# Chapter 1

# The Evolution of Cloud Computing

## 1.1 Chapter Overview

It is important to understand the evolution of computing in order to get an appreciation of how we got into the cloud environment. Looking at the evolution of the computing hardware itself, from the first generation to the current (fourth) generation of computers, shows how we got from there to here. The hardware, however, was only part of the evolutionary process. As hardware evolved, so did software. As networking evolved, so did the rules for how computers communicate. The development of such rules, or protocols, also helped drive the evolution of Internet software.

Establishing a common protocol for the Internet led directly to rapid growth in the number of users online. This has driven technologists to make even more changes in current protocols and to create new ones. Today, we talk about the use of IPv6 (Internet Protocol version 6) to mitigate addressing concerns and for improving the methods we use to communicate over the Internet. Over time, our ability to build a common interface to the Internet has evolved with the improvements in hardware and software. Using web browsers has led to a steady migration away from the traditional data center model to a cloud-based model. Using technologies such as server virtualization, parallel processing, vector processing, symmetric multiprocessing, and massively parallel processing has fueled radical change. Let's take a look at how this happened, so we can begin to understand more about the cloud.

In order to discuss some of the issues of the cloud concept, it is important to place the development of computational technology in a historical context. Looking at the Internet cloud's evolutionary development,[1] and the problems encountered along the way, provides some key reference points to help us understand the challenges that had to be overcome to develop the Internet and the World Wide Web (WWW) today. These challenges fell

into two primary areas, hardware and software. We will look first at the hardware side.

## 1.2 Hardware Evolution

Our lives today would be different, and probably difficult, without the benefits of modern computers. Computerization has permeated nearly every facet of our personal and professional lives. Computer evolution has been both rapid and fascinating. The first step along the evolutionary path of computers occurred in 1930, when binary arithmetic was developed and became the foundation of computer processing technology, terminology, and programming languages. Calculating devices date back to at least as early as 1642, when a device that could mechanically add numbers was invented. Adding devices evolved from the abacus. It was a significant milestone in the history of computers. In 1939, the Berry brothers invented an electronic computer capable of operating digitally. Computations were performed using vacuum-tube technology.

In 1941, the introduction of Konrad Zuse's Z3 at the German Laboratory for Aviation in Berlin was one of the most significant events in the evolution of computers because this machine supported both floating-point and binary arithmetic. Because it was a "Turing-complete" device,[2] it is considered to be the very first computer that was fully operational. A programming language is considered Turing-complete if it falls into the same computational class as a Turing machine, meaning that it can perform any calculation a universal Turing machine can perform. This is especially significant because, under the Church-Turing thesis,[3] a Turing machine is the embodiment of the intuitive notion of an algorithm. Over the course of the next two years, computer prototypes were built to decode secret German messages by the U.S. Army.

1. Paul Wallis, "A Brief History of Cloud Computing: Is the Cloud There Yet? A Look at the Cloud's Forerunners and the Problems They Encountered," http://soa.sys-con.com/node/581838, 22 Aug 2008, retrieved 7 Jan 2009.
2. According to the online encyclopedia Wikipedia, "A computational system that can compute every Turing-computable function is called Turing-complete (or Turing-powerful). Alternatively, such a system is one that can simulate a universal Turing machine." http://en.wikipedia.org/wiki/Turing_complete, retrieved 17 Mar 2009.
3. http://esolangs.org/wiki/Church-Turing_thesis, retrieved 10 Jan 2009.

### 1.2.1 First-Generation Computers

The first generation of modern computers can be traced to 1943, when the Mark I and Colossus computers (see Figures 1.1 and 1.2) were developed,[4] albeit for quite different purposes. With financial backing from IBM (then International Business Machines Corporation), the Mark I was designed and developed at Harvard University. It was a general-purpose electromechanical programmable computer. Colossus, on the other hand, was an electronic computer built in Britain at the end 1943. Colossus was the world's first programmable, digital, electronic, computing device. First-generation computers were built using hard-wired circuits and vacuum tubes (thermionic valves). Data was stored using paper punch cards. Colossus was used in secret during World War II to help decipher teleprinter messages encrypted by German forces using the Lorenz SZ40/42 machine. British code breakers referred to encrypted German teleprinter traffic as "Fish" and called the SZ40/42 machine and its traffic "Tunny."[5]

To accomplish its deciphering task, Colossus compared two data streams read at high speed from a paper tape. Colossus evaluated one data stream representing the encrypted "Tunny," counting each match that was discovered based on a programmable Boolean function. A comparison with the other data stream was then made. The second data stream was generated internally and designed to be an electronic simulation of the

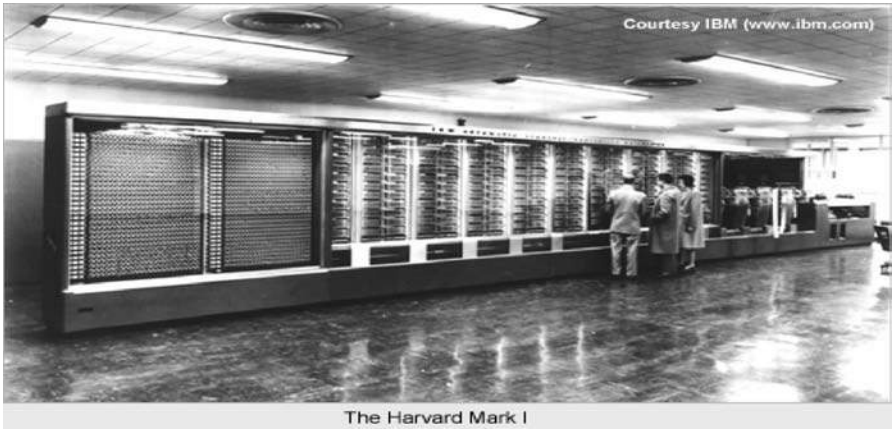


Figure 1.1 The Harvard Mark I computer. (Image from www.columbia.edu/acis/history/mark1.html, retrieved 9 Jan 2009.)

4. http://trillian.randomstuff.org.uk/~stephen/history, retrieved 5 Jan 2009.
5. http://en.wikipedia.org/wiki/Colossus_computer, retrieved 7 Jan 2009.

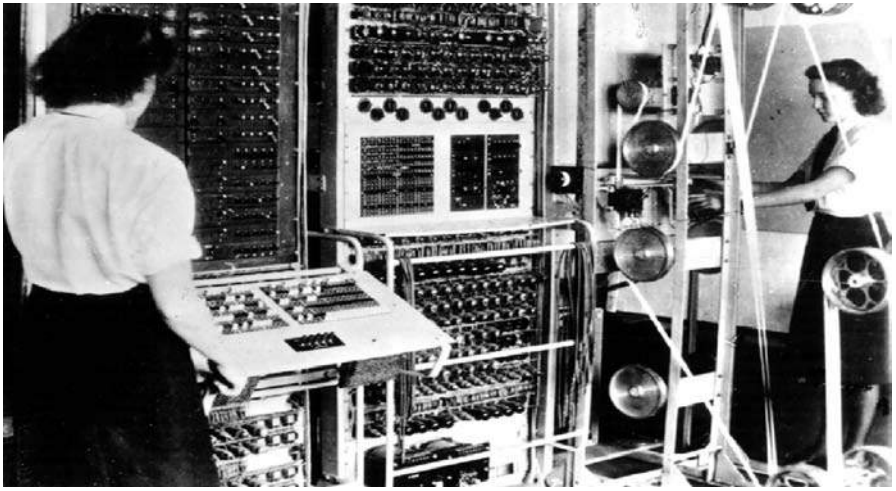

Figure 1.2 The British-developed Colossus computer. (Image from www.computerhistory.org, retrieved 9 Jan 2009.)

