



Woodrow Wilson
International
Center
for Scholars



**Foresight and Governance Project,
Woodrow Wilson International Center
for Scholars**

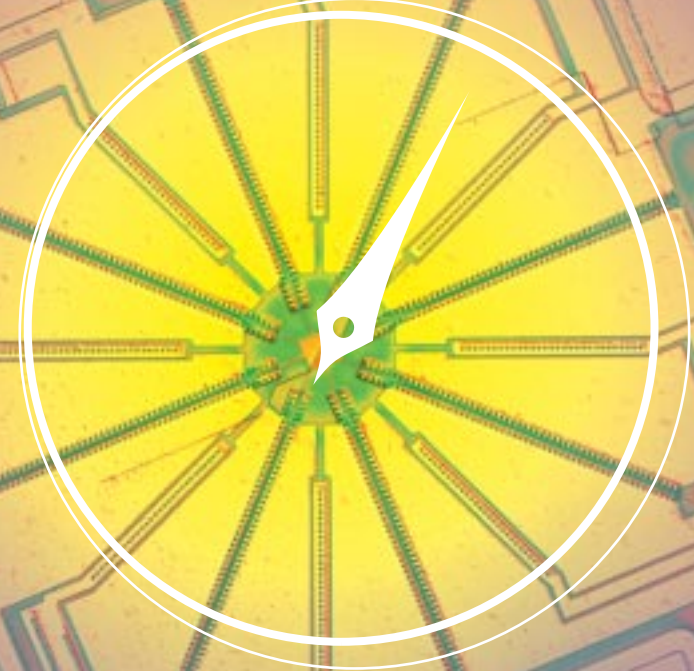
1300 Pennsylvania Avenue, NW
Washington, DC 20004-3027

(202) 691-4000

www.wilsoncenter.org/foresight
www.foresightandgovernance.org
www.ibm.com/ibm/gp

www.thefutureofcomputing.org

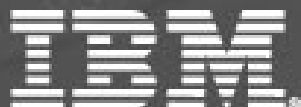
The Future of Computing



**Author: M. Mitchell Waldrop
Editors: David Rejeski, Greg Waddell**



Woodrow Wilson
International
Center
for Scholars



The Future of **Computing**

Author: M. Mitchell Waldrop

Editors: David Rejeski, Greg Waddell

Contents

This paper is part of a project on The Future of Computing at the Woodrow Wilson International Center of Scholars. This series of seminars and background papers is being developed by the Center's Foresight and Governance Project with support from IBM.

You can learn more at the web site: www.thefutureofcomputing.org. Cover illustration is from Masterfile (RM), Hans Blohm, "Close-up of IC Chip at 45× Magnification."

Editorial Offices

Foresight and Governance Project
Woodrow Wilson International Center for Scholars
1300 Pennsylvania Avenue, N.W.
Washington, DC 20004-3027

Order Form

There are limited copies of this book available for distribution. If you would like to receive one, please send an e-mail request to: foresight@wwic.si.edu.

About the Wilson Center:

The Woodrow Wilson International Center for Scholars is the living, national memorial to President Wilson established by Congress in 1968 and headquartered in Washington, D.C. It is a nonpartisan institution, supported by public and private funds, engaged in the study of national and world affairs. The Wilson Center establishes and maintains a neutral forum for free, open, and informed dialogue. The Center commemorates the ideals and concerns of Woodrow Wilson by providing a link between the world of ideas and the world of policy and fostering research, study, discussion, and collaboration among a full spectrum of individuals concerned with policy and scholarship in national and world affairs. Directions to the Wilson Center can be found at: www.wilsoncenter.org/directions.

Printed 2004

Introduction	v
1. Pervasive Computing—An Overview of the Concept and Exploration of the Public Policy Implications	1
2. Grid Computing	25
3. Autonomic Computing: The Technology of Self-Management	51
About the Author	71
Appendix:	73
The Future of Computing Events and Speakers	
Pervasive Computing	
1. <i>Rod Adkins, General Manager, Pervasive Computing Division, IBM</i>	
2. <i>Victor Zue, Director, MIT Laboratory for Computer Science</i>	
3. <i>David Brin, Author of "The Transparent Society"</i>	

Autonomic Computing

1. *Alan Ganek, Vice President, Autonomic Computing, IBM*
2. *Gail Kaiser, Director, Programming Systems Laboratory, Columbia University*
3. *Peter M. Hughes, Assistant Chief for Technology, Information Systems Division, NASA/Goddard*

Grid Computing

1. *Ian Foster, Argon National Laboratory*
2. *Andrew Grimshaw, University of Virginia*

Introduction

The *Atlantic Monthly* circa 1945 was hardly the technophiles' favorite bedtime reading material. However, in July of that year, as World War II wound down, the *Monthly* published what was certainly one of the most prophetic articles of the computer age. Written by the then Director of the Office of Scientific Research and Development, Vannevar Bush, the article had the simple, yet provocative, title: "As We May Think."

In this piece, Bush laid out a panoramic view of the future of computing. His vision included the personal computer, which he dubbed the "memex," and contained the broad outlines of such technologies as digital photography, bar codes, hypertext, and search engines. Much of his vision was, as he put it, an "extension of the known," yet in 1945 little beyond the most primitive computational devices were known. It would be another two years before the transistor was invented.

Thirty years earlier, in 1915, a company was being built by Thomas Watson, Sr. that would come to dominate the computer industry in the middle part of the 20th century. Watson and IBM adopted the motto "Think," and this concept has driven the company ever since. IBM has, for over a century, met the challenges of change, making innovation one of its core values. ThinkPads, which recently passed the "20 million sold" mark, epitomize this legacy of innovation. Today, IBM leads the world in invention and continues to extend the boundaries of innovation in areas of computing envisioned 50 years ago by Bush and, in many ways, realized by Watson's company.

If Vannevar Bush stepped into today's world of computing, much would be familiar to him, but much would elicit surprise. In his day, computer-based thinking took place in centralized facilities, or in his future vision, at the desk. Increasingly, thinking is becoming a distributed function spread across multiple devices, many of them having little to do with our common notion of a computer. In many ways the network is becoming the computer. Everything thinks—cars, toasters, phones, refrigerators—and increasingly, everything will share thoughts across ever-expanding networks. These networks will soon begin to manage themselves, reconfigure themselves, repair themselves, and optimize themselves, in many cases without human intervention. The world of person-to-person computing, or peer-to-peer (P2P), as it is commonly known, will become embedded and interconnected with a much larger world of machine-to-machine computing.

As computing moves beyond the desktop, we will have the ability to place computation where we need it to tackle a variety of important problems, from understanding ecosystems to detecting disease outbreaks or

managing complex logistics challenges. Increasingly, we will be able to bring computing power to applications on demand. This pervasive intelligence will also pose possible threats to civil liberties, and these need to be addressed proactively.

This collection of essays is intended to explore the future of computing and its possible implications for public policy. The three articles view the future through different lenses, each offering a different perspective on the rapidly evolving computing infrastructure and its related hardware and software. Taken together, the pieces converge on a world where computing is pervasive, autonomic, and on-demand; a world that diverges from the desktop environment most of us use on a daily basis.

The articles in this book formed the basis for three public seminars held in 2003 and 2004 at the Woodrow Wilson International Center for Scholars. These events were planned in conjunction with IBM and with support from the IBM International Foundation. The purpose of the series was to stimulate a broad dialogue on the future of computing among policy makers, academics, industry, and representatives from non-governmental organizations and the press. A dedicated web site was developed to support the project, and it contains presentations and videos of the speakers who took part in the seminars (www.thefutureofcomputing.org). Speakers came from a wide variety of institutions, both public and private, and represented a range of disciplines and viewpoints (a list of all the speakers is included in the appendix).

Any dialogue on the future of computing can have no beginning or end. Given the rate of scientific advance and technological innovation in the computer sector, the future approaches us at an ever-accelerating rate. It is the speed of change and the possibility of discontinuities and surprises that make such a dialogue critical from the standpoint of public policy.

We hope that this book will be the impetus for further discussions and debates. We are indebted to author Mitch Waldrop, who took on the task of describing a complex future in readable terms. Both the Wilson Center and IBM encouraged this exploration and provided the support to make it happen. Lianne Hepler at the Wilson Center designed the cover and EEI Communications helped with the layout of the book. We are also indebted to the speakers and people who attended the seminars and shared their work and thoughts about the future of computing.

Project Directors

David Rejeski

Director, Foresight and Governance Project
Woodrow Wilson Center

Greg Waddell

Governmental Programs Executive

1. Pervasive Computing: An Overview of the Concept and Exploration of the Public Policy Implications

The Convergence

Think of it as technology's Perfect Storm: the convergence of three massive waves of innovation, each driving the others on to new heights. Somewhere in the collision zone we're witnessing the birth of "pervasive computing," a.k.a. "ubiquitous computing," or "ambient computing." The three waves:

- A Proliferation of Devices
- Wireless Networking
- Mobile Software

A Proliferation of Devices

Back in the 1960s, the word "computer" still meant a big, expensive mainframe sitting off in an air-conditioned room somewhere, with lots of users lining up to submit their jobs on punch cards. Or, if the users were lucky and the mainframe was actually a time-sharing machine, they might be interacting with it in real time via remote terminals. Either way, the 1960s was the era of *many-to-one* computing: many users sharing one machine. Yet even then, just as Intel co-founder

Gordon Moore had first predicted in 1965, engineers were cramming more and more processing power into smaller and smaller packages; indeed, Moore noted, the power that was available for any given price was doubling every 18 months or so.¹ By the late 1970s, this evolution had progressed to where vendors could put an entire small computer on the user's desktop—and build it cheaply enough that that user could actually afford to buy it. This brought us into the era of *one-to-one* computing: one person, one personal computer. The ensuing PC revolution, together with the Internet revolution of the 1990s, has transformed the industry, and has dominated our thinking about the technology ever since.

But now Moore's Law has taken us so far that a typical user not only owns a desktop or laptop PC—or two, or three—but may find herself carrying around a cell phone, a PDA (Personal Digital Assistant), a pager, a smart card, a GPS (Global Positioning Satellite) receiver, an MP3 player, and more. At her office she may likewise find herself surrounded by printers, copiers, fax machines, scanners, servers, and routers, all of which have some form of on-board processing power. And that's not even counting all the microprocessors and sensors that may be operating invisibly inside her microwave, her refrigerator, her car, or even her wristwatch.

As we enter the new millennium, in short, the era of *one-to-one* computing is fast giving way to the *one-to-many* era. Each of us has more and more processors at our fingertips, know it or not, and the devices they power are becoming increasingly mobile.

Wireless Networking

Not only are the processors themselves proliferating, but more and more often they're connected. In fact, this is one of the hottest areas of innovation in high tech today, with several wireless technologies already competing in the marketplace. (A group at the University of Michigan offers a useful online tutorial.²) Wi-Fi,³ for example, is a wireless analog of Ethernet, meaning that it's designed to create a

“local area network” linking computers and other devices within an office or building. First introduced in 1997, it is rapidly growing in popularity for home and office use, and has recently begun to pop up in airports, coffee shops, and other such public spaces. (The idea is to provide Wi-Fi “hotspots” where anyone with a properly equipped laptop or PDA—and a subscription to the correct account—can access the Internet at will.) The current Wi-Fi standard, known formally as 802.11b, has an indoor range of roughly 150 feet. It operates at the same 2.4 gigahertz frequency band used by high-end cordless telephones, and can deliver data at up to 11 million bits per second, or just a bit higher than standard Ethernet. Products based on a new 802.11g standard, approved in early 2003, promise data rates of up to 54 million bits per second.

As we enter the new millennium... the era of one-to-one computing is fast giving way to the one-to-many era.

Bluetooth,⁴ meanwhile, is designed to be a wireless replacement for that rat's nest of cables going into the back of your PC. Originally devised by Stockholm-based cell-phone giant Ericsson, Bluetooth was named after the tenth-century Danish king who united the unruly tribes of Scandinavia. It could be used in the obvious way—linking computers to peripheral devices such as printers and scanners, for example—but could also allow for less familiar applications, such as using your cell phone as an office cordless phone. It operates in the same 2.4 gigahertz band as 802.11b and g, but has a much shorter range and data rate (no more than 1 megabit per second). On the plus side, Bluetooth uses considerably less power and is supposedly more resistant to interference.

Finally, there are the many technologies that allow for data transmission over cellular telephone networks. Data rates tend to be comparatively low—the “second-generation” digital cell phones that most of now use can't do much better than a 14.4 kilobits per second dial-up modem—but the range is effectively nationwide, if not worldwide. Think of the cell system as a wireless wide-area network; certainly there are already millions of people, especially in Europe

and Asia, who are happily accessing the Internet this way. And work is proceeding apace on a cluster of “third-generation,” or 3G⁵ technologies that promise 144 kilobits per second for highly mobile receivers moving at automobile speeds; 384 kilobits per second (roughly equivalent to a cable modem) for pedestrians moving at walking speed; and up to 2 megabits per second for stationary receivers.

It’s anyone’s guess how all of this will shake out in the marketplace. The Wi-Fi, Bluetooth, and 3G technologies may each find a niche of their own, and co-exist. Or one may end up displacing the others. (Many observers, for example, think that Wi-Fi has too much of a head start for 3G to overcome.) Or in the long run they may all be deposed by some new technology, such as the proposed Ultra Wide Band.⁶ But whatever happens, we can expect mobile devices in the future to be in almost constant communication with one another.

Mobile Software

Much of the Internet’s magic and power stems from the fact that it’s an information commons. Indeed, that’s been a central design goal ever since the network first took life as the Arpanet, back in the 1960s: you could tap into it from anywhere there was a portal, and get access to any information it contained (or at any rate, any information that wasn’t fenced off behind some kind of firewall). This magic was democratized in the 1990s by the World Wide Web, which not only gave the Internet a graphical interface, but hyperlinks. Suddenly, we could go browsing through online documents by the billions, almost as if the documents had all taken up residence on our local hard disk. And now, in the new millennium, a cluster of rapidly emerging technologies is allowing us to work much the same magic with other computational resources—everything from databases, simulation packages, and visualization tools, to the number-crunching power of the computers themselves. Known variously as web services,⁷ grid protocols,⁸ and peer-to-peer computing,⁹ these new technologies emerged in the 1990s quite independently, but are so similar in spirit and purpose that they are now in

the process of coalescing. Whatever the name, they promise to give our multiple connected devices the power to reach out into cyberspace, find computational resources wherever they may be, and then assemble them on the fly into whatever applications we need—without our ever having to know or care where the resources are, what computers they are running on, or how any of this occurs.

Pervasive Computing Is...

The trick, of course, is to figure out what will actually emerge from this Perfect Storm. It’s conceivable that the maelstrom of mobile devices, wireless connectivity, and mobile software will just continue on in chaos, leaving in its wake nothing more than a bigger pile of gadgets. But to an increasing number of people in the computer and telecommunications field, it seems much more likely that the collision zone will produce a discontinuity: a change in our relationship with technology that will be at least as radical as the PC revolution.

