4 IT Infrastructure -- Overview of Enabling Technologies

4.1 Introduction

A major challenge for the industry is to harness information technology (IT) to survive and succeed in the extremely competitive e-business environments. Everyone wants to use IT to reduce time to market, increase customer satisfaction, and improve employee productivity. The IT infrastructure (i.e., middleware, networks, operating systems, and computing hardware) enables e-business applications strategies. The purpose of this chapter is to review this enabling infrastructure quickly before discussing how applications can be engineered or reengineered by using this infrastructure. The information in this chapter is intentionally high level and is an abbreviation of the enabling technologies modules of this book ("Networks", "Middleware", "Platforms") that dig deeply into the individual technologies.

Figure 4-1 will serve as a general framework for discussion. This framework, introduced in the first chapter of this book, illustrates the role of the following main IT infrastructure building blocks:

- Networks that provide the network transport between remote parties and are responsible for routing and flow/error control support. The networks may be the private value added networks (VANs), Public Internet, and/or Extranets that utilize the wired or wireless transmission media. Network services are reviewed in Section 4.3.
- Computing platforms that consist of operating systems and computing hardware to provide the basic scheduling and hardware services. The computing platforms also include local system software services such as database managers, transaction managers, language translators, and utilities (see Section 4.4).
- Middleware that interconnects remotely located users, databases and applications. Middleware components are *business/industry unaware* software modules that provide a variety of services such as Web services, directory services, email, and remote data access services. We will review the general middleware in Section 4.5 and specialized (higher level) middleware services specifically intended for EC/EB in Section 4.6.

An interesting trend at present is to package a variety of IT building blocks into "application support environments" that can support the current and future breed of distributed applications. An example of such dominant platform is Sun's J2EE (Java2 Enterprise Edition). Another example is Microsoft's Dot Net environment. These two environments are discussed briefly in Section 4.7.