Lorenz SZ40/42 as it ranged through various trial settings. If the match count for a setting was above a predetermined threshold, that data match would be sent as character output to an electric typewriter.

### 1.2.2 Second-Generation Computers

Another general-purpose computer of this era was ENIAC (Electronic Numerical Integrator and Computer, shown in Figure 1.3), which was built in 1946. This was the first Turing-complete, digital computer capable of being reprogrammed to solve a full range of computing problems,[6] although earlier machines had been built with some of these properties. ENIAC's original purpose was to calculate artillery firing tables for the U.S. Army's Ballistic Research Laboratory. ENIAC contained 18,000 thermionic valves, weighed over 60,000 pounds, and consumed 25 kilowatts of electrical power per hour. ENIAC was capable of performing 100,000 calculations a second. Within a year after its completion, however, the invention of the transistor meant that the inefficient thermionic valves could be replaced with smaller, more reliable components, thus marking another major step in the history of computing.

6. Joel Shurkin, Engines of the Mind: The Evolution of the Computer from Mainframes to Microprocessors, New York: W. W. Norton, 1996.

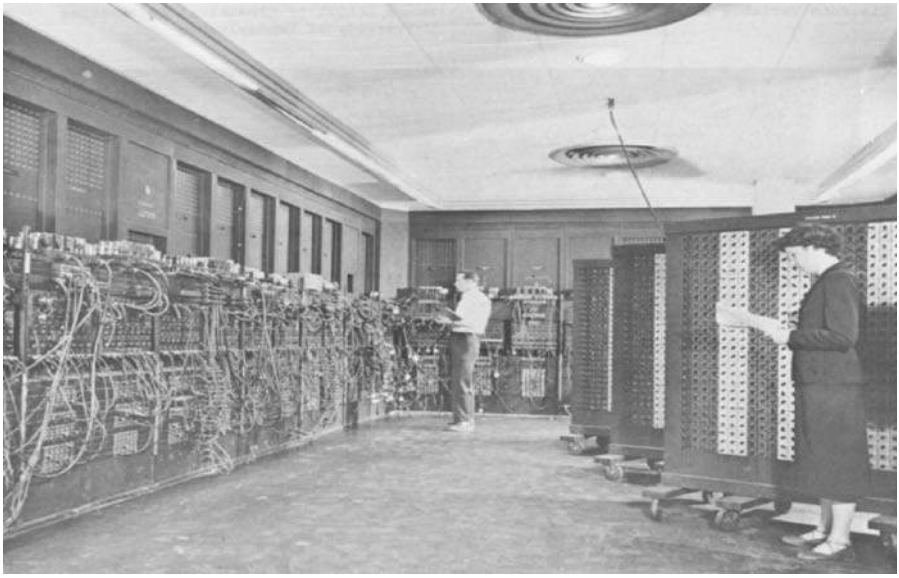

Figure 1.3 The ENIAC computer. (Image from www.mrsec.wisc.edu/.../computer/eniac.html, retrieved 9 Jan 2009.)

Transistorized computers marked the advent of second-generation computers, which dominated in the late 1950s and early 1960s. Despite using transistors and printed circuits, these computers were still bulky and expensive. They were therefore used mainly by universities and government agencies.

The integrated circuit or microchip was developed by Jack St. Claire Kilby, an achievement for which he received the Nobel Prize in Physics in 2000.[7] In congratulating him, U.S. President Bill Clinton wrote, "You can take pride in the knowledge that your work will help to improve lives for generations to come." It was a relatively simple device that Mr. Kilby showed to a handful of co-workers gathered in the semiconductor lab at Texas Instruments more than half a century ago. It was just a transistor and a few other components on a slice of germanium. Little did this group realize that Kilby's invention was about to revolutionize the electronics industry.

### 1.2.3 Third-Generation Computers

Kilby's invention started an explosion in third-generation computers. Even though the first integrated circuit was produced in September 1958,

7. http://www.ti.com/corp/docs/kilbyctr/jackstclair.shtml, retrieved 7 Jan 2009.


Figure 1.4 The Intel 4004 processor. (Image from www.thg.ru/cpu/20051118/index.html, retrieved 9 Jan 2009.)

microchips were not used in computers until 1963. While mainframe computers like the IBM 360 increased storage and processing capabilities even further, the integrated circuit allowed the development of minicomputers that began to bring computing into many smaller businesses. Large-scale integration of circuits led to the development of very small processing units, the next step along the evolutionary trail of computing. In November 1971, Intel released the world's first commercial microprocessor, the Intel 4004 (Figure 1.4). The 4004 was the first complete CPU on one chip and became the first commercially available microprocessor. It was possible because of the development of new silicon gate technology that enabled engineers to integrate a much greater number of transistors on a chip that would perform at a much faster speed. This development enabled the rise of the fourth-generation computer platforms.

### 1.2.4 Fourth-Generation Computers

The fourth-generation computers that were being developed at this time utilized a microprocessor that put the computer's processing capabilities on a single integrated circuit chip. By combining random access memory (RAM), developed by Intel, fourth-generation computers were faster than ever before and had much smaller footprints. The 4004 processor was capable of "only" 60,000 instructions per second. As technology progressed, however, new processors brought even more speed and computing capability to users. The microprocessors that evolved from the 4004 allowed manufacturers to begin developing personal computers small enough and cheap enough to be purchased by the general public. The first commercially available personal computer was the MITS Altair 8800, released at the end of 1974. What followed was a flurry of other personal computers to market, such as the Apple I and II, the Commodore PET, the

VIC-20, the Commodore 64, and eventually the original IBM PC in 1981. The PC era had begun in earnest by the mid-1980s. During this time, the IBM PC and IBM PC compatibles, the Commodore Amiga, and the Atari ST computers were the most prevalent PC platforms available to the public. Computer manufacturers produced various models of IBM PC compatibles. Even though microprocessing power, memory and data storage capacities have increased by many orders of magnitude since the invention of the 4004 processor, the technology for large-scale integration (LSI) or very-large-scale integration (VLSI) microchips has not changed all that much. For this reason, most of today's computers still fall into the category of fourth-generation computers.

## 1.3 Internet Software Evolution

The Internet is named after the Internet Protocol, the standard communications protocol used by every computer on the Internet. The conceptual foundation for creation of the Internet was significantly developed by three individuals. The first, Vannevar Bush,[8] wrote a visionary description of the potential uses for information technology with his description of an automated library system named MEMEX (see Figure 1.5). Bush introduced the concept of the MEMEX in the 1930s as a microfilm-based "device in which an individual stores all his books, records, and communications, and which is mechanized so that it may be consulted with exceeding speed and flexibility."[9]

Figure 1.5 Vannevar Bush's MEMEX. (Image from www.icesi.edu.co/blogs_estudiantes/luisaulestia, retrieved 9 Jan 2009.)

