based economy.

So, here's a seat-of-the-pants argument in the other direction (by the way, I first advanced this hypothesis long before the Year 2000 event, so I; at least, am definitely not one of the people who predicted the end of civilization as we know it). Suppose that we accept that modern computing systems are far too complex to really be understood, and moreover that the people actually operating and maintaining such systems very rarely even know what they are doing or how these programs work. Nonetheless, over time, the average computer does experience outages, and from time to time, some minor change has to be made and a new version of just about anything has to be installed. When these things happen, that average computer program has an opportunity to malfunction and to trigger some form of cascaded outage. Indeed, as we've mentioned, just about any complex piece of software fails from time to time. So here we have a very complex system and pieces fail, as a matter of course, now and then.

Cascaded outages are extremely annoying to people who operate big, complex computing systems and if one ever does occur, they go to great lengths to prevent the same kind of failure from ever again rippling through their network. My belief is that without intentionally doing so, this set of conditions created a social context that tended to harden computing systems, making cascaded failure pretty unlikely. Obviously, if Year 2000 bugs had not been repaired at all, so many systems would have failed simultaneously that it really would have been a catastrophe. But having fixed most of these "cascading" problems long beforehand, my guess is that the remaining issues were down at the same noise level typical of steady-state operation of these systems. The residual failure rate was probably not so different from the rate of problems seen as programs fail and are restarted for mundane reasons, or software is upgraded in completely minor ways. This is almost like evolutionary selection: it seems that the way software is used in very complex settings selects software robust to perturbations (below some threshold). Programs that aren't this robust just don't survive. So, given the vigorous effort to repair Year 2000 software problems, we apparently pushed the level of disruption well below the limit needed to provoke this intrinsically robust (albeit poorly understood) tangle of mutually dependent systems into failure. Canetti would have been fascinated.

At least as interesting was the public response to the Year 2000 event, which rapidly took on religious overtones. As the end of the Millennium approached, growing numbers of apocalyptic books, movies and television shows appeared. The idea of endings resonated strongly with the religious theme of Messianic judgment and the approaching end of times. And the Year 2000 problem thus was elevated by popular culture to Biblical proportions. Nostradamus, we were told, had predicted computers, and that they would fail with catastrophic effect at the end of 1999 (I wonder if he also gave investment advice?). One of my neighbors mentioned casually that he was stocking food and water for a long siege and confidently predicted that the stock market would collapse, that electricity and water supplies would shut down for months if not longer, and that anarchy would rule the streets. He was anything but alone in this view that our technological society would soon be judged and found wanting.

Not many of us in the field of Computer Science were especially worried about all of this, although I did go through a period of concern when I realized that my brokerage wasn't paying attention to the issue. Happily, as 1999 started, they woke up and began to factor Year 2000 expenditures into their evaluations of investments, and made a sensible decision that the risk was manageable. In effect, they went from doing nothing because they weren't paying attention, to doing nothing but having a rationale, yet I still felt reassured.

It was fascinating to see that when nothing happened, the newspapers were filled with angry criticism of the scientists and engineers who had "deceived" the country. Articles appeared suggesting that 600 billion dollars or so of national investment had more or less vanished into the pockets of computer programmers, beneficiaries of a sort of mass-delusion that they had perpetrated upon the rest of us. The writers seemed almost disappointed that the world hadn't ended. But perhaps they were just angry because the local supermarket was unwilling to accept returns on 5-gallon tins of peanut butter and 100-pound bags of beans.

All of this played out against the background of yet another mass-cultural phenomenon: an immense run-up of stock valuations for companies playing even the most peripheral role in the Internet revolution. The stock market gave the most absurd valuations to companies with no hope for revenues within decades. One company that I followed closely for a while was valued at nearly $8 billion by the market and yet was losing $15 million per year on revenues of about $12 million per year. This kind of valuation is disconnected from any clear financial rationale, unless one looks at the possible value of these companies twenty years from now (such a long time frame that many will have vanished before the bet can be called in), a point that was driven home when, during 2000, the valuation of that company suddenly plummeted by a factor of nearly 20! And so we had the spectacle of preposterous valuations for companies, apparently purely because of a near hysteria by the investing community to get a piece of the action, at any price at all. This was followed by a crash, then a rebound, than an even deeper crash. Why are market forces so disconnected from economics?

We saw a similar issue in quite a different setting. Recall that earlier, we observed the sense in which technology is threatening personal privacy, primarily because the security properties of modern networked computer systems are inadequate to protect the kinds of records now being put online. Politicians have been quick to decry this trend, but of course if the technology just isn't up to the task, the real issue revolves around the decision to put information online, not some sort of callous disregard for personal privacy by developers or operators of the systems (as many would suggest). The fundamental problem isn't technical. We need to ask why the community is encouraging these trends, rather than urging the industries involved to go slowly out of respect for privacy. While it would be easy to say that "we have the highest expectations of privacy", this appears to be another situation within which the *individual's* expectations and those of the *broader community* lead to very different places, for data of this sort.

As individuals, at least in the United States, contemporary expectations for personal privacy have rarely been stronger. We are deeply resentful of perceived encroachments – the one that bothers me most is the flood of unsolicited email I receive daily, particularly the solicitations to visit pornographic Web sites, although I'm not much happier with the invitations to save money on kitchen equipment or hot tips about the stock market. Yet I also realize that our society is tolerant of advertising and views email as just one more medium.

Privacy of my medical records, or personal financial information, is another matter. One has a strong expectation that such things will be treated privately – for you and me, that is, but perhaps not for "celebrities", who find their annual incomes and net worth routinely trotted out for public examination. The courts have tended to support this two-sided system, arguing that by exposing oneself to public scrutiny in choosing a career in media or otherwise pursuing publicity, one also yields a great degree of the normative guarantees of privacy to which the average citizen would be entitled. In some sense, it would seem that for one who wishes to live as a hermit, eschewing all contact with society, society will reciprocate with a strong effort to preserve privacy. The average citizen is protected against intrusive neighbors rooting through the garbage or sneaking around the house late at night. But the guarantee of protection erodes as the issue shifts from one of protection of the individual against her neighbors to information demanded by the community at large. And so President Clinton's sexual exploits emerge as fair game, even the most excruciating, sordid details, and the courts do nothing. I doubt that the police would step in if Madonna complained that journalists were searching through *her* garbage in search of discarded love letters.

Do we actually have a right to privacy for data that finds its way onto the network? Unfortunately, the answer is far from evident and seems likely to be negative, if current trends continue. Merely by collectively demanding the rapid evolution of technology, to the point where technology is deployed in advance of our ability to protect the information that it will gather and manage, such as in a medical setting, we seem to create a weak expectation for privacy. Over time, it seems likely that such weak guarantees become de-facto regulations. If, as a matter of course, one's history of treatment for sexually transmitted illnesses becomes a matter of public record – and if this is true for a great many of us, not merely celebrities – I would be surprised if the courts, with a sigh, didn't walk away from the matter, enshrining the public airing of one's sins merely because the circumstances have already rendered judgment.

The power and impact of mass-market phenomena are pervasive. While they emerge out of our individual views and beliefs, they can also be amplified out of all proportion. It seems plausible that this insight also sheds some dim light on the phenomenon of violence by youths in America, which became very striking in the same time period when these other mass trends emerged, with growing numbers of racist incidents and a rash of shooting outbreaks in high schools. (Paradoxically, during the same period, other forms of school violence declined sharply). Politicians are quick to assert that violent video games and other forms of media are at the core of the school shootings, but perhaps the picture is a much more complex one, where the emergence of these new forms of media are just one element of a very complex story that also involves new modes of behavior and communication, new social groupings and new ways of thinking.

In Western culture and society today, technology has emerged as the dominant force behind cultural transformation. I see this both as a function of the absence of other pervasive issues (there aren't any major wars underway, for example), and because of the role of technology as a vehicle for communication and in delivery of media. Humans are very much defined by communication and interaction, and the nature of the communication reaching us shapes our perceptions and ways of viewing the world. Canetti's thinking about crowds and their tendency to selectively amplify some behaviors while suppressing others surely offers insights into a broad spectrum of modern technology-driven social phenomena. But one of those insights is that there may be no simple way to extrapolate from our individual experiences to understand our crowd behaviors.

Mass amplification of individual experiences seems capable of explaining many kinds of generalized perceptions or expectations, offering a way to understand why how we as individuals may on the one hand contribute to large-scale trends and yet on the other, feel powerless to influence them. And indeed, as individuals, we may be powerless, although our views do contribute in apparently unfathomable ways to the behavior of our society. How then can we hope to sway the course of society? At the risk of cliché, it would seem that the only answer lies in educating the public, so as to inculcate a generalized perception and, through this, generalized pressure in a positive direction.

A Fraying Net

The most dramatic recent technology development has been the sudden emergence of the Internet as a central tool in electronic commerce and business. In the preceding pages, we've pointed to the difficulty of developing secure and reliable network applications, a difficulty that stems from issues with the Internet itself. This raises some questions: why does the Internet work the way it does, and what causes the problems that can be identified within it? Could we build a better Internet? Or perhaps we should be thinking small: Could a major project, like the air traffic control system or a big hospital or the electric power grid build a little *Intra*net of its own, to somehow circumvent those aspects of the public Internet that are hard to fix mostly because of its large size and the normal inertia associated with success?

One doesn't think of the Internet as suffering from the kinds of problems arising from the restructuring of the electric power grid; unlike the future power grid, the Internet is a known quantity. In fact, there were many predictions that the network would collapse under its own weight in 1996 and 1997, but nothing happened. Alarmist predictions come in cycles: more warnings were sounded early in 1999, but nothing happened then, either. As the millennium approached, the pundits must have assumed that between the existing pressures on the network, the surge in use as people went online to watch the festivities, and the Y2K bug, they couldn't go wrong. The predictions took on an air of certainty. Nothing. The network has remained relatively functional while growing enormously. The Internet must be about as good an example of a "scalable technology" as one could find: It keeps getting larger and is exposed to more and more demand, yet it continues to work.

The broader trend feeding the negativism of the press relates to several issues that the average Web user might not be aware of. To be sure, one of these issues concerns the rate of growth, both in numbers of users and in their demand for "bandwidth" – the amount of data that the network is asked to carry. Like computer speeds and capacity, these measures of demand have been growing exponentially; indeed, growth in network use has been considerably faster than the speedup of computers. This disparity fuels most predictions of imminent collapse. Basically, the experts reason that if computers are doubling in speed every 18 months, but Internet load is doubling every 9 months, the Internet won't keep up.

The other issue, though, stems from the observation my friend Robbert made about the distinction between a thing working, and it working *well:* in some ways, the Internet doesn't work very well, and we are witnessing the displacement onto the Internet of a great number of applications that really need the Internet to work well if *they* are to work well. Here, the issue is not so much that the Internet is about to break down in a catastrophic way, but rather that we may begin to expect more and more from it – to require it to have properties that it simply doesn't have and isn't likely to acquire. This second concern is probably the more serious (and perhaps the more legitimate) of the two.

If we focus for the moment on the question of growth, one can easily dispute the sort of analysis that predicts imminent collapse of the network. For one thing, most measures of Internet traffic add everything together, as if all the information in the Internet needed to travel over one communications line ("link"). In reality, the Internet really is like a net and there may be huge numbers of links between your computer and any site you might try to access, so the speed of a single link is just a part of the picture. For each user trying to access a Web site or to send an email, the "route" from that user to the target computer will traverse a few lines, and the performance seen by the user depends on the performance of the links and routing devices along that particular route. Thus, it only makes sense to worry about growth in load when we know that that load *will travel over the same link*. Viewed this way, the rate of improvements in the speed of computer networking hardware seems adequate to avoid severe overload in the foreseeable future. To be more specific: if we consider the rate of growth in the network itself (since more links means more possible routes), the rate of performance improvements in the technology out of which the Internet is built, and the rate of increase in demand, for the time being things seem to be in balance.

Those of us who work with computers use a rule of thumb called "Moore's law" to describe the evolution of the machines that sit on our desks (or in our pockets). In 1965, one of the founders of Intel, Gordon Moore, noticed that computers were advancing according to a surprisingly steady progression. Moore's Law, as it came to be known, was formulated as an observation about the density of electronic components on computer chips, which translates more or less directly to speed and capacity. But, remarkably, it turns out to apply to just about everything, because computing has permeated just about everything.

Moore's original insight was that computer speeds had been doubling every year or so for some time, and were likely to continue to do so. Later he adjusted his estimate to a doubling of speed every 18 months, and over the subsequent 35 years, this rule of thumb has continued to apply. Moreover, it applies not just to computer speeds but also to memory sizes and speeds, communication speeds, disk sizes – you name it. I find it astonishing that anyone could have predicted the future so accurately – just try to predict the future for the company of your choice over even the next year or two! Yet, Moore managed to describe the future of the electronics industry for at least half a century, because we still seem to be on the same curve. His prediction wasn't based on any sort of arcane knowledge: back in 1965, nobody had any idea *how* the industry would manage to speed up the chips and squeeze more memory into them. In fact, most people at the time believed that speedups would come in fits and starts – long periods without change, followed by dramatic, revolutionary advances. But progress has been as steady and as predictable as the seasons or the tides.

All of us have read the comparisons – that if automobile mileage had tracked computer speeds, the average car could drive to the moon and back on a gallon of gas, or the like. Certainly, computers have come a long way. The usual rejoinder is that yes, cars would have great mileage, but from time to time the wheels would fall off for no good reason at all, to start the ignition it would be necessary to turn the key while also depressing the cigarette lighter, that cars wouldn't start with the rear window open, that one would need a different model of car for driving on highways, in the city, or to the store down the road, and so forth. Cars "know" something that computers don't know: speed isn't everything; safety and reliability and ease of use count for a lot, too.

At any rate, Moore's law doesn't seem to describe networks very well. In contrast to the situation for the computers on the desk and the devices connected to them, network connections have generally come in just a few flavors at any point in time: "slow" (telephone modems), "fast" (Internet connections) or "really fast" (the Internet "backbone" links, which carry Internet traffic across the country). The slow connections have improved at a rate much slower than Moore's law over the past twenty years: 1200 baud modems were common in 1980, and 56,000 baud is the fastest you can find today. But the "fast' connections are another story: these have made big leaps, as has the capacity of the shared Internet links that comprise the backbone. Moreover, whereas one used to have to go to work to get one of those fast connections, they are now widely available for users at home. The effect of this is that computer networking bandwidth has advanced in big surges now and then (for example, when fast connections finally made it into the home) but then tended to stay steady for rather long periods.

One striking things about Moore's law is that while it predicts steady improvements in computer capabilities, the various components have all improved at roughly comparable rates. This has encouraged a rather incremental kind of improvement within the computer industry: first, some company introduces an exciting new product, say a spreadsheet. Then as time goes by, more and more great features are rolled out, using up the new capacity of the new computers, but without any sort of radical, revolutionary change. Eventually we all end up needing spreadsheets, and as we install new versions, are forced to upgrade our machines to keep up with the demand for speed and capacity of the newest and best spreadsheet software on the market. I'm sure you're familiar with the story.

In contrast, networking has evolved in a much less continuous manner, with long periods of stability disrupted by sudden, dramatic change. One of those changes is happening right now, as people switch from telephone modems to DSL and cable-modems: Overnight, an average home computer will go from having, perhaps, a 9600 baud modem connection to one that can receive data at 10Mbits/second (the transmission rates will remain low for a while longer, however). Consider that this trend began just as the Web emerged. Could it be that the Web is an application that was simply waiting to emerge as soon as the necessary communications capacity was available? Many of us in the field would argue that this is the right way to understand the recent history of networks.

Meanwhile, a change in the way that Web sites work is reducing the demand for long-distance Internet communication. The idea works this way. In the past, each company would have its own Web site, presumably close to its corporate headquarters, so that anyone using the Internet from far away (in the sense that many network links separate you from the site) would experience long delays. Today, however, many Web sites are actually duplicated with copies on lots of computers – often, rented computers. For example, one popular company, Akamai, makes a living by renting space on its Web servers to other companies. Akamai has a huge number of computers, and there is probably at least one right in your neighborhood (at least, in your Internet neighborhood). Thus, if you try to access a Web site that Akamai is managing, your requests won't have to travel very far over the Internet – at least, in principle. (In practice, things are a bit more complex, but I don't want to wander off on a tangent). Other companies are exploring similar tricks. To the extent that such schemes catch on, aggregated load on the Internet could expand enormously, and yet the traffic over typical links within the network might even decrease because typical users will be talking mostly to Web servers close to their home computer within the network. Thus, even if we view Moore's law as a limit on the likely speed of future computers, it probably isn't predictive of the speed of the Internet as we look into the future.

But rising loads aren't the only reason for the doom and gloom forecasts. Pessimists also cite the emergence of cyber-terrorism against the network itself. Curiously, even during the various cyber-attacks reported in recent years, I've had trouble confirming that anything out of the ordinary was happening. I use email and visit my home page from time to time, and I'll sometimes download an interesting article from MSNBC or the New York Times. Perhaps these Web sites were a little slow, but I often find them balky and slow, even when the network isn't under attack. There were a few times when my home page wouldn't come up at all, but even this isn't really so uncommon either. Nonetheless, the FBI and various major Web corporations assure us that the attacks happened, so I'll take their word for it. In fact, the dirty little secret within my community is that the hackers haven't even discovered the most serious weaknesses of the network! Hackers could certainly shut the network down sometime in the future. On the other hand, few hackers have experienced an FBI manhunt; people who turn to the dark side may end up as fugitives.

It turns out that the fundamental issues – the insecurity and instability of the Internet – run deep. The network wasn't really designed to be very secure in the first place, and as for performance… well, it isn't much of an exaggeration to say that nobody truly understands the behavior of the Internet. The problem is that what we call the Internet is really a hodge-podge of networks (about 50,000 of them last time I checked), glued together like a much-mended fishing net. There are rules that govern the way all of this should work, designed to automatically correct problems seen when a network link crashes for some reason, or becomes overloaded, or something else goes wrong. But even very simple rules, when scaled to millions of computers and thousands of networks, can produce incomprehensible behavior.

One can gain some intuition by thinking about the common experience of driving on a crowded highway. I'm sure you've been perplexed by this too: at some mixtures of speeds and density of cars, congestion suddenly occurs and traffic jams spontaneously arise, even if there haven't been any accidents. Technically speaking, one could drive bumper-to-bumper at 65mph. In fact I've seen a somewhat alarming technology proposal to develop automated highways and cars that would leave the driving to computers. The claim is that computers could handle densely spaced, high speed driving better than people, improving highway capacity and decreasing accidents. But I'm a skeptic; my guess is that people already come pretty close to the limits. If the density of cars rises and the spacing between them drops, eventually you either slow down for reasons of safety, or an accident occurs and everybody will have to stop. Whether people or computers sit behind the wheels, if there is enough traffic, a highway will inevitably jam up.

In a very similar manner, the Internet works really well when traffic is light, but is prone to congestion when lots of users happen to be working all at the same time. In fact, if you were to spend the time needed to really measure traffic within the Internet, you would discover wild fluctuations in performance, localized outages, and surprisingly frequent regional failures. Under any sort of real stress, the behavior is almost chaotic. For long periods, I may be able to download the equivalent of the New York Times to my computer in seconds, but now and then, the network itself may be basically disconnected from the Times Web site.

In some ways this may seem counterintuitive. For example, whereas packing cars close together on a freeway creates obvious safety issues, Internet messages aren't cars. So perhaps you would argue that it really *should* be possible to pack messages tightly together and blast them through the network at the speed of light. The reason that this doesn't work is that the network is full of intersections where fiber-optic "links" come together, and when these switching points become overloaded, they behave erratically.

Actually, erratic is perhaps too generous a word. Unlike a highway, when a network switch becomes overloaded, it will typically throw away lots of messages. Imagine a city surrounded by bridges that, during rush hour, suddenly start dumping the occasional car into the rivers below. In the Internet, if a region becomes overloaded, the odds that a message will manage to get through that region can become very low, because messages take so many hops on their way from wherever they started to wherever they are going, and at each hop, can have the bad luck to be dumped into the drink. This isn't a very graceful phenomenon: the Internet tends to either be working reasonably well, or not at all – when it gets sufficiently overloaded, nothing gets through. On the other hand, for reasons that will become clear shortly, it usually fixes itself pretty quickly, sometimes in just a fraction of a second.

In the early days of the Internet, most applications were designed to immediately resend missing data when loss was detected, the idea being that if my message gets lost, I should greedily try to get a new copy through as quickly as possible. The strategy reminds me of Hardin's description of the Commons in the good old days when we had just a few sheep and lots of grass: self-interest works well with a lot of spare resources. Much as on the Commons, as the number of users rises and they start to bang into one-another, the strategy eventually breaks down.

About fifteen years ago researchers realized that greedy resending of messages that get lost in the Internet was no longer working. Instead of overcoming the loss, the approach was provoking long-lasting regional collapses within the network: a few messages would be dropped because of load, triggering a surge in load when these were retransmitted along with whatever was already in the network, causing even more to be dropped, and so forth. The problem could persist for a long time once established.

To fix this, a community of researchers led by Van Jacobson, who has become something of a guru of modern networking, recommended modifying the part of the Internet called the "TCP protocol," so that when data loss occurs TCP will *slow down.* Applications like email and the Web are based on TCP. Thus, when the network gets overloaded, the applications currently using the network run at slower and slower speeds. The effect is to reduce load on the congested region, which usually recovers a few moments later. I'm sure you've experienced this problem – things are working smoothly when suddenly your Web browser may seem to stop. When this happens, I usually switch to some other Web site. However, a user who really cares – one who *depends* on the network being up – would see such problems as very serious.

This highlights the real issue. Whereas most current use of the network isn't terribly critical (a day trader, who uses the Internet to manage a stock portfolio, would probably disagree), this won't be true forever. We've talked about the electric power grid: the industry may create a little Internet of its own, but that network will still act like the Internet. The same goes for air traffic control and hospitals. The government will soon be using it to run all sorts of benefit programs. Companies are using the network in all sorts of e-commerce roles.

Thus, not long in the future, computer networks built like the Internet, or directly on the Internet, will become as vital as electricity, heat, water or telephones.

Now, you might wonder why these little Internets should be built in the same manner as the big one. Here, the issue comes back to one we've encountered several times in other settings. Basically, the selection of computer equipment in the local computer store pretty much determines what you can build. The electric power industry could, presumably, put out requests for proposals to develop a completely new "Powernet" with totally different properties and hardware, but the responses might represent some very expensive one-time solutions. And this whole way of thinking presupposes a degree of cooperation and agreement within the utilities that operate the power grid; companies that compete tooth-and-nail with one-another for advantage in the market. Thus such a scenario is very unlikely. In practice, anyone who wants to build a network ends up buying the same kinds of hardware (and software to run that hardware) as is used to build office networks and the Internet itself. Sure, the electric power network might not be connected to the Internet (in fact, connecting it to the public Internet would be a pretty dumb idea, although it may be harder to *avoid* such connections than one would expect). But disconnected from the public Internet or not, any network built like the Internet, with large enough numbers of users, and heavy enough load, will start to act like the public Internet.

❧

What will it take to make the Internet secure and reliable? After all, it's one thing for an Internet user to tolerate some security and reliability risks when placing an order for a gourmet selection of French cheeses, fresh from the Loire valley: it isn't a catastrophe if the order page won't open up right now or an order gets registered twice. It would be another matter if a hospital computing network experiences bizarre outages that disrupt patient monitoring or cause a dose of insulin to be injected twice. How do we get from the current network to one that can be used with confidence in critical settings?

To answer a question like this, one needs to start by understanding the community that built the network and that is responsible for it today. Indeed, one needs to understand an entire engineering culture, because the networking community is much more than a set of individuals or a steering board: it's a very large and informally structured collection of participants, some from companies that provide network services or hardware, some academic, and some from the government agencies funding advanced research. One part of this community is a group called the Internet Engineering Task Force, or IETF, which is often described as setting "standards" but actually exists just to structure dialog without any real control.

A real highway is controlled by the highway authority, patrolled by police cars and maintained by professional work-crews. Nobody controls the Internet. The Internet emerged from consensus between the varied players, and consensus still rules the thing. The operators and hardware vendors meet periodically to discuss problems they are seeing and to think about how they can be solved. When a proposal is made to change the network, the associated group will typically record its views in a type of memo called a "request for comments" (an RFC). A process then ensures: there is a period during which comments are collected, then the RFC is typically revised by the authors, but eventually they issue the definitive version without any sort of vote. It is appealing to think of an RFC as a law governing the legal way to operate a piece of the Internet, but unlike a normal law, an RFC has no legal authority. Some are ignored, others are followed by just some of the companies managing parts of the network, and relatively few are ever put into widespread use. Indeed, most RFCs turn out to be controversial and are either ignored or sent back for more discussion and revisions. The fate of an RFC ultimately revolves around the market demand for the associated features and their cost. Compliance is voluntary and there are no penalties for violations.

This may come as a surprise, since the Internet was a government creation, and one normally expects the government to insist upon rules and regulations. Initially called ARPANET, the Internet emerged from a research program created by the government funding agency called DARPA, which at the time was interested in exploring options for linking computers in support of email and document sharing. Back then, email was a breakthrough technology, and people were still trying to imagine other ways of using networks. From the outset, ARPANET was perceived as a high risk, purely experimental undertaking, and the early IETF was basically a collection of DARPA funded researchers who met periodically to discuss their work and to ensure that one person's new experimental application wasn't causing a lot of disruption for everyone else. If any one organization really had control over the ARPANET, it was a company named BBN (now a division of GTE), because they operated the computers that formed what came to be known as the network "backbone." But BBN ruled with a light hand, and thus at the very outset, a culture of consensus became established.

Basic decisions made in the early stages of the network turned out to have important implications today. The first developers never expected that ARPANET would simply evolve over time into the Internet. They were relatively unconcerned with security: everyone knew everyone else, and attacking the Internet would have been child's play (keeping it running was the hard job)! In this culture of mutual trust one finds the origins of the current insecurity both of data traveling over the network and also of the network infrastructure itself. Not many people realize this, but the network is highly trusting of its own components. A person who breaks into one of the computers that "speaks" the underlying Internet protocols can easily disrupt the network in dozens of ways, because these computers currently trust one-another. But we'll return to this topic later.