Among the earliest and most influential advocates of this view was the late Mark Weiser,¹⁰ chief technologist at PARC, the Xerox Palo Alto Research Center. In 1988, Weiser articulated a vision of “ubiquitous computing,” which he later described this way:

For thirty years most interface design, and most computer design, has been headed down the path of the “dramatic” machine. Its highest ideal is to make a computer so exciting, so wonderful, so interesting, that we never want to be without it. A less-traveled path I call the “invisible”; its highest ideal is to make a computer so imbedded, so fitting, so natural, that we use it without even thinking about it.¹¹

...Our preliminary approach: Activate the world. Provide hundreds of wireless computing devices per person per office, of all scales (from 1” displays to wall sized). This has required new work in operating systems, user interfaces,

*networks, wireless, displays, and many other areas. We call our work “ubiquitous computing”...*¹²

As Weiser often pointed out, this was the exact inverse of that much-hyped notion, “virtual reality”: instead of asking us to enter into the computer’s world, ubiquitous computing would bring computers into *our* world. By the early 1990s, he and his colleagues had an experimental environment operating within PARC, using a wireless network and three types of computers: “tabs,” which were essentially smart tags that could be attached to a book, say, or used as a personal ID badge; “pads,” which were handheld machines that could be written on with a stylus, much like today’s tablet computers; and “boards,” which could be mounted on a wall and used like a conventional whiteboard—except that the “ink” could be stored and processed electronically. (The latter was later marketed as the Xerox LiveBoard.) In the process, they began to work out many of the ideas discussed below.

Sadly, Weiser died in April 1999 after a brief battle with cancer. By that point, however, his ideas had already achieved widespread currency among other researchers—albeit often under the alternative name “pervasive computing.” (Many people found the word “ubiquitous” to be a mouthful and didn’t like the frequently used contraction “ubicom” much better.) With the three technological trends mentioned above becoming abundantly apparent, industry heavyweights such as IBM and Hewlett-Packard were already beginning to make significant investments in the area, as were federal funding agencies such as the National Science Foundation and the Defense Advanced Research Projects Agency. And a wide variety of demonstration projects were getting underway—most of which are still going strong even in the face of the downturn. Among the most notable are IBM’s Pervasive Computing Lab¹³ in Austin, Texas; Hewlett Packard’s Cooltown¹⁴ in Palo Alto; MIT’s Project Oxygen¹⁵; the California Institute

...instead of asking us to enter into the computer’s world, ubiquitous computing would bring computers into our world

for Telecommunications and Information Technology—Cal-(IT)²—in San Diego¹⁶; the Georgia Tech Aware Home¹⁷; and Royal Philips Electronics’ Homelab¹⁸ in Eindhoven, The Netherlands (where it’s called “ambient computing”).

Of course, because the research is still in its blind-men-and-the-*elephant* phase, each of these projects (and each of the people involved in the projects) has a different view of precisely what pervasive computing will be. Speaking generally, however, most of the properties they describe can be grouped under three headings: **nomadic**, **embedded**, and **invisible**.

- **Nomadic**

The idea here is to let users set up their tents anywhere they like, so to speak, instead of being tied to a specific place or specific machine. Or to put it another way, the idea is to let users’ software and data be just as mobile as they are.

To some extent, this is starting to happen already. In 2002, for example, the Toronto Police Service began its deployment of eCOPS¹⁹: an integrated database system in which (among other things) officers on patrol can use wireless-equipped laptops to get an immediate listing of any prior occurrences at a particular address, any outstanding warrants, the records and mug shots of any individuals they’ve placed under arrest, and other such critical information. In the Washington, DC, area, meanwhile, Maryland, Virginia, District, and federal authorities are pushing ahead with CapWIN, the Capital Wireless Integrated Network.²⁰ Inspired in part by 9/11, CapWIN is intended to provide fully integrated communication and data access among the region’s multitudinous federal, state, and local law enforcement organizations, fire and emergency medical services, transportation, and other public safety agencies.

Similar efforts are underway in the business world, where there’s widespread interest in taking full advantage of all those smart phones, wireless PDAs, and in-vehicle information systems. “Given the current economic environment, deployment is not as robust as could be,” admits Rod Adkins, IBM’s vice president for pervasive computing. “But in some areas, the advantages are very compelling. For example,

keeping your sales force mobile in the field.” With wireless connectivity, he says, “they can handle more customers and make more service calls, while still getting the data they need at point of engagement.” Indeed, says Adkins, IBM—which is lead contractor on both the eCOPS and CapWIN efforts—sees potential markets for pervasive computing in areas that range from financial services, to product distribution, to travel planning. In healthcare, to take a major example, physicians who carry PDAs and/or tablet computers on their rounds could get immediate access to critical lab results, as well as a menu-based prescription system that minimizes the chances of an error—a major source of adverse reactions and even death in hospitals.

As useful as such applications may be, however, they’re just the beginning of real data mobility. Say you need to catch up on some work while you’re on the road. In the pervasive world, say the visionaries, you’ll be able to walk up to any free PC and beam a network address at it from your favorite handheld device. That PC will then pull down your entire desktop environment, along with all your files and applications, so that you can work on them exactly as you would at home. (All that personal information will be stored back to the network, of course. And once you’re finished, security being as vital as it is, it will vanish from the host machine without a trace.) Or say you’ve been invited to give a seminar. You’ll be able to walk into the conference room, beam another network address at the projector, and have it instantly bring up your PowerPoint slides. At the same time, with the projector feeding instructions back to your handheld device telling it how to become a controller, the device’s display will suddenly show touch-sensitive buttons for *forward*, *back*, and all the rest, and you’ll use it to launch right into your presentation.

Applications such as these are currently under development at HP Cooltown, Project Oxygen, and many other pervasive computing labs. However, the technical challenges are large. Take that handheld device, for example. The proliferation of such devices is out of control already, which is why there’s strong market pressure for integration; no one, with the possible exception of the geekiest computer nerds, wants to walk around wearing a utility belt that’s full of PDAs, cell phones, MP3 players, and whatnot. Thus we’re seeing PDAs that are also cell

phones, cell phones that can browse the Web, and so on. But in a truly nomadic world, where we’ll be using the things constantly, the challenge to integration will be extreme. The actual shape of our favorite gadget will no doubt be a matter of personal preference; one model may look a lot like a current-generation tablet PC, another like a PDA, and so on. But with one or two of the things, at most, we’ll want to have the functionality of a cordless phone at home, a cell phone on the road, a tablet PC, a PDA, a GPS receiver, a projector controller, and much more. We’ll also want our devices to automatically upgrade themselves whenever some new functionality comes along, so that we don’t have to run out and buy a new gadget every time.

In short, we’ll want what some researchers have dubbed the universal information appliance.²¹ What makes this hard is that designers can’t just keep on cramming in more and more specialized chips, which is how integration is done now; not only does that approach make for a tough upgrade path, but even with Moore’s Law, there are limits. One possible alternative is the RAW chip architecture²² developed for Project Oxygen by MIT’s Anant Agarwal. Think of it as “soft hardware.” Instead of making just one very complicated microprocessor per chip, as Intel does when it fabricates, say, a Pentium, Agarwal’s design calls for laying down a few zillion very small and very simple processors in a regular array, each of them identical. By routing and rerouting the way data flows through this array—in effect, rewiring the chip on the fly—the software can turn the RAW chip into whatever processor it needs to be for the task at hand.

And then there’s the knotty issue of connectivity. The world of pervasive computing will be much like the world is today: full of people, cars, printers, handhelds, and myriad other entities, all of which are constantly moving (or being moved) from place to place. The difference is that every one of those entities will need at least intermittent connections to the network, on demand, often *while* they are moving. This means that each device will have to reach out when necessary, and automatically hook itself into the cellular phone system, or a conference room’s local Wi-Fi network, or a desktop computer’s Bluetooth connection, or whatever else gives it the best data rate and most reliable connection at that particular moment. If it’s moving around, it will

also have to shift to new connections as needed—and do so in mid-conversation, without the user’s being aware of anything except a possible speeding up or slowing down of the data rate. As it enters each new location, moreover, each device will have to reach out and identify any other devices and software resources that are there, while simultaneously introducing itself to *them*; soon enough, after all, they may be given a new task that they will have to carry out as a team (e.g., “Send this document to the nearest printer”). In short, as the Project Oxygen web site puts it, the networks of the pervasive world will become “*ad hoc* collaborating communities of people and computing devices, [which] configure and reconfigure themselves automatically, as nodes appear, migrate, and disappear.”²³

Unfortunately, this is *not* how networks operate today. The basic “middleware” that controls the Internet builds in the tacit assumption that computers are stationary, meaning that they can be served by a more or less static set of data pipes that feed a more or less static set of data outlets—each of which has an absolutely static, fixed-in-stone “IP address.” (That’s the numeric address that your web browser looks up when you type in, say, *www.ibm.com*.) Moreover, that address exists only in a kind of abstract network space; there’s no way to direct something to “the nearest printer” because the system itself has no concept of physical location.

Of course, it’s certainly possible to transcend those limitations. For example, devices can establish their unique identity through radio frequency ID tags, or RFIDs,²⁴ which have been commercially available since the 1980s. They can monitor their location and context through GPS signals. (Or, if they are indoors where GPS is unreliable, they can triangulate the signals from ceiling-mounted radio beacons, as in MIT’s Cricket system.²⁵) And they can learn about other devices in their vicinity—“What’s in this room? How do I access it?”—via the kind of resource discovery mechanisms that are already built into web services, grid protocols, and peer-to-peer networking.

Nonetheless, it’s still a cutting-edge challenge to get all these capabilities working together as a robust and seamless system. Indeed, it’s a challenge that extends beyond networking middleware to encompass software in general—and arguably even computer science

in general. After all, a standard shrink-wrapped application like Microsoft Office builds in some equally strong assumptions about the kind of system it’s going to be running on. (“Pentium 166 or higher processor required...”) But in a pervasive computing environment, there will be no way to know in advance what kind of processors will be present, not to mention what kind of memory resources, network connections, and all the rest. The “applications” that carry out any given task will have to be put together on the fly by the system itself, with the available devices collectively making use of the available resources as best they can.

Call it the challenge of Computing in Uncertain Environments. “It’s changing teaching and research all through computer science,” declares Victor Zue, director of MIT’s Laboratory for Computer Science. And not just in academia. In 2001, to take a notable example, IBM announced that it was reorienting its advanced research program around the notion of autonomic computing²⁶: building “computer systems that regulate themselves much in the same way our autonomic nervous system regulates and protects our bodies.” The IBM vision was that these autonomic systems would be *self-configuring*, meaning that each new device would automatically fit itself into the existing community of devices in the way described above (think Plug-and-Play on steroids); *self-optimizing*, meaning that the system would automatically distribute tasks and subtasks to the various devices in the most efficient way possible; *self-protecting*, meaning that the system would detect and guard itself against damage from accidents, equipment failure, or outside attacks; and *self-healing*, meaning that the system could detect, diagnose, and recover from any damage that did occur.

This is an ambitious and long-term research agenda, to put it mildly. But in the meantime, the nervous-system metaphor brings us to the second characteristic of pervasive computing—

- **Embedded**

Consider the modern refrigerator, which has a built-in electronics package that beeps at you if the door is slightly ajar and warns you if there’s been a power failure that might have let food spoil. Or consider the modern automobile, which uses dozens of embedded microproces-

sors and micro-sensors to adjust the fuel mix, trigger warning lights if an air bag needs attention, and generally keep a constant watch on the vehicle's health and performance—and then, if need be, provide diagnostic readouts for the technicians at the garage.

The idea behind “embedded” pervasive computing is to extend that kind of electronic nervous system to virtually everything else in the world.

A prime example is the Ecological Observatory²⁷ now being demonstrated by Cal(IT)²: the California Institute for Telecommunications and Information Technology. (The institute, which has emerged as a leading testbed for the embedded approach, is one of the four California Institutes for Science and Innovation first proposed in December 2000 by Governor Gray Davis. It has 40 industrial participants, including such heavyweights as IBM, Hewlett-Packard, and Microsoft, and encompasses some 200 faculty members and students at UC San Diego and UC Irvine.) The Ecological Observatory is located in the 4000-acre Santa Margarita Ecological Reserve, in northeastern San Diego County. The goal is to monitor the environment there by deploying the widest possible range of sensing devices: a “sensornet” whose elements will communicate with headquarters (and each other) using the cell network, satellite links, and a variety of other wireless connections. According to the observatory web site, the sensornet will keep watch on “pollutants in the Santa Margarita River; seismic events; the moisture ‘load’ of the vegetation (which affects the degree of fire danger); air quality; the presence and spread of invasive botanical species; and the movement, behavior, and numbers of animal populations, such as mountain lions and bats.” The Cal(IT)² observatory will also serve as a prototype for the National Science Foundation's much broader-scale National Ecological Observatory Network.²⁸

Another example from Cal(IT)² is Autonet,²⁹ which seeks to create a central nervous system for urban transportation. Researchers on the project are still developing the basic technology. But their long-term vision includes 1) a dense network of roadside stations that will collect second-by-second traffic flow data, while simultaneously broadcasting that data to passing vehicles; 2) on-board navigation devices in each vehicle that can augment the real-time data with advanced traffic-pre-

dition algorithms, presumably downloaded from commercial route-guidance services; and 3) wireless vehicle-to-vehicle communication that will allow the on-board devices to share traffic and other data. (As Cal(IT)² founding director Larry Smarr asks in mock horror, “How can it be that all the cars today don't know where all the other cars are, and what speed they're going?”) The hoped-for result: a much more efficient traffic flow—a prospect that has considerable appeal in a state where traffic congestion is chronic and one of the biggest obstacles to continued economic growth.

Meanwhile, it's easy to imagine similar electronic nervous systems monitoring the roadways and bridges themselves, as well as other elements of the civil infrastructure. After an earthquake, for example, embedded sensors could instantly alert repair crews (and drivers) that such-and-such a bridge was no longer safe. Likewise, in the much-discussed “smart home” and “smart office,” embedded electronic nervous systems could monitor and manage each building for energy efficiency, safety, and security. Say you're planning a vacation, says Bill Bodin, director of IBM's Pervasive Computing Lab in Austin: a properly wired house could not only tell you whether it was safe to leave—Did you really turn that iron off?—but whether it was safe to come back: Has a window been broken? Are there warm human bodies waiting inside a house that's supposed to be empty?

Indeed, the notion of an electronic nervous system could even be extended to individuals. In principle, at least, an array of unobtrusive and non-invasive sensors built into your watch, jewelry, and/or clothing could monitor our risk of, say, a heart attack—and then automatically alert the emergency rescue services via wireless if your warning signs suddenly spiked into the red zone. Indeed, declares Smarr, “to me, it's clear that everybody should be having their body read out on the Internet.” True, he adds, “people tend to have a horrified reaction to that idea”—not least because of the privacy implications. But if strong privacy safeguards were built in from the outset, he says, such a system could not only safeguard personal health, but could provide an incomparable trove of statistics for public health. For example, within a few years, given the way the technology is progressing, getting your own, personal genome sequenced could become as standard as getting

a regular physical. “So you could take that genomic sequence, and combine it with the metabolic readout as you interacted with environment,” says Smarr. “And now if you had that kind of information for millions of people, you could use very sophisticated data mining techniques to understand how different populations of people will react to a new drug.”