8. http://en.wikipedia.org/wiki/Vannevar_Bush, retrieved 7 Jan 2009.
9. http://www.livinginternet.com/i/ii_summary.htm, retrieved 7 Jan 2009.

After thinking about the potential of augmented memory for several years, Bush wrote an essay entitled "As We May Think" in 1936. It was finally published in July 1945 in the *Atlantic Monthly*. In the article, Bush predicted: "Wholly new forms of encyclopedias will appear, ready made with a mesh of associative trails running through them, ready to be dropped into the MEMEX and there amplified."[10] In September 1945, *Life* magazine published a condensed version of "As We May Think" that was accompanied by several graphic illustrations showing what a MEMEX machine might look like, along with its companion devices.

The second individual to have a profound effect in shaping the Internet was Norbert Wiener. Wiener was an early pioneer in the study of stochastic and noise processes. His work in stochastic and noise processes was relevant to electronic engineering, communication, and control systems.[11] He also founded the field of cybernetics. This field of study formalized notions of feedback and influenced research in many other fields, such as engineering, systems control, computer science, biology, philosophy, etc. His work in cybernetics inspired future researchers to focus on extending human capabilities with technology. Influenced by Wiener, Marshall McLuhan put forth the idea of a *global village* that was interconnected by an electronic nervous system as part of our popular culture.

In 1957, the Soviet Union launched the first satellite, *Sputnik I,* prompting U.S. President Dwight Eisenhower to create the Advanced Research Projects Agency (ARPA) agency to regain the technological lead in the arms race. ARPA (renamed DARPA, the Defense Advanced Research Projects Agency, in 1972) appointed J. C. R. Licklider to head the new Information Processing Techniques Office (IPTO). Licklider was given a mandate to further the research of the SAGE system. The SAGE system (see Figure 1.6) was a continental air-defense network commissioned by the U.S. military and designed to help protect the United States against a space-based nuclear attack. SAGE stood for Semi-Automatic Ground Environment.[12] SAGE was the most ambitious computer project ever undertaken at the time, and it required over 800 programmers and the technical resources of some of America's largest corporations. SAGE was started in the 1950s and became operational by 1963. It remained in continuous operation for over 20 years, until 1983.

10. http://www.theatlantic.com/doc/194507/bush, retrieved 7 Jan 2009.
11. http://en.wikipedia.org/wiki/Norbert_Wiener, retrieved 7 Jan 2009.
12. http://www.computermuseum.li/Testpage/IBM-SAGE-computer.htm, retrieved 7 Jan 2009.

Figure 1.6 The SAGE system. (Image from USAF Archives, retrieved from http://history.sandiego.edu/GEN/recording/images5/PDRM0380.jpg.)

While working at ITPO, Licklider evangelized the potential benefits of a country-wide communications network. His chief contribution to the development of the Internet was his ideas, not specific inventions. He foresaw the need for networked computers with easy user interfaces. His ideas foretold of graphical computing, point-and-click interfaces, digital libraries, e-commerce, online banking, and software that would exist on a network and migrate to wherever it was needed. Licklider worked for several years at ARPA, where he set the stage for the creation of the ARPANET. He also worked at Bolt Beranek and Newman (BBN), the company that supplied the first computers connected on the ARPANET.

After he had left ARPA, Licklider succeeded in convincing his replacement to hire a man named Lawrence Roberts, believing that Roberts was just the person to implement Licklider's vision of the future network computing environment. Roberts led the development of the network. His efforts were based on a novel idea of "packet switching" that had been developed by Paul Baran while working at RAND Corporation. The idea for a common interface to the ARPANET was first suggested in Ann Arbor, Michigan, by Wesley Clark at an ARPANET design session set up by Lawrence Roberts in April 1967. Roberts's implementation plan called for each site that was to connect to the ARPANET to write the software necessary to connect its computer to the network. To the attendees,

this approach seemed like a lot of work. There were so many different kinds of computers and operating systems in use throughout the DARPA community that every piece of code would have to be individually written, tested, implemented, and maintained. Clark told Roberts that he thought the design was "bass-ackwards."[13]

After the meeting, Roberts stayed behind and listened as Clark elaborated on his concept to deploy a minicomputer called an Interface Message Processor (IMP, see Figure 1.7) at each site. The IMP would handle the interface to the ARPANET network. The physical layer, the data link layer, and the network layer protocols used internally on the ARPANET were implemented on this IMP. Using this approach, each site would only have to write one interface to the commonly deployed IMP. The host at each site connected itself to the IMP using another type of interface that had different physical, data link, and network layer specifications. These were specified by the Host/IMP Protocol in BBN Report 1822.[14]

So, as it turned out, the first networking protocol that was used on the ARPANET was the Network Control Program (NCP). The NCP provided the middle layers of a protocol stack running on an ARPANET-connected host computer.[15] The NCP managed the connections and flow control among the various processes running on different ARPANET host computers. An application layer, built on top of the NCP, provided services such as email and file transfer. These applications used the NCP to handle connections to other host computers.

A minicomputer was created specifically to realize the design of the Interface Message Processor. This approach provided a system-independent interface to the ARPANET that could be used by any computer system. Because of this approach, the Internet architecture was an open architecture from the very beginning. The Interface Message Processor interface for the ARPANET went live in early October 1969. The implementation of the architecture is depicted in Figure 1.8.

13. http://www.urbandictionary.com/define.php?term=Bass+Ackwards defined this as "The art and science of hurtling blindly in the wrong direction with no sense of the impending doom about to be inflicted on one's sorry ass. Usually applied to procedures, processes, or theories based on faulty logic, or faulty personnel." Retrieved 8 Jan 2009.
14. Frank Heart, Robert Kahn, Severo Ornstein, William Crowther, and David Walden, "The Interface Message Processor for the ARPA Computer Network," Proc. 1970 Spring Joint Computer Conference 36:551–567, AFIPS, 1970.
15. http://www.answers.com/topic/network-control-program, retrieved 8 Jan 2009.