The most far-reaching decision made by early network developers was that the Internet wouldn't be based on the telephone system. This may seem obvious today, but at the time, the telephone system was the main model for connecting things together – computer A would connect to computer B by dialing to it, and then their modems would convert any data to be exchanged into a form that the telephone company could carry over a normal telephone circuit. This telephony model is sometimes called "virtual circuits" because one can think of a telephone call as establishing a private wire between caller and callee: an electrical circuit. We use the term "virtual" because the circuit isn't a real one; actually, the circuit connecting your telephone to your mother's is superimposed on the same wires (fiber-optic cables, to be precise) that carry my telephone call to my friend out in California. Yet the telephone company treats the call as a circuit, and all sorts of telephony design decisions reflect this. For example, the company carefully arranges that enough capacity is available to carry your speech, in digital form, back and forth. It happens that this doesn't take terribly much bandwidth, but the necessary bandwidth is reserved just for you, and if a telephone link is shared, the telephone company switching system does this in a way that cleanly divides it between the different users so they shouldn't ever be aware of one-another. (The ghostly conversations one sometimes notices in the background are something else, a kind of malfunction unrelated to this business of sending digital traffic from many calls over the same lines).

In contrast, the Internet developers adopted a message-oriented design. In their approach, each computer was given an electronic address, like a street name and address, and if computer A wanted to send some form of message to computer B, it would send the equivalent of a postcard: a small digital message containing B's address, A's address (so that B can reply), and some limited amount of text or data. Over time, for technical reasons, the term "packet" came to replace "message", so the modern network is called a "packet switching" system, and the computer systems that direct these packets from their source to their destination "routers". Billions of packets per second pour through the Internet, routed electronically from a myriad of sources to their many destinations. The routers, which continuously estimate the "travel time" to each possible destination, play an endless relaying game: grabbing an incoming packet, glancing at the destination, and directing it out on the link that should get it to that destination as fast as possible. Think of the Internet as an electronic express-mail service and you won't be far off.

The designers of the Internet made another basic decision at that time. In effect, they decided that since the telephone system worked quite adequately for telephone calls, the Internet didn't need to worry about voice data. Now, digitized voice packets need to be delivered very rapidly and steadily – otherwise, you get echoes and long delays and drop-outs, all familiar to anyone who made transatlantic telephone calls fifteen years ago, before the era of transatlantic fiber-optic cables. But if you set that problem to the side, the remaining applications involve things that much less stringent demands on the network infrastructure.

With this approach came a number of big simplifications. For starters, the Internet is not designed to transport packets reliably. Instead, it simply makes a best effort, like a postal carrier who wanders around collecting mail and delivering it, but is told that if her bag ever fills up, she can dump any excess mail into the next garbage bin she passes. This may sound appalling, but in fact the Internet loses packets all the time. Indeed, those of us who develop software for the Internet think of packet loss as the network's way of signaling overload to applications running on it – the network tosses out some packets, and the applications realize that they are being asked to send data at a slower pace.

If we aren't in any great hurry to get our data through, a computer can hide the problem: anything sent on the Internet is silently and automatically stored at the sending computer until the receiver acknowledges that it has gotten through. If acknowledgement is not forthcoming within a short time, additional copies are sent. Since this is happening at unbelievably high speeds, you never even realize that some of your packets are lost, while others arrive out of order or even in duplicate. Hidden from you, your computer and the sites it talks number each packet, so that any problem can be detected and corrected. If necessary, a new copy of any missing information is automatically requested on your behalf.

Thus, when you access a Web page, you won't see little gaps in the pictures even if the packets that contained those parts of the image got lost in transmission. Your computer notices that something is missing, asks the remote Web site to resend it, and the retransmitted packet is slotted in automatically to repair the gap. If you are looking closely, you might see a brief pause while the screen is being rendered, since this back-and-forth dialog over the network takes some time, but eventually the image pops up, complete and intact. Well, at least, that's what usually happens. Sometimes after trying lots of times to recover a chunk of missing information, your computer just gives up. In these cases you get an error message, leaving you to puzzle over what might be broken. After all, perhaps the cat just knocked the network cable out of the socket, or perhaps something crashed down at your local network service provider's office, or there could be a problem on the Internet backbone, or an overload, or perhaps something crashed in the computer room of the Web site you were talking to. You can't distinguish the cause, because any of them would have the same symptom, so your Web browser simply posts one of those annoying error messages and gives up.

People call this the "end-to-end" approach to network design, because it puts the onus for reliability on the applications using the network, not the network itself. The network is understood to be fairly reliable but not absolutely so. If you want reliability, your computer needs to work with the Web servers and email servers and corporate database servers it talks to in order to obtain the desired guarantees, between one "end point" and the other. The same philosophy is applied to security: if you don't want intruders to be able to read your packets, your computer and the computers it talks must cooperate to encrypt the sensitive data before sending it, and to decrypt it on reception. In fact, there isn't even a guarantee that the sender's address is correct: my machine can send your machine a message claiming it came from some other source, entirely. The end-to-end view is that if you want to be sure who sent a packet, you should use some form of security mechanism (encryption) to accomplish this. This gets complicated and most computers trust the source address, which is why "spoofing" (sending floods of disruptive packets containing fake sender addresses) has turned out to be such an effective way to harass Web sites.

The end-to-end philosophy has limits. If the network becomes severely overloaded, routers may begin to drop so many packets for such a long period of time that the simple-minded retransmission scheme no longer works. Perhaps, the requests for retransmission won't get through, or the retransmitted data will also be lost, and your browser will find itself waiting longer and longer for missing data. Your Web browser would then slow to a crawl, or even freeze up completely. The same behavior occurs if the Web site you are talking to crashes – you'll see an error message on the screen. Dozens of conditions can cause these kinds of failures. Moreover, not many applications implement end-to-end security. By and large, networked computers are very trusting of one-another. But the key insight is that these things don't happen very frequently, so we don't think of them as typical behaviors of the Internet.

Problems of the sort we're discussing here are really phenomena of scale. Suppose that you were to stand at the back of a big room filled with hundreds of people using the Web at a furious pace, all clicking from page to page as quickly as possible. With the Internet working the way it does today, at any given moment, you would see a few people cursing and banging on their machines. From time to time, everyone in the room would suddenly freeze up, all waiting for a response from their computers. From your new vantage point, you would realize that the Internet isn't all that reliable because the reliability problems would be obvious in such a setting. But the *individual* experience is quite another matter. When you surf the Internet from the privacy of your office or your home computer, you are much less likely to be aware of the pervasive nature of disruptions and access problems on the Web. After all, it takes time to read the stuff you fetch, and to scroll around in it, so you probably don't click to new URLs all that often, and you would only notice an Internet problem at the moment of clicking to a new page. It may seem as if a slow response only happens once in an hour or so. Thus, the way we use the Web tends to prevent us from noticing network problems, just as the Web browser itself is designed to repair Internet data loss automatically whenever possible. All of this perpetuates the sense that Web browsers really work pretty well. For the purpose, in fact, I guess they do.

But given the relative fragility of the whole structure, if a malicious person wants to do so, it is relatively easy to disrupt the network or to eavesdrop electronically on the data being sent between any one computer and any other – a sort of digital wiretap. There isn't any way to get better reliability, and if you want stronger security, you need to activate data encryption, so that the contents of the packets you sent are first mixed up in a way that looks completely undecipherable to anyone lacking the secret keys needed to decrypt the data. Unfortunately, except in some special cases (making purchases on the Web is one of them), encryption is hard to turn on because each application is expected to have its own way of managing security keys. Recall that the end-to-end philosophy dictates that security is your problem, not the Internet's job. Thus, the application developer or user must do something to enable security. The task is simplified if all of the pieces of the application run on identical computers, but it gets difficult if the various pieces might be using software from different vendors. In such settings, many systems administrators simply turn the security mechanisms off, rather than deal with the complexity of getting them running on and between such a variety of systems.

Lest we wander off on a tangent, it is important to keep in mind that despite its limitations, the Internet is one of the shining successes of the late 20'th century. Since the invention of the automobile and the telephone, it's hard to identify an engineering project with greater accomplishments or more impact! During the past decade or so, the major focus of the Internet community has been on scaling the network to larger and larger numbers of users, and supporting faster and faster packet switching. In the early days, sending an email from Cornell to Berkeley involved perhaps four or five switching points, each able to handle perhaps a few hundred packets per second, and the entire action took perhaps twenty seconds if the email was small. Today, there are at least twenty or twenty-five switches between me and Berkeley, but many of them can switch hundreds of thousands of packets per second, and under good conditions, a message from me to Berkeley would arrive about one-tenth of a second after I send it. From my home in Ithaca I have the sort of connectivity that only Bill Gates could afford even a few years ago. The future promises even greater speeds.

The deeper issue is not necessarily that the Internet has its little problems, but rather that the technology of the Internet has nearly ceased to evolve. Sure, links and routers are getting faster, but the basic architecture hasn't really chanced since Van Jacobson's changes to TCP, mentioned earlier. Prior to that, the last really big event[3] occurred around 1980, when a completely new way of maintaining Internet "routing" tables was introduced. In contrast, waves of radical reform have swept just about every other aspect of the computing industry with stunning regularity. People talk about "Internet time" to connote the idea of an

[3] The executives of large Internet companies are fond of claiming that their companies are "reinventing" the Internet. But just as you don't really "reinvent" a car by fitting it with oversized tires, there isn't any shortcut to a better Internet in the works today, or even on the horizon.

exceptionally rapid pace of change but, ironically, the Internet itself is changing at roughly the speed of the national highway system!

This is not to say that there haven't been good reasons to make changes. For example, about ten years ago a proposal was made to add encryption to the messages used within the Internet itself to manage its own configuration. Only now, very gradually, is this feature being introduced. Indeed, one of the most pressing problems is that we are running out of addresses for computers connected to the network, yet while there has been a solution available for ages, there is little evidence that it will actually be adopted anytime soon. You can contrast this with the introduction of, say, Microsoft's Windows 98, where millions of period *per day* upgraded to the new software.

Even fixing serious bugs can take an eternity. A few years ago University of Michigan researchers studying the Internet discovered that it was being swamped by all sorts of nonsense messages, such as messages from Ithaca to the world as a whole announcing that "there is no direct route for packets connecting Ithaca and Brazil." This sort of message makes no sense, since Ithaca normally isn't on the Internet route to Brazil, and to make matters worse, the nonsense packets were produced in great volume. The problem was eventually tracked down to a very subtle bug in the software operating certain packet routers. It took a few months for the bug to be found and fixed, but the problem persisted for more than a year because operators of the networks that make up the Internet were slow to pick up and install the necessary fix. So one gets a pretty clear a sense both of the kinds of evolution still occurring, and the pace. Nothing really major has happened in years.

There are two dominant reasons for this slow pace of change. First, unlike the situation for laptop computers or cellular telephones, there have not been any good opportunities to compete by offering a "better Internet." The Internet is already so large – several hundred-million computers are connected to it – that any better Internet would initially cover only a tiny portion of the space. As a practical matter, only computers talking to other computers connected to the better Internet (call it the Supernet) could benefit. If you signed up for Supernet but then used it to talk to a computer on the Internet, that last "hop" would subject your traffic to all the usual issues.

The hardware infrastructure investment in the Internet is already staggering, so even if a Supernet came along, it would be very hard to convince Internet providers to throw away their existing routers and replace them with Supernet routers. There may be opportunities associated with big shifts; for example if all computers suddenly go wireless, the wireless companies could possibly build a nationwide wireless infrastructure that would operate differently from the Internet. Later, in fact, I'll suggest that the government really should try to build a Supernet, since industry isn't likely to do the job, but might be convinced by a proof of success. But for the time being, we'll just see continued expansion of the Internet: bigger and faster, but "unsafe at any speed", to paraphrase Ralph Nader.

A second important form of resistance comes from the Internet development community itself. We mentioned the end-to-end philosophy. It wouldn't be much of an exaggeration to call this a religion. Any proposal for new Internet features or extensions is rapidly abandoned if the end-to-end community determines it to be at odds with the philosophy. As a practical matter, I can't really blame them. Like Moore's law, the end-to-end philosophy has been a huge success. The end-to-end philosophy has become a kind of guiding principle for endorsing or rejecting each new proposal for improving the network and for whatever reason, "end-to-endism" has given us a network that works well enough to support the web, email, and thousands of other important ways of computing and using networks. When you think about it, the philosophy has done an astonishing job: after all, the modern Internet is doing all sorts of tasks never imagined by its developers. So, it is easy to understand why the Internet engineering community would be somewhat cautious about change. If we suddenly set out to redesign the network to work better for some specific purpose, like telephone calls over network connections, we simply break the network itself, or somehow deny ourselves the next break-through application.

Moreover, as noted earlier, nobody really understands why the Internet works to the degree that it does, or why the problems that are seen occur. We lack a really scientific way to design a complex large system like the Internet with any confidence that the result would be an improvement. Or perhaps this is a misstatement. Rather, we lack any methodology for improving the *current* Internet, a proposition somewhat like upgrading the engines on a jet plane while flying it. There is a reasonable chance we could build a better one from scratch, but it isn't certain, and the means of arriving at a better Internet within the current structure remain elusive.

Butler Lampson, a Turing Award winner, spoke to this point in a talk he gave late in 1999. Butler asked why the computer science research community didn't invent the World-Wide-Web (as you may know, the Web emerged from the physics community, which needed to build a form of digital library in support of their collaborations). He pointed out that the basic mechanisms needed to build a Web browser were in place by 1990 if not earlier, and that any of us in the field could have built a primitive browser by simply sitting down for a month or so and hacking. Butler speculated that this didn't happen because the computer science research community has traditionally shied away from problems unless they can be solved in complete generality, with all sorts of detailed guarantees. The essential success of the Web stems from the realization that *"it only needs to work most of the time."* Butler argues that those of us who could have easily build a Web browser probably didn't see doing so as a good research topic, since we would have wanted to build browsers that work really well, and yet when you come down to it, Web browsers really don't work all that well. Perhaps he is right, and this explains why the Web was discovered not by the academic computing research community, but rather by a computer scientist working in support of the physics community, where researchers were sharing tremendous amounts of data and publishing electronically long before anyone else. But many of us still wonder how we missed it.

Butler's point comes with a counterpoint: on the current Internet, one simply can't build a Web browser that works predictably, reliably, and securely. The underlying properties of the network are inadequate, and there is nothing much that can be done to overcome its limitations.

We now find ourselves in a curious bind. The Internet consists of a vast and very mature infrastructure, representing an immense capital investment and operated by hundreds of independent participating companies. Everyone understands the benefits of improving the speed of the thing, so this, at least, is certain to happen. To a more limited degree, the security of the network infrastructure itself will also advance. But a completely different dynamic controls progress towards an infrastructure that might better support very demanding applications, which need strong guarantees of security together with ways of guaranteeing reliability (for example, assurance that a medical computing system can't be shut down by a bored medical intern or by the failure of a packet switch or link somewhere).

The end-to-end philosophy insists that these applications should be on their own: that it isn't the job of the Internet to solve their problems. And the current trends offer us an infrastructure that is certainly faster and faster, but not safer or more reliable, except to the extent that it is less likely that a faster infrastructure will become congested. That observation may sound to you like a compelling argument that the future will bring a network that actually is safe because it will be faster. But the rejoinder is that this is only true if the number and speed of the users remains constant. In fact, the numbers of users and the level of their use is growing even faster than the network is gaining speed, and the speed of the typical connection is also rising. Moreover, many existing applications send data as fast as possible and are just waiting to soak up any extra speed. This tends to counterbalance trends that reduce load, like the tendency to place Web sites close to the user. Going forward, the Internet will groan under greater and greater load, and is unlikely to be substantially more robust than it is now.

I work in this field, but it isn't obvious to me (or anyone else) how all of this will play out. If the Internet were to melt down, fulfilling the most catastrophic scenarios one reads about, I could imagine the government deciding to intervene in some absurd, heavy-handed way: a law, for example, could be passed to require that providers of Internet services guarantee reliability and security. But if the Internet continues to work most of the time, and hacker attacks don't become a continuous disabling presence, it makes more sense to imagine a continuation of the status quo. Anyhow, even if the government tried to become involved, it isn't at all clear how it would do so, since there is no organization controlling the thing! Given a system that more or less works, I doubt that the public or the industry would look favorably upon a government regulatory body like the FCC stepping in, and in any case, the FCC, like the American Federal Aviation Administration with its air-traffic control problems, lacks the technical ability to guide things in the desired direction.

Many people in the field would argue that the best hope for a better Internet comes from the commercial pressure to run telephone connections over the Internet. In fact there are quite a few proposals to do this, and in smaller settings (say, a single building or a University campus) one can already build a single network that can carry both voice and data with good results. But as we mentioned earlier, voice is really a pretty easy case: the bandwidth needed to accurately reproduce speech is tiny compared to the data rates associated with typical networked applications. So a success for voice telephony wouldn't really help very much for an application like a future medical computing system. Moreover, given the huge existing voice infrastructure, any network-based solution would face stiff competition. My guess is that the Internet is here to stay, and that its limitations won't be eliminated any time soon.

As mentioned earlier, some companies have recently emerged with plans to build very large networks of their own, the idea being that all the Web sites of importance to you would be hosted on their computers at locations physically close to your point of access. This way, when you access a Web site, your request won't need to travel very far, and hopefully you'll be spared the vagaries of the rough-and-tumble world of the network as a whole. They need to update these pages and hence end up networking them together, typically using dedicated lines so as to ensure very high performance. A variation on the theme is to offer a very high speed connection from your desktop to the Internet backbone (at a correspondingly high price). As you might expect, this also gives rather snappy performance when you talk to Web sites that are similarly well connected. But solving connectivity problems for Web sites wouldn't really solve it for anything else: the community health information systems we've talked about, for example, involve much more than fast access to the web. So these companies are a positive step for the typical commercial user interested in offering network-based services, but perhaps not for anything else.

What about security? Here, at least, there are some promising initial steps. One involves what are called "virtual private networks." Without getting overly technical, a VPN (the usual shorthand) lets you create what seems to be a private network, superimposed on any sort of public Internet. Under the surface, VPNs work by encrypting packets so that only legitimate users of the VPN can make sense of the data sent within them, and so that any fake packets can easily be discarded at low cost. A VPN would let a medical network run right on top of the Internet, and while this would not guarantee bandwidth for applications like remote monitoring or controlling a remote insulin pump, it would still represent a first step. A mixture of VPN's, faster Internet technology, and a way to create cross-connections into the standard telephone network really might do the trick. I can imagine nicer looking solutions, but if I had to speculate on what the future really will bring, my bet would be on this kind of spaghetti – a torturous mixture of just about everything, but adequate in a patched-together Rube-Goldberg sense.

Lacking in this future, if this is indeed what the future holds in store, will be the kinds of robustness that could protect against a determined intruder who wants to compromise security or bring the system to its knees. Anything as complex as what I am imagining here – and it would be hard to imagine a more complex structure than what you would get by combining the Internet, replicated Web sites, VPN technology and the telephone network – is certain to be fragile and hence attackable by a sophisticated and motivated intruder. After all, even the telephone network is known to be fragile and insecure; it works well mostly because so few hackers understand how to make their computers mimic telephone company control computers. In fact, some have done so. The most famous incident involved a hacker named Kevin Mitnick, who was ultimately apprehended and jailed; he manipulated the telephone system to create illegal wiretaps, steal credit card numbers, and committed all sorts of pranks over an extended period. The more complex a system gets, the harder it is to protect; if my vision were to become a reality, future Mitnicks will find easy pickings.

Up to now I've talked about the technology of the network and left the sociology of the network to the side. The topic is an interesting one, with many ramifications. For example, by creating new forms of connectivity, the Web is changing our traditional notion of societal groupings. Take my own name, "Birman", which is also the name of a lovely variety of Persian cat. It used to be that the owners of Birman cats were an esoteric and far-flung crowd. Today, type "Birman" into any Web browser and you'll find thousands of Web pages devoted to the cute little critters. The Web has created a close-knit society where there was once a scattered group of isolated devotees.

There are positive and negative aspects to these trends: on the negative side, the white supremacist and neo-Nazi movements has never been better organized; on the positive side, perhaps people are just a little less lonely than they were in the past. The Web is creating whole new illnesses: to substance abuse problems, we can now add "Internet abuse." I bet the support groups are already forming.

But while the cultural issue is fascinating, and would probably lead to a whole different book if one were inclined to really study it, it leads away from our real topic here: the role of big technology projects in running the world we live in, day to day. With respect to that goal, the main cultural insight to take away from this quick glimpse of the Internet is related to the way we perceive the network. On the one hand, we have the network as it really is: a somewhat strained edifice evolving under the pressure of tremendously rapid growth, and in many ways, keeping up with the demands. Yet while the Internet works, it certainly doesn't work well, if by "well" we mean securely and reliably enough to support generations of extremely critical uses on which lives might depend. This confronts developers with a serious dilemma: either they should walk away from projects that require reliable networking, or they need to somehow try to build a reliable system around an unreliable network.

To me, the frustration is that the Internet probably could be made to work well. But of course this is a technical comment: as a practical matter, it may be impossible to find a path that gets us from here to there and that the commercial world could ever navigate. A further frustration is that the Internet works just well enough to encourage us in imagining all sorts of ambitious ways of using it. How many of the world's real decision makers appreciate that those occasional Web outages have a far-reaching significance, with implications that go well beyond the minor irritation of being unable to get to the New York Times Web site, leaving nothing much to do except for a trip to the coffee machine? My guess is that the vast majority of casual network users are quite impressed with the capabilities of email and Web browsing, and if asked, would enumerate all sorts of societally critical functions that we really should move to the Web as soon as possible. We have a peculiar situation in which the Internet, for all its flaws and limitations, glistens invitingly, urging everyone to jump right in.

Barbarians at the Gate

One of the more interesting issues in computing concerns the culture of hacking, and "cracking". Up to now, I've used the term indiscriminately, but real hackers are software builders with a special talent for throwing together large, complicated computing systems out of disparate pieces – they hack out new programs, sometimes at astonishing speeds. One cannot but be awed by a hacker's programming skills. Even the term evokes the image of a fearless machete-wielding trail guide, hacking a path through dense jungle undergrowth while battling poisonous snakes and fierce animals. I used to be quite a hacker myself, although the experience of launching a small software company in 1989 pretty much burned it out of me. A true hacker works late into the night, defining and then elaborating entire worlds of his own imagining – literally, because when one develops very complex, large software systems, one conceives of them as a kind of a self-created universe, with actual places within it that can be visited, things that can be done, a sort of elegance or even beauty to the structures. (I'm using the masculine pronoun intentionally: there are women hackers, but most are men).

I'm sure that most people have difficulty with the idea that a program can be somehow visualized – that dry lines of code might somehow evoke a world as strange and complex as any one encounters in daily life, somewhat like the world of a video game or a fancy Web page. Yet not only is this the case, but many programs are really perceived as beautiful (especially by the creator) – vast, sweeping sculptures, with subtle allusions to objects found in the real world. Other programs are profoundly unattractive creations, little more than junk held together by ropes and chewing gum and bandages. The idea that software might be beautiful – particularly the internals of a complex program that normal people only see from the outside, talking to an interface it presents – is a fundamental world-view that separates hackers from everyone else. I think that relatively few people ever achieve this shift in perspective. Even the average programmer never makes the leap to begin to conceive of a program as a world in its own right. To the hacker, a program and the world within which it lives can take on a vivid reality that is in many ways more real than day-to-day life. The hacker can define data structures that spiral through spaces limited only by his ability to visualize; worlds within worlds.

This, I think, accounts for much of the fascination some of us feel for computing. The real world is not under one's own control, but the world defined by a complex computing system is limited only by the artist's vision. Yet the hacker's virtual world is also a capricious place, savagely punishing even the most minor mistakes, rewarding occasional leaps of intuition with completely unexpected behaviors or insights. Programs can seem willful, refusing to cooperate for the most obscure reasons, then suddenly working properly, in a vast leap, such one might feel when piloting a starship as it suddenly warps space to leap across the galaxy. The hacker builds a thing that is greater than himself, and it gains life of its own. He feels the power of creation. He becomes a minor deity.

There are very few hackers, by comparison to the numbers of computing professionals in the world. In my direct experience, I would say that less than one programmer in a thousand is a true hacker – and relatively few people can program in the first place. And so, being a hacker is an elite experience, like membership in an extremely exclusive club. Hackers often come to know each other (through the Web and email, of course – rarely in person) and impress one-another with feats of gnarly programming success. They measure ability by sheer volume of code produced, and by difficulty of the tasks accomplished.

There is a fringe at the edge of the hacker community, populated by a mixture of hackers who turned to the dark side, and by wanna-be hackers who simply couldn't deliver. This community of "crackers" focuses not on creating elaborate software edifices, but rather on their defeat. The term is one that deliberately rhymes with hacker, but evokes the image of a safecracker, because breaking into things is what crackers do best.

The fascination with breaking into complex software systems is, I suspect, somewhat like the thrill that drives the graffiti artist to climb past coils of razor-wire and over electric rails to paint the sides of trains. Most crackers break into computing systems precisely because they purport to be secure, somewhat like Sir Edmund Hillary, who climbed Mount Everest "because it was there." By and large (although there are exceptions), crackers have no criminal intent. They simply can't resist a challenge.

In the south of France there stands a walled town, Carcasonne, that overlooks the plains between the Massif Central and the Pyrenees. Carcasonne was one of Europe's most valuable strongholds during most of the Middle Ages. Although the town came under assault untold times, it never fell. It is the prototype walled city against which all others measured themselves, the very essence of security in a time when crusades and massive armies swept through with regularity. (More recently, the town is also known for its sensational Cassoulet, best enjoyed with a bottle of the local Cahors red wine).

Those who build sensitive computing systems often find themselves challenged by the contradictory need to place their systems on the network, exposed to the masses, and yet to protect the integrity of the application itself or of the data it manages. For the crackers of the world, such a system is an irresistible attraction. Like Carcasonne, it stands on the plain, defying all who pass by to penetrate its thick walls of stone, fitted so tightly that even the cracks between them are nearly invisible, perhaps hiding inconceivable riches. Simply to enter and be able to see the jealously guarded data – this is the cracker's fantasy.

The cracker who approaches such a target draws upon a varied arsenal of tools. Many of these, ironically, were developed by the same systems hackers who build new security solutions. They include password-finding programs that try to log in as some legitimate user by guessing his or her password using vast dictionaries and detailed knowledge of how password checking is done on various existing systems. There are sophisticated computing systems designed to test for great numbers of known security flaws in old versions of various applications and operating systems – easily five or ten percent of all programs that have ever been connected to the network have turned out to have some form of security weakness that can be exploited in an attack, and not many systems administrators are sufficiently diligent to track down and plug every hole. Even the most cautious administrator may be loath to replace a program that has known problems with the very latest version, since this often entails a major upgrade that will impact all sorts of things, and in any case, the newest version of just about any program may have its own problems. In effect, an upgrade can be a bit like rolling a Trojan horse into the castle keep: you merely trade a problem you knew about for one that you won't find out about until later. This is why most experts doubt that any networked computer system can ever be truly secure. In fact, one of the most active military research programs in the security area is focused not on building software to be more secure, but rather on rapidly detecting intrusions and counterattacking! The idea is that perfect security may be unachievable, but a form of "digital immune system" could compensate for the inadequacies.