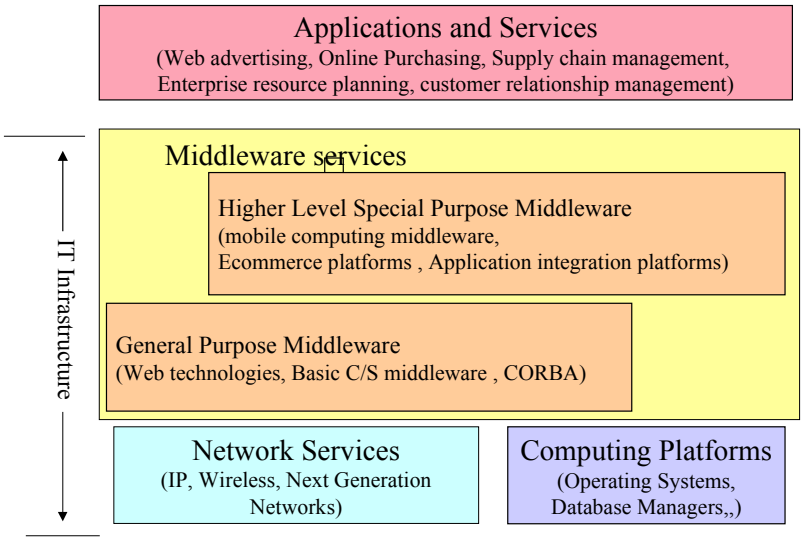

Figure 4-1: IT Infrastructure

4.2 Case Study: XYZCorp Review of IT Infrastructure Architecture

XYZCorp needs to review its existing IT infrastructure architecture and determine if it will enable all the services and applications to be provided by XYZCorp. The IT Planning Committee has asked you to produce a high level vision of the future infrastructure architecture. This vision will be refined later. In particular, the vision should show how the networking services, the middleware components, and other services will support the emerging e-business applications that will also utilize mobile computing and groupware. This architecture should allow different applications and users at different sites to communicate with each other ("any data from any application anywhere in the company"). The current platforms consist of a variety of devices. The regional offices house minicomputers (mainly UNIX) which are connected to the corporate mainframe (MVS). The regional computers maintain regional inventory, customer information and prices of items sold in the region. Some regions (e.g., Atlanta and San Francisco) are very UNIX oriented. Other regions have Novell LANs. The corporate headquarters are IBM mainframe oriented (MVS, DB2, IMS, SNA, Token Ring). PCs are used commonly throughout the organization. The company has not figured out how the new platforms such as the Sun J2EE and Microsoft Dot Net will fit into the big picture. The company specifically wants to know how will AICS (introduced in the case study at the end of previous chapter) reside on this infrastructure.

The Agenda
- Network Overview
- Computing Platforms Overview
- Middleware Services
- 3G Distributed Computing Platforms

4.3 Network Services and the Internet

4.3.1 Overview of Network Services

Network services, as shown in Figure 4-2, provide a transport mechanism that is at the bottom of the IT infrastructure. Network services are provided through communication networks that are a collection of equipment and physical media, viewed as one autonomous whole, that interconnect two or more stations. A station is an end-point (source/sink) in a communication network and can be a terminal, computer, telephone, sensor or a TV. A network can be configured as a wide area network (WAN) for long distance communications, a local area network (LAN) for short-haul communications within a building, or a metropolitan area network (MAN) for communications within a city. Most enterprise networks are a combination of LANs, MANs, and WANs. Physically speaking, the communication between stations on a network can use analog or digital transmission facilities over copper, wireless or fiber optic media.

The major growth in networking is interconnection and integration of LANs, MANs and WANs into large and high data rate (known as "broadband") networks. Network interconnections are based on network architectures that describe the physical components, the functions performed by the components and the interfaces between the components of a network. Network architecture standards are needed to interconnect different networks from different vendors with different capabilities.

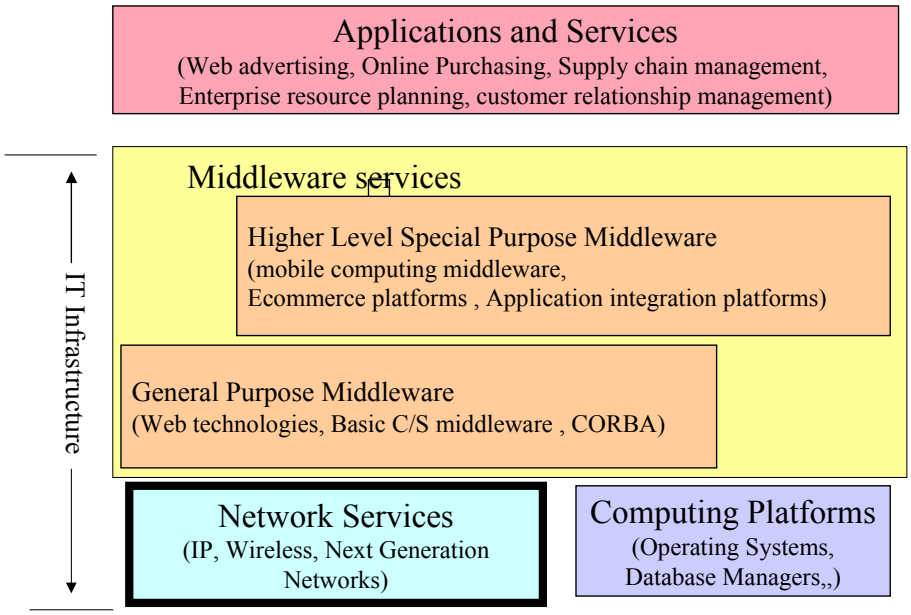

Figure 4-2: Network Services as part of IT Infrastructure

The Open System Interconnection (OSI) Reference Model, shown in Figure 4-3 specifies standards for networks from different vendors to exchange information with each other. The model describes network services in terms of 7 layers, but the lowest 4 layers are devoted to network services. These services are responsible for routing and transporting your messages in an error-free manner across a network. A common example of layered network protocols is the TCP/IP (Transmission Control Protocol/Internet Protocol) that is at the foundation of the Internet. Network Services are discussed in detail in the Network Module of this book. The following gives a brief overview.