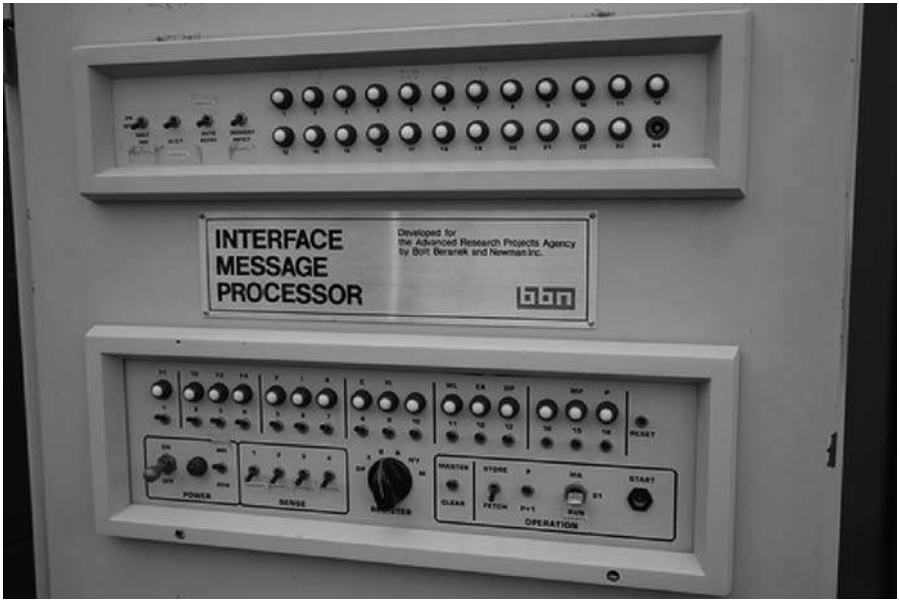


Figure 1.7 An Interface Message Processor. (Image from luni.net/wp-content/uploads/2007/02/bbn-imp.jpg, retrieved 9 Jan 2009.)

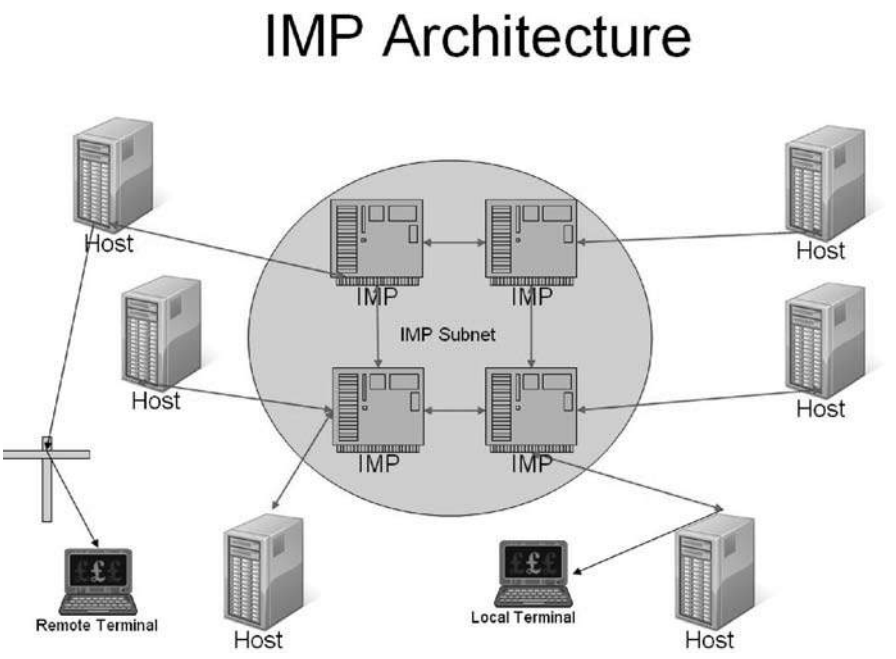


Figure 1.8 Overview of the IMP architecture.

### 1.3.1 Establishing a Common Protocol for the Internet

Since the lower-level protocol layers were provided by the IMP host interface, the NCP essentially provided a transport layer consisting of the ARPANET Host-to-Host Protocol (AHHP) and the Initial Connection Protocol (ICP). The AHHP specified how to transmit a unidirectional, flow-controlled data stream between two hosts. The ICP specified how to establish a bidirectional pair of data streams between a pair of connected host processes. Application protocols such as File Transfer Protocol (FTP), used for file transfers, and Simple Mail Transfer Protocol (SMTP), used for sending email, accessed network services through an interface to the top layer of the NCP. On January 1, 1983, known as Flag Day, NCP was rendered obsolete when the ARPANET changed its core networking protocols from NCP to the more flexible and powerful TCP/IP protocol suite, marking the start of the Internet as we know it today.

It was actually Robert Kahn and Vinton Cerf who built on what was learned with NCP to develop the TCP/IP networking protocol we use today. TCP/IP quickly became the most widely used network protocol in the world. The Internet's open nature and use of the more efficient TCP/IP protocol became the cornerstone of an internetworking design that has become the most widely used network protocol in the world. The history of TCP/IP reflects an interdependent design. Development of this protocol was conducted by many people. Over time, there evolved four increasingly better versions of TCP/IP (TCP v1, TCP v2, a split into TCP v3 and IP v3, and TCP v4 and IPv4). Today, IPv4 is the standard protocol, but it is in the process of being replaced by IPv6, which is described later in this chapter.

The TCP/IP protocol was deployed to the ARPANET, but not all sites were all that willing to convert to the new protocol. To force the matter to a head, the TCP/IP team turned off the NCP network channel numbers on the ARPANET IMPs twice. The first time they turned it off for a full day in mid-1982, so that only sites using TCP/IP could still operate. The second time, later that fall, they disabled NCP again for two days. The full switchover to TCP/IP happened on January 1, 1983, without much hassle. Even after that, however, there were still a few ARPANET sites that were down for as long as three months while their systems were retrofitted to use the new protocol. In 1984, the U.S. Department of Defense made TCP/IP the standard for all military computer networking, which gave it a high profile and stable funding. By 1990, the ARPANET was retired and transferred to the NSFNET. The NSFNET was soon connected to the CSNET, which

linked universities around North America, and then to the EUnet, which connected research facilities in Europe. Thanks in part to the National Science Foundation's enlightened management, and fueled by the growing popularity of the web, the use of the Internet exploded after 1990, prompting the U.S. government to transfer management to independent organizations starting in 1995.

### 1.3.2 Evolution of Ipv6

The amazing growth of the Internet throughout the 1990s caused a vast reduction in the number of free IP addresses available under IPv4. IPv4 was never designed to scale to global levels. To increase available address space, it had to process data packets that were larger (i.e., that contained more bits of data). This resulted in a longer IP address and that caused problems for existing hardware and software. Solving those problems required the design, development, and implementation of a new architecture and new hardware to support it. It also required changes to all of the TCP/IP routing software. After examining a number of proposals, the Internet Engineering Task Force (IETF) settled on IPv6, which was released in January 1995 as RFC 1752. Ipv6 is sometimes called the Next Generation Internet Protocol (IPNG) or TCP/IP v6. Following release of the RFP, a number of organizations began working toward making the new protocol the *de facto* standard. Fast-forward nearly a decade later, and by 2004, IPv6 was widely available from industry as an integrated TCP/IP protocol and was supported by most new Internet networking equipment.