But this form of digital intrusion detection remains esoteric. To the cracker, that castle in the plain is undermined by tunnels and there are secret passages running through the walls. Some of the stones may be lose, and it is rumored that the wizard Gandalf knows a magic incantation that will open locked doors, and reveal staircases invisible to all but the cognoscenti.

Curiously, many crackers are very weak programmers, and indeed some of the most successful crackers are completely incapable of programming. For example, in their book *@Large: The Strange Case of the World's Biggest Internet Invasion*, authors Freedman and Mann describe a bizarre episode that occurred during an 2-year period starting in 1994, when thousands of computers connected to the Internet were penetrated, including machines used by the Bureau of Land Management to control flood-gates on dams in the North West. The authors point out that from these machines, the cracker could with a few keystrokes have flooded millions of acres and quite possibly caused a significant loss of life.

Should you doubt that anyone might actually want to do such a thing, it is interesting to realize that these specific dams are at the core of a tremendous controversy now, because native salmon are at the edge of extinction in the rivers on which these dams are sited. Were an "eco-terrorist" to gain access to the same computers, I don't think he or she would hesitate for long before enlivening the debate with a preemptive strike on the dams, especially if it could be done in a way that might put them completely out of commission.

But the culprit was no eco-terrorist. In a bizarre twist, he turned out to be a mildly retarded teenager who spent hours each day on an aging computer in his room, and proved completely incapable of understanding what he was doing. Investigators were puzzled when, for example, they tapped his computer line and observed him typing commands that only work on Unix computer systems on all sorts of computers that didn't run Unix, and doing this day after day. But, as they ultimately learned, the boy behind the crimes simply had no idea of what these commands did, nor did he appreciate that there are many kinds of computer systems. He tried everything he knew, downloading all sorts of obscure cracker tools on every computer he could break into, and in this way, slowly and painfully, broke into one machine after another – thousands of them.

Similarly, one reads of rather unsophisticated but effective attacks on popular Web sites, for example by bombarding them with high rates of nuisance requests – the computer equivalent of making a phone call and hanging up just as the phone is answered. A child who has gained access to a few computers on the network could easily launch such an attack, and yet it is hard to protect against and can be very effective in shutting down sections of the network for periods of time. He wouldn't need to understand the tools to be able to find them and to learn how to use them. Moreover, once the perpetrator logs off, it can be remarkably hard to track him down.

Early in February of 2000, these kinds of nuisance attacks were coordinated in a way that crippled several of the most popular Internet commerce sites. As I write these pages the perpetrators had not been apprehended, but many people from the edges of the cracker community have stepped forward in their defense. "All information should be free," they argue, "these crackers are only attacking commercial sites to protest against commercialization of the Internet." This willingness to destroy a thing in order to save it seems unique to the cracker community, and the military.

Of course, there are also criminal crackers, out to steal corporate secrets, extort money under threat of revealing flaws in a computer Web site or credit card numbers of other precious data, seeking to transfer money from bank to bank or to engage in illegal forms of insider trading, and even trying to hurt competitors for business advantage. The government fears a wave of international terrorism: crackers funded by rogue states, working from the comfort and security of well appointed computer rooms buried deep beneath desert hills, who would reach out through the network to threaten the very heart of our increasingly technology-dependent society.

I had a really odd interaction with someone from the White House on precisely this issue. This occurred at the end of the DARPA ISAT study mentioned earlier. At the outset of that study I was just one of the participants, but over time my role shifted, and I ended up with responsibility for presenting our findings, together with some recommendations for a research agenda, in government and military briefings. One of these was a presentation to a group called the Jasons, which originated within the physicists and scientists who created the first nuclear weapons, and then lived on in an advisory role to the government on sensitive technology and science issues.

Now, I should perhaps mention that our group's findings were rather tame as such things go. We concluded that the United States is certainly becoming more and more dependent upon computing and networks, and that the state of reliability and security technologies for these networks was deplorable. But we also expressed skepticism about any sort of imminent disaster. Our position was that there was a problem, slowly but steadily growing worse, and that the government needed to respond by improving the technology base so that it would become easier and easier to make critical systems secure and reliable. While some members of our study group were concerned that a terrorist attack could easily succeed (at least at the time of the study), others were doubtful. Although we identified specific kinds of attacks that appeared to be feasible, the level of sophistication they required from the attacker would be high, and they required that the attacker go undetected until quite a bit of damage was done[4]. Within our group, there was no consensus that the typical attacker could pull this off. We agreed, however, that the more serious risk was of an increasingly unstable infrastructure within which major breakdowns would become almost mundane events.

I arrived early for my talk, which turned out to be a problem. A classified presentation was underway, and because I didn't have the proper clearances, I wasn't allowed anywhere near the auditorium. This turned out to mean that I was confined to the front door of the building and the cloak room for forty-five minutes, until finally a big red light over the door of the auditorium went out and my host came to find me. I gave my talk, which was well received, and then they announced a break for refreshments. I found myself in conversation with a man named Mike Nelson, who was then in the White House Office for Science Policy. Nelson explained to me that "if I knew the kinds of things he knew" I would be far more concerned about the terrorist threat. This dialog occurred shortly after the 1996 Presidential election, and Nelson confided that he had watched the returns with the President and Vice President, worrying the entire time that the national power and telephone grids would suddenly be shut down, the election would shift in favor of Mr. Dole, and then everyone in the room would turn to him – his nightmare was of an unexpectedly defeated President Clinton asking, sadly, "Mike, how could you have let this happen?"

This type of paranoia is difficult to counter. Nelson kept alluding to secret information not available to me (the world's spooky people have a habit of this sort of thing, and one suspects that they also tend to exaggerate the significance of obscure but secret information that they happen to possess, but under no

[4] The Internet attacks of early 2000 proved us both right and wrong. On the one hand, the Internet was not crippled and after a few days, the targeted Web sites were back to normal. On the other hand, the sophistication of the attackers was not very high, and terrorists could easily have done the same thing, or worse. I remain convinced that the kind of attack that could really hurt the country is very unlikely. As the Internet sector of the economy grows, the problem will grow too, but my guess is that at least until 2010 or 2015 the degree of dependency on the Internet will still be low enough so that any form of really serious consequences would remain unlikely. As we look to the more remote future, of course, the potential gravity of an attack rises.

circumstances could share). I gamely defended the position that disaster wasn't really imminent. Nelson and a crowd of hangers-on surrounding him more or less agreed that I was a poor fool and would someday regret it. But as it transpired, at least up to the present, the cyber-warfare threat remains somewhat remote, and people like Nelson moved on to other worries.

Of course, all of this is not to say that the emergence of a technology-anarchist community couldn't occur. In fact, as we saw during the Internet attacks in early 2000, there is certainly a community of crackers ready to attack Web sites or even the Internet as a whole to make a statement. But despite the noise this creates, I think the more real concern would require the emergence of "professional" terrorists. For this to happen in the United States, something would need to cause a great deal of anger within the community of Internet hackers and crackers at the fringes of technology society. Right now, quite a few of those people are involved in the Linux movement and a fair number are actually getting quite wealthy, so I don't yet see circumstances that might motivate them to go underground and become real dangers. Meanwhile, one can imagine remote desert warlords who might be prepared to spend large sums if they could destabilize the Internet by so doing, but it seems doubtful that they could assemble a strong enough technical team to cause more than minor problems.

Yet I could be wrong. For example, I remember very odd confrontation that occurred when I was lecturing in Brazil as part of a short summer course being offered to Brazilian college students in Campinas. Among the lecturers was a fellow named Richard Stallman, who founded and heads a group called the GNU Foundation located in Boston. The GNU people are famous for having built a completely free version of some of the same kinds of software that Microsoft was founded to sell. They distribute the software in what we call "source code" form: you don't simply get the ability to run these programs, you actually get the programs themselves in a form that you can modify, if this is your inclination. People call this "open source", and the idea dates back to the early versions of the Unix system, which was popular back in the 1980's and remains very widely used under the name "Linux" today. Stallman wrote a great deal of software for GNU and Linux, founded the GNU organization, and is a hacker par-excellence, with very strong political convictions.

In particular, Stallman believes in something he calls "copyleft", as an alternative to "copyright." With conventional copyright, the owner of a work retains the license and lets people use it only for a licensing fee. In contrast, the idea of copyleft is that the developer of a piece of software agrees to share it for free, in "source code" form, on condition that anyone who makes use of it and changes or extends it will give back their extensions to the community at large under the identical terms. Linux, the operating system getting so much press these days, emerged from the GNU community, although it was built by people other than Stallman himself. (This bugs him; in Brazil, Stallman repeatedly referred to Linux as "that operating system they couldn't have built without the GNU tools"). The whole notion of open source code with copyleft licensing, has thus created a very substantial worldwide culture and defined a community that includes tens of thousands of hackers.

When comparing the GNU community with the more prevalent commercial software community, it is impossible not to recall the political tensions between Capitalism and Socialism in the 1950's, at the start of the cold war. Microsoft is typical of the capitalists: they make money on software, create jobs, and are extremely protective about their whole line of business. The GNU community are the socialists: they believe in sharing software, and argue that there is something sinful about keeping the source code secret in order to protect one's market. The GNU community has no issue with people making money by selling technology (or at least, if they have an issue with this, it is moderated by their own financial successes doing so), but they have a passionate conviction that the software itself should be available in open source form for all to see, understand, and modify,. Of course, the players in this debate aren't separated along geographic boundaries – both sides are right here around us. Yet in a world that has increasingly seen geographic boundaries give way to conceptual ones, the tension between these communities is every bit as real and sharp-edged as any tension between America and the then-Soviet Union.

In Brazil, Stallman and I got into a rather nasty debate. Stallman organized an evening talk on copyleft licensing and GNU, and I attended. From the audience I asked him how he felt about a completely unencumbered license that Cornell was using at the time – unlike copyleft, ours lets the user do anything at all with our technology, even sell it (or sell modified versions). I intended the question innocently, but I guess it hit a sore point, because Stallman went ballistic. He reviled Cornell as "evil", say that "next to Microsoft, people like you (me!) are the greatest enemies that the free world faces today." He swore that GNU would dedicate itself to developing copyleft versions of all of Cornell's software, so that nobody would ever use our versions again. I suggested that under Cornell's license, he could just make a copy and proclaim it to be subject to copyleft from GNU, but at this he became almost apoplectic. He swore that GNU would dedicate itself to the destruction of groups like mine, and of companies like Microsoft. So absurd did this tirade become that finally, I had to leave the evening session. As for Stallman, I'm told that he subsequently calmed down, having exorcized the devil. He avoided me for the remainder of the week, and I haven't seen him since.

Now, I don't want to suggest that the GNU group, or Richard Stallman, are potential terrorists. Quite the opposite: they have been hugely successful, and in fact many are associated with the Linux phenomenon, which has minted dozens of multi-millionaires. However, the mixture of raw fanaticism and extreme talent does give one pause. Imagine a scenario wherein some significant number of people with these abilities were to turn against the system, go underground, and then set out to attack it in a serious way.

There are some isolated glimpses of what such a world might be like. Dan Farmer, a hacker/cracker/anarchist who is about as tattooed and body-pierced as one can be, is renowned for having developed a powerful tool he calls *Satan,* for breaking into computer systems, explaining that it was offered to the world as a form of "self-test" so that administrators would be able to identify the weaknesses in their systems. But in an interview with *Scientific American* Farmer comments that he is personally fascinated by pain, and I can't help wondering if he thinks of his software in terms of some sort of ritualized mechanical S&M activity: a program to torture other programs? We mentioned Kevin Mitnick, who developed tools to manipulate the telephone system (which is entirely controlled by software), exploiting this to harass his enemies and steal from banks and other online credit outlets. A Cornell graduate student, Robbert Morris Jr., launched an "Internet worm" program that was designed as a form of electronic parasite, intended to quietly infect computer systems and to spread, but not do damage. It got out of control and brought the whole Internet to its knees[5].

The whole topic raises fascinating questions about the ethics of breaking into computing systems, particularly if one's goals are non-malicious. There is something of a spectrum of attacks in this domain: some computing systems are wide open to intruders and provide essentially no protection at all, while others are the cybernetic equivalent of an armored tank. Breaking into the former can be as simple a matter as asking it to do something for you; the latter may require sophisticated anti-tank weaponry, and one would be hard pressed to pretend that the intrusion was accidental.

The goals of the intruder also enter into the picture. At the extreme of innocence are intruders who simply notice a system on the network and probe it out of curiosity, wondering what the thing does, but with no intention of damaging it or subverting it in any way. But the spectrum extends to individuals intent upon corporate or even state-sponsored espionage and even to hackers intent upon doing damage, like a group of software suppliers who tried to cripple a major cosmetic company's computers when a billing dispute escalated into serious acrimony. And there have been some widely reported incidents of corporate terrorism, blackmail and extortion, although these seem rare.

There are two perspectives upon these incidents. I've often wondered how many hospital computing systems were infected by the Internet worm, and whether any played potentially critical monitoring or data management roles. Probably, when the worm was launched, there were far fewer such systems than today; one might wonder if the same attack, perpetrated now, would result in injuries or even

[5] Two curious footnotes to this affair: Morris' father, Robert Morris Sr., was at the time one of America's top experts on computer security at the NSA. I'm told that he helped develop NSA's "Cyber Warfare" technology capability. As for Morris Jr., he was asked to leave Cornell and indeed, I've never even met him. But he resurfaced at MIT, where he eventually received his Ph.D. and is now employed as a faculty member. These days, he continues to work on network security, but he limits himself to publishing the results in conferences and journals.

fatalities. One can and should ask if hospitals have the moral right to even connect insecure computers to networks. Yet the action of the hospital often seems innocent – perhaps, the goal was simply to allow physicians to monitor patient status from home, or to enable Web access to pharmaceutical company Web sites. Can one really expect every hospital in the country to build up a military-strength counter-cyber-espionage unit? And if not, why do we feel surprised if a hospital takes what turn out to be typical but inadequate security measures to protect its systems?

Making matters worse, a surprising number of security weaknesses are built into computer systems by their own designers – Trojan Horses intended for the designer's own use, or amusement. When I was a graduate student at U.C. Berkeley, one of my colleagues was a programmer named Eric Allman, who later founded a company named UUnet and became hugely wealthy. Allman was, at the time, working on the core of the network email system – a program named "sendmail" that was among the first of the world's really successful network applications. Not surprisingly, sendmail sometimes got into trouble, and people would call Eric to ask if he could have a look and perhaps help out. Some of these computers weren't connected to dialup lines, so he developed a debugging feature with which he could send mail to his program from a remote location, ask it about its health, and even download new versions of the program or the configuration files it depended upon. Back in the days when the entire world operated on GNU-like principles, this was a completely normal and reasonable thing to do. But as time went by and the networking area became a huge commercial enterprise, Allman's back door became a notorious security flaw. And because computing systems often have surprisingly long lifetimes, even now there must be a remarkable number of computers on which the security flaw remains, waiting for a cracker to poke at it.

The shifting morality of the computing arena raises still further imponderables. People like me grew up during a period when most computing systems were open. We tended to view IBM as being inherently evil, because the company was large and stodgy, used old-fashioned software technologies, and generally adopted a dull, commercial approach to things. When Sun Microsystems emerged, we rooted for them because they represented the best of what our generation was capable of, and watching IBM stock collapse in price as Sun soared was a real triumph. But of course one grows up.

Today, Microsoft – a company that was founded the year I started in graduate school – is perceived as evil, yet while Microsoft certainly is guilty of some very aggressive business tactics, I find it hard to see them as deserving of this label – no more so, at any rate, than IBM. And, in retrospect, IBM was simply a very large company following what it believed to be the best available business path. When one considers the number of jobs represented by the IBMs and Microsofts of the world, it is hard to see why the GNU approach, which employs relatively few people, should be intrinsically preferable. Sun Microsystems, over this period, has begun to seem little different from the others to me, and while the company remains virtuous in the eyes of the free software community, they also confessed to having rigged the outcome of a performance evaluation of their "Java" platform just a few years ago. Is this company quite the embodiment of virtue that some perceive?

The connections between technology and politics, and public perception, merit much more study. There is little doubt that the hacker community perceives the larger companies of the world in terms that equate them with conservative politics: resistant to change, disinterested in the less fortunate, profoundly self-serving. Hackers seem to favor not merely a liberal model, but one that comes close to anarchy. But all of this points to a rather deep dilemma for the hacker, who perhaps feel a need to dominate artificial systems because the alternative is to accept that one is ultimately hostage to it. After all, the artificial worlds within the computers and networks are self-created in the eyes of the developer who builds them, but to other hackers can seem rigid. These free spirits are troubled by the externally imposed stricture, which conflicts with their own insistence upon unlimited freedom of expression.

To the extent that hackers perceive this stricture as associated with big companies and conservative forces, it is easy to understand the more damaging forms of the hacker/cracker ethic. In this situation, the hacker achieves power over his adversary by mastering the adversary's system and subjugating it. The hacker who launches a worm or a virus makes a mockery of his imagined enemies – the virus masters their best technology, then crushes it, taking utter control. The expression of power and domination in this situation is almost primal.

This is especially evident within what is called the "gray hat" cracker community: whereas "white hats" work to protect systems and "black hats" work to break into them, gray hats break into systems, then publicize both the attack and the fix. The claim of these groups is that they are doing an important public service by identifying and repairing vulnerabilities. But they also gain a form of power over the target: some hapless systems administrator is compelled to repair such and such a problem in such and such a manner, *right now*, under pain of an onslaught from the black hats, who by the way have just been tipped off. To me this is as clear a demonstration of raw power by one group over another as could be imagined, and it probably speaks volumes about black hats, too.

Finally, there are the hackers who view the majority of software with disdain and think of the Internet as some sort of projection of an unworthy society: cold, massive, indifferent. Such a hacker finds himself among a tiny community of similarly enlightened individuals, struggling to inflict a symbolic blow on what they view as an inhuman life form, populated by drones incapable of the slightest creativity, living their ant-like lives in the hive. To them, the machine – the whole system –is the very embodiment of societal rigidity, rules, and authoritarianism. The machine demands conformity and the hacker is one of the very few who stand between it and utter domination. And so they strike back, to save us all from a monster of our own creation.

Pandora's Digital Strongbox

The biggest threat associated with the growth of networking may be the erosion of personal privacy. As we begin to migrate more and more functions to the Internet, it becomes easier and easier to build electronic profiles of individuals that link their diverse activities into a composite picture that might include a great percentage of all their financial transactions, buying habits, credit record, taste in food and wine and clothing and perfume, reading and musical interests, political preferences, lifestyle choices, income and tax records, arrest and prison records (not to mention parking tickets and traffic accidents), current health and medical history, over-the-counter medication preferences… you name it. For some of us, in fact, the future holds the potential of being "personally" online, as medical monitoring devices and treatment devices gradually move to remote-controlled networked architectures.

There is already a sophisticated industry aimed at gathering this type of information into huge databases, which can be "mined" for insights. One can "investigate" just about anyone for a small fee, and the advertising we receive is increasingly targeted. In some ways I suppose this could be a good thing: perhaps purveyors of fine wines will write to offer me direct access to some obscure bottle that I would otherwise have to hunt down at considerable time and expense. I have no real objection to law enforcement officials tracing emails between suspected drug dealers, as long as they first get permission from a judge. But in the bigger picture, I prefer my privacy.

This isn't the setting for an extended exposition on our right to privacy and the issues that arise when privacy is considered in a networked context, but there are a few points on which we should touch briefly. First, it isn't at all clear that we have a right to privacy that extends to the contents of files on our computers, emails we send, or other information that, in effect, we voluntarily release without imposing explicit restrictions first. In the United States, the right to privacy stems primarily from the Constitution, which includes protection against unreasonable search and seizure of property. If I keep my computer locked in my study, and never connect it to the Internet, and then put a file on it, this does offer me some protection. But suppose that I connect my computer to a network. As the author of this book, I'm obviously aware that computer networks offer poor security. Do you violate my right to privacy if you exploit a security loophole to access my file? Is the problem changed if I take some special precautions to protect my computer or my file, perhaps by encrypting it? Or did the mere decision to place the file in a setting where security really can't be guaranteed imply that I was prepared to forgo the privacy of the file?

While courts have increasingly cracked down on hackers, the rationale often revolves around the economic damage that hackers often do, such as the costs associated with tracking them down, fixing the security problems they exploited, and otherwise restoring the integrity of a system. Cornell University's Law School maintains a very comprehensive online legal database system, but in searching it, I'm not able to find any sign that courts are convicting intruders who merely look at a file that someone else created, but then placed in a public place. With a good Web search engine, you can easily retrieve all sorts of files. What if one of them was intended to be private? Is it potentially illegal to merely follow a link?

To some extent, these matters are issues of public policy. For example, we should look to Congress to establish the rules under which companies can create databases that contain information about us. Legislation is needed that can spell out what my options are if that information is incorrect: How can I find out what these databases contain, and under what conditions can I require that errors be corrected? Are there different categories of information? Presumably yes: medical information has long been treated as especially sensitive, while personal buying preferences are viewed as public. But suppose that from your over-the-counter purchases of pregnancy tests at the local drug store, an insurance company is able to *deduce* that you are considering having a baby. Would that company be justified in raising your rates, in anticipation of higher bills? What if it was an over-the-counter HIV test? Remember, when you pay at the register, first they scan the bar-code, then your credit card. These kinds of payment records are closer and closer to being accessible to sophisticated users of the data-mining software and tools.

Late in 2000 it was reported in national newspapers that law enforcement officials had developed a technology that, in effect, permits them to search email in transit on the Internet. The propriety of such searches is very much a question that we as a society should debate (at present, the search system is apparently treated much like a wiretapping technology that can only be used with a search warrant, and only used to search for emails between specific individuals). As we saw in the previous chapter, once a tool is created, it rarely takes long before hackers get their hands on it. Will we soon see "black market" email traces?

In fact, the government has been pressing for fuller access to email and other computer-produced information, arguing that criminals are increasingly dependent upon modern technology. Of course, criminals encrypt their emails (at least, the smart ones do). Thus, during much of the past decade a debate has raged about something called "digital key escrow", which basically involves automatic registration of the means to decrypt files. The idea is that if the criminals buy their software from the same sources as everyone else, and don't find a way to defeat the escrow mechanism, law enforcement officers would be in a position to access encrypted data when the need arises. So far, key escrow hasn't happened, but the question has a way of popping up again and again, and I wouldn't rule it out if a sufficiently conservative, law-and-order oriented administration ever comes to power.

A second side of the equation is more personal. Clearly, to the extent that we as individuals chose to use computers in our work and lives, we should understand how computers protect information, what the limitations are on these forms of protection, and to what degree we can control access to information about ourselves. The same mechanisms protect your medical records at the local doctor's office, and your banking records down at the local savings bank. In what follows, I want to focus on this question. To what extent can one build a digital strongbox?

To read the advertising associated with popular Web sites that sell things like computer equipment or wine, or run Web auctions, one might well believe that the security of the Internet is on a par with that of one of those New York City banks that maintain massive vaults in the basement, surrounded by steel walls ten feet thick, monitored by state of the art sensors, and pretty much impregnable: a digital Fort Knox. Yet President Clinton goes on television to warn about the dangers posed by cyber-terrorists and less professional crackers, and one also reads newspaper articles about thefts of credit card numbers and even unauthorized wire transfers from banks connected to the network. I try to keep my PC software current, which involves downloading a "security patch" once every month or two and upgrading my anti-virus scanning software at least once a week.

We've talked about the difficulty of building genuinely secure computer systems, but why is it so hard to secure specific kinds of sensitive data? Does it really matter that the whole computer system be secure? Perhaps security is an esoteric requirement for the cyberbanks of the future, and the rest of us can continue to manage quite well with more limited protection for things like credit cards.

To answer such a question, one wants to first ask what sorts of information needs to be secured. Let's focus on the case of purchasing something on the Web. When a Web site claims to offer security, there are really several things going on. First, the Web site needs to identify its customers in a way that can be trusted. Some Web sites do this using passwords invented by the customer him or herself, but this is risky: it turns out that most people use birthdays or children's names for passwords if given the choice, and consequently end up with passwords that are very easy to guess. Other Web sites assign a password to the user, which is why most computers have a collection of stick-on notes somewhere close by, listing lots of accounts and passwords. If someone slips into your office, they might well end up with many of these. But the problem is more subtle, because a person who wiretaps the network could potentially capture the electronic messages containing them and in this way, steal access to your accounts.

Surprisingly, it isn't even clear that this type of wiretapping is illegal: this is another of those unclear privacy questions. Wiretapping laws make sense for telephone lines, which we think of as private. But computer networks are fundamentally shared: the electronic messages between my computer and my bank's Web site share the same wires as the ones between your computer and your favorite on-line clothing store. In some sense, the potential for someone to wiretap this type of electronic traffic is a basic property of the way that networks work. It would be disingenuous to proclaim it illegal; somewhat like saying that it is illegal to overhear a conversation between people standing next to you on a crowded subway train or bus.

Or, suppose that as we've predicted, there are settings where telephony travels over the same computer networks as computer data. Now telephone conversations are private, but computer data is inherently somewhat public: am I introducing a digital wiretap if I install software that scans these packets, perhaps for some legitimate reason like to check for viruses? What if my computer system automatically logs copies of data that travels on the network: has it illegally recorded telephone conversations? Murky territory!

So much for passwords. But computer networks also use a new form of password that one would not normally see in a direct sense. Most Web browsers and email systems have a built-in notion of security based upon what are called "digital keys" which are authenticated using "digital certificates." A digital key works much like any other kind of key: you can use it to "lock" information up (we do this by encrypting the information using the key), or to "unlock" (decrypt) information that was "locked" using the key. Keys can also be used as a form of proof of identity, similar to a password: you know that I'm *me* because I am able to decrypt things that were encrypted using a key that (theoretically) only you and I posess, or because I can produce encrypted copies of things upon request.