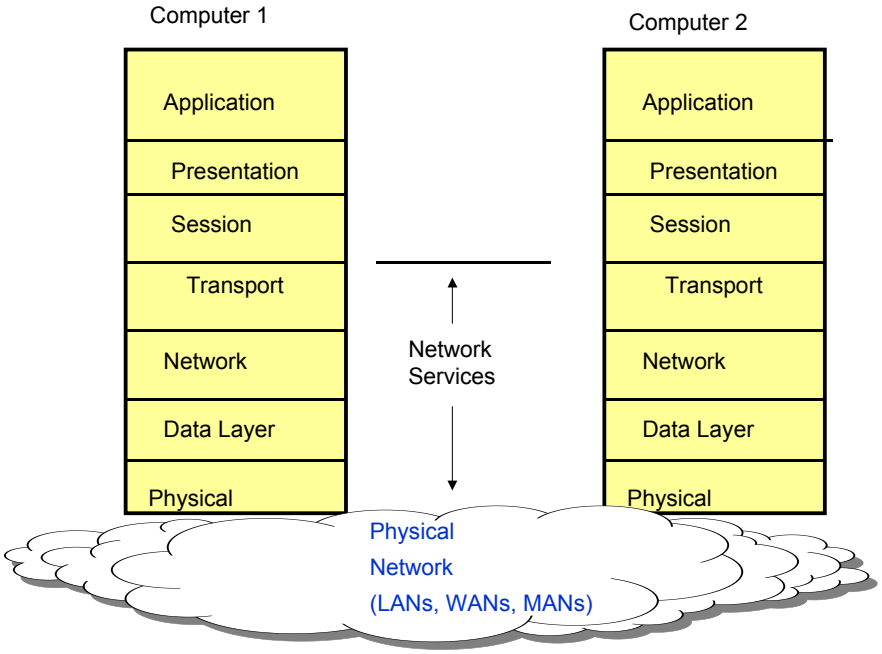

Figure 4-3: Conceptual View of OSI Network Model

4.3.2 Network Interconnectivity

Network interconnectivity is needed in large networks to provide interfaces and transport of messages between remotely located users, applications, databases, and devices. For example, if you access the Paris University Web site from Chicago, then many interconnectivity devices are used to get you from Chicago to Paris. The two principal network interconnectivity devices are:

- **Routers** find a path for a message in larger networks and then send the message over the selected path. Routers use very sophisticated routing algorithms and provide functionality such as "firewalls" for security checking.
- **Gateways** translate one type of protocol to another. In most large networks, protocols of some subnetworks need to be converted to protocols of other subnetworks for end to end communications. A gateway connects two dissimilar network architectures and is essentially a protocol converter. A gateway may be a special purpose computer, a workstation with associated software (e.g., a PC with gateway software), or a software module which runs as a task in a mainframe. An example of gateways for network interconnectivity is the TCP/IP to Novell LANs.

Many routers and gateways are used commonly in enterprise networks and the general Public Internet. For example, if a salesman in Detroit needs to access a customer database in New York, then a series of routers and gateways would be needed to find the path between the two cities. Figure 4-4 shows a realistic enterprise network that uses TCP/IP very heavily, except the IBM SNA network (an old network technology) at the mainframe. The routers are used between all TCP/IP network segments and gateways are used to convert the TCP/IP messages to SNA and the Novell protocol.