### 1.3.3 Finding a Common Method to Communicate Using the Internet Protocol

In the 1960s, twenty years after Vannevar Bush proposed MEMEX, the word *hypertext* was coined by Ted Nelson. Ted Nelson was one of the major visionaries of the coming hypertext revolution. He knew that the technology of his time could never handle the explosive growth of information that was proliferating across the planet. Nelson popularized the hypertext concept, but it was Douglas Engelbart who developed the first working hypertext systems. At the end of World War II, Douglas Engelbart was a 20-year-old U.S. Navy radar technician in the Philippines. One day, in a Red Cross library, he picked up a copy of the *Atlantic Monthly* dated July 1945. He happened to come across Vannevar Bush's article about the MEMEX automated library system and was strongly influenced by this vision of the future of information technology. Sixteen years later, Engelbart published his own

version of Bush's vision in a paper prepared for the Air Force Office of Scientific Research and Development. In Englebart's paper, "Augmenting Human Intellect: A Conceptual Framework," he described an advanced electronic information system:

> Most of the structuring forms I'll show you stem from the simple capability of being able to establish arbitrary linkages between different substructures, and of directing the computer subsequently to display a set of linked substructures with any relative positioning we might designate among the different substructures. You can designate as many different kinds of links as you wish, so that you can specify different display or manipulative treatment for the different types.[16]

Engelbart joined Stanford Research Institute in 1962. His first project was *Augment,* and its purpose was to develop computer tools to augment human capabilities. Part of this effort required that he developed the mouse, the graphical user interface (GUI), and the first working hypertext system, named NLS (derived from o**N**-**L**ine **S**ystem). NLS was designed to cross-reference research papers for sharing among geographically distributed researchers. NLS provided groupware capabilities, screen sharing among remote users, and reference links for moving between sentences within a research paper and from one research paper to another. Engelbart's NLS system was chosen as the second node on the ARPANET, giving him a role in the invention of the Internet as well as the World Wide Web.

In the 1980s, a precursor to the web as we know it today was developed in Europe by Tim Berners-Lee and Robert Cailliau. Its popularity skyrocketed, in large part because Apple Computer delivered its HyperCard product free with every Macintosh bought at that time. In 1987, the effects of hypertext rippled through the industrial community. HyperCard was the first hypertext editing system available to the general public, and it caught on very quickly. In the 1990s, Marc Andreessen and a team at the National Center for Supercomputer Applications (NCSA), a research institute at the University of Illinois, developed the Mosaic and Netscape browsers. A technology revolution few saw coming was in its infancy at this point in time.

16. Douglas Engelbart, "Augmenting Human Intellect: A Conceptual Framework," in a report for the Air Force Office of Scientific Research and Development, October 1962.

### 1.3.4 Building a Common Interface to the Internet

While Marc Andreessen and the NCSA team were working on their browsers, Robert Cailliau at CERN independently proposed a project to develop a hypertext system. He joined forces with Berners-Lee to get the web initiative into high gear. Cailliau rewrote his original proposal and lobbied CERN management for funding for programmers. He and Berners-Lee worked on papers and presentations in collaboration, and Cailliau helped run the very first WWW conference.

In the fall of 1990, Berners-Lee developed the first web browser (Figure 1.9) featuring an integrated editor that could create hypertext documents. He installed the application on his and Cailliau's computers, and they both began communicating via the world's first web server, at info.cern.ch, on December 25, 1990.

A few months later, in August 1991, Berners-Lee posted a notice on a newsgroup called alt.hypertext that provided information about where one could download the web server (Figure 1.10) and browser. Once this information hit the newsgroup, new web servers began appearing all over the world almost immediately.

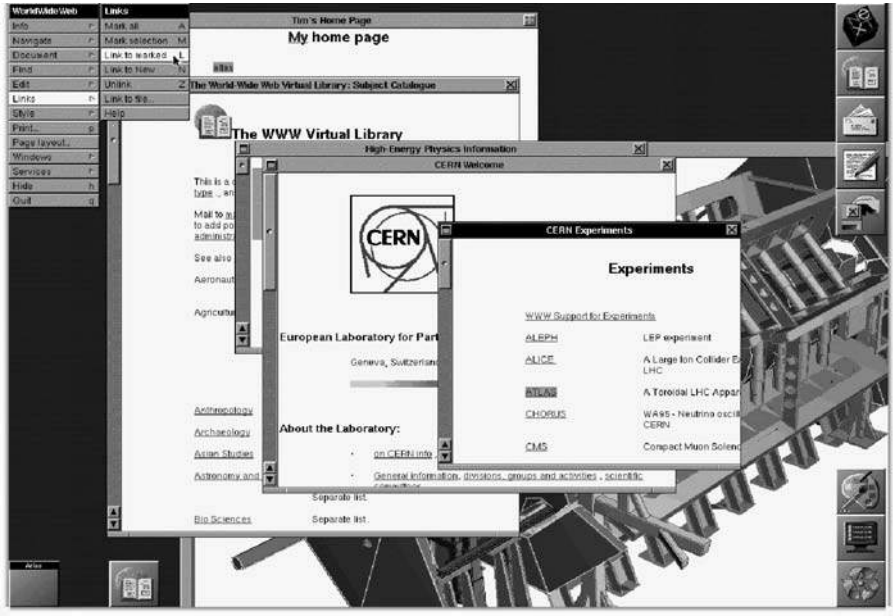


Figure 1.9 The first web browser, created by Tim Berners-Lee. (Image from www.tranquileye.com/cyber/index.html, retrieved 9 Jan 2009.)

Figure 1.10 Tim Berners-Lee's first web server.

Following this initial success, Berners-Lee enhanced the server and browser by adding support for the FTP protocol. This made a wide range of existing FTP directories and Usenet newsgroups instantly accessible via a web page displayed in his browser. He also added a Telnet server on info.cern.ch, making a simple line browser available to anyone with a Telnet client.

The first public demonstration of Berners-Lee's web server was at a conference called Hypertext 91. This web server came to be known as CERN httpd (short for hypertext transfer protocol daemon), and work in it continued until July 1996. Before work stopped on the CERN httpd, Berners-Lee managed to get CERN to provide a certification on April 30, 1993, that the web technology and program code was in the public domain so that anyone could use and improve it. This was an important decision that helped the web to grow to enormous proportions.

In 1992, Joseph Hardin and Dave Thompson were working at the NCSA. When Hardin and Thompson heard about Berners-Lee's work, they downloaded the Viola WWW browser and demonstrated it to NCSA's Software Design Group by connecting to the web server at CERN over the Internet.[17] The Software Design Group was impressed by what they saw. Two students from the group, Marc Andreessen and Eric Bina, began work on a browser version for X-Windows on Unix computers, first released as version 0.5 on January 23, 1993 (Figure 1.11). Within a week, Andreeson's release message was forwarded to various newsgroups by Berners-Lee. This generated a huge swell in the user base and subsequent redistribution ensued, creating a wider awareness of the product. Working together to support the product, Bina provided expert coding support while Andreessen