That’s a ways in the future, of course. But in the meantime, it’s clear enough that much of the embedded infrastructure will actually be mobile, and that nomadic devices will actually spend much of their time communicating with embedded ones. These two characteristics of pervasive computing are completely complementary and synergistic—which leaves us with the third characteristic—

- **Invisible**

Go back to the modern automobile again and consider the anti-lock braking system: the cluster of embedded sensors and microprocessors that makes the brakes pulse on and off during an emergency stop, so that the car doesn’t skid. This is a classic example of what Mark Weiser meant by “invisible” computing: pulsing the brakes isn’t something you do *on* the computer. It’s something that just happens in the background—automatically—while you stay focused on the primary task of stopping the car.

The goal is to make this kind of invisibility the norm in pervasive computing. After all, goes the argument, computation is hardly a scarce resource anymore. What *is* in chronically short supply is human attention—a resource that our current-generation cell phones, pagers, Web browsers, email clients, TVs, etc. tend to squander. And when you multiply that by the overwhelmingly larger number of processors that could surround us in the pervasive world, a certain invisibility seems essential. Better to save our attention for what really matters, instead of frittering it away on tools that are forever asking us to scroll through menus or find the right button to click. In the mid-1990s, in fact, Weiser and his then-boss, PARC director John Seely Brown, suggested that we should be working toward a whole new generation of “calm technology”: stuff that would sooth the users and actually help them stay focused.³⁰

In practice, this means that our pervasive computational environments will have to interact and respond to us in a much more humanly intuitive and natural way. Specifically, they will have to become 1) *multimodal*, responding appropriately to speech, gesture, and perhaps even facial expressions; 2) *proactive*, anticipating our needs and requests in context, and fulfilling them before we have to ask explicitly; and 3) *semantic*, understanding and doing what we mean, without insisting on a precise definition—or violating common sense. (“The nearest printer,” for example, probably doesn’t mean “the printer that’s directly below this office three floors down,” even if it actually is the closest in a straight line.)

Of course, none of this is going to happen easily, or soon; artificial intelligence researchers learned a long time ago that “common sense” is a lot slipperier and more subtle than it looks, to take just one example, and they’ve been struggling to get decent speech understanding for at least fifty years now. True invisibility is by far the toughest challenge in pervasive computing. Nonetheless, any number of people are trying.

At MIT’s Project Oxygen, for example, Howard Shrobe and his co-workers are developing technology for an “Intelligent Room.” What’s been achieved so far is admittedly pretty mundane, says Shrobe. When one or more people walk

into such a space—the group has installed prototypes in several offices and conference rooms—microphone and camera arrays embedded in the walls and ceiling allow

...our pervasive computational environments will have to interact and respond to us in a much more humanly intuitive and natural way.

the room to respond in certain ways. (*Human*: “Computer, what’s the weather tomorrow?” *Room*: THE FORECAST FOR WEDNESDAY IS PARTLY CLOUDY, WITH A HIGH OF 67° FAHRENHEIT.) On request, the room can lower the screens for a presentation and dim the lights, or videotape a meeting. But hopefully, says Shrobe, this is just the walk-before-you-run stage, preparing for a much more sophisticated kind of human-environment interaction. In the future, according to the group’s web site, a meeting might go something like this: “[the con-

ference room environment] knows the interests, organizational roles, and skills of all team members, and it understands the application domain within which the team functions. For example, it tracks action items within the group and dependencies with other groups, retrieving relevant information and bringing it to the attention of the most appropriate individuals. The collaboration system plays the role of an active participant, noticing tasks that need to be undertaken, noticing when information required for those tasks has been developed, and making conclusions when appropriate.”

Meanwhile, the MIT researchers hope that one of the most useful furnishings in any Intelligent Room of the future will be the “smart surface,” which will be able to take freehand sketches and turn them into working simulations. MIT’s Christine Alvarado and Randall Davis have devised a working prototype of such a system called Assist,³¹ a.k.a. “Magic Paper.” Sketch on the wall with a special wireless-equipped “pen,” and the system will project a trail of digital “ink.” It’s almost as if you were drawing on an ordinary whiteboard—except that your rough circles automatically become perfect circles, your shaky lines become straight, and so on. Now sketch a little cart with wheels, sitting at the top of an inclined plane, and tap the button labeled “Run”: the cart rolls down the plane and falls off the end. In a similar way, if you draw an electric circuit, the system will recognize the various capacitors, resistors, and so on, pretty them up, and then simulate the flow of current through the circuit.

It’s a beginning, says Davis. But think of the possibilities down the road for, say, education: “Part of the dream is that in the textbook of the future, you’ll never see a static diagram. You’ll always be able to ask it, “I wonder what would happen if...?”

Policy Challenges

Indeed, the techno-visionaries have given us a beautiful dream of what pervasive computing could be. But how much of that dream will ever become real?

As we’ve seen, the technical challenges are daunting. Pervasive computing in 2003 is roughly where personal computers were circa 1976, when the machines existed only as hobbyist kits that you had to put together yourself. But then, for an optimist, that’s good news—a strong indication from history that pervasive computing’s growth curve from here on out will be at least as explosive as PCs in the 1980s. And indeed, there’s evidence to support that view. In New York’s ultra-trendy Soho district, for example, each item of clothing in Prada’s new Epicenter store has an RFID tag attached. When you find, say, a suit that you want to try on, and hang it in the dressing room’s “smart closet,” antennas in the walls scan the tag and activate the room’s LCD touch screen—whereupon you suddenly find yourself paging through a display of the same item in different colors, or suggestions for accessories that would go with the item’s look. “That’s a very interesting first experiment in creating a brand new experience in the physical places people go,” says HP Cooltown director Gene Becker. “And there are plenty of other companies experimenting with this technology, as well. So when the economy picks up, you can expect to see people deploying it quite rapidly.”

Be that as it may, however, history is a decidedly imperfect guide to the future. There’s also good reason to think that we *won’t* see a repeat of the wide-open, wild and woolly days of the early PC revolution (or for that matter, the early Internet revolution.) After all, the PC pioneers were not only operating in a consumer electronics market that was largely unregulated, so that competition had free rein, but they were offering up a technology that seemed to have no serious downside. You may or may not have *liked* PCs, but there was no strong reason to be afraid of them.

Neither is the case with pervasive computing. The concept is utterly dependent on wireless networking, after all—and wireless is part of the (partially) regulated telecom industry. Moreover, the very pervasiveness of the technology raises obvious concerns about privacy, security, and trust. Some of the issues that policy-makers and business planners may want to consider:

Spectrum allocation. Nature gave us only so much space in the radio spectrum, and there’s no way to make more of it. Thus the FCC,

and the perennial headache of trying to satisfy more and more users with smaller and smaller slices of a fixed resource—a problem that could be greatly exacerbated if the market for pervasive-style devices really does take off. On the other hand, thanks to the basic physics of channel capacity, which can be quite subtle and counter-intuitive, it's theoretically possible for distributed networks of devices to make collective use of the available bandwidth in ways that individual devices cannot. So maybe technology can keep ahead of the game. And maybe the FCC will need to do some fundamental rethinking of its whole approach to allocation.

Interoperability, open standards, and the business model.

“This is one of the big issues that [pervasive computing] has to face as an industry,” says HP Cooltown's Becker. “The last thing we want is a bunch of competing, completely incompatible infrastructures, so that, when I'm at home, I have a company A experience, but when I go to the office, none of my home devices work; I can only have a company B experience. And then when I go to my dentist, I can only have a company C experience.” Indeed, that kind of fragmentation could cripple pervasive computing before it really gets started; much of the dream appeal comes from its vision of utterly seamless connectivity, with each device having open access to cell networks, Wi-Fi links, and all the rest, as needed.

For better or worse, however, we live in a world in which fragmentation is the norm. “‘Open’ is *not* the way the wireless world works,” emphasizes MIT's David Clark, one of the architects of the modern Internet. “If I change cell providers, I have to throw away my phone! The market is vertical as they can make it.” Likewise the rapidly expanding network of Wi-Fi hotspots: unless you've got a paid-up subscription with the particular service that operates the particular hotspot you're in, don't bother trying to log on.

The challenge, says Clark, is to find a way for vendors to provide universal service—and still make money.

***...pervasive computing...
is utterly dependent on
wireless networking...
and wireless is part of the
(partially) regulated tele-
com industry.***

That's trickier than it sounds, he says. Witness the Internet, which is the only universal standard that's ever been a real success. “The reason the Internet looks the way it does is because it was funded by the government,” says Clark, “which meant that the academics who built it had nothing to cut off their idealized bias towards open access and open standards.” And that was a good thing, too, since the very openness of the Internet was what allowed it to function as an information commons, thus paving the way for the World Wide Web, e-commerce, and all the rest. “But the open architecture of the Internet also means that the guys at the bottom, the ones deploying the infrastructure, have no way to capture the money being made at the top,” says Clark. Quite the opposite: the Internet's open architecture has given rise to open competition at the infrastructure level, which has tended to drive the profit margins toward zero.

Thus the wireless providers' desire to keep their own markets as segmented as they can, as a way to get a healthy return on their initial investment. And thus the need to find a better way. After all, says Clark, “history suggests that what's good for an individual player isn't necessarily good for the structure of the industry.” One top priority is the search for alternative business models for pervasive computing—models that would allow for an open architecture *and* profits. (Clark is looking at several possibilities in collaboration with researchers from the MIT's Sloan School of Business.) Another is a top-to-bottom rethinking of the regulatory environment, starting with the Telecommunications Act of 1996.

But the one thing we should *not* do, says Clark, is assume that the pervasive infrastructure will just spring to life of its own accord. “There is no Moore's Law for deployment,” he says. “If we get this wrong in one way, we'll get an infrastructure that's completely vertical. And if we get it wrong in the other way, we won't have any infrastructure at all.”

Privacy. This is the elephant in the room—the one issue that everybody involved with pervasive computing lists as a key concern. The word “pervasive” can have quite a sinister connotation, after all. And a world full of electronic devices that are always on and always connected can sound very much like a world that will monitor exactly

where you are and exactly what you're doing—every minute of every day. What could your friendly neighborhood hacker do with such a system? What would J. Edgar Hoover have done?

“None of us wants to be the guy who invented Big Brother,” says MIT's Shrobe. Victor Zue, director of MIT's Laboratory for Computer Science, couldn't agree more: “All of us are committed to privacy and security as being not the last thing we add on, but something that's built into the system from the start.”

Indeed, there is a lot that technology could do. As mentioned earlier, for example, pervasive technology could build in a certain anonymity: once you've finished using a projector, or a guest PC, or whatever, that particular device would retain no information whatsoever indicating that you've been there. Likewise, once your personal, handheld device has finished asking the surrounding intelligent environment about its location, that environment would retain no information whatsoever about the device, the time, the location, or you.

Still, you don't want to pin all your hopes on well-behaved technology, not when that technology is the creation of fallible (or duplicitous) human beings. And in any case, technology alone can only *safeguard* privacy; figuring out what the safeguards ought to be, and where our zone of privacy actually lies, is a matter of policy, law, and ultimately social norms.

To put the issue in perspective, says Rodney Brooks, director of the MIT Artificial Intelligence Laboratory, “How would you like to have a live microphone in your bedroom 24 hours a day? Well, many of us do: it's called a telephone. Or how would you like to carry around a computer that always knows where you are to within a few hundred meters? Again, many of us do: it's called a cell phone.” We've come to accept such “Big Brother-ish” technologies without worrying too much about them, mainly because we trust that the phone companies won't use the technology against us. And that, says Brooks, is a social compact as much as anything.

Of course, new technology does have a way of confronting us with new social dilemmas; the norms for the pervasive world will have to be worked out as we go along. For example, says Cooltown's Becker, “there should be consistent ways for people to know what's going on—say, that a meeting they're in will be recorded by cameras and microphones hidden in the ceiling.” Or, as MIT's Shrobe puts it, “What rights do people have *not* to be on camera?”

Eventually, of course, we'll have to address the question of law: to what extent should these evolving expectations about privacy in the pervasive world be legislated? Today, for example, you have a legal right to know if you're being taped. So in the future, will offices be required to put up signs, “Caution—you are entering a camera zone”? In that context, perhaps it's worth pointing out that the concern about privacy is ultimately part of the larger concern about control: being able to have some say in what information is collected about you, and how that information is going to be used. “One possible way to handle the privacy issue is via the debate going on now about who owns your data,” says Larry Smarr. “For example, who owns your DNA sample? So maybe that's part of it: you have a right to your data stream.”

But however we handle the issue, notes Becker, “the one thing we can't afford to do is stick our heads in the sand. Technology is becoming pervasive whether we want it to or not.”

***The word “pervasive”
can have quite a
sinister connotation.***

Endnotes

- ¹ Moore, Gordon E. “Cramming More Components Onto Integrated Circuits.” *Electronics* (1965); <http://www.intel.com/research/silicon/mooreslaw.htm>
- ² <http://www-personal.umich.edu/~acarbon/Alejandro/wifi1.htm>
- ³ http://www.webopedia.com/TERM/W/Wi_Fi.html
- ⁴ <http://www.bluetooth.com/>
- ⁵ <http://80211-planet.webopedia.com/TERM/3/3G.html>
- ⁶ <http://www.webopedia.com/TERM/u/UWB.html>
- ⁷ http://www.webopedia.com/TERM/W/Web_services.html
- ⁸ http://www.webopedia.com/TERM/g/grid_computing.html
- ⁹ http://www.webopedia.com/TERM/p/peer_to_peer_architecture.html
- ¹⁰ <http://www.ubiq.com/hypertext/weiser/UbiHome.html>; Weiser, Mark. “The Computer for the Twenty-First Century.” *Scientific American* (1991): 94-100.
- ¹¹ Weiser, Mark. “Creating the Invisible Interface.” *Symposium on User Interface Software and Technology* New York, NY: ACM Press, 1994.
- ¹² Weiser, Mark. “The World Is Not a Desktop.” *ACM Interactions* 1, no. 1 (1994): 7-8.
- ¹³ <http://www-106.ibm.com/developerworks/wireless/library/wi-pvc/>
- ¹⁴ <http://cooltown.hp.com/cooltownhome/index.asp>
- ¹⁵ <http://oxygen.lcs.mit.edu>
- ¹⁶ <http://www.calit2.net>
- ¹⁷ <http://www.cc.gatech.edu/fce/ahri/>
- ¹⁸ <http://www.philips.com/homelab>
- ¹⁹ <http://www-3.ibm.com/software/success/cssdb.nsf/cs/NAVO-4U634A?OpenDocument&Site=indwireless>
- ²⁰ <http://www.capwinproject.com/index.html>
- ²¹ K.F. Eustice, et al. “A Universal Information Appliance.” *IBM Systems Journal* 38, no. 4 (1999): 575-601; <http://www.research.ibm.com/journal/sj38-4.html>
- ²² <http://cag.lcs.mit.edu/raw/>
- ²³ <http://oxygen.lcs.mit.edu/Network.html>
- ²⁴ <http://www.aimglobal.org/technologies/rfid/>
- ²⁵ <http://nms.lcs.mit.edu/projects/cricket/>
- ²⁶ <http://www.research.ibm.com/autonomic/>

²⁷ <http://www.calit2.net/observatory/index.html>

²⁸ <http://www.nsf.gov/search97cgi/vtopic>

²⁹ <http://www.calit2.net/transportation/index.html>

³⁰ <http://www.ubiq.com/hypertext/weiser/acmfuture2endnote.htm>; Weiser, Mark, and John Seely Brown. “The Coming Age of Calm Technology.” *Beyond Calculation: The Next Fifty Years of Computing*. Editors Peter J. Denning, and Robert M. Metcalfe, 75-86. New York: Copernicus, 1997.