Technically, a certificate is a kind of container within which various things can be securely stored. In the most common use, a certificate contains a key somewhat analogous to the one you may have for a bank safe-deposit box. As you know, safe-deposit boxes actually can't be opened without *two* keys: one that the bank keeps, and one that they give you. Similarly, a certificate normally contains one key drawn from a pair of related keys. The idea is that the key *inside* the certificate can be used to validate the authenticity of a document sent by the holder of the *other* key. In one common way of using certificates, we say that the certificate contains a "public" key, and that the computer from which it originated holds a different but matching "private" key.

For example, suppose that you decide to purchase a book from the large Internet bookseller, Amazon.com. The first thing that happens when you click the "check out" button involves switching to what is called the "secure Web server" for Amazon; when using that server, all interactions are automatically and invisibly secured on your behalf. To do so, your computer first obtains a copy of the certificate for Amazon from a service that it trusts, like the one maintained by a company named Verisign. It does this by sending a message to Verisign requesting a copy of Amazon.com's current certificate, and Verisign sends back the certificate. This is done in such a way that any tampering with the certificate would immediately be detected.

This certificate will contain a key that your computer can use to make sure it is really dealing with Amazon.com, and not with some sort of imposter. For example, suppose that you download a Web page with payment options from the secure Web server. When sending it, Amazon, uses its secret (private) key to sign the Web page in a way that your computer, using the public key in your copy of the certificate, is able to verify.

The term "signature" may surprise you here: how can a computer sign a document? It turns out that when using digital keys, there is an easy way to do so. Basically, one compresses the document into a very short string of characters, using a tricky compression scheme. Then, the sender's private key is used to encrypt this string of characters – to encode it, using a special coding scheme. The result is sent side-by-side with the original document. The main advantage of a signature, relative to encrypting the entire document, is that a signature is cheap to compute, so it won't slow your Web interaction down very much.

Now, suppose that an intruder shows up. Basically, he has two options. He can modify the document itself, or can somehow prevent it from reaching you.

Digital signatures won't block the second kind of attack, but they offer a way to prevent the first kind. The idea is that upon receiving the document, your computer repeats the identical compression operation that was done by the sender. Next, using the key in the certificate it holds for Amazon.com, your computer decodes the digital signature. If the two compressed strings match, the document is considered to be intact. Thus, if an intruder were to tamper with the document, the odds are overwhelming that the intrusion would be detected. This form of security isn't perfect: one problem with "compression" is that information is lost, and there can actually be multiple documents that would produce the same compressed signature. But the compression scheme makes it extremely unlikely that an intruder could modify a document without it being detected.

Another sensible question to ask is why the Amazon.com certificate itself should be trusted. After all, if an intruder were to compromise the certificate, he could easily trick you into dealing with a bogus Amazon.com billing server and could then steal your credit card or otherwise compromise your purchase. Here, things get a little hairy. Recall that your computer obtained the certificate by making a request to a certification company – Verisign in our example. It turns out that Verisign's computer is also secured, using another key that can be authenticated with another certificate, the "Verisign certificate" – thus creating a chain of certificates. To trick you into accepting a fake Amazon.com certificate, the intruder would need to compromise your database of certificates, or to steal Verisign's key. This starts to seem a bit implausible, and in fact a big part of the security of the whole approach is that an intruder would need to carry out a form of digital fraud on an impossibly large scale to compromise that last step when you place your order with Amazon.

The very last certificate in this chain is typically built into the Web browser itself; you get it when you install the browser software. So: with that hard-wired certificate, you validate the Verisign computer from which you obtain Amazon's certificate, which in turn leads you to trust Amazon's Web site. In some sense, your computer trusts Microsoft (or Netscape), Microsoft trusts Verisign, and Verisign trusts Amazon. Finally, you trust the authenticity of the Amazon Web page because your browser can use the key inside the certificate to validate that the page hasn't been tampered with.

I've emphasized digital signatures, because these are the most common form of security. They are easy and fast to compute, and using them is simple. But one problem with a signature is that the Web page itself travels through the network in a normal text form that the intruder might be able to observe by wire-tapping the network. Sometimes, you wouldn't want to run that risk, in which case the Web page itself can be encrypted in such a way that while traveling on the network, it looks like a random string of meaningless characters. This, however, gets expensive, and the keys employed for encryption would often be manufactured just for your interaction with Amazon.com, a costly step as well. So while encryption is an option, it isn't used unless the information at hand is especially sensitive, and when it is used, the performance of the application will probably be significantly impacted.

All of this works in two directions. Certificates can also be constructed on *your* behalf, when you first register your computer, and used to validate you to the Web sites that you visit. When a certificate is used this way, the keys involved behave somewhat like passwords, except that you never actually show your password to anyone else. Just like in the cases described above, two keys are created. One key is kept private on your computer, and used to encrypt sensitive data or to electronically sign unencrypted messages, thereby proving that you actually sent the message and that it hasn't been modified or tampered with. The other key is placed into a certificate that your computer registers with a company like Verisign, and when you interact with a Web site like Amazon.com, is retrieved by Amazon's accounting software to verify that your computer is genuinely being used by you.

But the details don't really matter. The main point to take away from all of this is that computer security revolves around possession of the private keys and the ability to download certificates containing the decryption keys used to decode messages or verify signatures. A computer hacker (or a virus) who breaks into your computer could mess up your certificates, in which case you might be prevented from accessing secure Web sites, or tricked into trusting a bogus Amazon.com. A hacker might even steal your private keys, permitting him to pretend he is you. But as long as these certificates remain secure, they can be used to prove the authenticity of various kinds of requests. Most Web sites that work with credit cards use this approach – on your Web browser, you'll see a little padlock if encryption is in use.

Digital security is quite powerful. As we've seen, digital keys make it possible to authenticate the parties to a transaction (or at least, their computers), and to protect the integrity and even confidentiality of information sent over a network. But this still raises some issues, because very few computer users are aware of the security features available in designing Web sites or using email. So to the extent that digital security is available, it is still used in a very haphazard manner. For example, one needs to refresh certificates periodically. Otherwise, they eventually expire, meaning that there is a significant risk that a malicious person on the network may have broken the security scheme. Not many people think about this, so even computers set up to use the best available security may be quite a bit less secure than their owners realize. Moreover, security is remarkably hard to set up, so even a motivated computer user might find it hard to select the right combination of options and certificates to correctly secure his or her computer.

As an aside, you may have read that some Web sites keep "cookies" on your computer. The idea here is very simple: when you first visit the Web site, it computes a kind of visitor identification number, and usually the cookie simply consists of this number. Each time you revisit the identical site, that cookie (but no others) is automatically presented back to the site, so that it can look up information associated with your past visits and customize the Web pages you see. A cookie can even store short-term information; this is one of the ways to maintain a shopping cart when you shop at a Web site – the things you purchase end up in a compact list associated with the Web site's cookie.

Cookies are sometimes used as part of a security system too, for example to skip past the initial log-in screen of a digital newspaper: when the newspaper Web site sees the cookie, it realizes that you are a regular visitor and lets you go directly to the article you are trying to read. The cookie doesn't provide any information about who you are: if you chose to provide this kind of information, you do it explicitly when first signing up, and the cookie is simply used to look it up in a big database. So, even though your home computer does know things like your name and email address, this information is not *directly* available to the Web sites you visit unless you decide to register your name when asked to do so. However, there can be many ways to get such information *indirectly*. For example, suppose that you visit some Web page where you previously registered, and it displays an advertisement on the banner of your browser. Now you go to a Web page where you prefer to be anonymous. It turns out that there are some easy technical tricks whereby the companies that generate the ads can match you up across the two pages, and quite possibly can match your access to the registration data you gave to the first of the two sites. Your anonymity may well be an illusion!

As computing advances, it seems likely that anonymity will be harder and harder to achieve. The largest threat is the potential that certain types of companies, particularly those involved in marketing and customization of advertising materials, will have ways to build sophisticated profiles up that would let them identify you even when you take actions that should be anonymous. Right now this is much less ominous than it probably sounds, because let's face it: nobody really cares if you visit an on-line casino or a Web site selling Viagra online. Down the road, though, one has to wonder. As we put more and more stuff into our digital strongboxes, it seems more and more worrisome that they are actually so full of leaks.

At any rate, now we have a view of security that comes down to forms of electronic keys. The digital certificate is basically a container for a way to validate that someone who sent you something has the right key for their own computer, and the certificates that are created on your own behalf can be used by the services you connect to over the Web as a proof that you have your own "key", which is as close as most computers come to establishing that you are really *you*. And as for cookies, it is best to understand them as a way that the Web sites you visit can leave a small memo to themselves, which your computer agrees to maintain on behalf of that site (and only that site).

What about security for the computer systems that you visit on the network, or the computers at work, or in a bank, or a hospital? Again, there are a few answers. Surprisingly many computer systems have no real security at all: the computer is connected directly to the network, and anyone who knows how to connect to it and can provide a legitimate password is able to log on. This approach is also the most risky, because so many computer systems have software bugs associated with their protection scheme. A good cracker can often get into a computer if he or she can learn its electronic address (its "IP address"), and can send electronic messages to it. This even extends to computers that accept incoming telephone connections using a modem: when computers talk to each other over a modem, the situation is like any other kind of network connection, albeit a little slower.

Some institutions need stronger security, although one might be surprised by the number of very sensitive computing systems that do accept dialup and other direct connections. For example, in 1994 Citibank was attacked by a cracker who discovered a way to dial in, then instructed one of their computers to transfer $10M to some remote location in Russia. He was foiled because he made a small mistake: such transactions require sending a type of confirming message, and he apparently didn't realize that a separate computer command was needed to issue it. As a consequence the transfer was only partially completed, drawing the attention of a bank supervisor. Meanwhile, when the perpetrator presented himself to withdraw the money, the sum turned out to be well in excess of the cash reserves of the little Russian bank to which the wire transfer was made. It was only able to give him a small amount of the money. Within a day or two, Citibank had managed to stop the transaction, and he was arrested. The point, however, is that this particular kind of transfer, at the time, required only that one dial-in to the correct telephone number and know a password, which had not been changed in years.

Similarly, after early reports warned that the electric power grid was vulnerable to intrusion, the United States government began to wonder if the problem was really as severe as the pundits and experts were claiming. Accordingly, they assembled what is called a "red team" at the NSA and assigned them the task of breaking into control computers for the power grid using only knowledge that could be obtained by a potential terrorist. This exercise, called "Eligible Receiver", was quite a success: the NSA demonstrated convincingly that it, at least, would be able to shut down a large percentage of the nation's power plants over telephone connections – basically, they found ways to dial in and pretend to be plant supervisors working from home. I'm told that the security flaws exploited in this attack were subsequently patched, but it is easy to believe that other undiscovered flaws may still remain.

If one wants to build a genuinely secure computing system, the best place to start is with a computer that has no connections to the outside world. But isolating a computer or a network is a harder problem than you might expect. For example, many of us have laptop computers that can be connected to the network at work, or to the Internet at home. Such a machine is never connected to the Internet and to the network at work *simultaneously*. Yet it turns out that even so, it can serve as a conduit for a hacker, or a virus. That laptop is basically a big box full of data and software and can easily carry sensitive information out, a point driven home in when the ex-Director of the CIA, John Deutch, was stripped of his Top Secret clearance and nearly prosecuted for precisely this sort of lapse.

But not only do most computers allow people to bring laptops to work, many are continuously connected to the Internet. In sensitive settings, the number of "gateways" to the Internet is kept to a minimum, and the gateway computers are typically protected by so-called "firewalls:" software designed to allow legitimate users to talk to the computer while repelling messages originating at unauthorized sources. Firewalls represent a real barrier to unsophisticated attackers, but much less of one to crackers armed with all the latest technology. At one time or another, most commercial firewalls have been circumvented by the cracker community, sometimes by exploiting flaws in the firewall itself, but more often by exploiting common flaws in the way that administrators install and configure these very complex pieces of software. Even a flawless firewall (if such a thing could be built) can easily be compromised by a system administrator who installs it incorrectly. Moreover, like the walls around a castle, a firewall lives at the periphery of a sensitive network, not within it. So, like an attacker who tunnels under the walls of the castle to break in, or sends a trojan horse loaded with soldiers, there are sometimes ways to tunnel into a network and then attack it from within. Without getting dull and technical, I'll simply say that firewalls can help, but they are certainly not a silver bullet.

If we only worry about secrecy and what can be called "integrity", today's strongest forms of security are associated with the esoteric security technology mentioned in the previous chapter: the "virtual private network" or VPN. Now we can see how these work. A VPN distributes a copy of a digital key to each authenticated user of a network, and then uses that key to sign every single message sent on the network. Incoming messages that lack a legitimate signature (or have been modified) are discarded. Virtual private networks, however, have quite a few limitations. The most obvious is that some kinds of applications will need to interact both with computers within the corporation and with customers on the outside, for example to accept orders or even send and receive email. This means that there still needs to be some form of gateway between the virtually private network and the public network, and some of these technologies have been attacked through their gateways. There have also been successful attacks through hidden back-doors, like the ones my friend Eric Allman left in the sendmail program we talked about earlier. These potentially offer the attacker a route that can bypass the signature mechanisms used to protect the virtual private network. Moreover, VPNs don't guarantee fault-tolerance, tend to be slow, and lack any defense against denial-of-service attacks, where the intruder tries to shut down the network by overwhelming it with huge volumes of nonsense packets.

This pretty much summarizes the state of the art for computer security. The picture is best described as a mixture of mechanisms that really do work, but that can sometimes be circumvented, mechanisms that really don't work, although they may look as if they work from the perspective of the naive user and systems that aren't secured at all, but should have been. For example, a Web site that uses passwords and cookies may look relatively secure to the user (that little padlock would be missing on your browser, of course), and yet even an unsophisticated intruder would find it child's play to break in.

But why should we be so worried about security, in any case? One answer can be found by considering the kinds of electronic documents and services that are now on the Internet, or heading onto it. These include bank records and financial documents, which used to be mostly on paper, but will soon be mostly electronic. In fact, it you think about it, even "money" is increasingly just an electronic record. When I grew up, they were phasing out gold certificates – paper currency backed by the guarantee that the paper could be exchanged for gold upon demand. Today, you could argue, we have an economy based on electronic money backed by paper certificates. For example, if your salary is direct-deposited into your bank, and then you make purchases using a Visa debit card, you could live for months with only minor need for actual paper currency. Abstractly, your salary can be converted to paper currency upon demand, but as a practical matter, the kind of person who opts for direct deposit probably uses credit and debit cards for most transactions. Money, in the western economic system, is more and more abstract: a digital record stored in the memory of the bank's computer. Our confidence in the integrity of the system, and the many checks and balances built into it, are the fundamental reasons that it all holds together.

So we have a situation in which much of the world's financial system has quietly been converted from a system based on the physical presence of bars of gold in the basement of a bank, to one in which money is represented by electronic records maintained by banks and exchanged between banks over a network. It is hard to imagine the consequences of a widespread collapse of trust in this e-money scheme.

The Internet also offers cost-savings opportunities to the government. More and more people are entering their tax returns over the network, and it is expected that most government benefits such as Social Security, Welfare and Medicare/Medicaid will be electronic, too. Department of Justice databases covering everything from fingerprints to criminal records are all headed this way, as are insurance records and the kinds of forms that one currently fills out on paper to request reimbursements. In fact, one of the largest Internet startups in recent years is proposing to revolutionize medical paperwork, and has been backed enthusiastically by banks, insurance companies, health management organizations and banks. Everyone is hoping that computers will sweep the current mountain of regulations and paperwork to the side by just doing the stuff automatically.

The 2000 Presidential vote resulted in widespread calls for new and modernized voting machines. Suppose that the government were to experiment with putting elections online, through a Web page of some sort that you could access provided that you had the right kind of key (most likely, a secret registration number that would be given to you on a card at the time you register to vote). Given such a technology, people could vote from the comfort of a office or home.

I can see many reasons that this might appeal to the government. First, the current political structure reflects relatively higher voter participation within the educated, upper class sector of society: the same people who use computers most heavily. Electronic voting would favor participation by educated upper-class voters, and perhaps by the elderly, both groups that tend to support the status quo, and hence the major parties would presumably see electronic voting as a good thing. In contrast, many existing parties are threatened by other forms of increased participation, since the disenfranchised have a disturbing tendency to vote for change! So, it seems that one can anticipate that electronic voting would be quite popular with the electorate.

But the idea also raises some really troubling security issues. Suppose that you register but don't get around to voting. Could the system be "hacked" to show that you *did* vote, without you even knowing that your computer had been hijacked, in effect? Could someone change your vote electronically? Is there any risk that voting might lose its anonymity, so that people might have a way to figure out who voted for whom? My guess is that unless computer security improves dramatically, this type of sophisticated intrusion would be a real risk with Web-based voting. Indeed, even computerized voting machines would concern me, although I suppose that they have become inevitable: the days of punch-card voting systems are clearly numbered.

As hospitals go online, community health information networks will be more and more common; we've already talked about the prospects for getting direct healthcare from home. Soon, all your medical records may be online, with all your test results, and in some cases, your insulin pump may be online too. Down the road, it is easy to imagine all sorts of medical devices that would be connected to a network, sending data continuously about your condition, and even receiving remote controlled instructions. Merely by putting a computer into a medical device and configuring it to talk to a standard network, that computer is potentially at risk. So here the security of the system may play a role in the safety of your medical care, and certainly plays a big role in protecting the confidentiality of your records.

We've focused on security, but similar things can be said about "reliability" – the art of constructing computer networks to evade the problem to which Leslie Lamport alluded in his slightly cynical definition of a computer network. Lamport, as you recall, pointed out that in a network, your computer can fail because of the failure of something else; some remote component upon which it depends, but that you might not have even known was a part of the network. Now, "failure" has a nice solid sound to it and evokes the image of a hardware failure – a disk making a whimpering sound as wisps of smoke emerge. Those of us who used to work in computer machine rooms in the 1980's became experts at nosing out failed components, which have a very characteristic sweet smell, smoky but also slightly caramelized. These days, however, a failure is more often caused by a software problem – a crash, or a bug, or simply an oversight of some sort, which might correct itself, or might require that a computer systems operator track down the malfunctioning component and do something to fix it.

Broadly speaking, we consider a system to be adequately secure and reliable if it provides the desired service, to appropriate users, when and where the service is desired, and does so correctly. Such a definition treats reliability as a dimension of security. But in a world of networks and complex interdependencies between computing systems, a statement like this is much easier to write down than to translate into any sort of rules or guidelines for actually building the desired software.

As we've seen, reliability *can* be achieved, using practical techniques that are well understood, but it isn't necessarily easy. The trick is to accept that things are going to fail – but not often, and that it is really unusual for a lot of things to fail all at once. With this in mind, you can design a computer system to keep multiple copies of critical information and to monitor its own health, on the theory that even if something does go wrong, its impact will be limited to a small corner of the network and the remainder will remain healthy enough to initiate a repair action. Unfortunately, the FAA project serves as a reminder that it isn't trivial to build systems so that they can do this – the average computer program is such a snarl of interconnected chunks of software that there is no meaningful way to isolate a problem. But if a system is sufficiently "modular", faults can usually be isolated, and the system can automatically restart the dead component, or start a new but equivalent one somewhere else, after which the rest of the network will resume healthy operation.

Today, major vendors like Microsoft and Sun Microsystems are only beginning to treat security and reliability as major issues. Market pressure continues to favor being first to market with a sophisticated new capability, while making systems reliable and secure remains secondary – an afterthought. Until these issues are treated as first-class requirements, my guess is that reliability of complex systems will remain low, and that security will only be a little better.

Meanwhile, the rollout of critical computing applications continues unabated. All sorts of things will soon be in computers connected to the network. There will be records of purchases you've made, and travel, and lists of the movies you've rented or purchased, and of the books you've checked out of libraries. And there'll be plenty of confusion. For example, it is illegal for a library to keep long-term records showing who read which book or to release this information, but nobody knows whether or not the same practice is legal for a digital library (a form of Web site). Worse still, as a practical matter, most library computer systems *do* keep track of who has read which books; they simply don't provide a way to print this information out. Your email is online and many companies keep copy. It isn't wise to send the sort of emails that you might regret seeing on the front page of the newspaper.

It goes without saying that our society presumes that most of this sort of information should be protected and treated as private. But however easy it is to make such a statement, routinely protecting everything is very difficult; indeed, if we interpret the term narrowly, it is an impossibility. Protection, in any practical sense of the term, is currently as much an accident of data being spread around in a vast number of places as it is of any form of security. Each medical office runs its own software, each bank has its own proprietary internal systems, and so while the data is online, it is not accessible in any practical form. For perhaps another ten or twenty years this kind of protection will continue to represent a huge practical barrier to systematic abuse. But as corporations grow in size and merge, and as standards emerge for more and more of the word of electronic commerce and electronic information exchange, data will tend to collect in larger and larger databases and these practical barriers to abuse will gradually be reduced. Over time, it is nearly inevitable that more or less everything that computers ever "capture" in the first place will find its way into large databases that he government, private organizations, and criminals might be able to mine for sensitive information.

The sad aspect of this situation is that it has such an air of inevitability. As a researcher working on distributed computing systems, and especially on issues of reliability and security, I'm under no illusions about the obstacles to greatly improved computer network security. It hasn't happened up to now because, in some sense, security is perceived (rightly or wrongly) as anathema to the goals that lead one to build networks, and networks are driving the emerging electronic economy. Security is about barriers, and networks are about communication; if we want the latter, we forgo the former. As for reliability, well, "reliable software" just sounds ponderous and slow and old-fashioned, and with this mindset, not many developers are clambering for reliability-enhancing tools; instead, they just complain that the newest version of Java or Windows has bugs and stability problems. How can we inculcate a mindset that promotes reliability in positive terms?

Moreover, if the best available tools and technology have limitations, most applications will have vulnerabilities arising out of those limitations. To the degree that widely accepted standards are sweeping almost anything proprietary to the side, larger and larger swathes of the future electronic marketplace and world will suffer from the same limitations. And so the inexorable pressure from our society to adopt the hot new technologies currently presages an inevitable loss of security and privacy.

Yet, we build these systems, limitations and all, and roll them out with incredible enthusiasm and vigor. By way of analogy, if we propose to live in houses with windows, there are probably limits to the degree of privacy possible within our homes. A motivated eavesdropper could sneak up to the windows and peak in, wiretap the phones, bug our rooms, steam open the mail and read it. Likewise, if we build with two-by-fours, a strong wind will probably cause a lot of damage.

Ultimately, the protection against intrusive actions comes from their illegality. I don't need to move into an underground bunker to ensure my privacy: If a person trespasses on my property, I can have him arrested and he'll find himself explaining his actions to a skeptical judge. Similarly, the need is for broad legal protections that would allow us to prosecute more or less any individual, agency or organization for violation of our right to privacy in the event that systematic intrusions are detected. Today, these protections are surprisingly weak, although recent court cases against crackers are gradually building a body of precedent that favors protection of the individual. In the United States, the Supreme Court has yet to rule in this area, however, and one might legitimately worry that until sufficiently strong laws are enacted, the quality of protections will remain limited and somewhat ad-hoc. Suppose that for the foreseeable future, a great deal of private information will be poorly protected because of a technical deficiency. Does a third party have the right to accidentally "overhear" it? Does anyone have the right to actively seek it out?

Reliability is another matter. But one could look to a form of legal protection here, too. Houses need to comply with building codes. Could we come up with a system of "software building codes" that would encourage reliability without mandating something absurd or unpopular?

These kinds of fundamental questions need to be answered before the basic protections of the emerging information society will be at all clear. One can only hope that governments will appreciate the nature of the emerging risk and will act wisely to ensure that on the one hand, developers are made responsible for using generally accepted standards to secure sensitive information, and that on the other, individuals who maliciously intrude upon private information will face stringent penalties.

The delicate challenge for the legislative representatives who will make these kinds of laws is to strike an appropriate balance. On the one hand we have the potential for laws that overreach: demanding something impossible, like perfect security or flawless reliability. In a world of software that inevitably contains bugs, we must expect that computing systems will have some security bugs and that crashes will continue to occur. Yet systems often can be designed to anticipate such possibilities, to detect them, and to take corrective action when they occur. Developers cannot be compelled to build perfect software, but we can expect a high degree of professionalism from them: it is reasonable to demand that developers use "best standard practice" in developing systems where one might reasonably expect that the system will have life- or safety-critical (or even business-critical) applications, so that a developer who exhibits complete indifference to the usual reliability and security expectations for applications of the sort he or she is building would be viewed as negligent. These types of small, limited steps could have a tremendously positive impact. But we'll have more to say on this subject shortly.

Into the Machine

rojects like the electric power grid restructuring, the air traffic control effort, or the kinds of medical critical care networks we've mentioned, defy popular expectation. Things like this *should* be possible. The assertion that a mere technical challenge might represent a serious obstacle has a slightly blasphemous ring about it. After all, we put men on the moon, created the Internet, and cured polio!

Contemporary culture elevates technology to sit upon a pedestal. David Gelernter, who you may remember as the target of the Unabomber, wrote at some length on the topic of technology and society in his book *1939: The Lost World of The Fair*. The book evokes the changing attitudes toward technology evident today by offering a comparison with prevailing attitudes at the time of the 1939 World Fair. The Fair focuses on futuristic technology and science, and Gelernter interviews people who recall the experience of visiting it in terms that I would call almost reverent (Gelernter hesitated between a career in computer science and studying for the rabbinate, he chose computer science but like GTE's Michael Brodie, tends to see a deeper significance in technology). Viewed from the 1939 perspective, Gelernter suggests, our world is a utopia: we've accomplished much (although not everything) that the World's Fair simply dreamed about. And yet we are a dissatisfied, unappreciative bunch, morally corrupt and degenerate. It would seem that we got to heaven, all right, but we weren't worthy and messed up the place. Gelernter believes that a loss of faith in technology and science has deprived modern society of something to believe in – to him, science and technology offer reasons for hope and optimism.

Yet even now, popular belief in the infallibility of technology runs very deep; I think that one could still argue that science is the most ubiquitous modern religion. Relative to the world that Gelernter evokes, however, the modern world has had to grapple with a new fickle side of the world of science and technology: the limitations and failings of science. In 1939, people didn't know about global warming or the impact of DDT on songbirds. Poisonous algae had not yet started to bloom along the coasts, and medicine had not yet encountered microbes resistant to every drug in the arsenal, or illnesses like AIDS and mad-cow disease.

All of this seems to have created a modern schizophrenia in the way that technology and progress are viewed. On the one hand, society remains quick to laud advances in science and technology. But in 1939, people carried this view to the extreme – technology and science would reshape the world. Entering the third millennium, this optimism is tempered by suspicion that at least some technologies may not be all that beneficial. I see the new attitude as one that tends to blame the developers for any deficiency in their technology – in effect, we believe that most technologies and scientific advances *can and should be* beneficial; if not, the engineer is probably to blame.