17. Marc Andreessen, NCSA Mosaic Technical Summary, 20 Feb 1993.

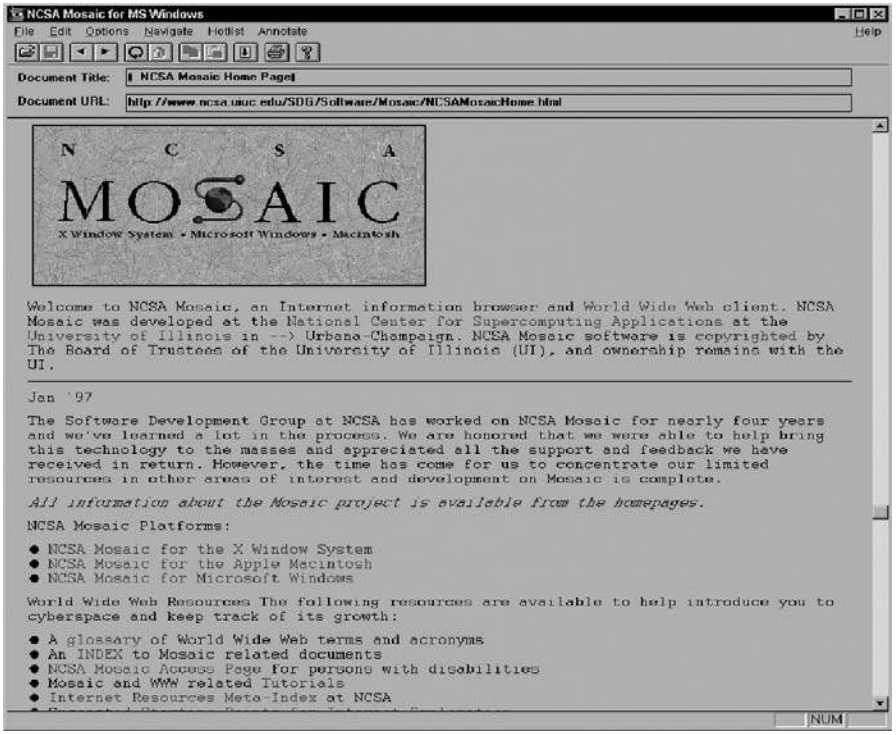


Figure 1.11 The original NCSA Mosaic browser. (Image from http://www.nsf.gov/od/lpa/news/03/images/mosaic.6beta.jpg.)

provided excellent customer support. They monitored the newsgroups continuously to ensure that they knew about and could fix any bugs reported and make the desired enhancements pointed out by the user base.

Mosaic was the first widely popular web browser available to the general public. It helped spread use and knowledge of the web across the world. Mosaic provided support for graphics, sound, and video clips. An early version of Mosaic introduced forms support, enabling many powerful new uses and applications. Innovations including the use of bookmarks and history files were added. Mosaic became even more popular, helping further the growth of the World Wide Web. In mid-1994, after Andreessen had graduated from the University of Illinois, Silicon Graphics founder Jim Clark collaborated with Andreessen to found Mosaic Communications, which was later renamed Netscape Communications.

In October 1994, Netscape released the first beta version of its browser, Mozilla 0.96b, over the Internet. The final version, named Mozilla 1.0, was


Figure 1.12 The original Netscape browser. (Image from http://browser.netscape.com/downloads/archive.)

released in December 1994. It became the very first commercial web browser. The Mosaic programming team then developed another web browser, which they named Netscape Navigator. Netscape Navigator was later renamed Netscape Communicator, then renamed back to just Netscape. See Figure 1.12.

During this period, Microsoft was not asleep at the wheel. Bill Gates realized that the WWW was the future and focused vast resources to begin developing a product to compete with Netscape. In 1995, Microsoft hosted an Internet Strategy Day[18] and announced its commitment to adding Internet capabilities to all its products. In fulfillment of that announcement, Microsoft Internet Explorer arrived as both a graphical Web browser and the name for a set of technologies.

18. http://www.microsoft.com/windows/WinHistoryIE.mspx, retrieved 8 Jan 2009.

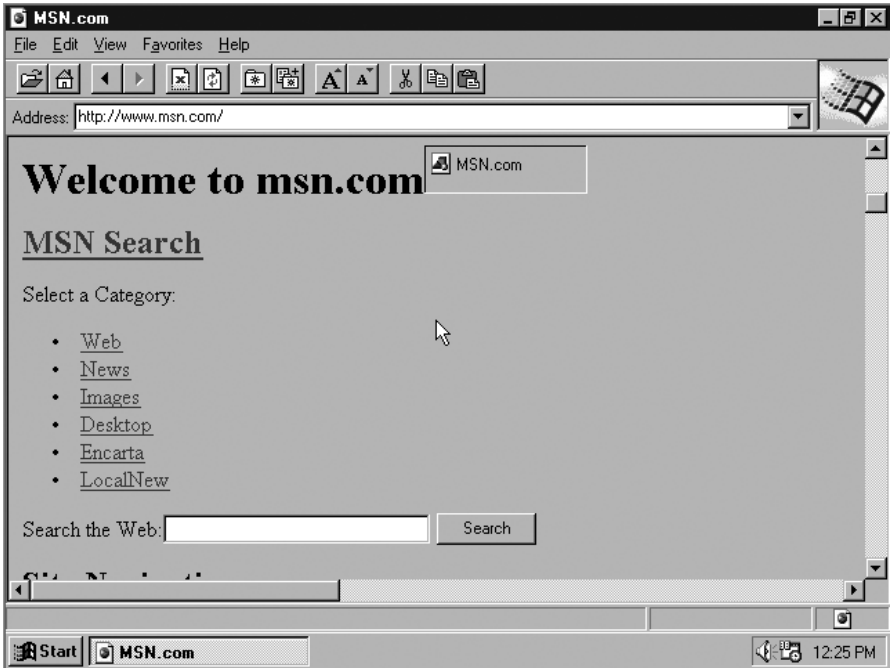


Figure 1.13 Internet Explorer version 1.0. (Image from http://www.microsoft.com/library/media/1033/windows/IE/images/community/columns/old_ie.gif, retrieved 9 Jan 2009.)

In July 1995, Microsoft released the Windows 95 operating system, which included built-in support for dial-up networking and TCP/IP, two key technologies for connecting a PC to the Internet. It also included an add-on to the operating system called Internet Explorer 1.0 (Figure 1.13). When Windows 95 with Internet Explorer debuted, the WWW became accessible to a great many more people. Internet Explorer technology originally shipped as the Internet Jumpstart Kit in Microsoft Plus! for Windows 95.

One of the key factors in the success of Internet Explorer was that it eliminated the need for cumbersome manual installation that was required by many of the existing shareware browsers. Users embraced the "do-it-for-me" installation model provided by Microsoft, and browser loyalty went out the window. The Netscape browser led in user and market share until Microsoft released Internet Explorer, but the latter product took the market lead in 1999. This was due mainly to its distribution advantage, because it was included in every version of Microsoft Windows. The browser wars had begun, and the battlefield was the Internet. In response to Microsoft's move,


Figure 1.14 The open source version of Netscape, named Mozilla. (Image from http://browser.netscape.com/downloads/archive.)

Netscape decided in 2002 to release a free, open source software version of Netscape named Mozilla (which was the internal name for the old Netscape browser; see Figure 1.14). Mozilla has steadily gained market share, particularly on non-Windows platforms such as Linux, largely because of its open source foundation. Mozilla Firefox, released in November 2004, became very popular almost immediately.

### 1.3.5 The Appearance of Cloud Formations—From One Computer to a Grid of Many

Two decades ago, computers were clustered together to form a single larger computer in order to simulate a supercomputer and harness greater processing power. This technique was common and was used by many IT departments. *Clustering*, as it was called, allowed one to configure computers using special protocols so they could "talk" to each other. The purpose

was to balance the computational load across several machines, divvying up units of work and spreading it across multiple processors. To the user, it made little difference which CPU executed an application. Cluster management software ensured that the CPU with the most available processing capability at that time was used to run the code. A key to efficient cluster management was engineering where the data was to be held. This process became known as *data residency.* Computers in the cluster were usually physically connected to magnetic disks that stored and retrieved a data while the CPUs performed input/output (I/O) processes quickly and efficiently.