³¹ http://www.ai.mit.edu/projects/rationale/project_assist

2. Grid Computing

Just within the past few years, even as the tech sector has been cycling from exuberance to ruin to slow recovery, a growing coalition of researchers, entrepreneurs, funding agencies, computer industry heavyweights, and corporate CIOs have begun to place their bets on a new, and hopefully more solid vision of computing's future.

As befits an idea that has many elements and that has been reinvented many times, this vision goes by many names. Among them are distributed computing, Web services, Peer-to-Peer, organic IT, utility computing, and—the name we'll adopt in this paper for simplicity's sake—grid computing. Indeed, the multiplicity of names has often obscured the fundamental unity of the idea: the notion that in an increasingly networked world, computation need no longer be confined to computers. Instead, computation can be recast as a collection of processes that can move among desktops, PDAs, servers, and any number of other devices, assembling and reassembling themselves on the fly to meet the task at hand. In effect, the network itself really *will* become the computer—a seamless computational device with resources that span the planet.

This is what the Internet has been building toward for the past three decades, declares Larry Smarr, director of the California Institute for Telecommunications and Information Technology¹ in San Diego, and one of grid computing's most forceful advocates. "In the

first phase,” Smarr explains, “starting in the 1970s, we got the wires up and hooked in all the computers. Then with the World Wide Web, we started hooking in all the on-line documents.” Now, he says, with grid computing, we’ll be hooking in everything else. Applications, databases, sensors, video and audio streams—all will be reborn as services that live in cyberspace. “It’s going to be completely transformational,” Smarr says. “What we’re seeing is the emergence of a whole new infrastructure upon which first science, and then the whole economy will be built.”

Grid computing could have an impact at least as great as that of the World Wide Web itself, agrees Daniel Reed, former director of the National Center for Supercomputer Applications at the University of Illinois, and a principal investigator of a major grid demonstration project, the National Science Foundation’s TeraGrid.² After all, he says, “a Web browser didn’t actually make anything possible on the Internet that a technically savvy person couldn’t have done before, with effort. But what the Web *did* do was to vastly expand the number of people who had access to on-line data, because it greatly lowered the entry barrier.” And that’s what grid computing promises to do again.

Imagine, for example, that you’re the head of an emergency response team that’s trying to deal with a major chemical spill or even a terrorist attack. You’ll probably want to know things like, What chemicals are involved? What’s the weather forecast and how will it affect the pattern of dispersal? What’s the current traffic situation, and how will it affect the evacuation routes? If you tried to answer those questions on the Internet today, you’d quickly bog down in arcane log-in procedures and incompatible software—assuming that you could find the databases and simulations you needed in the first place. But with grid computing it would be straightforward. The system would already have standard mechanisms in place for discovering, accessing, and invoking just about any online resource you could imagine,

...in an increasingly networked world, computation need no longer be confined to computers.

while also building in all the necessary safeguards for security and authentication.

Likewise in the commercial arena. “Say your company wants to respond to a new business opportunity,” says grid-computing pioneers, Carl Kesselman of the University of Southern California’s Information Sciences Institute. “Suddenly you’re starting new partnerships you didn’t know you were going to have, getting new subcontractors, and creating a whole new ‘virtual organization’ of people who have to work together to get the job done. Well, right now, the way we share things is very primitive. We email stuff back and forth, and so on.” But with grid computing, he says, you and your new partners could share your whole underlying infrastructure of software, simulation tools, and databases, while still protecting the parts that were proprietary, because online security would already be built in. “Grid computing will allow us to interact in a much deeper, richer, more enabled way,” says Kesselman, “And that could have a profound impact.”

Certainly that’s the hope at IBM, which is the prime contractor for the NSF’s TeraGrid, as well as for similar national grids in Europe. David Turek, director of IBM’s server division, compares grid computing to the familiar grid of electrical power: “To use a hair dryer, you just plug it into a wall socket,” he says. “You don’t have to worry about how the turbine is designed up in Niagara Falls, or the physics of power transmission.” That’s exactly how Turek wants people to think about computing power. “In our vision of the future, if you’re a customer who occasionally needs 10 teraflops, for example, don’t buy a machine that’s underutilized most of the time; buy it from the grid. So grid computing will play into our vision of computing as a utility.” (In the 2003 U.S. Open Tennis Tournament, for example, IBM fielded a cluster of servers to support the U.S. Open web site. At any given time, the various web site applications would be running on however many servers were needed, while the rest of the servers were doing a protein-folding calculation for biomedical research. But when there was a spike in the web traffic, as often happened, the extra servers would suspend their protein-folding and pitch in.)

In addition to these utility grids, says Turek, many users will want to set up grids of their own. “You might see 10 to 20 departments coming together to create a campus-wide, or company-wide grid, each contributing some of the computer power they control,” he says. In another scenario, several independent companies such as defense contractors might do much the same thing to create an ad hoc grid that would allow them to use each other’s proprietary data and software to prepare, say, a proposal for a new military aircraft.

“That’s why we’re not going to espouse the grid as something that can be done only with IBM technology,” Turek explains. After all, he says, “if you get five companies wanting to come together on a grid, the likelihood of all five having the same servers is pretty slim.” To facilitate such flexibility, he says, IBM “endorses open-source protocols wholeheartedly.”

Protocols: The Long Road to the Grid

Protocols, of course, are the key. From a technical point of view, after all, what actually triggered the World Wide Web revolution of the 1990s wasn’t the Web browser per se, but the existence of widely

Protocols, of course, are the key.

accepted software standards for creating on-line documents and then linking them—respectively, the hypertext markup language, or HTML, and the hypertext transfer protocol, or HTTP. And so it is with grid computing. What’s begun to make it seem real is the emergence of widely accepted software standards that support it.

That said, however, there is also a difference. The Web protocols were unified and universal from the start, largely because they were created by one person—Tim Berners-Lee—working at one particular place and time: CERN, the European Center for Particle Physics, in 1990. But the grid protocols, as befits an idea that’s been reinvented again and again, have emerged in a much more decentralized and chaotic fashion.

Globus: The Utility Model

Perhaps the most straightforward thread of the story is the saga of the Globus³ Toolkit, and the utility model of grid computing.

“To put this in context,” says ISI’s Kesselman, who developed the toolkit in collaboration with Ian Foster of Argonne National Laboratory, “it’s worth remembering that the idea of being able to share computers goes back for quite a long time. The Arpanet [the ancestor of today’s Internet] was built in the 1960s to give users on one campus shared access to resources on a different campus. And people have been demonstrating various forms of resource-sharing ever since.” During the early 1990s, in particular, there was a flurry of experimentation with “metacomputing,” in which specialists working in the high-performance supercomputer field would try to make many distributed computers like one giant computer. One supercomputer might act as the meta-machine’s central processor, for example, while a total immersion virtual-reality facility halfway across the country might act as its display screen, and so on. The problem, says Kesselman, was that these efforts were largely ad hoc, with the experimenters having to reinvent the wheel every

time: “There was still no standard software for distributed computing, no infrastructure to support it.”

The watershed event came in 1995, at the annual supercomputing conference. “There’s always a big demonstration of some sort at these conferences,” Kesselman

explains. “Well, by this time, we had begun to see a variety of high-speed networks,” including one funded by NASA and a whole series of “gigabit testbeds” funded by DARPA, the Defense Advanced Research Projects Agency. “So a group of us thought it would be interesting to demonstrate what you could do with them.” He and Foster already had the software to support such an effort, Kesselman adds, “Ian and I had

During the early 1990s... there was a flurry of experimentation with “metacomputing,” in which specialists working in the high-performance supercomputer field would try to make many distributed computers like one giant computer.

been working together for ten years. We had some library tools and very basic security tools for high-performance computing applications, and we realized that they could work just as well over networks.”

The result was the I-Way demonstration, in which 11 separate high-speed networks were briefly connected into one. “Some 60 groups came forward and had applications running at I-Way,” recalls Foster, who was especially struck by some of the collaborative design applications. “For example, Argonne and an industrial group did a virtual reality simulation for the collaborative design of industrial incinerators. Users at different sites could see each other as ‘avatars’ while they flew through the incinerator, placed injectors in it at various points, and jointly studied the effect on its output.”

In any case, says Foster, “I-way was what convinced people that grid computing had great potential.” And one result was that in October 1996, DARPA funded them to develop a set of protocols that would serve as a solid foundation for it. The basic idea was to let the network itself take care of the low-level chores, like moving data around, while the “Globus” protocols, as they came to be known, managed the higher-level tasks required for resource-sharing. Among other things, Globus would provide software tools for resource discovery, which meant automatically finding out where a required database, program, or other resource could be found on the network; tools for one-time login, so that the user wouldn’t constantly be asked for passwords to site after site after site; tools for automatically allocating sub-tasks among the various resources; and most importantly, tools for implementing fundamental security procedures, so that you could be sure that an outside program trying to interact with your machine was actually serving a legitimate purpose, and hadn’t been sent by some hacker.

That was a tall order, says Foster, and they soon had a programming team numbering in the dozens trying to fill it. Nonetheless, he says, “at the 1997 supercomputer conference we were able to demonstrate a grid with some 80 sites worldwide running Globus software. That was another thing that convinced people it was worthwhile and real.” By that point, moreover, they had even started to *call* it grid computing.

“We were looking to make a clean break with the term ‘metacomputing,’” explains Kesselman. “We’d eventually begun to understand that what we were really doing was collaborative problem-solving over networks—working together with other *people*, on problems where you might need to share certain resources. So we wanted a new name to capture that idea. Well, I can’t remember if it was Ian and me, or Larry Smarr, or the three of us together. But we came up with ‘grid computing’ from the analogy to the electric grid, where you’re sharing access to generators. And then the use of the term was codified in the book.”

“The book”—*The Grid: Blueprint for a New Computing Infrastructure*—was essentially the proceedings of a workshop that Foster and Kesselman held at Argonne in September 1997: a community-building exercise that featured presentations from individuals working on the whole range of grid-related technologies. “Now, a phrase like ‘new infrastructure’ has lots of chutzpah,” says Smarr, who wrote the book’s introduction. “But oddly enough, it’s turned out to be accurate. Out of that meeting and the book, this entire grid movement emerged.”

Certainly the timing was perfect; with the book as a manifesto, grid computing suddenly seemed to fill a need for scientists all over the world. In Geneva, Switzerland, for example, CERN was

already planning its next-generation particle accelerator, the Large Hadron Collider. “We estimated that when the Collider started running in 2006,” says Fabrizio Gagliardi, director of the CERN School of Computing, “it would produce particle collision data at the rate of 8 to 10 petabytes per year. That’s *peta*, as in a million gigabytes. Now, a large fraction of those data would have to be preserved over the entire experiment life, which would be about 10 years. At the same time, portions would have to be distributed to all the people at all the institutions that participate in CERN—all over the world. And, since the most interesting physics tends to be found in the rarest events, they

...what we were really doing was collaborative problem-solving over networks—working together with other people, on problems where you might need to share certain resources.

would be processing every bit of that data in multiple ways.” In short, says Gagliardi, the data-handling promised to be far more of a challenge than the computing itself. “So we defined a computational architecture for what we would need, then went shopping for a system of tools to build it—and discovered that while CERN had been debating, the computer scientists had come up with solutions.”

Several solutions, actually. At the University of Virginia, to take the most notable alternative, Andrew Grimshaw had been working since 1993 on an attractive and well-thought out set of grid-computing protocols known as Legion.⁴ (Legion is now being marketed by the Avaki⁵ Corporation of Cambridge, Massachusetts.) On the other hand, says Gagliardi, “we liked it that Globus was not a closed system”—meaning that Globus was backward-compatible with software that the physicists had already spent many years developing. “So we could migrate into grid computing step by step.” Indeed, in the interests of getting their system adopted as widely and as rapidly as possible, Foster and Kesselman had decided to emulate the now-famous Linux operating system and make the Globus source code available to any users who wanted it, so that they could study it, experiment with it, and suggest improvements.

So in the end, says Gagliardi, “we picked Globus.” The result was the European DataGrid,⁶ a three-year demonstration and infrastructure development project that began on January 1, 2001, with a commitment of 13.5 million from the European Union. In addition to particle physics, says Gagliardi, who is now the DataGrid’s director, the project will also be developing grid applications for two fields that face similarly daunting data challenges: earth observation and biology. By the beginning of 2002, he adds, “we had deployed 20 computers at CERN, as well as 10 to 20 machines at our other sites around the continent. So we’re already running in excess of 100 machines—all using Globus.”

Meanwhile, the grid-computing idea has been finding an even warmer welcome among scientists in the United States—with Globus once again being the choice of virtually every large project. One of the first was GriPhyN, the Grid Physics Network,⁷ which was organized by Foster and Paul Avery of the University of Florida, and

launched in September 2000 with a commitment of \$11.9 million from the NSF. Like the DataGrid, GriPhyN is a demonstration and development project for grid technology—only in this case, focusing on four different data-intensive physics experiments: the CMS and ATLAS detectors at the Large Hadron Collider; LIGO, the Laser Interferometer Gravitational-wave Observatory, which will detect gravitational waves from pulsars, supernovae, and the like; and the Sloan Digital Sky Survey, which is automatically mapping the faintest possible stars, galaxies, and nebulae. A more recent initiative is NASA’s Information Power Grid⁸ (IPG): an effort to seamlessly integrate the space agency’s far-flung supercomputers, data bases, and high-end instruments, so that a user will perceive them as running inside his or her desktop machine. The IPG has been running since 2001. Another is NSF’s Network for Earthquake Engineering Simulation Grid⁹ (NEESGrid): an effort to provide similar integration among computer simulations, physical model studies, and observational data that are currently scattered among some 20 earthquake engineering labs. The foundation hopes to have the NEESGrid fully operational by September 2004.

And then there’s the TeraGrid,¹⁰ announced by the NSF in August 2001, and arguably the most significant of the lot. “This is the put-your-money-where-your-mouth-is grid,” declares Argonne’s Charles Catlett, executive director of the project.

The TeraGrid, he explains, builds on the foundation’s Partnership for Advanced Computational Infrastructure,¹¹ which in turn was an outgrowth of the network of supercomputer centers that NSF has been operating since 1985. In the initial phase, funded at \$53 million, the TeraGrid’s computational resources will be divided among four sites: NCSA at the University of Illinois,¹² which will provide much of the actual processing power; Argonne, which will focus on high-resolution rendering and data visualization; the California Institute of Technology,¹³ which will focus on applications; and the San Diego Supercomputer Center,¹⁴ which will focus on data-intensive operations. Each of those sites, in turn, will be powered by a cluster of high-end microcomputers based on Intel’s new 64-bit “Itanium” processor, and running the Linux operating system. Collectively the

sites in this first phase should have a capacity of 13.6 trillion floating point operations per second, or 13.6 “teraflops.” (Thus the project’s name.) But to accomplish that, says Catlett, they will have to be integrated through a dedicated network running at 40 gigabits per second. “We’ve been talking for years,” he says, “but this will show us a lot about how the software really works in a production environment—not just the Globus software, but the Linux kernels, the TCP/IP stacks, the routers.”