There is a powerful connection between the belief that we merely need to throw money at a problem to vanquish it, and the prevalent underlying belief in the infallibility of technology. Having grown up in a world where astonishing progress occurs with astonishing regularity, it becomes difficult to accept that technology has more than transitory limitations. Magic bullets have been developed to cure syphilis, smallpox has been completely eliminated (except from bioweapon arsenals), and flu shots protect us against the annual flu epidemics. From such triumphs it is a small step to feeling outraged when progress turns out to be unexpectedly slow.

Our relationship with machines and technology is so old, and has stood us so well, that we feel comfortable and justified in the expectation that machines won't let us down. Yet we are also deceived by almost every technology, once the patina wears off. The irony is that we should be simultaneously entranced by technology and yet betrayed, like a lover who naively ignores all faults in her paramour, although the audience knows that sooner or later, he'll cheat on her.

One can trace both the buildup and the letdown intrinsic to this dynamic to the economic system within which we live. The buildup is easy to understand; we see it all the time. It takes the form of a mixture of genuine advances and marketing "hype." Consider the Internet. The network is genuinely creating a new form of commerce. But how new is it? In many ways, we've ended up with a new and improved catalog sales system. Now, these are certainly better catalogs: customized to our individual interests and even interactive – all most impressive. And the ordering systems are also better. Not long ago, I bought a new camera this way, and was fascinated to track it almost hour by hour across country. I watched as my camera was loaded in a truck in California, winced as the truck developed engine trouble, sighed with relief as it resumed progress towards a trans-shipping depot in some city I've never heard of, then watched it inch towards the east, until it reached Ithaca and was delivered to my home at 5:27pm, with the notation that my wife signed for the package!

No question that all of this is remarkable. But it would be hard to argue that I've been fundamentally transformed by my ability to purchase goods through electronic catalogs. After all, I was already a regular user of printed catalogues. I'm sure that the Internet will put a great number of telephone answering services out of business. Yet this is hardly the sort of social transformation of real concern in this book. During the next few years, it is hard to see the Internet as being as great a transformation as the newspapers and marketing hype suggests that it is (eventually, it may well be of this scale, but we'll need quite a bit longer to get there). My point, then, is not that the Internet isn't a dramatic new technology opening the door to a revolution in the ways we work and live, and in the ways that companies interact with one another. I'll grant all of these points. Rather, my concern is that the public somehow expects these things to happen overnight, and that these expectations are disproportionate to the *short-term* potential for the area.

There have been many examples of comparable marketing hype. The computer industry itself experienced a huge run-up in the 1970's, then sagged dramatically in the 1980's as blue chips such as IBM and Digital Equipment stumbled, only to recover again in the 1990's. The bioengineering and drug industry was viewed as a can't-lose investment opportunity in the 1980's, but investors soon discovered that creating better drugs is a hit-and-miss affair. The Internet revolution has much the same feel to it.

I think we are changed in very subtle ways by these developments, but not necessarily in the sweeping, all-inclusive ways that the marketing people would have us believe. But after all, marketing people market dreams. Their job is to sell things. The rest of us let ourselves be seduced, only to be let down when the reality proves to be less spotless and shiny and amazing, less transforming, than the dream predicted. We behave like children waiting so eagerly for the arrival of some new toy that almost everything else is pushed to the side. Given such inflated expectations, small wonder that the toy ultimately disappoints.

The difficulty with this dynamic, when we look at the broader version of the same phenomenon, is that public policy often reflects the same naïvely positive expectations that we adopt as individuals, but lacks a way to quickly abandon one dream in favor of a new one. When public projects are based on unrealistic perceptions of technology, this leaves us saddled with generations of nonsensical policies, which often systematically overlook downstream costs because they were not yet recognized when the policy was conceived.

The classic example of a downstream cost arises in the context of a Mid-Western power plant belching sulfurous smoke and ash high into the air as it inefficiently cranks out electricity for consumption in the region. The smoke and fumes are wafted high into the atmosphere and are carried east, eventually falling as acidified rain onto the forests and lakes of the eastern seaboard, where they poison the environment. Economists see such a situation as one in which there is a high downstream cost that our system fails to attribute back to the producer. In effect, we accept a hidden tax on the environment that subsidizes power production by Mid-Western producers.

To correct such hidden technology burdens one wishes to somehow charge them back to the producer. For example, the Federal Clean Air Act (enacted decades after this example was first recognized) required power producers to clean up their plants, creating economic incentives for doing so, but also offering some flexibility by permitting plants to buy and sell the right to pollute. The plant that continues to pollute thus pays for doing so, and runs at higher costs than a clean plant, which may even reap financial benefit by exceeding the standards.

But to intervene in this way, the government needs pretty dramatic evidence. Acid rain in the Northeast was only documented after decades of scientific study. Rachel Carson's book, *Silent Spring*, was the key to banning DDT, but the ban didn't occur for many years after the scientific community recognized the linkage between DDT and declining bird populations. Global warming and depletion of the ozone layer have only started to provoke some form of response, and it may be decades before the two phenomena have the degree of impact and visibility needed to really motivate the sort of concerted global action that will be needed to stem the trend.

We tend to believe that technology is good, unless a clear proof to the contrary is presented and even then, with the exception of certain types of particularly terrifying maladies or threats, any action can occur after very long delays. Thus when a technology is just a little defective, or has a very subtle negative impact, we seem completely at a loss to react to the problem. The prevailing view, in such cases, is that we live in a free society, and consumers are free to vote with their wallets.

Small wonder that many technologies overlook drawbacks or downstream costs. Our society doesn't really offer companies the incentive to do better. Instead, the dominant pressure is to be first to market. Only a very clear, easily demonstrated health or safety risk can provoke a credible concern on the part of the manufacturer.

The mass market also introduces a second factor into the picture, namely the tendency of any market to contract around a small set of majority producers. In most markets a relatively small number of vendors produce the great majority of the technology, and they hang onto their market shares by being fastest to market, most efficient as volume producers, and having the most effective advertising channels.

One can see the impact of such phenomena by looking at the evolution of the computing market over the past 25 or 30 years. In the 1960's we had many companies each selling its own proprietary design. Computers were full of wires and circuit boards and this made sense. The mass-market components were transistors, resistors, capacitors – and all forms of computers used them in great numbers. The plants needed to produce these components were relatively inexpensive and many companies operated them.

The market began to shift in the 1970's and 1980's with the emergence of VLSI. Today, a typical factory for building computer "CPU" and memory chips may cost *billions* of dollars. Designing a new chip can cost hundreds of millions of dollars. To be able to justify such staggering investments, any company that builds a VLSI fabrication line must target an enormous market. The costs of entry dictate that even if the chip market enlarges hugely, fewer vendors will be selling more standardized chips over time. This creates a world within which only the current generation of electronic products can be purchased, at any price. As it happens, each generation is also more capable than the one it replaces, but the pressure to compete in the mass market also dictates short product lifetimes and a certain "immaturity" of the product.

Now, one might wish that electronic components should somehow be independent of the computers and electronic devices that incorporate them, but of course this is not the case. The computers need to take advantage of the capabilities of each new generation of chips, and their sales create the market for the chip, so it is axiomatic that advances in chips will drive waves of change in the computers. These, in turn, will drive waves of change in the software that runs on the computers: the "operating system" that controls the machine, and the applications available on them.

Thus the mass market drives a cycle of innovation, one that on the one hand contracts the market around a small number of dominant suppliers, and yet on the other brings important benefits to the consumer. The evidence of this is visible in the rapid growth of the electronics industry, which has actually accelerated in recent years. Time to market has become a dominant consideration for any major company in the field. And as time passes, more and more companies, no matter what product they offer, are replacing their current generation of products with new digital product lines. Thus, as we look to the future, we can expect the same cycle of innovation and change to grip more and more of the country as a whole. More and more technologies, of every kind, are assembled from the same core components, sharing their advantages and also their limitations. Like a world built of Lego blocks, everything is increasingly standardized.

The downstream costs associated with this rapid cycle of innovation are, primarily, consequences of the relatively low reliability and security of many products. We'll have more to say about this later, but the key insight isn't very deep. It takes time and effort to "harden" a computer program or system, because software development is intrinsically error prone. I don't use the word "intrinsically" lightly: there simply isn't any way to develop bug-free software and I doubt that there will be anytime in the foreseeable future. The best development methodologies still yield software with enormous numbers of bugs and problems[6], which one then tries to resolve by extensive testing (and this is true not just of Microsoft but also Sun, and not just of programming languages like C but also Java – software is always buggy and no matter what anyone tells you, this is just a consequence of human limitations). In the types of networked systems with which I work, we actually plan ways of dealing with failures and (in effect) build systems that are reliable in the large not because the individual components work better than the PC's on your desk, but because the system has

[6] I don't want to become technical in this book, yet I realize that there is widespread skepticism concerning the feasibility of building bug-free or very secure software. Somehow, the prevailing mindset seems to assume that the major vendors – take Microsoft since they are a favorite target of such criticism – are doing a poor job of building software, because new releases of Windows are often buggy. Presumably, this reflects the general experience that most mature software products are fairly stable, leading the consumer to believe that if a new system is somehow buggy, the vendor is clearly doing a sloppy job. Yet what the public is rarely aware of is that there simply is no way to build large bug-free software systems, and that for all the hype about new languages like Java, the problems are fundamental to the way that software works. Even in Java one can code a program with a loop that never terminates, leaving the computer "stuck", or can accidentally try to divide a number by zero. Any program can be written incorrectly, so that the "save" button sometimes scrambles the file. Testing helps, but even with thorough testing, the best developers only manage to reduce their bug rates. Every program of any substantial size has bugs – and this includes bugs in the security mechanisms used to protect against intrusion. Indeed, considering the size of the systems they build and sell, I find it amazing that companies like Sun and Microsoft build products that are as *reliable* as they seem to be!

the capacity to detect failures and to repair itself, especially when a failure only impacts a small part of the system. But even this goal is hard to achieve in a technical sense, and it takes time to introduce the necessary mechanisms into a system. If the system is sufficiently complex, it can be very hard to make it reliable. Thus, pressure to rush new software to market more or less guarantees that the resulting systems will be unstable and hard to work with until they have time to "settle in" as users begin to work with them, and the vendors issue corrections for the more serious problems encountered.

Notice that this dynamic rewards consumers willing to tolerate product instability. Although they experience the early problems – we use the term "on the bleeding edge" to convey the way it can feel to be an early adopter of some technologies – companies prepared to do so can seize a big competitive advantage, because their products run on the hot new platforms sooner, and they have more time to work around any instabilities that may survive into later versions of the platform. Since early customers *expect* problems, they typically don't insist upon extreme reliability, hence the market as a whole is not really under very much pressure to provide such properties; the focus, instead, is on new features. One could imagine a different market cycle, but this is the one that has become established.

Hidden in this dynamic is a deeper, broader problem. While the typical customer of a new operating system or computer may be tolerant of instability, unreliability and a lack of security, what about the applications requiring special security or reliability properties? These represent a small market (how many computer systems will the world's hospitals install, in total, as a percentage of the overall world market for computer systems?), but their aggregated impact on our lives and environment is considerable. Developers find themselves confronted with an unending stream of new versions of products, none terribly reliable or secure, and just as a platform becomes stable, the vendor typically introduces a completely new product line with features that more or less demand that one upgrade. Much like the environmental impact of a non-biodegradable pesticide, we lack effective mechanisms for dealing with situations where a small effect is amplified by large numbers. Thus, the tolerance of technical deficiencies in early releases of products causes problems both by enshrining a lower degree of security and reliability than might otherwise be feasible, and by reducing overall market demand for high security, high reliability products.

I suspect that this is why the FAA didn't even begin to build an upgraded air traffic control system until 1981. Starting in the 1960's, the economic underpinnings that permitted the agency to build dedicated, one-time systems eroded. As early as the 1970's we saw the emergence of a new market predicated on the use of ubiquitous components, in which powerful commercial products could be purchased off the shelf, and easily combined into systems provided, however, that one used the most standard tools. Less appreciated was that the necessary infrastructure had become enormously unwieldy and hence that any non-standard way of using these components would be hugely expensive and unlikely to succeed. The FAA presumably watched this new pattern emerge with mixed feelings, since the commercial product base was much more cost effective, yet lacked the properties expected of an air traffic control technology. Simultaneously, the kinds of products aimed specifically at their needs were vanishing from the market rather than evolving to keep pace with the less costly, less specialized, mass-market products. When the issue was forced upon the agency in 1981, it must have seen IBM's research in fault-tolerance as a form of magic bullet: even if IBM itself treated the idea as a small selling point in the content of a big proposal, in the eyes of the FAA this represented a huge advance, because a major company was proposing to use standard components in standard ways, and yet claiming it could arrive at an ultra-reliable solution. For its part, IBM underestimated the broader market resistance to non-standard ways of *using* standard components. As it became clear that the approach was no magic bullet after all, and that the FAA's specialized needs didn't represent a major new market for the company, IBM divested the group running the project and it began a slow death spiral.

This example, illuminating the broader pattern, helps explains how we can live in a society where on the one hand, marketing people hype the Internet as the greatest advance since the printing press, and yet the government commissions one study after another to investigate the massive threat to our "nationally critical communications and computing infrastructure" posed precisely by the growing use of insecure and unreliable Internet computing technologies, in just about every sector of the modern economy.

There is an additional trend that raises further questions. This involves technologies that undergo what might be called a "creeping transition" whereby, over time, something that was originally considered relatively unimportant and non-critical becomes critical to life or safety, and yet by virtue of its gradual evolution, escapes the normal scrutiny to which life or safety-critical technologies are normally subjected.

Anyone who has ever spent time in a hospital intensive care unit or watched one of the many medical "dramas" on television is aware that hospitals have become increasingly technology-dependent. In medical computing systems one can distinguish two rather different ways of using technology. The first involves medical devices that "practice medicine" in some loose sense of the word, like a cardiac pacemaker, or an IV drip that administers a controlled amount of a drug, or even the cardiac rhythm monitors that sound alarms if anything goes wrong. These kinds of devices could cause injury if they malfunction, and consequently are tightly regulated, much like a drug. When software or computers are involved, the Food and Drug Administration (FDA) insists that an exhaustive safety validation process be followed[7]. I'm told that the results can be measured in "pounds of paperwork per line of code", because so many different tests and evaluations are required that simply certifying that the task was done can involve literally pounds of documentation. A single document may represent hours of work by a testing group. The process limits the size of these kinds of computer programs: they rarely exceed a few thousand lines of code, which by modern standards is minuscule.

[7] These tiny, exhaustively tested programs are the exception that proves the rule: they cost a fortune to develop but really are safe and reliable. The problem is that the same methods could never be used for something really big, like the Linux operating system or Windows.

The second type of medical computing system plays a much less critical role, at least in the eyes of the FDA and the industry. Examples include the computers that keep track of medical reports and laboratory reports, handle billing (the hospital may view this as a critical task, but the FDA doesn't worry about it!), keep track of which type of meals should be given to which patient, and maintain the keep lists of medications that each patient is currently receiving. Overall, one could say that this second category of computer systems maintains the records.

One's intuition might suggest that the medical record is every bit as critical as a pacemaker, but at least in a historical sense, this has not been the case. The record was traditionally kept on paper or in folders down in the hospital records office, and since a person needs to approve any entries, the record can easily lag behind. Moreover, laboratory results can be incorrect, and a patient's condition can change suddenly. For all of these reasons, no doctor would want to base treatment decisions purely on the medical record – seeing the patient and confirming his or her condition through a direct examination remains the central element of medical decision making, and the record simply documents what the doctors and nurses are doing in "real time."

But things have begun to change. Over the past few years, more and more medical monitoring devices and laboratories have been connected directly to the patient record-keeping system, creating a more integrated record that comes closer to tracking the state of the patient in a continuous way. Under pressure to cut costs, hospitals have increased the load on doctors and nurses, and they don't have as much time to spend at the bedside as in the past. Moreover, as noted earlier, some devices can even be controlled remotely over the network, although not many medical settings take advantage of these features.

More and more hospitals are combining traditionally separate functions on a single computer platform: the bedside machine might support some aspects of the medical monitoring system and some aspects of the patient record-keeping system and may even provide a web-based browser that can access the medical library or drug-company information sites in a pinch. A good friend of mine, Wes Blauvelt, is Vice President for Development at one of the big regional hospitals for upstate New York. He tells me that this integration will probably continue until the disparate elements of the medical computing network merge into one relatively seamless distributed system stretching over the network into the various devices, out through the community to laboratories and clinics, into the homes of patients and doctors, and out to the insurance and HMO computing systems that pay the bills. He says that the trend is well established and is letting the hospital do things that were never possible in the past, like treating a patient who is living at home. Moreover, by linking the various care providers and laboratories together, the hospital can reduce duplication of services and can operate more efficiently.

So far so good. But where does this leave the FDA? In the past, it was easy to separate the critical computing systems from the non-critical ones. Suddenly, the critical roles of the single blended-together computing system are side-by-side with the non-critical parts, and one must worry that somehow the behavior of the non-critical systems could have a troublesome impact on the critical pieces. The distinction between the computing systems that practice medicine and the ones that just keep reports is less and less clear. Doctors and nurses are basing real clinical decisions on the computerized versions of records, and with the emergence of telemedicine – the practice of medical care at a distance, over a network – this dependence will only grow.

To me, this suggests that the same logic that demands certification for medical monitoring devices should now require the FDA to certify these integrated solutions. But this leads to a dilemma: how, exactly, would one go about certifying the safety of, say, a PC running Windows 2000 and Microsoft Office, over which some very complex database system is running side-by-side with a web-browser and a few older monitoring applications (not to mention a few computer games if there is a kid in the vicinity)? Techniques used to reach high levels of assurance for a few hundred lines of code make little sense in such a system, which aggregate millions of lines of code. Moreover, whereas the traditional monitoring system was isolated and hence could be validated without worrying much about the rest of the hospital, here we see a kind of system that extends like an octopus, so that no small piece can be considered completely independently from the remainder. Not only must each part be correct, but there also needs to be some certainty that the pieces can't interfere with one-another when more than one event occurs at the same time.

The technical group at Hewlett Packard's medical division told me a story that illustrates this problem. This team is responsible for a product that sends medical telemetry to remote monitoring systems, so that a nursing station can be configured with a duplicate of the cardiac monitoring data seen at bedside. Since the monitoring system also sets off alarms if a patient's heart rhythm deviates from what the nurses are expecting, it has a life-critical role and hence is subject to FDA regulation, although less stringently than for a pacemaker.

In the past, the HP system used to run on dedicated computers and had a network of its own. But hospitals complained that using separate computers just for this purpose was causing a maintenance headache, and that running a separate network was totally out of the question. So, a few years ago, the company was forced to move their system onto traditional PC's shared with other applications, and running over the same kinds of networks that we use in our offices.

As one might expect, this led to all sorts of nasty surprises. HP found it necessary to create its own in-house testing group, which receives new versions of software from vendors like Microsoft and Netscape and evaluates security and reliability from the standpoint of critical medical uses. They tell me that they've fixed countless serious problems in almost everything they've tested. Even so, some problems slip through.

In particular, one hospital complained that sometimes the cardiac monitoring systems would shut themselves down in the middle of the night, when nothing seemed to be wrong. After exhausting all other options, HP sent technicians to investigate. Sure enough, everything was fine during the day, but late at night, the displays would shut down. It wasn't hard for them to track the problem down to a network overload: something on the network was using so much bandwidth that there wasn't enough left over for the monitoring system. But why should this happen late at night, rather than during the day when everybody is hard at work? Having worked in a hospital with friends who were medical students, I wasn't terribly surprised to learn the answer. It seems that this was a teaching hospital, full of medical interns working long hours. Now, one might imagine that doctors really stay on their feet for 36 hours at a time, but the reality is normally pretty boring. And on those long boring nights, it seemed that some of these medical interns turned out to be in the habit of downloading the very highest quality, um, medical imagery. What dedication: laboring day and night to master even the most arcane details of anatomy and physiology!

Once HP understood the problem, they had little difficulty fixing their system, but I wonder about the situation for some of their competitors. The really big companies, like HP and EMTEK, are facing competition from small startups that throw systems together and aim at the lower end of the market. These kinds of smaller companies can't assemble in-house testing groups or even think through some of the scenarios that might arise when huge quantities of prebuilt software from many sources are glued together to create the hospital computing system of the future. Presumably, their technologies are quite shaky, despite having the same impressive look and feel as do other modern applications running on PCs.

The HP experience highlights a technical weakness of modern networks. If you think about it, it seems clear that HP needed a way to run their old software, designed for a private "dedicated" computing network, in a shared one with other users. Broadly, they could do this by modifying the system, or by somehow fixing the network to behave like a private one so that the old system could be used without change. Changing the application is only feasible if the requirement wasn't real in the first place; if safe operation of the system requires that such and such a level of communication traffic be possible, one can't simply change the application to have a weaker need. In our example, HP solved the problem by modifying the behavior of people using the network – but it seems obvious that this is not a good solution; one could easily imagine legitimate applications that would generate similar network traffic and couldn't be chased away so easily. Clearly, the need is for a way to somehow tell the network about the assumptions and needs of classes of applications, and for the network to dedicate some of its resources for private use by the application. In our example, if the monitoring system had a way to tell the network "this application needs to rapidly send messages from the monitoring system to the displays, continuously", the network could at least theoretically have walled off some bandwidth just for use by the monitoring application, and the background traffic wouldn't have posed a problem. Of course, the Internet images and videos would have downloaded a bit slower, but presumably, the interns wouldn't have complained. On the other hand, the resulting system would be safe in much the same sense as the older monitoring networks. The FDA could certify it.

Unfortunately, modern networks are built in a manner that precludes this sort of solution. There is no way for an application to tell the network what it needs, and the network has no way to even distinguish traffic associated with one application from traffic associated from another. Although some very sophisticated proposals for dedicating network resources to special purposes have been advanced, these are aimed mostly at running telephones over the Internet – a problem fairly remote from the needs of a critical computing system like HP's medical monitor. We thus are faced with a profound mismatch between the true needs of this class of applications and the capabilities of the networks on which they are to be used. The solution may seem to work, but at a deeper level, its safety and correctness has begun to depend on good behavior (or good luck).

It seems fair to say that the FDA will face quite a serious challenge as this trend continues and accelerates. The hospital of the future will be running a large, complex, critical computing system built using out-of-the box components that were created for non-critical office settings. One couldn't imagine a style of system more different from the sorts of small isolated devices the FDA is used to certifying, yet these systems are perhaps even more life-critical than the ones they replace. And making matters worse, while the FDA process normally involves small amounts of code built by the device designers, these systems include massive amounts of code built by other vendors, big chunks of Internet software, proprietary components that are simply not available in "source code" form, and even assumptions about human behavior!

The "creep of the critical role" is not unique to medical computing systems. The trends are very similar in the avionics industry. When you step aboard a modern airplane, like an Airbus 320 or a Boeing 777, you are stepping into a hugely complex machine controlled almost entirely by computers and software. Much like a medical monitoring system, these very critical pieces of software are stringently validated in isolation, and if anything, the number of pounds of paper per line of code is even higher than for the FDA process. Yet, much like a medical system, avionics systems are gradually turning into large networks of relatively conventional looking computers and programs, which work in concert to control the plane.

Ricky Butler works on avionics safety certification at NASA; I first met him back in 1994, when he headed a NASA workshop on reliability issues with what are called embedded computing systems – small computing systems that operate things like airplanes or medical equipment Ricky doesn't come across as worrier – a tall, lanky man with a buck-toothed, awe-shucks style, he seems incredibly relaxed, and when I attended a workshop he organized in Washington, I was extremely impressed at the degree to which he put people at ease. But Ricky isn't really all that relaxed on issues underlying aircraft safety, which is his main interest. At that workshop, and in subsequent dialog with me, Ricky described trends he sees as nothing short of scary: as fast as the industry can, it is moving away from the old style of hand-crafted systems into a new style of "integrated modular avionics" motivated by the plug-and-play approach to upgrading personal computing systems, by just plugging in new equipment.

Today, the airplanes you fly were probably certified as safe by FAA inspectors who viewed the entire plane as a single package. Tomorrow, if integrated modular avionics continues to gain ground, a plane might be certified as safe one year, and then upgraded with new equipment (say, new software to control the flaps) without recertifying the rest of the airplane. Obviously, industry is very enthusiastic about this prospect, because in a world of modular components and standard technologies, the need to recertify the entire plane each time a component changes is very cumbersome. But with this greater ease of upgrade comes a growing risk that someday, down the road, an upgrade will trigger some unanticipated problem elsewhere in a plane and render the whole package unsafe.

A series of accidents associated with the modular approach suggest that Ricky's concerns are grounded in some real issues. In Europe, the Arianne rocket is one of the most important recent products of the same multinational collaborations that brought us the Concorde supersonic jet and the Airbus. Well, during 1995, a new model of the Arianne exploded shortly after liftoff. It turned out that the rocket failed because of just the sort of mistake that Ricky is afraid might happen here: they decided to build the new rocket by upgrading some parts while reusing others from older rocket designs. One of those reused parts was built for a much slower model of rocket, and malfunctioned when subjected to the higher acceleration associated with the new design. The resulting error messages confused the guidance system, causing the flight as a whole to abort. In 1999, a similar problem caused a NASA mission to Mars to fail: a module built by one subcontractor produced telemetry data using English units, but was connected to a module that expected metric units. The mismatch caused the spacecraft to go off-course and it plunged to destruction in the Martian atmosphere.

Here we see that simply plugging a new module into an old system can cause catastrophe even if the new module works perfectly. One could argue that an analogous problem arises in medical monitoring systems that are moved from dedicated devices onto shared computer platforms. But the parallels with the medical situation go much further. Recall that the effort to upgrade the American air traffic control system stalled some years ago. It turns out that after the project failed, a decision was made to try and reduce load on controllers by taking them out of the loop where possible, resulting in something called "free flight." In this approach, human controllers are only involved in decision making close to airports and when planes appear to be drifting dangerously close together in the air. For longer distances, each pilot is freed to make his own navigational decisions, using an on-board navigational system that gets data about what other planes are doing from a ground-based computing system, which tracks progress of other planes in the air and has copies of their flight plans.