In the early 1990s, Ian Foster and Carl Kesselman presented their concept of "The Grid." They used an analogy to the electricity grid, where users could plug in and use a (metered) utility service. They reasoned that if companies cannot generate their own power, it would be reasonable to assume they would purchase that service from a third party capable of providing a steady electricity supply. So, they asked, "Why can't the same apply to computing resources?" If one node could plug itself into a grid of computers and pay only for the resources it used, it would be a more cost-effective solution for companies than buying and managing their own infrastructure. Grid computing expands on the techniques used in clustered computing models, where multiple independent clusters appear to act like a grid simply because they are not all located within the same domain.[19]

A major obstacle to overcome in the migration from a clustering model to grid computing was data residency. Because of the distributed nature of a grid, computational nodes could be anywhere in the world. Paul Wallis explained the data residency issue for a grid model like this:

> It was fine having all that CPU power available, but the data on which the CPU performed its operations could be thousands of miles away, causing a delay (latency) between data fetch and execution. CPUs need to be fed and watered with different volumes of data depending on the tasks they are processing. Running a data-intensive process with disparate data sources can create a bottleneck in the I/O, causing the CPU to run inefficiently, and affecting economic viability.[20]

19. Paul Wallis, "Keystones and Rivets," http://it.toolbox.com/blogs/keystones-and-rivets/understanding-cloud-computing-22611, retrieved 2 Jan 2009.
20. Ibid.

The issues of storage management, migration of data, and security provisioning were key to any proposed solution in order for a grid model to succeed. A toolkit called Globus[21] was created to solve these issues, but the infrastructure hardware available still has not progressed to a level where true grid computing can be wholly achieved.

The Globus Toolkit is an open source software toolkit used for building grid systems and applications. It is being developed and maintained by the Globus Alliance[22] and many others all over the world. The Globus Alliance has grown into community of organizations and individuals developing fundamental technologies to support the grid model. The toolkit provided by Globus allows people to share computing power, databases, instruments, and other online tools securely across corporate, institutional, and geographic boundaries without sacrificing local autonomy.

The cloud is helping to further propagate the grid computing model. Cloud-resident entities such as data centers have taken the concepts of grid computing and bundled them into service offerings that appeal to other entities that do not want the burden of infrastructure but do want the capabilities hosted from those data centers. One of the most well known of the new cloud service providers is Amazon's S3 (Simple Storage Service) third-party storage solution. Amazon S3 is storage for the Internet. According to the Amazon S3 website,[23] it provides a simple web services interface that can be used to store and retrieve any amount of data, at any time, from anywhere on the web. It gives any developer access to the same highly scalable, reliable, fast, inexpensive data storage infrastructure that Amazon uses to run its own global network of web sites. The service aims to maximize benefits of scale and to pass those benefits on to developers.

In 2002, EMC offered a Content Addressable Storage (CAS) solution called Centera as yet another cloud-based data storage service that competes with Amazon's offering. EMC's product creates a global network of data centers, each with massive storage capabilities. When a user creates a document, the application server sends it to the Centera storage system. The storage system then returns a unique content address to the server. The unique address allows the system to verify the integrity of the documents whenever a user moves or copies them. From that point, the application can request the document by submitting the address. Duplicates of documents

21. The reader is encouraged to visit http://globus.org/toolkit for more information.
22. http://www.globus.org, retrieved 6 Jan 2009.
23. The reader is encouraged to visit http://aws.amazon.com/s3.

are saved only once under the same address, leading to reduced storage requirements. Centera then retrieves the document regardless of where it may be physically located.

EMC's Centera product takes the sensible approach that no one can afford the risk of placing all of their data in one place, so the data is distributed around the globe. Their cloud will monitor data usage and automatically move data around in order to load-balance data requests and better manage the flow of Internet traffic. Centera is constantly self-tuning to react automatically to surges in demand. The Centera architecture functions as a cluster that automatically configures itself upon installation. The system also handles fail-over, load balancing, and failure notification.

There are some drawbacks to these cloud-based solutoins, however. An example is a recent problem at Amazon S3. They suffered a "massive" outage in February 2008, which served to highlight the risks involved with adopting such cloud-based service offerings. Amazon's technical representative from the Web Services Team commented publicly with the following press release:

> Early this morning, at 3:30am PST, we started seeing elevated levels of authenticated requests from multiple users in one of our locations. While we carefully monitor our overall request volumes and these remained within normal ranges, we had not been monitoring the proportion of authenticated requests. Importantly, these cryptographic requests consume more resources per call than other request types. Shortly before 4:00am PST, we began to see several other users significantly increase their volume of authenticated calls. The last of these pushed the authentication service over its maximum capacity before we could complete putting new capacity in place.
>
> In addition to processing authenticated requests, the authentication service also performs account validation on every request Amazon S3 handles. This caused Amazon S3 to be unable to process any requests in that location, beginning at 4:31am PST. By 6:48am PST, we had moved enough capacity online to resolve the issue.
>
> As we said earlier today, though we're proud of our uptime track record over the past two years with this service, any amount of downtime is unacceptable. As part of the post mortem for this event, we have identified a set of short-term actions as well as longer

> term improvements. We are taking immediate action on the following: (a) improving our monitoring of the proportion of authenticated requests; (b) further increasing our authentication service capacity; and (c) adding additional defensive measures around the authenticated calls. Additionally, we've begun work on a service health dashboard, and expect to release that shortly.
>
> Sincerely,
>
> The Amazon Web Services Team

The message above clearly points out the lesson one should take from this particular incident: *caveat emptor,* which is Latin for "Let the buyer beware."

## 1.4 Server Virtualization

*Virtualization* is a method of running multiple independent virtual operating systems on a single physical computer.[24] This approach maximizes the return on investment for the computer. The term was coined in the 1960s in reference to a virtual machine (sometimes called a pseudo-machine). The creation and management of virtual machines has often been called *platform virtualization.* Platform virtualization is performed on a given computer (hardware platform) by software called a control program. The control program creates a simulated environment, a virtual computer, which enables the device to use hosted software specific to the virtual environment, sometimes called guest software.

The guest software, which is often itself a complete operating system, runs just as if it were installed on a stand-alone computer. Frequently, more than one virtual machine is able to be simulated on a single physical computer, their number being limited only by the host device's physical hardware resources. Because the guest software often requires access to specific peripheral devices in order to function, the virtualized platform must support guest interfaces to those devices. Examples of such devices are the hard disk drive, CD-ROM, DVD, and network interface card. Virtualization technology is a way of reducing the majority of hardware acquisition and maintenance costs, which can result in significant savings for any company.

24. George Ou, "Introduction to Server Virtualization," http://articles.techrepublic.com.com/5100-10878_11-6074941.html, retrieved 6 Jan 2009.