We’ll know how well it works soon enough. After suffering through more than a year’s delay, largely caused by a slippage in Intel’s schedule for delivering the Itanium chips, the TeraGrid team got their first clusters up and running in late 2003. If all goes well, the rest will be in place by the first half of 2004. Assuming that they are, the all 13.6 teraflops will then be available to cosmologists, climate modelers, protein-folding experts—anyone who can convince the allocation committee that they have a project worth doing. “My slogan is, ‘We want to give the individual scientist the power of a national lab,’” says Argonne’s Rick Stevens, one of the co-principal investigators of the project.

And that will be just the beginning, Stevens adds: “the TeraGrid is also designed to expand.” To that end, the NSF has already planned two major expansions of the project. In October 2002, the foundation announced that the Pittsburgh Supercomputing Center¹⁵ (PSC) at Carnegie Mellon University and the University of Pittsburgh would become a partner in the TeraGrid—in the process, linking in an existing cluster there—and that the overall funding for the project would be upped by \$35 million. Then in September 2003 the NSF added yet another \$10 million and four additional partners: the Oak Ridge National Laboratory¹⁶; Purdue University¹⁷; Indiana University¹⁸; and the Texas Advanced Computing Center¹⁹ at The University of Texas at Austin.

On the technical side... one of the big challenges is scale-up...on the ‘social’ side are such nitty-gritty issues as standards-setting.

And after that, says Stevens, the model is the old NSFnet, which was created in the 1980s to link the foundation’s then-new supercomputer centers, but which turned out to provide a backbone for the growth of the Internet as a whole, as one smaller network after another linked in. The hope is that the TeraGrid will play a similar role, providing a backbone for smaller grids to accrete into a truly worldwide grid.

TeraGrid...all 13.6 teraflops will...be available to cosmologists, climate modelers, protein-folding experts—anyone who can convince the allocation committee that they have a project worth doing.

Indeed, that may be more than just a hope. Although the NSF hasn’t formally committed itself to support the TeraGrid beyond the first five years, the agency is already laying plans to make it the basis for a permanent, ever-expanding “cyberinfrastructure.”²⁰ According to the vision laid out in a January 2003 blue-ribbon panel report,²¹ the cyberinfrastructure would provide not just the high-speed networks, but access to a vast web of interconnected computational engines, data repositories, digital libraries, sensor networks, and remotely operated instruments such as telescopes and accelerators—anything that supports the U.S. (and world) research community as a whole. And that, if it ever comes to pass, could bring the vision of grid computing roughly up to where the Internet was in the late 1980s, on the eve of its sudden explosion into the popular consciousness.

So—will history repeat itself?

Maybe. But first, there’s still a lot of work to do. On the technical side, notes Foster, one of the big challenges is scale-up—“making sure that all the services and protocols can deal with hundreds or thousands of times more devices than they handle now, and understanding the failure modes that can occur in such systems.”

And then on the “social” side are such nitty-gritty issues as standards-setting. “Remember how much we’ve gained from the fact that every computer runs the Internet protocol,” says Foster: TCP/IP, as that protocol is known, is the very foundation of the Internet. So,

to ensure the same universality for grid computing, the grid communities in the United States, Europe, and Asia have formed the Global Grid Forum.²² Founded in 2001 and patterned after the Internet's standards-setting body, the Internet Engineering Task Force, the forum is quite explicitly *not* centered on Globus, even though that is by far the most commonly used protocol. Indeed, Legion creator Andrew Grimshaw was one of the original Grid Forum's founders. Instead, the organization's goal is to make sure that Globus, Legion, and any other grid protocol can interoperate seamlessly. "If every computer uses standard methods for managing authentication, authorization, describing resource capabilities, and negotiating access for resources," says Foster, "that's a big win."

Peer to Peer Computing: Computing at the Edge of the Net

At the same time, the grid computing pioneers have also gotten serious about their alliances with Peer-to-Peer (P2P) computing—the catch-all name for a variety of efforts that independently emerged in the 1990s to exploit the collective power of networked personal computers and workstations. If the utility model of grid computing discussed in the previous section is analogous to central-station electric power, in the sense that the services are usually provided by supercomputers and large server farms, then the Peer-to-Peer approach is the analog of solar power: it attempts to harvest resources that are diffuse and low-intensity, but collectively too vast to ignore.

The roots of P2P go back to the 1980s, when universities, research labs, and corporations were deploying increasingly high-powered PCs and workstations on as many desktops as possible, and linking them with Local Area Networks (LANs). Even then, it was obvious to everyone that those machines were wasting the vast majority of their computer cycles in idle mode, while their users were talking on the telephone, or off at lunch, or doing any of a thousand daily activities

that didn't involve computers. And so, by the end of the decade, many efforts were underway to capture those cycles.

At the Xerox Palo Alto Research Center (PARC), for example, Bernardo Huberman and his colleagues attacked one of the major challenges in that effort: how do you allocate the work? How do you make the most effective use of machines that may be all over the map in terms of processing power, memory, and storage capacity, and that will be constantly moving in and out of the labor pool? (Say for example, a user comes back from lunch and decides she needs her computer to draft a report.) The PARC group's Spawn system showed that resources could, in fact, be allocated very efficiently if a computer that needed help simply broadcast a "request for proposals" over the network, and all the other computers submitted "bids" based on how much free capacity they had at the moment.* The medium of exchange could be real dollars and cents. Or, as in the current-generation peer-to-peer system Mojo Nation,²³ which is similar in spirit, it could be some sort of virtual currency.

At roughly the same time, in 1988, Miron Livny and his group at the University of Wisconsin, Madison, began work on the Condor²⁴ protocols, which are able to harvest wasted computational power from idle desktop machines and then combine it with the power of dedicated servers and clusters. The project is still ongoing; today, the open-source Condor software is not only managing more than 1000 machines every day at the university, but also is managing similar installations for hundreds of other organizations in industry, government, and academia.

...the Peer-to-Peer (P2P) approach is the analog of solar power: it attempts to harvest resources that are diffuse and low-intensity, but collectively too vast to ignore.

* C.A. Waldspurger, et al. 1992. "Spawn: A Distributed Computational Economy," *IEEE Transactions on Software Engineering* 18(2):103-17. Available online at <http://www.computer.org/tse/ts1992/e0103abs.htm>.

Other development efforts followed, including Grimshaw's Legion. For the most part, however, these systems tended to be deployed on machines that were controlled by a single institution, with connections that were provided mostly by local area networks. That has a number of advantages, not the least of them being that everything is inside the firewall, so to speak, where security is not quite such an issue. But with the rise of the Internet in the mid-1990s, it didn't take long for the P2P notion to be reinvented yet again on a much grander scale.

One of the first such efforts, starting in 1995, was the Great Internet Mersenne Prime Search,²⁵ in which volunteers allow idle time on their PCs to be used in the hunt for a certain type of very large prime number. (The project's sixth and largest such prime number, discovered in November 2003, has more than 6 million digits.) A few years later GIMPS was an inspiration for SETI@home,²⁶ in which radio telescope data from the search for extraterrestrial intelligence project is parceled out across the Internet and processed by idle PCs running a special screen-saver program. The success of SETI@home—since its launch in May 1999, the project has sent data to nearly 5 million users at one time or another—has in turn inspired a number of others.²⁷ At Stanford University, for example, the Folding@home²⁸ project uses a very similar approach to calculate how newly made protein molecules fold—and mis-fold—inside the cell. (Mis-folded proteins are thought to be the problem behind a number of devastating neurological disorders, including Alzheimer's, Parkinson's, and Mad-Cow disease.)

Meanwhile—and again, totally independently—a plethora of file-sharing systems began springing up on the Internet, initially fueled by the younger digerati's enthusiasm for trading MP3-encoded music tracks, and quickly spreading to the trade of video files, image files, and documents. Among the first was Napster,²⁹ created by 19-year-old Shawn Fanning. Napster was put out of business in 2001, when the recording industry successfully argued in court that this kind of open and royalty-free file-sharing constituted massive copyright infringement. (The technology and the name were revived in 2003, when Napster was relaunched as a legal online music sales

service.) But by that point other royalty-free file-sharing systems were rising to take its place, among them Gnutella,³⁰ launched in late 1999, and KaZaA,³¹ “the world's most downloaded software,” launched in early 2002.

The Napster/SETI@home phenomenon fascinated academics, who quickly saw that these models of Internet-based distributed computing could be applied much more broadly. Indeed, the very scale of the Internet—more than 150 million host computers and growing—has encouraged the researchers to think big. Most notably, in the March 2002 issue of *Scientific American*, SETI@home director David P. Anderson and Berkeley computer scientist John Kubiawicz envisioned an “Internet Scale Operating System,” or ISOS,* that would encompass every one of those hosts, from the simplest PC (or handheld PDA) to the most powerful mainframe. Such an ISOS would turn the Internet into a collective computational resource vastly greater than anyone could afford in-house, they argued. Moreover, that power would only continue to grow as the Internet grew (and as the individual hosts got upgraded).

Of course, the technical challenges are large, as Anderson and Kubiawicz are the first to point out. How would the ISOS keep track of things, for example? Once it had chopped up your data and/or your computation into millions of fragments distributed all over the Internet, how would it find them again—and then make sure that what it had found was, in fact, the most recent update? For that matter, how would the ISOS distribute the fragments in the first place? How would it find the best allocation in a universe of Internet hosts that can vary from low-end laptops to world-class

The Napster/SETI@home phenomenon fascinated academics, who quickly saw that these models of Internet-based distributed computing could be applied much more broadly.

* Anderson, David P., and John Kubiawicz. March 2002. "The Worldwide Computer," *Scientific American* 286(3):40-7.

supercomputers? How would it take into account the fact that some hosts are hidden away behind firewalls and are extremely limited in what they can accept? That some hosts are available only sporadically (when, say, their user lets them go into screen saver mode)? Or that some are not available at all for certain types of jobs (with an owner who would allow, say, biomedical research, but not military applications)? And perhaps most fundamental of all from a sociological standpoint, how would the ISOS give computer owners an incentive to let their machines take part? Volunteerism is all very well for a sexy, high-profile project like SETI@home. But a long-term, workaday system might very well require a market-based solution along the lines pioneered by Spawn, in which users earn when they make their machines available to the ISOS, and pay when they tap into it.

But then, for researchers, challenges like this are half the fun. The technology required to implement such an ISOS is currently under development in a variety of research efforts, including Kubiawicz's own Oceanstore³² and Tapestry³³ projects at Berkeley, Microsoft's Pastry³⁴ and Farsite³⁵ projects, MIT's Chord³⁶ system, the Cosm³⁷ project, and many others—not to mention all the grid computing work mentioned in the previous section.

Back on the ground, meanwhile, the P2P phenomenon was also fascinating the business community. By the late 1990s, at the height of the tech boom, a number of startups were trying to bring the technology to market. And some, having carved out a substantial niche installing P2P systems for corporate clients, seem to have survived the downturn reasonably well. One good example is Avaki,³⁸ which markets Andrew Grimshaw's Legion. Another is Entropia,³⁹ which has been offering a commercial version of the Mersenne prime software since 1997. By 2000, in fact, there were enough companies in the field that they formed an industry consortium known as the P2P working group⁴⁰ to work on standards, interoperability, and other common problems.

And finally, not surprisingly, all this P2P activity was being closely watched by participants in the Global Grid Forum. There were some obvious differences in approach and philosophy. Whereas

the Globus protocols tended to emphasize security and the utility model of computing, for example, P2P tended to emphasize bottom-up self-organization and the ability of individual devices (and users) to interact directly without central coordination. (Thus the many references to resources at the “edge” of the net.) Nonetheless, the similarities were obvious to everyone, as was the potential for synergy. By 2001, both Condor and Entropia had integrated the Globus protocols into their offerings and the Globus project had adapted several sections of Condor code for version 2.0 of the Toolkit. By 2002, the P2P Working Group had merged itself into the Global Grid Forum, where it now exists as a special interest area. So in effect, at least when it comes to standards-setting efforts, the two movements have become one.

Web Services:* The Self-Assembly Model

Finally, even as the Global Grid Forum was emphasizing the utility model of grid computing, and the Peer-to-Peer movement was stressing bottom-up self-organization, an emerging web services movement was emphasizing what might be called the “ad hoc-ness” of grid computing: the ability to assemble processes on the fly.

Web services, like P2P, had its roots in the 1980s. The difference was that in this case the impetus came not from academics or technically adept audiophiles, but from increasingly desperate corporate IT officers. Over the years their organizations had saddled them to an increasingly hopeless hodgepodge of incompatible databases, incompatible software, and incompatible machines—from mainframes to minicomputers to PCs. With the spread of networking, it was becoming more and more obvious that this mess was not only a maintenance nightmare, but was actually undermining one of the greatest potential benefits of IT: its ability to integrate and streamline business processes.

* A number of background papers on web services, including a two-part history of the field, can be found online at <http://www.webservices.org/index.php/article/archive/61>.

It was clear that something had to be done to end the madness. And indeed, a variety of technical solutions were proposed. In 1991, to take a notable example, an industry consortium known as the Object Management Group introduced CORBA,⁴¹ the Common Object Request Broker Architecture. CORBA was intended to be an open, vendor-independent protocol that would allow business applications to communicate and interact seamlessly across the network. And indeed, CORBA achieved some currency during the early 1990s. In practice, however, the “vendor-independence” part left something to be desired. Implementations from different vendors would often not communicate properly one another, which meant that the “seamless” part didn’t do well; customers usually had to give in and buy all their software from a single company just to be sure that everything would work together as intended. It didn’t help when Microsoft introduced a rival, Windows-only protocol known as DCOM in 1996. The upshot was that this first wave of attempted integration didn’t come close to living up to its promise.*

Still, as the 1990s wore on, the impetus for that integration was only getting stronger. And at the same time, not incidentally, the Internet was providing a new and stunningly instructive reminder of the power that could come from common standards. So by the late 1990s, the ideas behind CORBA and DCOM had reemerged—this time oriented around the World Wide Web itself as a ready-made, universal standard for communication. True, the web had been conceived as a way for human users to browse through documents just by following one hyperlink after the next. But it was easy enough to generalize the idea and imagine using the underlying mechanisms of the web to let application talk to application. The hyperlinks, in that case, would link not just documents, but “web services” of all kinds—processes,

...an emerging web services movement was emphasizing what might be called the “ad hoc-ness” of grid computing: the ability to assemble processes on the fly.

data, visualization tools, whatever. It was just a matter of eliminating the user interface, meaning the browser and automating all the steps that are usually handled manually.

Inevitably, there were a great many opinions and rivalries about how to accomplish that last feat. The various competing standards began to coalesce (more or less) in April 2001, when web services became an official activity of the World Wide Web Consortium⁴² (W3C)—in effect, bringing it into the same standards-setting process that governs the rest of the web. But there were still a number of other heavyweights in the standards game, including an industry group known as OASIS, the Organization for the Advancement of Structured Information Standards,⁴³ and the Internet Engineering Task Force itself. So in February 2002, recognizing that the standards were still in flux and that the danger of fragmentation was still very real, Microsoft, IBM, and many other firms formed a consortium known as the Web Services Interoperability⁴⁴ organization, or WS-I, dedicated to achieving exactly what its title suggests.