I first realized that free flight was more than just a proposal when I took a flight to San Diego, which happened to pass over the Grand Canyon on a very clear day. The pilot suddenly announced that under the "new air traffic control rules" we were in for a treat: first those on the left of the plane, and then those on the right, would have a superb view of the Canyon! With no further warning, the plane rolled about 30 degrees to the left, then about 30 degrees to the right, and so forth, sweeping out gentle arcs over the Canyon at 600 miles an hour, 37,000 feet in the air. Something tells me that no human controller approved this plan. Now, this was your basic large jet on a flight from Pittsburgh to California, and people on such flights sometimes panic at the slightest turbulence (those of us living in the hinterlands are made of tougher stuff, and we need to be, considering the harrowing experience of flying to Ithaca in upstate New York on little prop planes in stormy weather). All around me, I heard a rustle of little paper bags, and tell-tale coughing sounds. So, this is free flight.

Consider the safety issue raised by free flight. The guidance of the plane is now tied not just to the software controlling the flaps or the engine, but also to the ground system that tells the pilot's navigational computer where everyone else is, and approves the safety (if not the wisdom) of the proposed maneuver or routing change. It may seem as if the on-board navigational system runs the show, but that system will only be as good as the data it receives from the ground-based system. The safety of the plane is, in effect, linked through the ground-based computing system to all the other planes in the air, and to all the software running that ground system, not merely directly but also through the network within which it lives. In some sense, free flight requires trust in the whole Web of ground based systems and airplanes and software used in the entire air traffic control network. Suddenly, as in the medical situation, validating the safety of the plane takes on a completely new dimension.

The obvious rejoinder is that the ground system was a critical technology too, and hence that this isn't really a more critical configuration of the avionics system and the air traffic control system than the one it replaces. But such arguments are slightly disingenuous, for several reasons. First, the air traffic control system derives much of its safety from the presence of various forms of backups in the loop: the controllers, the "raw" unprocessed radar images, and the pilots themselves. Indeed, I'm told that even if an air traffic control system fails completely, the controllers can manage using various backups. By connecting the software running the air traffic control database system directly into the plane, we've substantially reduced the number of independent checks and balances. At the same time, we've taken the avionics system – the flight management system of the plane itself – and connected it to a computer network. The potential for adverse interactions is inescapable in such a configuration, and previously, no such connection existed. So there is no doubt that these sorts of moves create a qualitatively different problem.

The list goes on, but it would become tedious to work through example after example in similar detail. Whether in medicine, avionics, banking, electronic commerce, electric power distribution, or any of hundreds of other critical uses, computer systems are becoming more and more interconnected, and what used to be mundane computing systems that were useful but not terribly important are evolving into what might be considered to be safety-critical or life-critical roles. The shear complexity of these systems defies the traditional ways of convincing ourselves that they are really safe and trustworthy, yet the emergence of these configurations and roles can be so slow as to go completely unnoticed until long after the point of no return. Each step makes sense in isolation, and may seem both minor and purely incremental. But the larger trend, somehow taking on a life of its own, is sweeping both in consequence and implication.

Living within the machine, I often wonder what counterbalance, if any, exists to prevent these kinds of developments from evolving in dangerous ways. Certainly, when a situation actually becomes critical and a catastrophe occurs, like the explosion of the Challenger, society suddenly awakens to the issue and can assert itself. With adequate oversight, it seems that even very complex technologies can be managed safely. Yet the evolution of technology often creates very complex structures that operate without any real oversight at all. When they work, like the telephone network – a hugely complex infrastructure – we marvel at the astonishing robustness of the thing, and perhaps wonder how it could have come about. Yet we don't hesitate to build these things even without assurance that they will work. Only rarely do we actually consider putting the brakes on before launching the experiment. The assumption is that technology projects will succeed if the engineers merely do their jobs competently.

Doing the Right Thing

Engineers are rarely popular culture's heroes or villains; more often, we're consigned to passive roles on the sidelines. For example, the world's most profound moral dilemma, at least in the past century or so, was the development of the atomic bomb. Both the responsibility for conceiving it and the actual development are typically ascribed to the physics community, and particularly to individuals such as Albert Einstein, Hans Bethe and Edward Teller. Engineers actually played a big role, and yet are portrayed in most accounts as oddly subservient. Physicists, in contrast, are often depicted as having struggled with the evident ethical conflict – Einstein, who wrote a letter to Roosevelt urging the development of the bomb out of fear that Hitler would build it first, subsequently campaigned valiantly against nuclear proliferation as the weapons race took on a life of its own in the postwar period. Bethe, father of the Hydrogen Bomb, has subsequently pressed for nuclear disarmament. Teller has argued just as passionately for the strongest possible offensive and defensive weaponry, and for years was the primary champion of the Star Wars missile defense proposal.

You may disagree with some of them, but at least the physicists were out there fighting for passionately-held beliefs. Few question the relative complacency of the engineers. One could perceive a conspiracy of silence and inaction: complicity by omission. But instead, society seems not to expect more from engineers. The engineer, it would seem, builds machines and is expected to approach such problems quite mechanically.

The same observation can be made much more broadly. Open the weekly health or science section of the newspaper, and you are likely to read of a doctor confronting a terrible ethical challenge. Such a story is often written vividly, starting with heart-wrenching interviews with patients suffering from some condition, and then gradually widening the topic of discussion to include the researchers struggling with a deep tradeoff posed by some emerging treatment or decision. Yet one never reads of engineers confronted by ethical dilemmas, fighting to do the right thing, except in the context of whistle-blowers who warn that a project is being mismanaged (and such an article would rarely omit to mention that under current legislation in the United States, a whistle-blower on a government contract is entitled to a substantial percentage of any recovered funds).

One could argue, I suppose, that physicists and doctors encounter ethical issues in a more clear-cut form. The physicist, after all, studies basic questions about the universe, often mathematically. Einstein's dilemma was primarily a struggle with the ethics of elaborating an implication of the more fundamental science that had already been revealed as he and his colleagues pressed for an ever-deeper understanding of the nature of matter and the forces that bind particles within an atomic nucleus. The question, then, was not so much whether or not to discover the possibility of an atomic bomb – this had already happened; it was an inevitable consequence of the study of the nature of the universe. For Einstein, the moral debate centered on whether or not to assist the military in transforming the basic physics into a weapon. Ultimately, having awakened Roosevelt to the potential, Einstein didn't actually participate in the development, and at the end of the war, fought to control the demons that the military community now unleashed.

In this view, the pure scientist is like a philosopher, revealing the potential of the universe through thought alone, while the engineer transforms science into usable technology and so creates the weapon. In a similar sense, I might look at a rock and realize that it could be used as a weapon, and perhaps even make this observation out loud, and yet my action seems minor in comparison to that of a person who seizes that rock and assaults his rival. Yet the creation of an atomic bomb was not an obvious thing and required a tremendous amount of scientific work, beyond the initial insight that it might be feasible, to complete. No doubt, the physicists of the time perceived a more complex terrain: the potential for clean, inexpensive, non-polluting power, for example (this was a period before the pernicious qualities of radioactive materials were widely appreciated). I leave it to the reader to speculate about where the fault lies when a scientific idea goes astray: did the physicist err by noticing the implication of the mathematics, or the engineer by building the device? Or are these things inevitable consequences of our society: a race to master science and technology, before a presumed enemy does so? Did the scientist and the engineer bear responsibility, or is the responsibility elsewhere, with the military perhaps, or with the soldier who actually seizes the rock and strikes?

The doctor confronts ethical decisions of an entirely different nature: the side-effects of the treatment may hurt the patient, or leave the patient with an unacceptable quality of life, or deprive some other patient of a life-saving treatment. In some ways, I think these are the easier questions to pose, if not to answer. But both kinds of ethical questions have a certain abstract clarity.

These sorts of issues take a very different form for the engineer. We've seen Moore's law and discussed the astonishing growth of the Internet, two different perspectives on the driving forces that sweep technology forward. Evolution and progress are the defining characteristics of the field. Thus the engineer is rarely faced with some simple choice, for example to cut corners so as to be first to market with a defective technology, as opposed to taking the time needed to construct an adequate one. More typically, a computing system should be viewed as a steadily advancing frontier: a basic kernel which will be extended over time with new features, new capabilities. If version 1 is a bit insecure and unstable, one can reasonably hope that the product will improve as its basic functionality is tested and subjected to the stresses of the real world. The engineer's challenge, perhaps in a dialog with management, is to ask what needs to be part of this version of a system and what should be delayed until the next version, to lay out a path of incremental improvement, and to see to it that the system cannot be used in inappropriate ways – ideally, by designing the system to actively enforce its own limitations, although this ideal is rarely achieved. At the least, the user needs to be warned, but perhaps this is as far as one can reasonably expect the engineer to go, particularly if he or she sees a path to improvement and is working to follow that path. After all, the knowledgeable user can always delay building the hospital computing network for a year, if the properties of version 1 of some technology are inadequate for the need, whereas version 2 is expected to plug the gaps.

In contemporary society, this simple perspective is heavily distorted by other considerations. Paramount is the financial one: early deployment of a product can determine its market success; thus, getting an inadequate product into the field and later improving it can result in a better outcome than delaying the product to improve it prior to release, while a competitor's inferior product seizes market share. We mentioned the perplexing disinterest of the market in better product quality; which can be traced to a form of self-reinforcing reward system that encourages market tolerance for product limitations. Nobody has perfect foresight; this is, after all, a world transformed by periodic revolutions, within which the flow of revenue ultimately determines which technologies will become standard and which will fade away. Thus, even if an engineer were to look to the future, the knowledge of potential future capabilities might not lead to an obvious decision to forgo some sort of technical development. Take the example of the Internet, which has now reached a precarious form of maturity, and yet continues to suffer many of the same deficiencies as did the early prototypes. Should an engineer aware of the limitations of the early architecture have refused to work on the the Internet, out of fear that society would seize upon it in ways that might ultimately lock in the weaknesses of the early design?

In my own experience, when such questions have arisen, they have rarely been clear-cut. For example, about twenty years ago I was hired as a college student to work on a medical computing problem at Columbia Presbyterian Hospital in New York City. Doctor Bigger, for whom I worked, was a leader in the development of new cardiac drugs, which one evaluates by measuring their impact on the abnormal cardiac rhythms exhibited by the ill patient.

The basic approach was the following. Doctor Bigger or one of his colleagues would start by having the patient wear a form of tape recorder for a day, or sometimes for a few days. The resulting tape would show that patient's cardiac rhythm through the entire day – along with all sorts of noise from the times the patient scratched at the wires, took a shower, and so forth. Then we would read this into a computer, and my job was to write software to reduce the full day of cardiac data to a concise report: everything was fine until 6:20pm when the patient had a series of brief episodes where her heart raced for as long as a minute at a time. The doctors would then try to correlate the influence of the drug with the manifestations of problems seen on the tape: perhaps, 6:20pm was precisely the period when the last dose of imipramine was wearing off.

Because we needed to do this on a large scale – clinical trials of new drugs typically involve hundreds or thousands of patients – the numbers of tapes to be analyzed was staggering. The situation was made for computers: if we could write software to rapidly and automatically scan these tapes, the drug evaluation would go faster and would be cheaper, letting them develop new and more effective therapies faster while ruling out the ineffective ones. But all that noise on the tapes – the scratching, the showers – made them remarkably hard to interpret. These weren't the clean, sharp images one sees when a television drama takes us into the cardiac intensive care unit, and we wait with bated breath to see whether the patient will open her eyes or the lifelines on the monitor will flatten. Rather, they were full of static and jumped around the screen and were often quite a mess. Each tape was a challenge in and of itself.

Software to try and solve this sort of problem uses what we call artificial intelligence techniques. The idea is to somehow capture the human process by which we as people make sense of these things, encode the rules into a form of logic that can be executed mechanically, and then turn it lose on the data. But while a team of world-class cardiologists can make sense of just about any cardiac rhythm, no matter how noisy the tape recording, explaining to a computer how they did this is quite another matter. At a glance, it would seem as if a computer would need to be able to see and think nearly as well as a person to solve such a problem (and a lot faster, since the goal was to analyze a 24 hour tape in a few minutes). After all, when we look at these tapes a person sorts out the cardiac signal from the background noise, realizes that this mess over here is from someone tugging on the leads, but that mess over there is a sign that the patient's cardiac problem is reasserting itself. Moreover, the cardiologist brings a vast arsenal of knowledge to the table: he or she knows everything there is to know about cardiac disease and the impact it can have on an electrocardiogram, and those who learn to interpret these signals pick up much of that arcane knowledge as well. Each cardiac condition has its associated symptoms, and one can sometimes recognize a peculiar glitch as nothing more than a glitch, or realize that a subtle distortion of the recording really has very serious implications, by drawing on this background knowledge of just what the patient's previous history has been, what her more recent problem seems to be, and how the problem was treated on the day the recording was made.

This is not the sort of information that modern computing systems are able to represent or work with. Perhaps you've read about the astonishing progress made by artificial intelligence researchers on problems like playing chess, but none of that work carries over to the sort of issues seen in analysis of an electrocardiogram. There has been little success in solving such problems, despite perhaps 30 or 40 years of research on them. Instead, computers have advanced by means of specialized, computer-oriented approaches to "vision" and "understanding". And this is the tack that I took when I worked on the problem at the time.

Basically, I came up with a bunch of rules-of-thumb that had very little to do with what doctors and other experts do when they look at these recordings. My rules looked for little spikes in the signal and measured distances between them, and through a remarkable number of contortions of this sort, came up with a surprisingly good interpretation of the cardiac data. The computer actually needed help – a person had to run my program and correct it when it made mistakes, but the speed was good, and with luck, an entire 24-hour tape recording could be analyzed within 20 minutes or so, giving what seemed to be a very accurate result. The doctors were thrilled.

I myself was very troubled, however, and ultimately left the field because I was unable to overcome the sense that I had built a computer program that was little more than a clever liar. Or perhaps "clever" is too generous a word, because it implies intelligence. More honest would be to say that it was a hugely complex, mechanized liar. True, my software did a pretty good job of rapidly reducing this huge pile of information to a few numbers the doctor could study. But it was also prone to weird, systematic mistakes –errors that no doctor would ever make, and it was quite capable of making these kinds of mistakes again and again if a patient's cardiac recording had the misfortune to fall outside of the norms, and the person operating my program wasn't astute enough to notice the problem.

I recall that the doctors, hugely impressed with the system, insisted that really, it did an astonishingly good job – some even argued that we should eliminate the operator entirely, so that the computing system could be used in the Cardiac Care Unit to monitor severely ill patients (and this was not just any CCU – Columbia was, and still is, among the very best in the field, so the CCU regularly treats some of the world's most famous heart patients). Fortunately, Doctor Bigger was much too knowledgeable about the limits of computers to ever casually take such a step, and in any case, we lacked the resources to take on the CCU monitoring task. But this came as a relief. What worried me was that his colleagues were naive about computers and their limitations. They were deceived by appearances. If my software did a remarkably good job on ten recordings in a row, they concluded that it would work well on all of them. Yet the reality was quite another matter: in fact, the eleventh recording, looking quite similar to the other ten, might be botched from start to end.

So here we face an ethical dilemma, although much less dramatic than some of the ones cited earlier. On the one hand, at that time there few options for figuring out how these long-term cardiac medications really worked. One could spend a fortune to have a trained technician sift through the entire 24-hour tape recording, at a rate of perhaps one or two per day, but the cost of such an approach was prohibitive. So, lacking the computer, drug studies had a big subjective component – the physician's impression of the impact of the drug on the patient carried great weight, despite being based on such things as the patient's own impression of whether the drug had helped reduce the frequency of such and such a symptom.

On the other hand, one had the approach facilitated by programs such as mine. The software looked very slick, ran at impressive speed, and often did an astonishingly good job – if the signal was clean and the patterns were easy, my program really nailed the thing. The kind of data that emerged was of enormous value in putting a quantitative underpinning on the table, so that drug evaluations no longer relied completely on impression. Yet the doctors using these systems were lulled into complacency, thinking of the analysis program as if it encapsulated what they would recognize as sound cardiology. And in reality, the program was little more than a clever simulacrum, mimicking the actions of a human but with none of the human's intelligence and reasoning ability at all.

At the time, I tried to be a good engineer – I worked hard to improve my program, and to introduce better ways of interpreting the signals without the knowledge available to an expert. As it happens, over the ensuing twenty years or so, this approach has paid off – modern systems, at the time of this writing, really do a fairly good job. Of course, the recording technology itself has also vastly improved, so noise levels are much reduced, and these days there are special hardware devices to assist with signal processing; we had nothing like them back in the early 1980's. Still, the problem is certainly not solved, because the computer has no real understanding of the task it is performing. From the 1980's to the present, most advances in "artificial intelligence" have had nothing to do with representing the kind of knowledge that a doctor uses to interpret a complex electrocardiogram. So, while many things have changed, you could probably claim that fundamentally, the systems have simply become better and better at fooling the user. Presumably, nurses are well aware that certain patients are "hard to monitor" because the systems "make a lot of mistakes" for reasons that are just not clear. Moreover, even if the ultimate solution to this problem were to surface today, more than two decades later, that wouldn't respond to the issue of how one should have dealt with it back then, in the early days, using systems that genuinely didn't work very well. This problem seems paradigmatic of the broader one. In general, does eventual progress or even success retrospectively validate aggressive use of an inadequate technology in the early days?

I don't know where I ultimately come down on this issue. Presumably, we should use the monitoring systems but should also make a point of training the users to ensure that they will be aware of the limitations of the technology. At the time, I might not have had so much discomfort if the typical doctor had shown more awareness that the technology was, fundamentally, very limited. I think that what troubled me was the sense of complicity in deception: seeing that some of the doctors were deeply naive, and realizing that as long as I continued to work on these things, I was contributing towards their misimpression, like it or not. It seems a bit insipid to say that "this doesn't work very well, so use it carefully." Yet isn't that what we were doing, twenty years ago?

And so I moved on to computer networks, a field that seemed relatively devoid of ethical dilemmas, where the problems seemed to be solvable and funding was easier to come by. Of course, the trends caught up with me, for now computer networks raise many of the same issues, albeit in very different forms!

The Golden Tower

ike most of my colleagues here at Cornell, the decision to pursue an academic research career wasn't obvious. I had been working on various kinds of medical monitoring software systems, and while I had decided not to continue work on electrocardiogram interpretation, there were other medical computing problems, like medical database systems and systems to assist physicians in interpreting medical images, that seemed at least as interesting. With this in mind, after earning my doctorate from U.C. Berkeley in 1981 I spent six months in Europe, working with a good friend in the field. But the long-term job opportunities in Europe at that time weren't exciting, and I found myself on the market in the United States early in 1982.

I interviewed with a half-dozen research laboratories and advanced product development groups in companies, mostly in the medical arena, but only applied for three or four academic positions, and these at the strong urging of friends of mine who were convinced that I should give academic life a try. From my perspective, I had had enough of school, and the idea of juggling so many balls at one time – teaching, hunting for research funding, advising students – didn't hold much appeal. My parents are both professors and I thought I knew what academic life would be like. Obviously, academia brings freedom not found in other kinds of jobs – one sets one's own research agenda, and there are long breaks when we don't have any teaching responsibilities and can devote ourselves to research. But pick the topics poorly, or stop publishing the kind of cutting-edge research results that make the community sit up and take notice, and you can easily find yourself out of a job or relegated to some sort of academic dustbin.

When I found myself with offers both from an interesting medical computing company and from a few strong universities, I was really torn. It seemed clear that in either setting, I could do exciting work and enjoy myself, and in the long run, the medical computing opportunity seemed like it might pay a much higher salary. But after a lot of soul searching, my wife and I decided to give the academic life a try. And so I became a junior faculty member in the emerging area of computer networks and distributed computing, while she began a program of studies that ultimately led to a medical career.

In those early years at Cornell, the field of computing systems was white-hot and I found myself competing with some of the very brightest people in computer science as a whole. Getting papers into the top conferences and journals was a tremendous thrill, and while the field is no less acrimonious than any other competitive discipline, it was also a lot of fun to come up with new approaches to hard problems and then to demonstrate that they really could work. The Department of Defense agreed to fund our project and suddenly, we seemed to be swimming in money – DARPA has always been generous in support of work felt to have potential military importance.

But in all of this, there was a background reality: that in choosing academics over industry, one was walking away from much higher salaries. After all, the fate of my own software (hacked out late into the night) was to be distributed free of charge from an Internet site we maintained at Cornell. It was exciting to have some users – at one point, we estimated that nearly 1000 people had "downloaded" my system and that we had perhaps 100 active users. But while this brought academic visibility and the chance to travel to some really interesting places, the numbers were tiny in comparison to the size of IBM or Microsoft's markets.

For me, the first sign that things were changing came when my phone rang one afternoon and I found myself talking to a person who seemed absolutely infuriated. I had no idea who he was, or why he should be quite so upset, but as the conversation progressed it became more clear. It developed that he was Vice President of a major New York City brokerage firm, with responsibilities for information technology. He seemed to be under the misimpression that I was associated with a company that had sold a defective product and was refusing to provide support or other consulting services, and he basically was calling to read me the riot act before blacklisting my company on Wall Street.

"Hold on a second!" I finally managed to interject. I didn't have a company or a product – there must be some confusion, perhaps because my free software distribution from Cornell had a similar name to a product of some sort that they had purchased elsewhere.

After profuse apologies (these Wall Street types have a way of turning emotion on or off – or at least, anger on or off), the situation only became more confusing when further dialog revealed that the software he was upset about was something I had developed after all: my free software system. It seemed that some of his people had downloaded it from the network, and were using it in a financial risk-management application. So far so good. But then the brokerage decided to use this system on a worldwide basis. Unfortunately, when they had tried to scale the application up, by running more and more copies of it, some sort of bug crept in and the application would freeze up or crash (not a surprise, because at Cornell I only had a handful of computers for use in testing, and his plans involved running this application on hundreds of machines). So now I found myself in a position of apologizing – I was gratified to have users who might even consider doing such a thing, but not at all surprised that it didn't work.

"So look," said my caller, who now seemed perversely calm. "Suppose we could put five-hundred thousand or a million dollars on the table. Could you make this thing work?" Now, at Cornell, my annual salary at that time was just over sixty thousand a year. Moreover, to convince the research community that my approach to reliable computer networking really worked, I needed this kind of real-world example. So here was one of those rare opportunities to make everybody happy (and make a bit of money too). It took a few weeks, but I soon found myself at the head of "Isis Distributed Systems", a small Ithaca-based company, employing me and two of my colleagues.

Time went by and the company managed to deliver (more or less) what our customers wanted. We closed bigger and bigger deals – this is how I played a role in developing software for the New York Stock Exchange, the Swiss Exchange, and the air traffic console project of the French air traffic control system. And we also had lots of less flashy customers.

Cornell was relatively happy with this sideline, provided that it didn't slow us down as researchers. In fact, it had the opposite effect: through the company, my research group gained credibility that we probably couldn't have come by in any other way. We managed to work experiences from our corporate lives into our research papers and choice of topics, and were able to justify our research as a response to the real needs of real networking users. I found myself spread a bit thinly, but there were only a few periods when the situation was really intolerable, and these were solved in the nick of time by just hiring more people and spreading the load. Soon we had fifteen, then twenty employees. Several of our customers left their jobs to start sales and support offices for us, and in no time at all, Isis Distributed Systems had offices in New York, then Paris, then San Francisco and Austin.

During this period, the company wasn't exactly a financial home run. Not only did I not pay myself a salary for the first two years, I actually ended up loaning money back to the company. I suppose we were terribly naïve – ten years later, it seems obvious that we should have gone to Wall Street and done an IPO, but at the time, we assumed that all the red ink would make that an impossibility. To make matters worse, one day we were suddenly sued, by an old friend of mine who had gone into industry and competed with us for some kinds of sales. His company claimed that we were infringing a patent they held – the case ultimately settled out of court, but only after years of costly litigation. At the start, faced with the prospect of millions in legal expenses, my company seemed to be at real risk of failing. And so it seemed like a godsend when we were approached by Stratus Computer, a hardware company in Boston that specialized in ultra-reliable computing systems. They were flush with cash, and wanted to acquire a company in our area, and it didn't take long for us to agree. Even after taxes, all of us did well financially – and so it all seemed worthwhile.

As it happened, the acquisition ran into problems. Our focus on software never matched well with the Stratus emphasis on hardware, and we had trouble developing a coherent business plan. Stratus closed down the Ithaca offices, and many of my friends were forced to move to Boston, or to take other jobs. Some of the big customers ran into scaling problems, similar to the ones encountered by our very first banking and brokerage customers, but now on such a large scale that fixing them took a tremendous amount of time and effort. Within the technical group, we began to argue that the system needed to be redesigned from scratch to deal with these very large-scale applications, but Stratus needed to make money out of the existing product and insisted on marketing it even in settings where it didn't work very well. Finally, just as the new product was ready for its launch, Stratus itself was acquired, than its new parent was acquired, and then it was spun back off. And, somewhere along the way, Stratus decided to pull the product off the general market. I had been consulting for them, but it seemed like time to move on. And so my experience with Isis ended.

As one of those who profited from the commercialization of an academic research project, I turned out to be representative of a whole generation of entrepreneurs with one foot in the hallways of the University and the other in Wall Street. It wasn't obvious at the time, but Isis was one of the first of what became an avalanche of companies spun out of the systems research community into the commercial world. Over the ensuing five years, the technology market suddenly exploded with the introduction of Internet Web browsers, and this was followed by some stunning financial deals. One of the professors with whom I had worked at Berkeley launched a database company, then sold it for $400M. (The company that bought his company, however, promptly had financial difficulties and "tanked," as the technology crowd would put it, earning my advisor a nasty lawsuit in which he was both defendant and accuser, the former as a member of the parent company's executive team, and the latter as a stockholder). Netscape went public with a boom, to be followed by company after company, each with a higher valuation. Jim Clark, a former Stanford faculty member, made billions and became the darling of the newspapers and financial magazines.

All of this excitement has had a tremendous impact on academic life. When I launched Isis, and indeed right up until I was totally out of the picture again, the company was an unusual thing – most computer science researchers in my area wrote papers, gave away software, and pursued consulting jobs on the side to bring their salaries into a competitive range. Suddenly, just about anyone in my area found that just about any idea might prove to be the seed of a multi-billion dollar company, at least in the eyes of Wall Street. Literally dozens of these companies were formed, and the halls of the conferences we attended were increasingly filled with people talking about stock options and IPOs and big deals.

The best research departments – Cornell included – found themselves under siege. More and more faculty members were lured out of academics into startups, and more and more graduate students were confronted with the choice between seeking a PhD degree or leaving early and getting rich. Few return after making their millions, or billions. Moreover, for those of us on the front lines, it became clear that there was less and less substance to these companies. The research community seemed increasingly populated by a form of new-age con-artist, focused on generating the maximum amount of publicity with the minimum amount of effort, doing the IPO, and hopefully managing to convert at least some of the inflated gains into real money before the public sours on the sensation of the moment and the stock deflates.