### 1.4.1 Parallel Processing

Parallel processing is performed by the simultaneous execution of program instructions that have been allocated across multiple processors with the objective of running a program in less time.[25] On the earliest computers, a user could run only one program at a time. This being the case, a computation-intensive program that took X minutes to run, using a tape system for data I/O that took X minutes to run, would take a total of X + X minutes to execute. To improve performance, early forms of parallel processing were developed to allow interleaved execution of both programs simultaneously. The computer would start an I/O operation (which is typically measured in milliseconds), and while it was waiting for the I/O operation to complete, it would execute the processor-intensive program (measured in nanoseconds). The total execution time for the two jobs combined became only slightly longer than the X minutes required for the I/O operations to complete.

The next advancement in parallel processing was multiprogramming. In a multiprogramming system, multiple programs submitted by users are each allowed to use the processor for a short time, each taking turns and having exclusive time with the processor in order to execute instructions. This approach is known as "round-robin scheduling" (RR scheduling). It is one of the oldest, simplest, fairest, and most widely used scheduling algorithms, designed especially for time-sharing systems.[26]

In RR scheduling, a small unit of time called a time slice (or quantum) is defined. All executable processes are held in a circular queue. The time slice is defined based on the number of executable processes that are in the queue. For example, if there are five user processes held in the queue and the time slice allocated for the queue to execute in total is 1 second, each user process is allocated 200 milliseconds of process execution time on the CPU before the scheduler begins moving to the next process in the queue. The CPU scheduler manages this queue, allocating the CPU to each process for a time interval of one time slice. New processes are always added to the end of the queue. The CPU scheduler picks the first process from the queue, sets its timer to interrupt the process after the expiration of the timer, and then dispatches the next process in the queue. The process whose time has expired is placed at the end of the queue. If a process is still running at the end of a time slice, the CPU is interrupted and the process goes to the end

25. http://searchdatacenter.techtarget.com/sDefinition/0,,sid80_gci212747,00.html, retrieved 10 Jan 2009.
26. http://choices.cs.uiuc.edu/~f-kon/RoundRobin/node1.html, retrieved 10 Jan 2009.

of the queue. If the process finishes before the end of the time-slice, it releases the CPU voluntarily. In either case, the CPU scheduler assigns the CPU to the next process in the queue. Every time a process is granted the CPU, a context switch occurs, which adds overhead to the process execution time. To users it appears that all of the programs are executing at the same time.

Resource contention problems often arose in these early systems. Explicit requests for resources led to a condition known as deadlock. Competition for resources on machines with no tie-breaking instructions led to the critical section routine. Contention occurs when several processes request access to the same resource. In order to detect deadlock situations, a counter for each processor keeps track of the number of consecutive requests from a process that have been rejected. Once that number reaches a predetermined threshold, a state machine that inhibits other processes from making requests to the main store is initiated until the deadlocked process is successful in gaining access to the resource.

### 1.4.2 Vector Processing

The next step in the evolution of parallel processing was the introduction of multiprocessing. Here, two or more processors share a common workload. The earliest versions of multiprocessing were designed as a master/slave model, where one processor (the master) was responsible for all of the tasks to be performed and it only off-loaded tasks to the other processor (the slave) when the master processor determined, based on a predetermined threshold, that work could be shifted to increase performance. This arrangement was necessary because it was not then understood how to program the machines so they could cooperate in managing the resources of the system.

Vector processing was developed to increase processing performance by operating in a multitasking manner. Matrix operations were added to computers to allow a single instruction to manipulate two arrays of numbers performing arithmetic operations. This was valuable in certain types of applications in which data occurred in the form of vectors or matrices. In applications with less well-formed data, vector processing was less valuable.

### 1.4.3 Symmetric Multiprocessing Systems

The next advancement was the development of symmetric multiprocessing systems (SMP) to address the problem of resource management in master/slave models. In SMP systems, each processor is equally capable and

responsible for managing the workflow as it passes through the system. The primary goal is to achieve *sequential consistency,* in other words, to make SMP systems appear to be exactly the same as a single-processor, multiprogramming platform. Engineers discovered that system performance could be increased nearly 10–20% by executing some instructions out of order. However, programmers had to deal with the increased complexity and cope with a situation where two or more programs might read and write the same operands simultaneously. This difficulty, however, is limited to a very few programmers, because it only occurs in rare circumstances. To this day, the question of how SMP machines should behave when accessing shared data remains unresolved.

Data propagation time increases in proportion to the number of processors added to SMP systems. After a certain number (usually somewhere around 40 to 50 processors), performance benefits gained by using even more processors do not justify the additional expense of adding such processors. To solve the problem of long data propagation times, message passing systems were created. In these systems, programs that share data send messages to each other to announce that particular operands have been assigned a new value. Instead of a global message announcing an operand's new value, the message is communicated only to those areas that need to know the change. There is a network designed to support the transfer of messages between applications. This allows a great number processors (as many as several thousand) to work in tandem in a system. These systems are highly scalable and are called massively parallel processing (MPP) systems.

### 1.4.4 Massively Parallel Processing Systems

*Massive parallel processing* is used in computer architecture circles to refer to a computer system with many independent arithmetic units or entire microprocessors, which run in parallel.[27] "Massive" connotes hundreds if not thousands of such units. In this form of computing, all the processing elements are interconnected to act as one very large computer. This approach is in contrast to a distributed computing model, where massive numbers of separate computers are used to solve a single problem (such as in the SETI project, mentioned previously). Early examples of MPP systems were the Distributed Array Processor, the Goodyear MPP, the Connection Machine, and the Ultracomputer. In data mining, there is a need to perform multiple searches of a static database. The earliest massively parallel

27. http://en.wikipedia.org/wiki/Massive_parallel_processing, retrieved 10 Jan 2009.

processing systems all used serial computers as individual processing units in order to maximize the number of units available for a given size and cost. Single-chip implementations of massively parallel processor arrays are becoming ever more cost effective due to the advancements in integrated-circuit technology.

An example of the use of MPP can be found in the field of artificial intelligence. For example, a chess application must analyze the outcomes of many possible alternatives and formulate the best course of action to take. Another example can be found in scientific environments, where certain simulations (such as molecular modeling) and complex mathematical problems can be split apart and each part processed simultaneously. Parallel data query (PDQ) is a technique used in business. This technique divides very large data stores into pieces based on various algorithms. Rather than searching sequentially through an entire database to resolve a query, 26 CPUs might be used simultaneously to perform a sequential search, each CPU individually evaluating a letter of the alphabet. MPP machines are not easy to program, but for certain applications, such as data mining, they are the best solution.

## 1.5 Chapter Summary

In this chapter, we stressed the importance of knowing about the evolution of computing in order to get an appreciation of how we got into the cloud environment. Examining the history of computing hardware and software helps us to understand why we are standing on the shoulders of giants. We discussed how the rules computers use to communicate came about, and how the development of networking and communications protocols has helped drive the Internet technology growth we have seen in the last 20-plus years. This, in turn, has driven even more changes in protocols and forced the creation of new technologies to mitigate addressing concerns and improve the methods used to communicate over the Internet. The use of web browsers has led to huge Internet growth and a migration away from the traditional data center. In the next chapter, we will begin to examine how services offered to Internet users has also evolved and changed the way business is done.