...hyperlinks...would link not just documents, but “web services” of all kinds—processes, data, visualization tools, whatever.

The emerging model of web services is based on four main conventions:

- XML, the eXtensible Markup Language,⁴⁵ is a standard way to describe data. It is similar to, but much more general than the Hypertext Markup Language used to create displayable web pages. Indeed, XML is deliberately designed to be as general as possible, so that specialized versions can then be created for specific applications.
- WSDL, the Web Services Description Language,⁴⁶ is one such specialized version of XML, used to express what a web service can do and how to communicate with it. If the web service were an applet for, say, obtaining stock quotes,

* A good discussion of this history is available online at <http://www-106.ibm.com/developerworks/webservices/library/ws-arc3/>.

its WSDL description would explain precisely what digital messages would activate the applet, and precisely what it would do in reply.

- SOAP, the Simple Object Access Protocol,⁴⁷ is another specialized version of XML—in this case, designed to give the web services a standard way to encode their messages and their responses for transmittal across the web.
- UDDI, the Universal Description, Discovery and Integration⁴⁸ protocol, is a kind of online yellow pages developed by the OASIS consortium. Using the XML-based standards described above, UDDI provides a web-based, distributed directory in which web services can list themselves, describe themselves, and find each other.

...all sides have been pushing hard to get grid computing and web services integrated.

If these standards sound a lot like the ones being promoted by the Global Grid Forum, they are. Indeed, as Foster and Kesselman have pointed out, the only reason they didn't base their Globus protocols on the web in the first place was that back in 1995, when they started, the web technology wasn't robust enough to support grid computing. But now it's a different story—which is why all sides have been pushing hard to get grid computing and web services integrated. In early 2002, for example, Kesselman, Foster, Foster's Argonne colleague Steven Tuecke, and IBM's Jeffery Nick took a big step toward that goal by proposing a set of integrated protocols they called the Open Grid Services Architecture.⁴⁹ That architecture was embraced with relatively little argument by the Global Grid Forum itself, and in early 2003 it was implemented in version 3.0 of the Globus Toolkit. On the industry side, meanwhile, Microsoft was arranging to have Globus ported to Windows XP—which in effect made grid computing a part of its own web services effort, known as the .NET strategy—while more than a dozen companies were announcing plans to build Open Grid Services-compliant versions of major database systems, Java-hosting environments, and the like.

And most recently, on January 20, 2004, the Globus Alliance teamed with IBM, Hewlett-Packard, and a number of other companies to propose a new set of standards, the Web Services Notification specification and the Web Services Resource Framework, that promise to take the convergence all the way.*

A Work in Progress

As should be clear from the foregoing, grid computing is still very much a work in progress. Nonetheless—and despite the downturn—industry interest and investment have been strong. The utility and P2P approaches have an obvious and potentially very large market in biotech, financial services, oil exploration, web hosting, online gaming,[†] and any other application that requires massive number crunching and data handling. And the web services approach has an equally obvious market in business-to-business activities such as supply chain management, in which companies have to get their applications to communicate across their mutual firewalls. As a result, most of the major vendors have already undertaken one or more initiatives in this area—among them the .NET⁵⁰ strategy at Microsoft, the N1⁵¹ architecture at Sun, the Adaptive Enterprise⁵² vision at Hewlett-Packard, and the On-demand⁵³ initiative at IBM. Two good sources of perspective and up-to-date industry news are CNET's "Get Up to Speed" pages on utility computing⁵⁴ and web services.⁵⁵ Also very useful is WebServices.org,⁵⁶ the "web services industry portal."

...grid computing is still very much a work in progress.

* A news release is here:

http://www.marketwire.com/mw/release_html_b1?release_id=61977. The Globus web site has presentations on the proposal by Ian Foster and IBM's Daniel Sabbah.

† In 2002, IBM agreed to develop grid-based online gaming in collaboration with the West Virginia startup Butterfly.net. A news story is here: <http://news.com.com/2100-1040-903691.html?tag=nl>.

The bottom line here is that grid computing does seem to have a solid future, both in scientific research and in the marketplace. However, what's considerably less clear is whether grid computing will ever become a utility in the way electric power is a utility. After all, goes the argument, it's one thing to talk about sharing resources that can take place inside the firewall, where you own the machines and have a measure of control, or inside a "virtual organization," where you're dealing with known and (reasonably) trusted partners. Either way, the advantages are clear. But it's quite something else to talk about plugging into a black box in the wall and buying your computational services from a third party.

From a business perspective, says Jeff Clarke,⁵⁷ vice president of Dell's product group, the reliability and security implications of doing that are the stuff of nightmares. Are those services really going to be there when you need them? Is the vendor really going to be able to guarantee an acceptable quality of service in the face of fluctuating demand? And do you really want to trust your mission-critical applications to a vendor that's drawing processor power from who knows where? How can you be sure that some kid hooking into the vendor's grid isn't going to bring along a virus that will infect the whole operation?

Then there's the economics of computing. According to Clarke, it isn't at all like the economics of electricity. Electric power really is a commodity with a mature technology and a relatively stable price. But IT is still in this weird evolutionary state of rapidly increasing capability at a rapidly decreasing cost. And in that environment, he argues, it's generally cheaper for a company to buy its own computers and operate them in-house—especially in light of those security and reliability concerns.

But hold on, says Philip Brittan,⁵⁸ chairman of graphical user interface developer Droplets: companies trust mission-critical services to outsiders all the time, from customer relations management to enterprise resource planning to payroll. Security and reliability are certainly concerns, but obviously aren't unmanageable. Even first-generation P2P systems like SETI@home have built-in safeguards and much tougher ones are under development.

And as for the economics, notes Patricia Suelz,⁵⁹ executive vice president of Sun Services, rapid changes in the technology are actually an argument for the utility model. "Once an organization buys a computer, there is no guarantee it will recoup its money's worth over the long run,"

she writes. In fact, it's often very hard to figure out what the benefits of a given IT investment actually are. More revenue? Greater productivity? What? The result is a disconnect between costs and benefits that leads to all kinds of management problems, she adds—not the least of them being expensive installations that never get used to their full potential and that can't easily be changed in the face of changing business conditions.

But now look what happens with utility computing, says Suelz. You can pay as you go instead of making a big up-front investment based on a guess. You can scale your computer usage up and down to match your workload. You can get away from endlessly chasing after the latest technology. And because you're monitoring on a pay-per-use basis, you can get a much better sense of how costs and benefits actually match. The result is a much more effective and efficient use of the technology.

Of course, it remains to be seen how these factors will shake out. But the argument does suggest a fairly predictable model for grid computing's evolution, with the earliest deployments generally taking the form of enterprise grids and virtual organizations—as has indeed been the case so far. Then only slowly, as the technology matures and as people gain experience and trust, will we begin to see true grid computing utilities.

Indeed, it could be quite a while before we see the kind of Internet Scale Operating System that Anderson and Kubiawicz envisioned. But one day it could be here, whether through the evolution of the NSF's Cyberinfrastructure, or something else. One day, argues Larry Smarr, we could see interconnected grids at every scale: "supernodes,

Every car will be a grid connected to the Internet. And with maybe two dozen microprocessors per car, times 100 million cars, that's already ten times the size of the Internet today.

mid-level nodes, and micronodes, all tied together by links of varying capacity. The supernodes will correspond to things like the TeraGrid, where you're creating the biggest possible supercomputers—but there will be very few of them. There will be a lot more mid-sized nodes, where you use software like Entropia to hook together a bunch of Windows PCs. If the TeraGrid is like central station electric power, this will be like gathering solar energy.”

And the micronodes will be everywhere, says Smarr, thanks to the wireless revolution: “Over the next 20 years we'll be extending the Internet throughout the physical world. So wherever there's air, there's bits. And because of the miniaturization of components, we'll have billions of endpoints that are sensors, actuators, and embedded processors. They'll be in everything, monitoring stress in bridges, monitoring the environment—ultimately, they'll even be in our bodies, monitoring our hearts. Every car will be a grid connected to the Internet. And with maybe two dozen microprocessors per car, times 100 million cars, that's already ten times the size of the Internet today.

“Now, to me, to first order, this wireless extension of the grid will have nothing to do with computing. Yes, the data will probably get computed many times as it flows through the grid. But the grid itself is going to be all about data flow. And that's why we'll have to build in security, and so on, from the base. We can't do it as an afterthought. The planet is assembling the grid infrastructure—at multiple scales—that it will live on for the rest of 21st century.”

Endnotes

- ¹ <http://www.calit2.net/>
- ² <http://www.teragrid.org/>
- ³ <http://www.globus.org/>
- ⁴ <http://www.cs.virginia.edu/~legion/>
- ⁵ <http://www.avaki.com/>
- ⁶ http://web.datagrid.cnr.it/servlet/page?_pageid=1407&_dad=portal30&_schema=PORTAL30&_mode=3
- ⁷ <http://www.griphyn.org/index.php>
- ⁸ <http://www.ipg.nasa.gov/>
- ⁹ <http://www.neesgrid.org/index.php>
- ¹⁰ <http://www.teragrid.org/>
- ¹¹ <http://www.paci.org/>
- ¹² <http://www.ncsa.uiuc.edu/>
- ¹³ <http://www.cacr.caltech.edu/resources/teragrid/>
- ¹⁴ <http://www.sdsc.edu/>
- ¹⁵ <http://www.psc.edu/>
- ¹⁶ <http://www.oml.gov/>
- ¹⁷ <http://www.purdue.edu/>
- ¹⁸ <http://www.indiana.edu/>
- ¹⁹ <http://www.tacc.utexas.edu/>
- ²⁰ <http://www.cise.nsf.gov/div/index.cfm?div=sci>
- ²¹ <http://www.cise.nsf.gov/sci/reports/toc.cfm>
- ²² <http://www.gridforum.org/>
- ²³ <http://www.mojonation.net/>
- ²⁴ <http://www.cs.wisc.edu/condor/>
- ²⁵ <http://www.mersenne.org/>
- ²⁶ <http://setiathome.ssl.berkeley.edu/>
- ²⁷ <http://www.intel.com/cure/research.htm>
- ²⁸ <http://www.stanford.edu/group/pandegroup/folding/>
- ²⁹ <http://www.napster.com/>
- ³⁰ <http://www.gnutella.com/>
- ³¹ <http://www.kazaa.com/us/index.htm>

- ³² <http://oceanstore.cs.berkeley.edu/>
- ³³ <http://www.cs.berkeley.edu/~ravenben/tapestry/>
- ³⁴ <http://www.research.microsoft.com/~antr/pastry/>
- ³⁵ <http://www.research.microsoft.com/sn/farsite/>
- ³⁶ <http://www.pdos.lcs.mit.edu/chord/>
- ³⁷ <http://www.mithral.com/projects/cosm/>
- ³⁸ <http://www.avaki.com/>
- ³⁹ <http://www.entropia.com/>
- ⁴⁰ <https://forge.gridforum.org/projects/p2p/>
- ⁴¹ <http://www.omg.org/gettingstarted/corbafaq.htm>
- ⁴² <http://www.w3.org/>
- ⁴³ <http://www.oasis-open.org/who/>
- ⁴⁴ <http://www.ws-i.org/>
- ⁴⁵ <http://www.w3schools.com/xml/default.asp>
- ⁴⁶ <http://www.w3schools.com/wsdl/default.asp>
- ⁴⁷ <http://www.w3schools.com/soap/default.asp>
- ⁴⁸ <http://www.uddi.org/about.html>
- ⁴⁹ <http://www.globus.org/research/papers/ogsa.pdf>
- ⁵⁰ <http://msdn.microsoft.com/webservices/>
- ⁵¹ <http://www.sun.com/software/solutions/n1/>
- ⁵² http://www.hp.com/products1/promos/adaptive_enterprise/us/adaptive_enterprise.html
- ⁵³ <http://www-1.ibm.com/services/ondemand/index.html>
- ⁵⁴ http://news.com.com/2001-7339_3-0.html?tag=nefd_guts
- ⁵⁵ http://news.com.com/2001-7345_3-0.html?tag=nefd_guts
- ⁵⁶ <http://www.webservices.org/>
- ⁵⁷ http://news.com.com/2010-7339_3-5091350.html?tag=guts_bi_7339
- ⁵⁸ http://news.com.com/2010-7784_3-5099858.html?tag=guts_bi_7339
- ⁵⁹ http://news.com.com/2010-7339_3-5092235.html?tag=guts_bi_7339

3. Autonomic Computing: The Technology of Self-Management

As Frederick P. Brooks described so vividly in *The Mythical Man-Month*,¹ his 1975 classic on the perils of software development, the invention of computers took engineering complexity into a whole new realm. Unlike a bridge, or a skyscraper, or even an electrical circuit, all of which are relatively static structures, the processors, devices, and subroutines that comprise a large, modern computer system exist to produce *behaviors*—which then go on to produce a whirlwind of contingencies, side-effects, and unanticipated interactions that can all too easily leave our finite human brains hopelessly overwhelmed. Still, by investing major effort in the development of programming languages, operating systems, and development environments, not to mention a whole new profession, software engineering, information technologists have usually managed to stay a step or two ahead. Usually. More often than not.

Lately, however, complexity has been moving up fast. As recently as a decade ago, says Alan Ganek, Vice President of IBM's Autonomic Computing initiative, the prototypical example of a large, complex computer system would have been a citywide ATM network supporting a few tens of thousands of customers. Yet now, says Ganek, "some of our clients have that many *servers*. Any big financial institution today is supporting millions of users logging in over the Internet." Worse, he says, that workload can be variable in the extreme. Traffic

on CNN's website during the U.S. Open Tennis tournament can surge to fifty times its normal levels. "Things can come and go in seconds," says Ganek, "which means that it's just not possible to have manual procedures."

Such examples can be multiplied endlessly, from corporate supply chains stretching across the planet to personal computer operating systems so Byzantine that they need patches once a month. Partly, of course, this explosion of complexity reflects the relentless workings of Moore's Law, together with our avid embrace of networking technology. Networked desktop and laptop PCs are almost everywhere, it seems, along with cell phones, Personal Digital Assistants, pagers, smart cards, Global Positioning Satellite receivers, MP3 players, and more. Behind the scenes, moreover, we're utterly dependent on all manner of servers and routers that most of us are only dimly aware of, not to mention a host of microprocessors and sensors operating invisibly inside our microwaves, our refrigerators, our cars, or even our wristwatches. And thanks to the rapid spread of connectivity in general—and of wireless technologies³ such as Wi-Fi and Bluetooth in particular—these processor-equipped devices are not only proliferating, but are more and more often talking to one another, and trying to cooperate (or compete) with one another.