All of this creates a strange mixture of pressures. In my own case, the climate more or less forced my research group to start another company. Over lunch, my colleagues and I analyzed our options and realized that in this situation, there was simply no other option. So far, we've tried to keep our vision modest, and we're hoping to build a small company dealing with a few large customers. Reality, though, tends to intrude: it is hard to have a technology company and not to dream of IPOs with billion dollar valuations. The more people we hire, the more we find ourselves acting like any other company; yet-another-Internet-play, forced to accept this role, although with our own special angle. One has the sense of being on a train accelerating down a track – you can't steer the thing and have absolutely no idea what lies around the next curve. Yet there isn't any viable alternative. Any other approach would probably break up my research team (since people would otherwise find the lure of industry irresistible) and could also threaten our ability to have real impact (these days, impact comes from major users, and they expect to deal with companies and support infrastructures).

Working simultaneously at an academic job and as an executive of a start-up company creates the potential for serious conflicts of interest. Cornell, like any University, has a policy governing "do's and don'ts" for people who consult or are involved in outside ventures. The basic philosophy is to require a full and honest disclosure of situations that could possibly be perceived as a conflict of interest. The University then sets up an impartial monitoring structure and, in extreme cases, intervenes to avoid potential abuses. Our system is relatively tolerant– many universities require that faculty members who have an executive role in a company go on leave of absence, recognizing that once individuals cross the boundary into the private sector, they rarely return. But Cornell has the advantage of being situated in upstate New York, where the whole venture-capital startup frenzy is somewhat at a distance. Cornell has rarely suffered serious abuses of its more permissive system, whereas some of our peer institutions in Boston or the Bay Area have been forced by bitter experience to lay down much tougher lines.

When I talk to people outside of Computer Science about the "dual life" of the entrepreneurial researcher, the first reaction is often to criticize the principle of accepting public research funds in support of work that might seed a company and hence reward the researcher with a rich payday. Yet the picture is not so simple. Most academic research, including everything that my group does, is placed more or less directly into the public domain. And while a skeptic might speculate that one first tosses a monkey wrench into the works, it turns out that there is little reason to do so – and every reason to do the opposite.

On reason for this is that research prototypes of a new technology are rarely suitable for mission-critical use in production settings, as those early users of my first Isis system discovered. I didn't need to toss in a monkey-wrench: the quality of academic software is simply below the expectations of demanding industrial settings. Even the best academic work typically takes the form of a proof of concept. We can also seek patents on our work, in the University's name. But, doing so doesn't preclude using the same idea in your own company: the University is happy to license any patents it obtains and rarely drives a hard bargain, since it has no products or market to protect. So, academic software finds its way into the public domain, and nobody loses when this happens.

A second type of reaction, also typical, is that it is inappropriate to use federal funding in support of work that might have any sort of commercial value. But this perspective is counterbalanced by the many benefits of having the government "invest" in technology. In the United States, we are enjoying the payoff from decades of technology investment, in the form of a massive boom of information technologies that are impacting almost every sector of the economy. These technologies have improved productivity, created huge amounts of new wealth, produced enormous numbers of high-paying jobs, and even generated unexpected revenue surpluses. The students trained by these research efforts are in tremendous demand by industry, and in fact are the dominant source of bodies to populate the executive and management ranks of this country's major corporations – the old ones as well as the startups. Were the government to invest primarily in technologies that have no commercial value, one would see none of these benefits. And while it makes a certain kind of sense to say that the investment should benefit anyone except the researcher who receives the funding, we live in a society strongly motivated by the potential of self-enrichment.

The more serious conflict of interest problems relate to issues such as working with one's colleagues under circumstances that might lead to conflicts between the goals of the company and the goals of the University, or to situations in which one pretends to be an impartial spokesman for the research community, and yet is quietly promoting a self-serving perspective.

I've had some experience with problems such as this. With my first company, Isis, one of my graduate students did some work for us as a consultant that became important to a major client. He decided to use this work as the basis of his Ph.D. dissertation, but joined the company on a full time basis before filing the final copy of his thesis. Now, this may seem like a minor matter, but a graduate student often has several weeks of full-time work to do after finishing all other requirements and before the thesis can be filed. Well, my student somehow wasn't able to find a few weeks to set aside for this task. And although I, in my role as his advisor, wanted him to finish, in my role at the company I needed him to work on the very priorities he was putting first. So I sat by and watched as month by month, some new emergency prevented him from finishing his thesis.

Moreover, he was very dedicated to the company and it would have taken a lot of pressure from me to convince him to do things differently! As it happened, he finally did finish the thesis, but it was filed some three years late. When I was later asked what sorts of rules I thought Cornell should be especially tough about enforcing, I felt that the one absolute boundary should be that no student should simultaneously have the same academic advisor and boss. Certainly, I wouldn't want to get back into a situation like that one: It came dangerously close to depriving a very strong student of his PhD, yet at each step, we had the best of intentions.

But there are other types of very serious conflicts that we've seen here at Cornell, and one reads of such things elsewhere. Not long ago, Cornell attracted a top researcher from industry, only to see him promptly start a company. We watched our new colleague become increasingly distracted, as his company demanded more and more time, and just as people began to wonder if he had come to Cornell merely as a springboard into private industry, he moved the whole company out west. The department lost a world-class researcher, but I can't say that I was all that sorry about the outcome. Many of us have some sort of outside venture underway, including me, but the atmosphere here at Cornell depends on the faculty putting the interests of their research, and the department, before other activities.

At MIT, two faculty members found their way into the news when one assigned a homework project that some students, working for the other's company (Akamai) refused to do, viewing the very statement of the problem as a form of corporate espionage. According to the Wall Street Journal, which ran a lengthy story about this episode, the two had competing companies, each focused on the best way to host Web pages in very large networks having hundreds or thousands of servers, and that the students in the class were also principals in Akamai. One of the two people involved in this tells me that the story was much exaggerated, and I have had enough experience with the press to believe him. Yet the story is certainly based on reality – the two companies really did compete, they really were founded by MIT faculty who work up the halls from one-another, and another issue reported by the Journal (that students in the company are advised by faculty members in the pay of Akamai, and that the conflict of interest was supervised by people with conflicts of their own) is certainly accurate.

As it turned out, Akamai went public with one of the largest IPOs in Wall Street history, enriching a huge wave of participants not just at MIT but also at other schools. Cisco later acquired the second company for eight-hundred million dollars. And the associated tidal waves of money even lapped at the doors here at Cornell, where some of our graduate students held stock options earned during summer internships at Akamai. I'm told that there was a rule in the stock grant contracts these people signed, requiring them to work a certain number of hours – full time in some cases – for periods of as much as two or three years to retain their immense gains. Not surprisingly, a considerable number of students and faculty members accepted this condition, and it is hard to imagine that many will be back.

In an overheated (and overvalued) technology sector, the very nature of the market is profoundly at odds with the basic expectations that we normally have of the academic community. At present, and perhaps for the next decade, entrepreneurs in Internet technology areas and related areas enter into business realizing that the job involves a mixture of showmanship and style – but relatively little substance. Companies with a handful of employees, no real products, and little prospect of ever earning substantial profits are given valuations exceeding those of major automotive companies. The academic entrepreneur learns to speak earnestly of the immense potential of the most minor ideas – the antithesis of academia, where work is supposed to be evaluated purely in terms of its merits, and where every claim must be independently reproducible. As far as I can tell, about 90% of modern Internet companies are the equivalent of the cold-fusion researcher from a few years ago: all hype, little substance. On the other hand, some subgroup within the remaining 10% will have a revolutionary impact, create the next generation of billionaires, and put healthy old-economy companies out of business overnight.

With the breakup of the Soviet Union, it has become common to read of the emergence of a new generation of Russian "robber-barons" – hugely wealthy beneficiaries of dubious deals, who manipulated the system to their own profit. Profligate spenders, they have scattered through Europe buying luxury items, Swiss chalets, and resorts on the Cote d'Azur. Here in the United States we pretend that our robber-barons are of different stock. Fortune adorns its covers with grinning professors, explaining how their brilliant insights are creating a whole new era of technologies and advances. Certainly, there is some substance to all of this – we really are seeing a massive technology revolution, and some companies have tremendous promise. But for every such real company, there seem to be dozens of also-beens. When a generation of professors create companies out of smoke and mirrors, swap up to trophy wives and private jets, rush off to confer with other billionaires on complex investment schemes, in what way are we sending the sort of message traditionally transmitted by the academic research community to future generations of innovators and scientists? What sort of role models are these modern faculty members, except of conspicuous consumption? The foundations of the ivory tower are being undermined.

The university itself can also end up in conflict of interest situations. MIT and Stanford are two institutions that have taken the lead by actually taking equity (stock) positions in companies founded by current faculty members. Such investments can be enormously rewarding, and there is a tremendous competition among venture capital sources to be allowed to make them. But these investments potentially compromise the impartiality of the University itself. First, one must wonder if the University is in a privileged position, being on the one hand in control over the faculty member (perhaps, even able to obstruct the new company on matters such as intellectual property ownership), and yet on the other offering a helping hand and an outstretched wallet if the faculty member is willing to play along. Worse still, that faculty member may not yet have received tenure, a decision that should be based purely on academic considerations. How impartial would the decision be? Suppose that an entrepreneur requests a reduced teaching load –with an investment at stake, the University has every reason to cooperate. When the University crosses the boundary to become both employer and investor, it too faces difficult conflict of interest problems that bring the very basis of the academic system into question.

Conflict of interest extends well beyond individual behavior as we look at critical applications of computing or of technology in general. Just as there is a long history of concerns that medical and biological researchers funded by drug companies have a motivation to misrepresent the efficacy of new treatments, one finds a similar issue when a computer scientist works as a consultant to such and such a company, and yet maintains a high public profile in the research community. Confidence in the very integrity of the system is eroded by such situations.

More broadly, it seems plausible that these trends, if they continue, could reach a point of bleeding the academic research community dry while also seriously damaging the public perception of the integrity and morals of the field. Of course, the public can be expected to feel some resentment simply because the amounts of money being earned have become so extreme – I can count perhaps a dozen hugely wealthy people, including a couple of billionaires, among my acquaintances. Billionaires are not much loved in contemporary society. Stories of this sort are at odds with the prevailing public view of the absent-minded professor, who may well be financially comfortable because of patents or spinoffs from work done in the past, but who is still viewed as disdaining money, having chosen a higher calling. Should this perception be seriously damaged, one must worry that the longer term public support for higher education and research could be placed at risk. Moreover, the continuing drain on the system, as the most talented young researchers yield to the irresistible attractions of the private sector, is gradually eating into the very core of the pipeline.

For the moment, the predations of the Internet frenzy seem not to extend beyond the computer networking, distributed computing and systems area, but it seems possible that the side-effects of changing perceptions of academics will soon extend to the entire discipline. Fewer and fewer new researchers are entering the field, and more and more of the new people are just visiting, on their way between getting their PhD and leaving to head their company. Those of us who feel more or less stable in academic settings, myself included, are nonetheless drawn into outside ventures that distract from one's work, and this is noticeable in a decreasing rate of publications from the academic research community, world-wide. Many of the top journals are literally running out of papers to publish, not because the work has gone elsewhere or because there are too many journals, but because the young entrepreneurial community finds that publishing in conferences is a better way to generate "buzz" about their work, and so they focus on getting work into conferences but rarely take the time to extend it into high quality, polished, journal publications. Anyhow, journals often insist that work be described in a level of detail incompatible with corporate secrecy. Over a six year period, when I was Editor in Chief of one of the most prestigious journals in the area of computer systems and software, each issue was a struggle. If "publish or perish" remains the mantra of the academic research community, it would seem that the community in this area actually has perished, or at least gone elsewhere!

One has to wonder how it will all play out. Today, the situation is reminiscent of the tale of King Midas, who wished that all he touched might turn into gold, and then died of thirst and starvation because gold, after all, is not edible. The roaring 20's, a period also known for conspicuous consumption, was followed by a profound collapse. Will the Internet boom calm down before it drains the fountain of innovation that emerged in the 1970's and 1980's and fueled the explosive expansion now underway? Presumably, we'll know within another decade. By then, those of us who remain in academics are unlikely to be productive enough to generate the sorts of crazy new ideas that really advance the field. Either we'll have been replaced by a new generation of bright, excited researchers, or the wellspring feeding this whole technology area really will have dried up. Meanwhile, industry continues to commercialize ideas, but there are fewer and fewer companies prepared to invest with a time-horizon of more than perhaps 18 months, and industry is not in a position to train the next generation of computer science researchers. So, if the current situation persists, there may not be anyone left to take the field forward as the next set of challenges emerges, leaving us with a community quite capable of doing "engineering" but less and less capable of "science".

Distance Learning

ne might expect that professors accept academic jobs in order to teach, but in fact the appeal of a job at a place like Cornell is the freedom it offers to do research. We get to work with fascinating students who bring energy, enthusiasm, and original ideas, and are constantly pushed to do something new, experiment with different perspectives on problems, to go our own way. When this succeeds and you manage to do something really original, the thrill is hard to beat.

Of course, one has to teach well – but we can't all be gifted teachers. The average professor's goal is just to teach very proficiently, and at least to be sure that we are exposing students to the most important material. After all, students don't come to Cornell's Computer Science classes primarily because the faculty are great instructors. Rather, they come because we have a reputation for forging new directions and for being at the very frontier of the field. They expect us to teach well, but also to share a perspective that can't easily be found at schools where teaching is the only priority, and the faculty members are so far from the research community that they've lost all sense of where the field is going, or even where it currently is centered.

Nonetheless, even teaching "well" is no small undertaking, particularly when one must also secure research funding, advise armies of students, and help with departmental administration issues. Moreover, teaching isn't really an inborn trait, although some people are obviously naturals. Over the years, the typical young faculty member learns to teach by watching colleagues, borrowing their notes, and slowly, with a great deal of effort, finding his or her own way to connect with a class, to pace a lecture so that it will stay exciting and interesting, to use a blackboard reasonably well. It isn't easy to learn to listen to the students, and to interact with a class.

Perhaps this seems obvious, and it would be for small classes. But at least in Computer Science, the problem is complicated by huge class sizes, even at an advanced level. My very first classes at Cornell involved an easy introductory course that was split into two "sections" (with identical lectures, homework and exams). There were a total of 1200 students in that class at its peak, about 300 in the early morning section, and the other 900 in the second section. I remember standing on a stage in front of a sea of anonymous faces, and needing to interrupt my lectures to ask the fellow with the dripping boots on the balcony to please take off the boots or at least stop dripping on the people down below, or to ask the owner of the enthusiastic dog to please take it outside.

I also teach a more advanced class to seniors – a tough class on a hard topic, where really working closely with the students seems like a very good idea if you want them to understand the material. This class had about 65 students the first time I taught it back in 1983; last year, it had 220 and although we've split it into two sections (spring and fall), enrollment may still rise to as many as 250. If I think back on the best classes I took as an undergraduate and graduate student, none of them had more than about a dozen students in the room. Thus, modern teaching is increasingly remote from what I experienced as a student. Economic pressures and the need to know some basic things about business and technology – not to mention computer networks and operating systems -- are driving vast numbers of students into very similar decisions about which courses to take. No matter what school you look at, the effect is that more and more classroom hours are being spent in huge classes.

If your reaction is that no university should tolerate classes of this size, we would have to agree with you. At Cornell, reducing these class sizes has been a top priority for as long as I've been on the faculty. But such things are easier said than done. We don't like to limit class sizes, so the only way to make a class smaller is to split it into multiple sections and this typically requires additional staff. My department is adamant about having faculty members do the teaching, so smaller classes would mean hiring a lot more faculty members. And this is where things break down: even hiring as fast as we can, my department has only managed to grow from 16 faculty members the year I joined to 32 today, almost

20 years later. This gets back to the lure of those vast piles of gold we talked about earlier!

Cornell is not the only school facing this type of pressure, although our counterparts have reacted in different ways. Some departments yield to the temptation of hiring instructors and lecturers to teach more and more of the classes, but when doing so, they lose the "edge" mentioned earlier. Some departments set draconian standards for admission to the major: perhaps, only people with straight-A grades will be permitted to major in Computer Science. But you can imagine how these kinds of elitist policies play out with the students, who, after all, are paying for their educations and feel some right to follow their interests. Some departments ask faculty members to teach large numbers of sections, but this ultimately erodes their ability to do research, triggering waves of departures and retirements. Another approach is to have large lectures complemented by smaller "recitation" sections, which is what we do at Cornell, but finding the people to staff the sections becomes a problem if enough classes get large enough.

Indeed, there have been studies here at Cornell showing that the great majority of students spend the majority of their time in huge classes, and yet paradoxically, that the majority of classes being offered by the University are quite small. The problem is that relatively few faculty members are qualified to teach the largest courses, simply because the topics can be so esoteric. My big course, "operating systems", is one that only five or six of us can teach. As it happens, the same group of faculty are also responsible for two other equally large courses, and we do like to teach graduate courses in our research areas too, from time to time. If you do the math, you discover that there is simply no way to break up those huge sections unless the people already teaching them agree to teach extra sections or to stop teaching those special graduate classes. Thus, Cornell is probably offering a greater number of small specialized classes and a greater variety than at any time in its history, and yet the average experience of our students is that they spend more and more time in huge classes.

Meanwhile, the trends are alarming. I mentioned that my department currently has about 32 faculty members. Cornell as a whole has about 2700 faculty members, so we represent about 1.5% of the total. But in recent years, as many as 20% of all incoming undergraduate students have indicated that they plan to major in Computer Science. So far, these numbers have dropped as these students advance in their studies. Just the same, if 1.5% of the faculty are expected to teach anything like 20% of the students, classes will be very large. Our view has been that given a choice between smaller classes taught by less qualified instructors, increased teaching loads on our current faculty members, and large but high quality classes, we should go with the latter.

So, what should the modern University do about this? Right now, the trend is to look to technology as the solution for a problem that, fundamentally, reflects the growing importance of technology in our society. The view we've taken here at Cornell, and that our peers are taking elsewhere, is that the "Information Revolution" will only continue and accelerate, and as this happens, it will become more and more important for our students to become proficient with technology and the information-oriented tools that matter in their areas of study.

This is not to suggest that Computer Science will somehow enlarge to encompass, say, Biology. The view is very much the contrary: that fastest-growing subarea of Biology will be Computational Biology, and that the fastest growth in Linguistics will be Computational Linguistics, and that in just about any field you might name, whether in Law or Languages or Art or Science, computational tools and new ideas about how to structure and organize information are yielding the most dramatic new advances. The Information Sciences and Digital Arts are clearly becoming critical to more or less every field of scholarly endeavor.

Such a perspective naturally leads one to the view that instruction in these kinds of tools needs to occur throughout the campus. It also leads to a complementary insight, which is that we need to use technology more effectively in the way we teach. If teaching remains so human-intensive but the labor pool remains so limited, continued growth in loads can only lead to a disaster. So the productivity of teaching simply has to rise through better use of technology.

But once one adopts this belief, it leads to a secondary insight. If we are going to bet on a huge burst in productivity associated with new ways of teaching that use technology more effectively, why not use those tools to extend the reach of the University so that our students might be able to take courses from their homes or offices—true distance learning, with the Internet carrying the lectures. For a cash-strapped institution, the lure of a new and potentially massive market is understandably irresistible. So we see what might be called a confluence of mutually catalytic trends: higher enrollments stimulating greater use of technology, which in turn enables the university to reach out to a much larger market. (This also raises all sorts of thorny intellectual property issues, but in the interests of brevity, I'll leave them for some other venue).

This vision of remote instruction is often called "distance learning", although it is unclear to me that remote students will actually learn anything out there. After all, my kids would happily watch television rather than go to school, but I am doubtful that they would learn very much if I permitted this. Nonetheless, if you study the remarks of the leaders of the nation's major educational institutions, you'll realize that distance learning has become a near obsession for the whole educational infrastructure. Just as the electric power industry is restructuring because of a popular perception that doing so is the right next step in increasing competition and profits, so is the educational industry poised to launch itself into in distance learning; an experiment that will surely transform education in America, whether for better or worse.

I don't want to become repetitious in this book, but one should notice that the issues we confront in distance learning are very similar to the ones seen in the other areas covered previously. In those other areas, we identified a pervasive tendency to simply bet on technology, for better or worse, in the belief that technology always rises to the occasion. So too in the case of distance learning. In fact, we lack the productivity tools needed even at the level of the basic classes a faculty member like me teaches here at Cornell. Somehow, from the recognition that technology could help us teach more effectively here at Cornell comes a series of increasingly large leaps of faith. First, we become convinced that the technology will soon emerge, then that when it does, it will also enable a solution to the even broader problem of teaching masses of students at remote locations, and finally, that they will have a good learning experience. The vision is certainly an appealing one, particularly for those who administer the biggest schools, but in leaping directly from vision to a commitment that may have irreversible consequences, we are engaging in another one of those big bets on technology, with consequences that are simply unforeseeable.

Yet having said this, it would be unjust not to acknowledge the administration's perspective. Here at Cornell the administration really feels that it has limited options. They see a problem that wasn't created by the university, but rather was imposed on it by a society that has embraced technology so suddenly and so fervently that overnight, all previous ways of solving problems have begun to pale beside the powerful new tools offered by computational and information-oriented researchers. The students are simply at the vanguard of an irresistible and inevitable development. So the administration sees itself not as originating a trend, but merely attempting to assert some control over it.

I'm reminded of a similar event from a few years back. You may recall that Microsoft was a bit slow to get started when the Internet Web browser phenomenon first occurred. The story goes that the catalytic event within the company was associated with a galvanizing memo written by Steve Sinofski, a Microsoft Vice President who happens to be a Cornell graduate. Steve was snowed into Ithaca on a recruiting trip, and wandered into the undergraduate computer laboratory to see what the students were up to do. To his shock, he discovered a whole room filled with students running the precursor to the Netscape Web browser. That very night Steve wrote an urgent memo to senior management: "Cornell is Wired!", Bill Gates bought in, and history was made!

To me, the analogy with our air traffic control scenario is overwhelming. Social trends have made certain technical steps inevitable, because there is simply no other way to respond to the trends. For better or worse, technology needs to find its way into the classroom. Distance learning, I suspect, will succeed or fail on the basis of its effectiveness, but larger and larger classes are here to stay, and fields really will be transformed by technology; this, it would seem, is simply a consequence of the fundamental success of new technologies in breaking through the limitations of older ways of expressing and solving problems.

In the end, I truly wonder where this will take us. I'm not one of those natural teachers you read about, and I have to put a lot of time into the job. I tend to use the blackboard when I lecture, old-fashioned or not, because for me, at least, it seems to be the most effective way to communicate with students – with high-tech slides I often have a sense of going to fast and losing my connection to the class.

But maintaining that sense of connection is more and more of a struggle: you want to get to know your students, and yet the soaring class sizes and the increasing physical size of the room and the crowd jammed into it is a real barrier that seems to be growing over time. So far, people like me have managed to keep up with a state-of-the art that changes at "Internet Time," and most of the students seem reasonably happy. But I find myself wondering if I'll be the next victim of this new dependency on a yet-to-be developed technology. They'll expect me to use it, but how well will it work? I guess we'll just have to keep our fingers crossed.

Fleeting Memories

hen writing this book, I had the frequent experience of turning to the Web as an information source – a natural thing to do if you are researching issues concerned with technology and the Internet. But if you do this, you'll make a surprising discovery: the Web turns out to have very little material that isn't "current" on it. While I can easily pull up pages concerned with today's breaking news story, finding information relating to things that are a bit older – say, an air traffic control outage from a few years ago – is very much a hit and miss affair. Nobody seems to be bothering to keep such information around and accessible.

This is troubling because we seem to be creating a whole generation of intellectual material that is certain to be washed away by the digital tides. Now, I don't imagine that we face the kind of cultural amnesia that might deny us memory of things that are really important. Yet the tools of the new cybernetic society, at least in these early days, seem to be particularly poorly suited for building up any form of record. One can always hope that a book, printed on paper, will survive even a new Dark Age. But what about a pattern of bits recorded on a plastic disk, with a lifetime of a few years, in a format unlikely to survive even that long? In the new world, anything that we don't explicitly preserve is immediately lost.

With the emergence of the Web, we've seen a nearly a decade during which each new Web technology was quickly swept to the side by a new generation of improved software. It seems entirely plausible that we are entering an extended cycle of innovation and transitions, during which much of the new intellectual content created may be outmoded by each succeeding generation of tools and content. This ceaseless shuffle cultivates the tolerance, and even embrace, of transience – as if this moment, now, were all that mattered. But when they finally sit down to write the history of the Information Age, will anyone remember how it happened?

The new, ephemeral media are having impact well beyond the Web. The entire publishing and media industry is endangered by currrent trends. Libraries are rapidly being outmoded – with the advent of digital libraries, one is increasingly forced to ask why the world needs more than *one* library, assuming that its Web site and Internet connection have adequate bandwidth. Publishers find themselves competing with vast quantities of Internet content – free content. Many academic journals are rapidly going out of business, as research moves online, but the preservation of online research archives is subject to the same concerns we just cited. There is ample reason to hope that the new technologies can play the same roles as did the previous generation of technology, but it is far from certain that this should be the case. Equally plausible is a scenario where the thing we have lost is replaced by a poor approximation, like a museum in which the old masters have been ripped from the walls and replaced by cheap prints.

The preservation of property rights in the new media is a particular subject for concern. Tradtionally, copyright was at the economic core of most media industries. If I like a book, I can't simply print copies of it for my friends and myself: I need to buy copies, and the income compensates the author, the publisher, the bookstore, and the others who play some role in the process. But when media moves to the Internet, the moment that a malicious customer downloads that object, unauthorized copies could escape.

As a technical problem, copyright is very difficult to enforce in our new Webbed world. Once information is placed into a form suitable for display on a computer's screen, copying that information is trivial. Since any form of media delivery over a network ends up with a display on a screen, we seem to be entering a period of time when the things that matter most will necessarily be placed into settings that offer very limited forms of protection for them. Moreover, the problem isn't limited to Web content: new films are being released on digital media, making it much easier to pirate them than at any time in the past. CDs containing music are, when you come down to it, just patterns of bits. Anyone with a computer and a CD reader can easily copy the information onto the Web, and we've already seen the emergence of a whole culture in which "ripping" music to share, free, with others is beginning to dominate other ways of obtaining access, like buying a copy of your own.