The result is a kind of digital cacophony that's growing louder by the day. Nonetheless, the complexity is being driven not just by the technology itself, but even more importantly, by the new ways people want to *use* the technology. Take enterprise integration, for example. As shoppers, we rarely think about what it takes to keep all those shelves stocked at the local Safeway. But at companies like consumer products giant Procter & Gamble, today's supply chain managers have found themselves building a system that's about as complex as they come. Remember, says Jake Barr, P&G's associate director for supply network innovation, every time the company introduces a new product, "we have to decide how rapidly we're going to introduce that product,

...we're utterly dependent on all manner of servers and routers that most of us are only dimly aware of...

where we're going to make it, where we're going to get the materials, and how we're going to get it to the retail outlets." Now think about doing that for some 250 product lines, ranging from Tide to Pringles to Pampers, while simultaneously coordinating a tangle of outsourced suppliers and strategic partnerships. This isn't easy, to put it mildly; make one apparently innocent change in the system, and you may suddenly find that the Pringles are piling up in one warehouse, while shelves are going empty somewhere else—the supply-chain equivalent of a traffic jam. "That's why we don't talk about a supply *chain* anymore," says Barr. "Today we use the words 'supply *network*.'" Within three to five years, moreover, thanks to the spread of Radio-Frequency ID tags—essentially, microchip-based "electronic product codes" that can be embedded right in the packaging—the industry expects to be tracking every single box of detergent or toothpaste through every stage in the product life cycle, from factory to warehouse to retailer's shelf to checkout line. So the pressure is on, says Barr: every manufacturer is scrambling to make its supply network as flexible and as responsive as possible, so that it can adapt to changes in consumer demand just as rapidly as the data come in.

Much the same story could be told about manufacturing, banking, utilities, and more. In an era of rampant globalization, with markets, technologies, and, occasionally, governments in a state of constant flux, more and more firms have made it a life-or-death priority to integrate their business processes end to end—to connect everyone from their suppliers to their customers in the most flexible way possible, so that they can respond to changes as rapidly as possible.

On the research frontier, meanwhile, there is pervasive computing—the subject of the first seminar in the Wilson Center/IBM series. Also known as "ubiquitous," or "ambient" computing, pervasive computing builds on the vision first articulated in 1988 by the late Mark Weiser,⁴ chief technologist at PARC, the Xerox Palo Alto Research Center. Weiser later described it this way: "For thirty years most interface design, and most computer design, has been headed down the path of the 'dramatic' machine. Its highest ideal is to make a computer so exciting, so wonderful, so interesting, that we never want to be without it. A less-traveled path I call the 'invisible'; its highest ideal is to

make a computer so imbedded, so fitting, so natural, that we use it without even thinking about it.”⁵

Weiser’s vision struck a strong chord in the research community. And today, as another 15 years’ technological advancement has begun to bring it into the realm of the almost possible, it’s under active development in a variety of demonstration projects.⁶ One thing that’s very clear, however, is that “invisibility” will require computers that can take on most of the management chores that are now left to the users—with all the internal complexity that implies. Consider the problem of integration, for example. Since no one, with the possible exception of the geekiest computer nerds, wants to walk around wearing a utility belt that’s full of PDAs, cell phones, MP3 players, and whatnot, researchers are trying to figure out how to create a universal information appliance: a single, handheld gadget that would automatically adapt itself to function as a cordless phone at home, a cell phone on the road, a tablet PC, a PDA, a GPS receiver, a projector controller, and much more.⁷ (The device would also upgrade itself automatically whenever some new functionality came along, so that we wouldn’t have to run out and buy a new gadget every time.)

Or consider the problem of connectivity. The world of pervasive computing will be much like the world is today: full of people, cars, printers, handhelds, and myriad other entities, all of which are constantly moving (or being moved) from place to place. The difference is that every one of those entities will need at least intermittent connections to the network, on demand, often *while* it is moving. This means that each device will have to reach out when necessary, and automatically hook itself into the cellular phone system, or the conference room’s local Wi-Fi network, or the nearest desktop computer’s Bluetooth connection, or whatever else gives it the best data rate and most reliable connection at that particular moment. If it’s moving around, it will also have to shift to new connections as needed—and do so in mid-conversation, without the user’s being aware of anything except a possible speeding up or slowing down of the data rate. As it enters each new location, moreover, each device will have to reach out and identify any other devices and software resources that are there, while simultaneously introducing itself to *them*; soon enough, after all,

they may be given a new task that they will have to carry out as a team (e.g., “Send this document to the nearest printer”). In short, as the Project Oxygen web site puts it, the networks of the pervasive world will become “*ad hoc* collaborating communities of people and computing devices, [which] configure and reconfigure themselves automatically, as nodes appear, migrate, and disappear.” Needless to say, that’s a far cry from our current-generation networks, in which a more or less static set of data pipes feeds a more or less stationary set of computers—each of which has an absolutely static, fixed-in-stone “IP address.” (That’s the numeric address that your web browser looks up when you type in, say, *www.ibm.com*.)

The bottom line in all this is that there aren’t enough management information specialists in the world to keep up with what we’re asking our computer systems to do. And the individual users certainly aren’t a substitute, not when so many of us already find it such a major hassle to install a simple software upgrade; imagine trying to do that level of installation, maintenance, synchronization, and upgrading on all the embedded and mobile devices involved in pervasive computing. Somehow, our fast-evolving information systems are going to have to do a much better job of managing *themselves*. A good analogy is the electric power grid, says David Turek, director of IBM’s server division: “To use a hair dryer, you don’t have to worry about how the turbine is designed up in Niagara Falls, or the physics of power transmission. You just plug it into a wall socket.” An even better analogy might be a star athlete charging down the basketball court, says IBM’s Ganek: “He’s not thinking about how to constrict the pupil of his eye, or how to elevate his breathing and heart rate”—all of which are handled below the conscious level by his autonomic nervous system. “He’s thinking about the strategy of the game.”

Unfortunately, Ganek adds, “today’s computer systems are remarkably unlike that.”

Three years ago, in recognition of this reality (and inspired by the nervous system analogy), IBM launched the research program in Autonomic Computing that Ganek now heads. The name “autonomic” hasn’t caught on everywhere, if only because it’s IBM’s. Microsoft, for example, talks about “trustworthy” computing, while others prefer

more generic, if less evocative, terms such as “self-managing.” But the idea itself most certainly *has* caught on, if only because the problem of complexity is very real, and far transcends any one company. By whatever name, moreover, there is broad-brush agreement about what an autonomic, self-managing system needs to accomplish. It should be *self-configuring*, meaning that each new device would automatically fit itself into the existing community of devices without requiring the user to go through some special installation procedure. It should be *self-optimizing*, meaning that the system would automatically distribute tasks and subtasks to the various devices in the most efficient way possible. It should be *self-protecting*, meaning that the systems would detect and guard itself against damage from accidents, equipment failure, or outside attacks by hackers and viruses. And it should be *self-healing*, meaning that the system could detect, diagnose, and recover from any damage that did occur.

The challenge, as always, is to figure out how to turn those requirements into working technology—technology that can function together seamlessly, at every level.

Strategies for Self-Management

The good news is that the developers aren’t starting from scratch; their predecessors, intentionally or not, have already given them an extensive legacy of autonomic technology to build on. Probably the oldest example is the operating system, a category of software that emerged in the 1950s as harried computer center operators began to equip their mainframes with routines to handle the most repetitive tasks, such as reading in those big decks of punch cards and spitting out the results on those thick stacks of fan-fold paper. In much the same spirit, mod-

The bottom line in all of this is that there aren’t enough management information specialists in the world to keep up with what we’re asking our computer systems to do.

ern operating systems such as Microsoft Windows and Linux give harried programmers a set of standard routines to handle the low-level details of memory and disk allocation, communication protocols, process scheduling, and peripheral interfaces. (In theory, at least, present-day systems also provide a measure of Plug-n-Play self-configuration, in that they can recognize a new peripheral device and install the correct drivers automatically. Sometimes.) Even more important, however, today’s operating systems can provide each program with a “virtual computing environment” that allows it to behave as if it had the machine all to itself. This simplifies the software development enormously, by minimizing the possibility of disastrous interactions between different programs. But it also confers a major element of self-protection, in that the OS can (usually) isolate and terminate a malfunctioning application without crashing the whole system. Still another layer of self-protection comes from the operating systems’ built-in provision for firewalls and anti-virus procedures.

An autonomic example of slightly more recent vintage is—of course—the Internet. The network has had a considerable amount of self-healing and self-protecting ability from the beginning, ever since it was launched in 1969 as the Arpanet. The minute a node or link goes down, the system’s packet-switching architecture automatically searches out an alternate route that will get the data packets through. And in case packets do go astray, the network software will automatically check each message on the receiving end and arrange to have any missing data resent from the source. Ever since the late 1970s, moreover, the basic “internetworking” protocol, TCP/IP, has given the network at least a limited kind of autonomic self-configuration, in the sense that anyone can add a network domain with minimal fuss. Once you have an address assignment, you just have to hook your network in. It doesn’t matter what bit rate or packet size your network uses, or what kind of transmission medium it employs—radio, satellite links, land lines, Wi-Fi, whatever. As long as the network knows how to speak TCP/IP at the interface, the bits will flow.

Finally, we have the Internet-based file-sharing services pioneered by the late Napster, and now carried on by successors such as Gnutella and KaZaA. Whatever one may think of their users’ attitude toward

copyright and intellectual property rights, it's hard to imagine a more effective demonstration of autonomic self-protection and self-repair.

The very fact that these services are totally decentralized, with perfect digital copies of songs, photographs, movies, pornography, and whatever else scattered through millions of computers, means that their world-wide database is effectively invulnerable to destruction—even when they are under attack by well-financed consortia of record companies and other such authorities.⁸ No matter many copies are lost to hard-disk crashes or legal action, more copies are always out there.

Likewise with Internet-based collective computation efforts such as SETI@home, in which radio telescope data from the search for extraterrestrial intelligence project is parceled out across the Internet and processed by idle PCs running a special screen-saver program; or the similar Folding@home, in which some 20,000 idle PCs around the world are used to carry out molecular-dynamics simulations of how proteins fold. The very fact that each piece of the computation is carried out independently means that the system is all but invulnerable to hardware failures. If one machine crashes or drops out of the program, another can fill in the gap. Indeed, the system is even “self-maintaining” in the sense that the individual owners take care of maintaining and upgrading their own machines.

Taking these examples together, then, one obvious strategy for autonomic computing is to combine them—to create what SETI@home director David P. Anderson and Berkeley computer scientist John Kubiawicz have called an “Internet Scale Operating System,” or ISOS.⁹ Such a system would be autonomic in the extreme, giving our multiple connected devices the power to reach out into cyberspace, find computational resources wherever they might be, and then assemble them on the fly into whatever applications we needed—without our

An autonomic example of slightly more recent vintage is—of course—the Internet. The network has had a considerable amount of self-healing and self-protecting ability from the beginning...

ever having to know or care where the resources were, what computers they were running on, or how any of this occurred. And indeed, the basic architecture of such a system is already taking shape, thanks to the convergence of three movements that sprang up independently in the 1990s.

Perhaps the best-known of these movements is peer-to-peer computing, which is an umbrella term that covers everything from grass-roots efforts such as KaZaA and SETI@home to industrial-grade systems such as the Condor protocols developed by Miron Livny and his group at the University of Wisconsin and the Entropia Platform from Entropia, Inc., of San Diego, both of which are designed to capture the unused capacity of desktop workstations. Almost as well known is the web services architecture, which offers access to far-flung computational resources through enhancements to the Web's hypertext protocol, and which is being embraced by Microsoft, IBM, Sun Microsystems, and many others in the industry. And then there is grid computing, which grew out of scientists' attempts to harness extended networks of supercomputers. Thanks to the Globus protocol suite, which is the de facto standard in the field, construction is already under way on dozens of distributed grid computers around the world, with applications ranging from genetics to particle physics to earthquake engineering. Among the largest is the U.S. National Science Foundation's recently completed \$53 million TeraGrid: a general-purpose, distributed supercomputer that links clusters of high-end workstations at five sites around the United States, and which is capable of some 13.6 trillion floating point operations per second.

Although peer-to-peer computing, grid computing, and web services arose independently, they are so similar in spirit and purpose that they are now in the process of coalescing. More to the point, emerging standards such as the Open Grid Services Architecture offer some powerful mechanisms for autonomic computing. Among them are software tools for resource “discovery”—automatically finding out where a required database, program, or other resource can be found on the network; tools to implement fundamental security procedures, so that you can be sure that an outside program trying to interact with your machine is actually serving a legitimate purpose, and hasn't been sent

by some hacker; and tools for automatically allocating sub-tasks among the various resources.

A recent example of the latter came during the U.S. Open Tennis Tournament, when IBM fielded a cluster of servers to support the CNN web site. “At any given time, we would allocate however many servers we needed to run the various web site applications,” says IBM’s Ganek. “And then we had another application that would use any left-over servers to run protein-folding calculations. But we had the system constructed so that when the tennis event needed more processing power, those extra servers would automatically suspend their work, save their partial results to disk, clear out their memory, and shift to tennis. A skilled systems administrator couldn’t have done that for all those servers in less than hours; we did it in minutes.” In short: autonomic self-optimization. And likewise with autonomic self-healing, notes Ganek: “If one server fails in this scenario, you just allocate a new one and the end user still has an acceptable quality of service.”

Although peer-to-peer computing, grid computing, and web services arose independently, they are so similar in spirit and purpose that they are now in the process of coalescing.

That’s a pretty impressive demonstration already. But then, as Ganek is the first to admit, all this allocating and re-allocating is taking place within a cluster of identical servers controlled by a single owner, IBM. The kind of Internet Scale Operating System that Anderson and Kubiawicz envisioned in the March 2002 issue of *Scientific American* is far more ambitious, in that it would encompass every kind of host computer on the Internet, from the simplest PC (or handheld device) to the most powerful mainframe.

They see two broad classes of applications for such a system. One, following in the footsteps of SETI@home, is distributed processing—physical simulations, signal analysis, genetic analysis, computer graphics rendering, financial modeling, and the like. The Internet currently has more than 150 million host computers, they point out, which represent a collective computational resource vastly greater than any-

one could afford in-house. Moreover, that power will only continue to grow as the Internet grows (and as the individual hosts get upgraded). The second category of applications, following in the footsteps of Napster, Gnutella, and KaZaA, are distributed online services—file storage, databases, hosting of Web sites, streaming media, advanced Web search engines, and the like. Just as with KaZaA and company, the distributed data would be available from any location. But through advanced coding techniques, that data would be even more resistant to accident or attack. With modern “*M-of-N*” codes, for example, a file or database can be broken up into a large number of pieces, denoted *N*, each of which is gibberish by itself—but then can be reconstructed by retrieving any subset of *M* pieces, where *M* can be much smaller than *N*. The bottom line is that such a distributed architecture could make a database very secure indeed, since an attacker would have to hack, say, 10,000 systems.

The technology required to implement such an ISOS is currently under development in a variety of research efforts, including Kubiawicz’ own Oceanstore and Tapestry projects at Berkeley; Microsoft’s Pastry and Farsite projects; MIT’s Chord system; the Cosm project; and many others. But the challenges are large, as Anderson and Kubiawicz are the first to point out. How will the ISOS keep track of things, for example? Once it has chopped up your data and/or your computation into millions of fragments distributed all over the Internet, how will it find them again—and then make sure that what it’s found is, in fact, the most recent update? For that matter, how will the ISOS distribute the fragments in the first place? How will it automatically “self-optimize” its allocation in a universe of Internet hosts that can vary from low-end laptops to world-class supercomputers? How will it take into account the fact that some hosts are hidden away behind firewalls, and are extremely limited in what they can accept? That some hosts are only sporadically available (when, say, their user lets them go into screen saver mode)? Or that some are not available at all for certain types of jobs (with an owner who will allow, say, biomedical research, but not military applications)?

And perhaps most fundamental of all from a sociological standpoint, how will the ISOS give computer owners an incentive to let their

machines take part? Volunteerism is all very well for a sexy, high-profile project like SETI@home. But a long-term, workaday system may very well require some kind of market-based solution, in which users earn when they make their machines available to the ISOS and pay when they tap into it. The medium of exchange could be real dollars and cents, or some sort of virtual currency, as in the peer-to-peer system Mojo Nation. Either way, however, a market-based approach might also be a big help in solving the self-optimization problem. A number of projects, dating back to the 1980s, have shown that resources can be allocated very efficiently if the computer that needs help simply broadcasts a “request for proposals” over the network, and all the various host computers submit “bids” based on how much free capacity they have at the moment.¹⁰

In the strategy for autonomic computing that we’ve been describing so far, most of the self-configuring, self-optimizing, self-defending, and self-healing behavior rests on the simple fact that the data and the computations are distributed. But that doesn’t help much with the individual components of the system. Achieving robust autonomic behavior at that level (and every other) will also require a complementary strategy that might be paraphrased as “Know Thyself!”

A more formal term is “continuous control loop.” The basic idea is that each component of the system—hardware and software alike—not only would know how to do its assigned tasks, but would have internal mechanisms that constantly monitor its own operation and make corrections as needed. More specifically, these mechanisms would include

- Some form of built-in sensor network (with the sensors made of hardware or software, as needed);
- Some form of built-in “knowledge” about how to integrate the sensor data, and how to interpret the results in light of what the device is supposed to be doing;
- Some form of built-in knowledge about how to recognize when things are going wrong, and to devise a repair plan; and
- Some form of internal “effectors” that will actually allow the device to make the fix.

Additional mechanisms might also allow the device to carry out that form of self-optimization known as learning. Over time, for exam-

ple, the device might be able to rearrange its sensors to keep a closer watch on sub-activities that are more prone to failure. But in any case, a critical feature of this scheme is that each device would handle as much as possible locally—and yet still have the means to call on the larger system when it needed help. Indeed, the scheme could easily be made recursive, so that when the call for help came, the larger system would already be watching for trouble using its own internal routines. It would take corrective action if it could, and call for help to a still larger system (or to the human operators) if it had to. And so it would go: a deeply recursive, autonomic system that ranged (in principle) from the individual hardware devices at the lowest level all the way up to the global Internet.

One approach that makes heavy use of this continuous control-loop strategy is the Recovery-Oriented Computing (ROC) initiative spearheaded by David Patterson at Berkeley and Armando Fox at Stanford. It should be stressed at the outset that recovery-oriented computing is not just a set of techniques, but an overarching philosophy of computer system design. The starting point is what Patterson and Fox like to call “Peres’ Law,” after a quotation from former Israeli Prime Minister Shimon Peres:

If a problem has no solution, it may not be a problem, but a fact—not to be solved, but to be coped with over time.

Their fundamental premise is that in computing, software bugs, equipment failure, and operator error are prime examples of Peres’ Law. Fifty years of effort to eliminate them, to engineer them out of existence, has failed miserably. Despite the software engineers’ best efforts, for example, the most carefully designed and thoroughly tested software on the market still averages more than one bug per

Despite the software engineers’ best efforts, for example, the most carefully designed and thoroughly tested software on the market still averages more than one bug per 1,000 lines of codes—in applications that can often run to tens of millions of lines.

1,000 lines of codes—in applications that can often run to tens of millions of lines. Likewise, well-manufactured hardware still has greater than a 1 percent failure rate. And well-motivated, well-trained operators still make mistakes in something like 10 percent of their actions. (Statistically, in fact, operator errors account for slightly more than half of all web site failures.) So the ROC mantra is to quit trying to solve the unsolvable. Accept the fact that things *are* going to go wrong, no matter what you do, and instead focus your efforts on designing systems that recover very rapidly.

In the June 2003 issue of *Scientific American*,¹¹ Fox and Patterson described a number of experiments now being carried out by the Berkeley/Stanford team to demonstrate how ROC could work in practice. Two examples that are particularly relevant for autonomic computing:

Making a quick comeback. When the going gets rough in the computer world—when the system crashes and burns, say, or when the screen freezes solid, or when the software just starts acting funky—then it doesn't matter if you're talking about a desktop PC or the server farm running a world-renown web site. The remedy is usually the same: reboot. Don't even try to find the error—just wipe the slate clean, and start over. The problem, of course, is that the slate-wiping process has a way of trashing valuable data, not to mention wasting the user's time. The ROC answer, which has already been demonstrated at Stanford, is "Micro-Rebooting." First, rewrite the application and the operating system software so that each module can be stopped and restarted independently, without sending the others into a tizzy. (This isn't easy, given how strongly software modules tend to interact with and depend on one another, but it seems to be doable. With an eye toward legacy systems, moreover, experiments are now under way to see if the rewriting can be done automatically.) Second, equip the operating system with routines that can recognize when one or more modules have gone awry, and automatically restart them. And finally, give the monitoring routines a database of historically useful fallback solutions, so that if the first restart doesn't work, it can try again with a larger group of modules. If all goes well, the system may never need to

reboot the entire collection of modules—and except for a very brief pause, the user may never even notice that anything has gone wrong.

Pinpointing the problem. Of course, it's one thing to apply a set of tried-and-true solutions for failure modes that you already know about. But how are the monitoring routines supposed to deal with the failure modes you *don't* know about—the ones that arise from, say, a newly installed application? The standard (and profoundly unhelpful) approach to such glitches is to flash the user with an error message. The ROC group's alternative is a program called "PinPoint," which has been successfully demonstrated in the Web site context. Whenever a user visits a PinPoint-enabled site, the program keeps track of which software modules participate in delivering services to that user. And over time, it analyses the growing list successful service requests, failures, and error messages using standard data-mining techniques. The result—an automatically generated, running tally of likely failure modes, as well as which modules seem to be causing the most trouble—can then be used to update the Micro-Reboot routines, or to inform the human operators. But either way, the beauty of it is that PinPoint automatically works with any set of software components, whether old, new, or upgraded.

Policy Challenges

Although the rise of autonomic computing (by whatever name) does pose some significant issues for policymakers, most of them—the need to encourage open standards and interoperability, for example, and making sure that the technology builds in privacy and security measures from the outset—are quite similar to those that arise in other contexts, such as pervasive computing. So we won't repeat that discussion here. However, there is one glaring issue that does seem unique to autonomic computing. Call it the question of *trust* and *responsibility*. How much authority should we delegate to these autonomic systems? How do we know that they will actually behave in the way we want them to? And who, in the last analysis, will be in charge?

At first blush, of course, the obvious answer to that last question—the answer that IBM and other vendors go out of their way to emphasize—is that humans will always be in charge. Indeed, that’s often touted as one of the greatest potential strengths of autonomic design. Ultimately (goes the argument), human managers will simply have to set the overall “policies” for the system—say, that

Gold-Class customers take priority and are guaranteed such and such a response time for Web services, while Silver-Class customers may have to wait and accept a somewhat slower response time. Then the system will automatically figure out how to enforce those policies. However (continues the argument), no matter how much of this detailed implementation the managers may delegate, the system will *never* set goals on its own.

Fair enough: such a policy-driven autonomic system would be undeniably convenient and, with any luck, might produce a lot fewer bugs than human programmers would. However, it’s worth remembering that systems designers have been down this road before, most notably during the Expert Systems boom of the 1970s and 1980s. The idea was to bring in human experts in, say, renal diagnosis, and have them go through exactly what they would do in each conceivable contingency—that is, to state a “policy” for that contingency. The programmers would then encode those bits of knowledge as IF-THEN rules. In practice, however, the experts always found it remarkably difficult to articulate their knowledge in such explicit terms. And even when they did, the resulting expert system was only as good as its rule base; as soon as it hit a situation that wasn’t covered by the rules, it was helpless.

We can expect much the same kind of issues to crop up in policy-driven autonomic systems. But it hardly ends there: thanks to modern networking, we can also expect a whole new level of “sociological” issues. What happens when independently created autonomic systems

...there is one glaring issue that does seem unique to autonomic computing. Call it the question of trust and responsibility. How much authority should we delegate to these autonomic systems?

from different departments, or different institutions, start to interact on the net—and discover that their policies are in conflict? Or what happens if my system autonomically reroutes traffic to resolve a congestion problem, and ends up creating a congestion problem on *your* network? Who adjudicates in such situations—the human operators? Maybe. Or maybe not, if the larger system was equipped with a layer of autonomic conflict-resolution software. But then, who is responsible for creating and maintaining that higher-level software, and setting its policies?

Whoever gets the job, it won’t be a responsibility to take lightly. As eastern North America was forcibly reminded in August 2003, large-scale technological systems like the electric power grid are not just enormously complex, but can have rare, catastrophic failure modes that are difficult or impossible to anticipate. Large-scale autonomic systems aren’t likely to be an exception to that rule—especially not when they are made up of individual subsystems following policies that may or may not be in agreement. If nothing else, we can ask that the deployment of such systems be accompanied by some very sophisticated modeling, simulation, and testing to determine what those failure modes might be. But then, who will be responsible for that effort? The vendors? The purchasers? The Federal Government?¹²

And finally, it’s probably worth spending a minute to contemplate a heretical thought:

maybe we *shouldn’t* automate everything in sight—especially not in places like aircraft cockpits, nuclear power plant control rooms, and even web sites, where we still need to rely on human operators as the ultimate authority. Again, designers have been down this

road before. In the 1970s and early 1980s, for example, during the first big wave of cockpit automation, the idea was that the computers would fly the plane, and the pilot would just monitor them in case anything

...we can also expect a whole new level of “sociological” issues. What happens when independently created autonomic systems from different departments, or different institutions, start to interact on the net—and discover that their policies are in conflict?

went wrong. The only problem with that scenario was that human beings are absolutely terrible at passive monitoring. Both common sense and rigorous psychological testing have shown that, no matter how motivated and well-trained they are, people get bored. Their attention flags. They start missing things. Worse, a passive pilot would often have to tackle an emergency cold, wasting precious seconds trying to figure out precisely where the aircraft was, and what the automatic systems had been doing.

In short, the early cockpit automators had run right into what Fox, Patterson, and the Recovery-Oriented Computing group like to call “The Automation Paradox”—the fact that (ill-considered, over-) automation can actually make the situation worse. (They also emphasize that automation doesn’t eliminate human error; it merely replaces operator errors with design errors.) In the case of cockpit automation, recognition of this paradox eventually led the aviation community through a 180-degree turn in their concept of what the computers should do. By the mid-1980s, they had moved to a philosophy of “human-centered” automation, in which the computers would leave the crew alone to fly the planes as they saw fit, thus allowing them to keep their flying skills sharp and to keep their attention focused on what was happening. Only if the electronics sensed that the aircraft were approaching a danger zone—if it were nearing a stall, for example—would the system issue a warning, or take over.

Instead of having people watch the machines, in other words, human-centered automation meant telling the machines to watch the people. It meant putting the pilots firmly back in command of the aircraft. And it meant putting automation back on the right track: as an assistant to the pilots.

Which brings us back to the heretical thought: perhaps what we should be contemplating here is a new age of human-centered autonomic computing.

Endnotes

¹ Brooks, Frederick P. 1975. *The Mythical Man-Month*. Reading, MA: Addison Wesley.

² Following a trend that Intel co-founder Gordon Moore first described in 1965, engineers have spent the past forty years cramming more and more processing power into smaller and smaller packages. Specifically, the power that’s available for any given price has been doubling roughly every 18 months, and should continue to do so for another decade. By the late 1970s, Moore’s Law had taken us from the era of *many-to-one* computing—many users submitting jobs to a single mainframe—to the era of *one-to-one* computing: one user, one personal computer. And by now, in the new millennium, it has long since brought us into the era of *one-to-many* computing, in which each person is served by a host of processors.

³ A group at the University of Michigan offers a useful online tutorial.

⁴ <http://www.ubiq.com/hypertext/weiser/UbiHome.html>; Weiser, Mark. “The Computer for the Twenty-First Century.” *Scientific American* (1991): 94–100.

⁵ Weiser, Mark. “Creating the Invisible Interface.” *Symposium on User Interface Software and Technology*, New York, NY: ACM Press, 1994.

⁶ Among the most notable are IBM’s Pervasive Computing Lab in Austin, Texas; Hewlett Packard’s Cooltown in Palo Alto; MIT’s Project Oxygen; the California Institute for Telecommunications and Information Technology—Cal-(IT)²—in San Diego; the Georgia Tech Aware Home; and Royal Philips Electronics’ Homelab in Eindhoven, The Netherlands.

⁷ The actual shape of our favorite gadget will no doubt be a matter of personal preference; one model may look a lot like a current-generation tablet PC, another like a PDA, and so on. But with one or two of the things, at most, we’ll have all the functionality we want.

⁸ Napster depended on a centralized database of users, which is how the authorities brought it down—by suing the central company that ran the database. Gnutella and KaZaA have no such Achilles’ Heel.

⁹ Anderson, David P., and John Kubiawicz. March 2002. “The Worldwide Computer,” *Scientific American*, 286(3):40–7.

¹⁰ C.A. Waldspurger, et al. 1992. “Spawn: A Distributed Computational Economy,” *IEEE Transactions on Software Engineering*, 18(2):103–17. Available online at <http://www.computer.org/tse/ts1992/e0103abs.htm>.

¹¹ Fox, Armando, and David Patterson. 2003. “Self-Repairing Computers,” *Scientific American*, 288(6):54–61.

¹² That’s actually not such a crazy possibility, considering that the Department of Homeland Security now operates a major critical infrastructure simulation facility at Sandia National Laboratory.

About the Author

Dr. M. Mitchell Waldrop

Dr. M. Mitchell Waldrop is the author of *Man-Made Minds* (1987), a book about artificial intelligence; *Complexity* (1992), a book about the Santa Fe Institute and the new sciences of complexity; and *The Dream Machine* (2001), a book about the history of computing. He earned his Ph.D. in elementary particle physics at the University of Wisconsin, and from 1980 to 1991 was a senior writer at *Science* magazine. He is currently with the National Science Foundation.

Appendix:

The Future of Computing Events and Speakers

Pervasive Computing

1. *Rod Adkins, General Manager, Pervasive Computing Division, IBM*
2. *Victor Zue, Director, MIT Laboratory for Computer Science*
3. *David Brin, Author of “The Transparent Society”*

Autonomic Computing

1. *Alan Ganek, Vice President, Autonomic Computing, IBM*
2. *Gail Kaiser, Director, Programming Systems Laboratory, Columbia University*
3. *Peter M. Hughes, Assistant Chief for Technology, Information Systems Division, NASA/Goddard*

Grid Computing

1. *Ian Foster, Argon National Laboratory*
2. *Andrew Grimshaw, University of Virginia*