As easy as it may be to say that we should simplify legislate stronger protection for copyrighted materials (and I favor this), I wonder if the Web isn't going to force us to confront a more significant change in the economics of information and content. The problem is a paradoxical one: on the one hand, possession of information, easy access to information, and the ability to organize information represent a big opportunities in the new digital world. If, for example, you invest in technology and I am a renowned forecaster of technology trends, I should be in a unique position to market my talents to the whole world over the network. Yet the very ability to get information to my customers is also a guarantee that my less scrupulous customers can potentially copy it and plaster it on Web sites worldwide, stripping my special wisdom of any value!

There have been a number of technical proposals for protecting digital information. For example, we can introduce digital watermarks: secret signatures inside documents that identify the path by which the document was obtained, so that if a copy is released, one can potentially pursue the original recipient for damages. On the other hand, having studied the difficulty of creating secure networks and systems, it should be obvious that if information is put online, unauthorized access becomes a real threat, and when information is actually sold to large numbers of users, unauthorized access becomes almost unavoidable. Another option that has been proposed revolves around encryption; DVD disks, for example, are encrypted so that only holders of a secret key can access them. Unfortunately, almost as soon as the DVD encryption approach reached the market, tools for decrypting disks surfaced on the Internet. Law enforcement officials have had little success in preventing their spread. Anyhow, we get back to the problem that in the very last step, a legitimate user needs to display data on a screen or play music through the computer's speakers. As this occurs, there is always an opportunity for an unscrupulous user to make a copy.

You may have read about Napster and Gnutella, two Web sites dedicated to helping their users share music and other forms of digital media. These technologies were designed to maximize anonymity: in effect, they promote a world in which crowds of people independently agree to share materials, but where no individual has any real feeling of directly causing the trend or any particular sense of responsibility for the consequences. Yes, I've downloaded my favorite Elvis Presley recording, but I have no idea who's computer had the copy I just obtained, and they don't know anything about me. While the courts have imposed restrictions on the way that both sites operate, this was possible only because the developers of these systems wanted to earn money by operating the sites. There is absolutely nothing to stop the emergence of a completely free, completely anonymous, decentralized mechanism with analogous properties. When that happens, perhaps you'll find yourself reading this book for free, with no indication of how the first copy came to be put online, and no indication of the set of participants who played roles in getting a copy to you. Short of shutting down the Internet, there seems to be no way to prevent this.

These same trends also offer a very positive prospect. Throughout history, new forms of media have yielded new forms of art and new kinds of cultural opportunities. Surely, the Web will bring its own basket of creative ideas and experiences, some perhaps as rich and exciting and transforming as anything ever done in the past. We grow, as a society, by expanding our horizons: the Web offers an unparalleled opportunity to experience new ways of viewing the world, to form new kinds of communities that can flourish without the limitations of space and time, and that can break through traditional political, cultural and mechanical barriers. Who can say how valuable this will prove to be?

The Web and the associated Information Revolution are poised to have a transformative impact on the economic and social structures associated with creating, selling, and preserving information. Just as the erosion of privacy may far-reaching implications, so too may the changes in the way that information is handled. Hopefully, there will still be incentive in the emerging system for authors to write books, musicians to sing songs, and directors to create films. But we certainly face the possibility that the economic rewards for such activities could be substantially eroded over time. The Internet revolution holds vast potential. What a shame if, in changing the world, we also discard some of our most precious possessions.

The Road Ahead

hen I set out to learn more about the failure of the American air traffic control project, I didn't expect onto stray to a digital version of Hardin's commons. As an engineer, I would have predicted a different outcome; a picture of communities so drawn to individual profit, or in such a hurry, that they were cutting corners and using shoddy techniques, thereby endangering what the original DARPA study termed the "survivability of our nationally critical information infrastructures". But like the government, or the popular press, anyone who focuses exclusively on the technical side of a major technology project risks missing the point. The fundamental drivers of technology are often remote from the specifics of a project. And the key to the whole picture, it seems, is the realization that as the density and ubiquity of technical solutions rises to surpass a threshold – one that we seem to have reached – the network and the various computing technologies strung onto it begin to amplify even very subtle effects, so that like the herdsman overgrazing the commons, we confront the necessity for rules to control the introduction of new technologies into this shared, increasingly sensitive space.

The entire topic turns out to be rich with issues that invite scrutiny. Technology has driven the evolution of society, undirected, since humanity first emerged on the plains of Africa or Asia, and it is difficult to talk about steering its course in more than a very tentative way. Yet gentle interventions can have a dramatic impact. When we survey the prospects for new technologies, especially those associated with the Internet, we confront scenarios with significant potential for societal dislocation and inequities. Today, proficiency with technology is largely concentrated within white, middle- and upper-class communities and this creates a huge and unnecessary barrier that, over time, invites catastrophe. The explosive growth of the Internet economy is poised to disrupt the economic fabric of the media industry – first in the area of music, where MP3 players are already widely available, and soon in the publishing and movie industries. The huge stock valuations Wall Street is conferring on Internet companies have drained a whole generation of researchers and graduate students from our top research institutions, robbing the information economy of its "seed corn." Volatility associated with the inevitable collapse of the resulting "bubbles" has roiled the international monetary system. And the trends also embody low societal expectations towards engineers, who are neither trained to think about ethical issues, nor viewed as responsible for technical lapses with apparent ethical implications or overtones.

There isn't any reason for excessive alarm, because these trends have been in place for a long time, and yet on the whole, we bumble along. In fact, American society seems to have a deep-seated societal mindset that eventually emerges to create backpressure on trends that run in the wrong direction for long enough, as in the case of environmental issues. Yet quite a bit of damage can be done before this type of pressure becomes powerful enough to impact on technology trends or practices. With this in mind, I would argue that government does have a vested interest in steering the system, but with a light hand, focused on persuasion and simple attention to the issues rather than on legislation and mandates. Here I differ from Hardin, who argued that the tragedy of the commons must be addressed by "mutual coercion, mutually agreed upon." Perhaps we will someday be forced to accept Hardin's conclusions, but I would favor a much less coercive approach, in which we might try to steer without imposing a suffocating blanket of regulations and rules.

One place to start is by getting the people who actually build technology to think a bit more about what they are developing. Consider the nature of engineering education in the United States. A typical University, such as Cornell, offers two undergraduate educational tracks, one in Engineering and one in Arts. My department, in fact, offers both kinds of degrees – a Cornell student can earn an undergraduate Computer Science degree in Engineering (a BSE) or in Arts (a BA), and the Computer Science requirements are identical in both cases. What distinguishes the two degrees are the remaining courses the student might take.

An Arts student is required to take a variety of courses that make up a basic Liberal Arts education: courses in English, philosophy, art or music, and even the classics. Not every student takes every such course, but any student who graduates with a BA degree has received this sort of broad exposure to issues very remote from his or her major. In contrast, an Engineering degree typically permits exclusive focus on technical courses. The required non-major courses can often be in closely related fields (for example, a student majoring in Computer Science might fulfill these requirements with Electrical Engineering courses), and it is even possible to graduate a little early, in three and one-half years (instead of the normal four) so as to earn an additional Masters of Engineering degree, by taking a few extra technical courses.

One consequence of this pattern, which is common among Engineering programs, is that Engineers are encouraged to focus on technology to the exclusion of all else, while Liberal Arts students are encouraged to broaden their perspective with courses drawn from, well, the Arts. Even without a specific course in ethics (although a course in the classics would often have quite a bit of content in these areas), an Arts student is very likely to have an exposure to issues that are basically ethical in nature. The Engineering student escapes all of this. It is time for a serious rethinking of the nature of Engineering education. Indeed, I wonder if the entire concept of a separate Engineering degree should be reconsidered.

In the past, fifty or a hundred years ago, an engineer needed a great breadth of scientific background to function, so engineering programs of study emerged during a period when it made sense for engineers to invest their time studying physics, chemistry, mathematics, electronics and so forth. Today, it is not at all clear why a degree in Aerospace Engineering should be fundamentally different from a degree in Physics, yet today, the Physics student gets a generous dose of courses drawn from remote disciplines, while the Aerospace Engineering student can confine his studies to engineering courses with mostly technical content. Should we be surprised that physicists took the lead in fighting nuclear proliferation, while engineers obediently designed better rocket guidance systems?

Not long ago, I read an article in the *New York Times* about MIT engineering students who call themselves *Borgs*, short for "cyborgs", a term referring to a person augmented by machines (the Borg are also a race of aliens in one of the Star Trek series). Today, these students achieve this by continuously wearing helmets topped by cameras, eyeglasses with small embedded screens, microphones and earphones. The article makes it clear that the result can be disconcerting: they look rather like insects, with all of this plastic and metal, and act rather oddly, constantly peering off into space or mumbling with great intensity as they walk down the corridors of MIT's Laboratory for Computer Science and Media Lab. Now, I have no real issue with the idea that one should experience future technologies if one wishes to play a leading role in developing them. Yet something seemed to be lacking in the MIT approach to the whole question: the article said nothing about the broader implications of such technology. Would one want to live in a world populated by large numbers of Borgs? What impact does this type of continual connection to a computing system have on the mental health of the person wearing the headset? Apparently, the topic never came up.

The problem here is dual: on the one hand, without anyone asking, it isn't terribly surprising if a group of students, caught up in the excitement of using and developing futuristic technology, might neglect to think about the broader picture. But at the same time, society as a whole is unlikely to reflect upon the implications of a technology if someone doesn't at least frame the question, as is so often done when a new medical technology or drug emerges. Who would be at fault if, later, we discover that spending much time as a Borg is psychologically damaging? As we have learned rather painfully over many decades, what society doesn't know can hurt all of us. We should expect more not just of engineers, but also of the press, universities and the government.

What about the sorry state of projects like the air traffic control one, or the emerging mess in the electric power grid control area? Here, it seems to me, the challenge (for us here in the United States, that is) is to learn from the French, yet to adapt their approach to an American model. Clearly it isn't realistic to expect

American government to suddenly gain the technical competence seen at high levels in the French government; outside of a few agencies, like Nasa and NIH, the necessary culture is simply lacking. But I wonder if we couldn't institute a process analogous to environment impact statements as a way to achieve a comparable result. The idea would work as follows: when undertaking a project likely to have "pervasive public impact", like a rebuilding of some part of our nationally critical infrastructure, the government would be required to produce and publish a form of impact statement during the early stages of the project. There would be a period for public comment, and the government would be required to respond before expenditures on the project could begin.

How would this help? Well, if we look closely at an environmental impact statement, it turns out that the expectations are really quite high for such things. I suspect that this wasn't true at the very beginning, but as time went by, the preparation of impact statements became a rather refined science, and the expectations rose steadily. Similarly, even if the first of these infrastructure impact statements was somewhat ad-hoc, one would certainly see a rapid progression towards statements constructed along the lines of the early stages of the French air traffic control project, in which the preparation of the statement involves doing fairly detailed experimental studies aimed at demonstrating that the technologies planned for the system are adequate to the task and will be used in appropriate ways – in their "technical sweet spot," as it were. While this approach wouldn't prevent projects from making mistakes, it would raise the stakes and force a degree of advance planning that we currently seem to skip. The US air traffic control project would probably have approached the problem differently if it had needed to present such a study for public review (keep in mind that the reviewers would include the original companies that lost the bid – presumably, rather motivated and sophisticated critics). Even if this kind of infrastructure impact statement doesn't solve the problem, it would at least institutionalize a type of preliminary introspection that would surely benefit any every large project. Projects that failed to do so would answer to a potentially irate Congressional inquiry later, when great sums of money had been lost – a form of pressure that could be quite convincing, I suspect, for the typical government employee or contractor!

In a similar sense, it seems to me that we need to institutionalize a more proactive role whereby some form of organization might guide the evolution of the Internet. With all the excitement about the Internet, the public has overlooked the weaknesses of the technology underlying the new world of ubiquitous communications and interconnectivity. Yet the Internet is profoundly limited: it lacks adequate security and the infrastructure itself is unreliable, subjecting applications running upon it to the vagaries of fortune. Even very unsophisticated crackers are able to bring the system to its knees, and theft of private data is child's play. And the bottom line is that the current way of administering the Internet leaves nobody in the drivers seat, with real responsibility for making the network more robust.

We certainly could build a better Internet today: one protected against attack (meaning that attackers couldn't disrupt the basic ability of the network to turn names of Web sites into electronic addresses, or to "route" messages from your machine to those sites), would offer the user much stronger security, and would support the kinds of tools needed by developers building demanding, mission-critical computing systems. We could build an Internet capable of guaranteeing isolation between applications, so that the traffic associated with your decision to download a video won't prevent me from checking the status of my bank account, or my doctor from checking my blood sugars and adjusting my insulin dosages. And this new "Supernet" could be made to look very much like the current Internet.

Yet industry won't do this on its own, and simply legislating that the network should work better won't give us what we need. The unregulated, nearly chaotic world of the Internet strongly favors the status quo, because it includes dozens of competing companies that agree on just one thing: any new products should be compatible with everything that was done in the past. We could try to build a Supernet by legislation: "thou shalt build a better network", but the results would certainly be unsatisfying. On the other hand, the government could create a new DARPA-supervised development program, underwrite the necessary technology research and then open the doors to commercial use and finally commercial control of the new infrastructure. This type of non-coercive leadership would surely succeed: if we build a better mousetrap, the world really would beat a path to the door. But only the government could pull this off. If I were to create "Supernet.com" tomorrow, I might manage to raise large sums of money, but it would be extremely hard to create the sort of expectation of success needed within the community of potential users, hence the project might succeed technically but fail commercially. The government could manage these expectations in a way that would produce not just technical but also widespread adoption of a solution.

ᘛ

In some settings, expectations should go even further. As we look at emerging critical applications, the expectations of society towards those kinds of projects potentially span a range from simple statements of values to much stronger requirements, entailing actual liability. In this respect, it is very striking that the notion of product liability has been largely missing from the software industry since its inception. This is probably a consequence of the impossibility of building bug-free software. Recognizing that all software will fail from time to time, companies have chosen to write product licenses that basically exonerate the developer from all consequences associated with the use of the product. (Next time you install new software on your home computer, leaf through the license, and you'll see what I mean).

It would be unrealistic and wrong to argue that software products must be reliable in quite the same sense as, say, an automobile must be safe. Legislating reliability, at least for non-critical settings, would simply cripple progress and create a barrier to entry into the field that might protect the software giants against competition from innovative new companies. Yet I do think that one could imagine a body of product liability law for software that could define stronger expectations without seeking to legislate the impossible. As an example, suppose that a company which knowingly uses inferior development or testing methods were to bear legal responsibility for consequences of failures that might have been prevented had better methods or more thorough testing been employed. In settings where a product is marketed with the expectation that it might play some sort of critical role, for example in electronic commerce or medicine, this seems like a fairly modest approach to product liability, yet it could have important consequences over time.

At the outset, suppose that Sun Microsystems and Microsoft compete to sell software for use on computers in hospitals. This type of liability requirement would have little impact on either company, because both use similar techniques to develop and test their software. However, over time, the impact could become more pronounced. By setting the expectation at the level of "best practices", Sun would be motivated to improve the state of the art for product testing and quality in these domains. Perhaps, using their new Java technology, Sun would seize the high ground, asserting in advertising and marketing materials that its products are safer and more secure and hence more suitable for critical uses. Microsoft could continue basing its products on other programming technologies, bun Sun will have raised the bar: to respond, Microsoft would need its own ways of showing that its products are indeed engineered to the "highest tolerances". On the positive side, the company would also be *protected* by such legislation, because it could win a product liability lawsuit by demonstrating that its engineering practices were as good as any in the business. The point here is that "best practices" need not imply that the products in question are completely free of defects, but rather that the company has been diligent in identifying and attempting to remedy them.

The obvious rejoinder is that both companies can simply license their software as "inappropriate for use in critical settings," which is precisely what they do today. But if the advertising and prevailing practice makes is clear that any technology offered for e-commerce will play a critical role in the business applications that employ it, it seems disingenuous to on the one hand advertise the product as the next big step, and on the other hide small print in the license that says the opposite. I'm convinced that a law could easily be formulated to mandate not absolute levels of reliability, but simply "due diligence" on the part of the vendor in achieving industry-standards for quality needed for the contemplated uses.

In contrast, today the buyer of a technology agrees, more or less, that even if the wheels of the car have a tendency to fail off at highway speeds, the manufacturer is not in any way responsible, because when you purchased the car, you signed an agreement that any and all consequences of driving the vehicle were at your own risk. Yet that same manufacturer, in effect, advertises its cars with ads that depict it rocketing around mountain curves at high speed, while the announcer whispers of 0 to 60 acceleration in less than 5 seconds, and the supermodel in the jump seat licks her lips suggestively. "NetMobiles: Driving at the Speed of the Web." The disconnection between claimed properties of products in the industry and the legal licensing associated with them needs to be replaced by a reasonable notion of liability: not onerous, but one that simply demands that when a company makes an implied representation in its product literature or marketing, it also make a serious attempt to validate the adequacy of the product.

I've recently heard about a similar idea, advanced as a possible response to the denial of service attacks we discussed earlier when considering the Internet. As you will recall, these attacks involve an intruder who takes control of a collection of computers, then uses them to launch a program that overwhelms a Web site or network with seemingly legitimate requests. The difficulty is that the requests look quite normal, the intruder is rarely using the computers in question when the attack occurs, and moreover that the systems under attack rarely employ state of the art defensive tools. The trouble-making messages can be blocked, but to do so, the targeted organization must install security software before its machines are brought back online.

The idea now floating around is that insurance companies might begin to offer insurance policies covering loss due to such attacks, provided however that the company buying the policy takes measures that the insurance company considers adequate, presumably by pre-installing these sorts of defense tools. The government might then exert a little additional pressure by requiring that any company selling services to public agencies obtain this form of insurance. The free market would dictate the costs of policies and the kinds of defensive mechanisms available; presumably, the most cost-effective and technically adequate solutions would eventually dominate. Over time, it is easily to imagine that the desire to minimize insurance premiums would motivate companies to deploy more and more powerful security mechanisms.

Recalling the idea that we now live in a digital commons, I would suggest that this is the sort of policy that makes the most sense for protecting our shared environment. The degree of coercion is relatively modest, and a tremendous amount of leeway is preserved for the market to exercise its magic: presumably, there would be a great number of insurance vendors in this market, and a vast array of security products. The structure creates a market incentive for the participants to take security seriously, but also to do so as inexpensively as possible. And there is room for innovation: build a better security product, and your customers will flock to you in the hope of reducing their insurance premiums. Such a policy appeals because it draws upon the special characteristics of the digital commons; the very mechanisms that create the commons operate in favor of the success of this kind of solution. In contrast, if one imagined a law that simply states "your systems must be secured", it seems clear that the market incentive is to evade the requirement at minimum cost, and one must imagine a complex government certification process and endless legal disputes.

Public policies could also help in other respects. We talked of some of the very serious consequences that might follow from widespread collapse of intellectual property protection for music, films and books. At the core of the problem is the assumption that if a person downloads, for example, a recording from a Web server, the legal responsibility for licensing fees rests with the owner of the server. This creates an incentive to locate a server in some obscure country where copyright law is unenforced. But suppose that the law treated such an incident much the way that speeding is treated: the person who downloads the file might be subjected to fines, perhaps substantial. It would not be all that difficult to develop law-enforcement tools for the Internet, which would randomly sample network activity looking for illegal downloads. While such a technology would have limits, MP3 players and similar technologies might be designed to warn the user that their action was potentially illegal and subject to penalties up to and including fines or imprisonment. None of this is any different from the warning or rules associated with rental videos, and in the case of videos, the rules seem adequate to protect the industry.

Obviously, such a vision raises issues. We allow random spot checks to detect drunk drivers. But how would we feel if the FBI announced that it was developing software to randomly check for copyright violations in Internet traffic? Most likely, people would immediately worry that the FBI (in a return to its Edgar Hoover proclivities) might also use that software to monitor political opinions: software that can do one kind of monitoring can probably do the other kind as well. Yet since we probably will *some* kind of monitoring, it seems to be time that the courts and government tackle these problems, writing laws that would set clear standards both for law enforcement, and also for protection of individual privacy. Lacking such legislation, we encourage a free-for-all in which the rights of both the author and the network user are being trampled. I'm not fond of excessive government regulation, but on the other hand, sometimes the government simply needs to step in, and I think we are reaching that point today.

The broad philosophical basis of laws directed at technology needs to be reexamined. Today, much as in the period before environmental issues first came to the political forefront, the downstream costs of technology are rarely charged back to the originator. Yet the failure to correctly attribute costs, and to hold technology ventures to the same standards applied in other settings, seems to be at the root of the most serious potential problems one can identify with many emerging technologies. As appealing as it may be to get all of one's music for free, the fact remains that in doing so, one is depriving the artist of the opportunity to earn income from producing music. When we allow a company to make strong claims about its products, and yet to disclaim responsibility at the moment that those products are actually delivered to the customer, we merely create an incentive for companies to exaggerate claims while saving money by using inferior development processes. The overriding principles governing financial responsibility for technology are clearly out of balance.

Better policies would require that government adopt a new and more sophisticated approach to technology. Today, the roles of government revolve around the procurement of technologies needed in certain settings: air traffic control, for example, or military applications. Where a broader societal interest in the *way* that technology evolves can be identified, it is rare for government to adopt policies reflecting that interest – rather, the general approach has been "full speed ahead and damn the torpedoes," presumably under the theory that anything giving the United States a technology lead in some sector is beneficial to this country.

To a great degree, this philosophy has been successful, yet it has also been responsible for many of the more egregious lapses of foresight, and for the development of many of the most troubling of contemporary technologies. As we work to foster sensitivity to impact of technology on society, there is a growing need for government attention to such issues. Moving as fast as possible may not always yield the best outcome. Government, and the engineering community, needs to inculcate a culture of quality, which in some ways may need to displace the current culture of aggressive forward movement.

The world's governments could also go a long way towards ensuring the presentation both of the past and of the present. Today, various national libraries are charged with obtaining copies of everything placed into print. Why not broaden this charge: each government could take on the mission of placing everything onto digital media and providing access over the Internet? Not just books, although one certainly would want this to include all the books in the vaults of the Library of Commerce. Such a challenge could extend to medieval manuscripts, early films, and even to certain kinds of contemporary Web pages. Let the government step in where industry is very unlikely to do so, and we can avoid a catastrophic loss of knowledge, while also creating national resources of incalculable value. I doubt that a Digital Library of Congress would even cost very much.

Antibiotics promised to eradicate infectious illness and the green revolution to wipe out hunger. Yet we've abused antibiotics to the point that we face a new wave of untreatable infectious diseases, and the green revolution has given way to lethal red tides. Such things don't need to happen, and now, with the writing on the wall, we need to ask if we will permit similar ravages at the hands of the most promising new technologies. I don't know about you, but I am not eager to live in a glass house, surrounded by neighbors with the electronic analogs of highpowered telescopes and microphones, opening my mail, studying my bank accounts and chatting about my medical records. We should certainly seize the promise of technology. But not in a manner indifferent to the side-effects of our choices.

It is easy to fall back to the old adage: if it isn't broken, why fix it? Hardin's essay offers one answer: on the digital commons, the question is forced by our changing needs and uses of the tools that support our society and world. On the digital commons, your actions may impact me in ways that would not have been the case when the density and degree of interconnectivity of technologies and systems – and social systems – was below some critical threshold; a threshold that we now have crossed. In this view, the system didn't break because it changed in a fundamental way, but simply because it succeeded. We came to rely upon technologies more and more heavily; to use them in more and more interconnected ways, and with this increasingly tangled Web of dependencies came a new phenomenon, one which we as a society must now confront.

Do machines exist to serve us, or are we at the mercy of technology trends outside of our direct control? One might imagine the answer to be obvious, but since the dawn of the information revolution, the trends have greatly outpaced our ability to control or manage them, to such an extent that they truly have gained a life of their own. Not all change is for the better, and this is as true for software and computer networks as it is for antibiotics, pesticides and genetically modified foods.

Sooner or later, I would suggest, we'll be forced to change our approach to technology. If we allow the advance of technology to be ruled purely by financial considerations, the market will inevitably trample many of the basic principles and expectations of society. And so one can easily foresee a time when at least in some modest ways, government and society will assert its prerogatives. But if this is, indeed, inevitable would it not be wise to see it happen sooner, rather than later? In the words of the old saying, if not now, when?

The drama and excitement of our times cannot fail to capture one's attention. This is a unique period in history, when the emergence of a series of new technologies – driven by Moore's law and likely to continue – are revolutionizing almost every enterprise we hold dear. A further revolution awaits, as the "new biology" emerges from laboratories and begin to impact health and lifestyle choices for the average person. Confronted by these developments, it is time to also recognize that change is not always for the best, and that far too many technologies have had serious negative consequences. Unless we learn from the past, and in particular learn to anticipate future issues with ethically balanced, sophisticated responses, we face wave upon wave of disruption, the loss of societal guarantees of great importance to all of us, and damage to the very things that have brought us to this juncture. And so it is time to chose the other path: the path of informed caution, not through a reactionary, Neo-Luddite rejection of technology, but by encouraging progress in ways and areas responsive to the greater good of society as a whole.

In his oft-quoted poem, Robert Frost wrote: "*Two roads diverged in a wood, and I – I took the one less traveled by, And that has made all the difference.*". Which shall we take?

Acknowledgements

I am grateful to a number of people who read early drafts of this book and provided helpful feedback and comments, as well as to the many who offered their thoughts in casual conversations on the topic. Particular thanks are due to Tibor Janosi, Jim Rothenberg and Zellman Warhaft, for their very detailed reading and suggestions. Each time I spoke to one of them, I ended up rewriting large parts of this book. I'm also grateful to Bob Constable, Fred Schneider, Charlie van Loan and Wayt Gibbs; their insights and perspectives encouraged me to write this book and shaped my own views. Finally, I want to thank many others who worked with me on the topics described here, notably Robbert van Renesse, Werner Vogels, Keith Marzullo, Ozalp Babaoglu, Sam Toueg, Robert Cooper, Andre Schiper, Bob Thomas, Massoud Amin, Anjan Bose, Jim Thorpe, Anita Jones, Howard Frank, Teresa Lunt, my colleagues both at Cornell and on the 1995 DARPA ISAT study, and the many graduate students with whom I've been privileged to work over the years. I also want to thank the individuals cited in the text, who were kind enough to explain their work to me in terms that helped me understand the role of technology in their varied disciplines. Finally, I am tremendously grateful to my wife and family, my parents and my in-laws. Their support was invaluable.