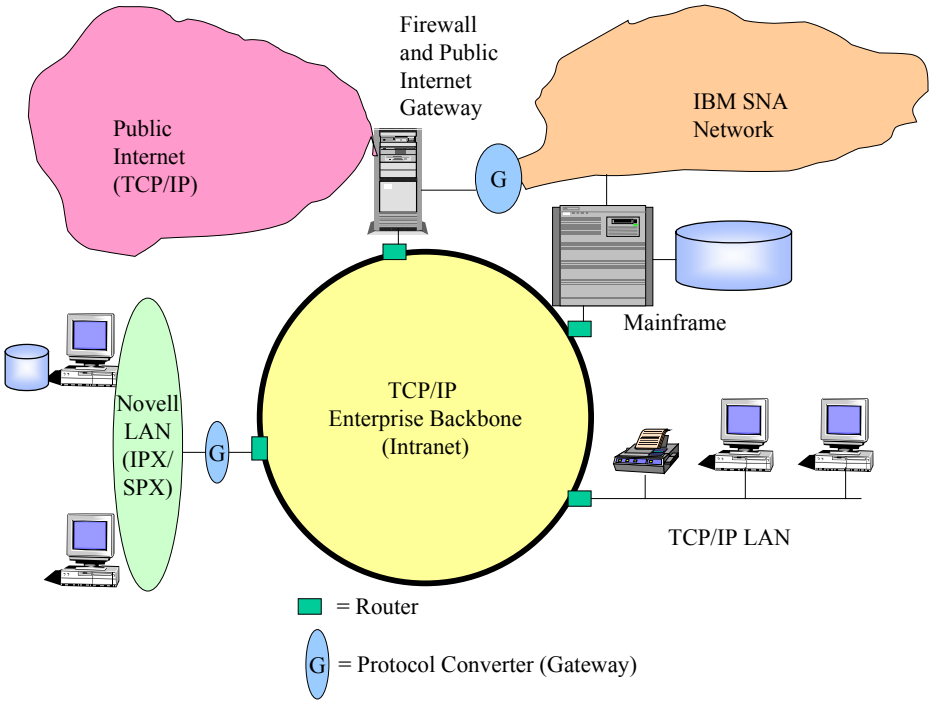

Figure 4-4: Network Interconnectivity in an Enterprise Environment

4.3.3 Broadband and Wireless Networks

The advancements in network transmission technologies have resulted in high data rate local and wide area transmissions, typically in the range of 100 million bits per second (Mbps) or higher (Gigabit networks). Examples of the evolving network communication technologies are Asynchronous Transfer Mode (ATM),

Frame Relay, Fiber Distributed Data Interface (FDDI), and wireless networks. In general, networks are becoming faster, ubiquitous and more reliable. Another area of advancement is the integration of voice, data and video images for multimedia applications such as teleconferencing and group problem solving, among others. In particular, Next Generation Networks (NGNs) combine the voice and data networks into an integrated high speed network.

Wireless networks, as the name implies, interconnect devices without using wires -- instead they use the air as the main transmission medium. Wireless networks are enjoying widespread public approval with a rapidly increasing demand. The increase in the number of cellular phones, palm pilots, laptops, notebooks, and other handheld devices is phenomenal. To meet this demand, mobile communications technologies are emerging with digital speech transmission and the ability to integrate cordless systems into other networks. In the meantime, researchers are developing the next generation of technologies for several years to come.

The unique features of the wireless networks are:

- The bandwidths, and consequently data rates, of communication channels are restricted by government regulations. The government policies allow only a few frequency ranges for wireless communications.
- The communication channel between senders/receivers is often impaired by noise, interference and weather fluctuations.
- The senders and receivers of information are not physically connected to a network. Thus the location of a sender/receiver is unknown prior to start of communication and can change during the conversation.

A very large body of work on wireless networks exists with emphasis on different aspects such as radio transmission technologies, standards, protocols, systems engineering, and carriers. For our purpose, wireless networks can be broadly classified in terms of wireless local area networks, wide area networks and metropolitan area networks (see Figure 4-5).

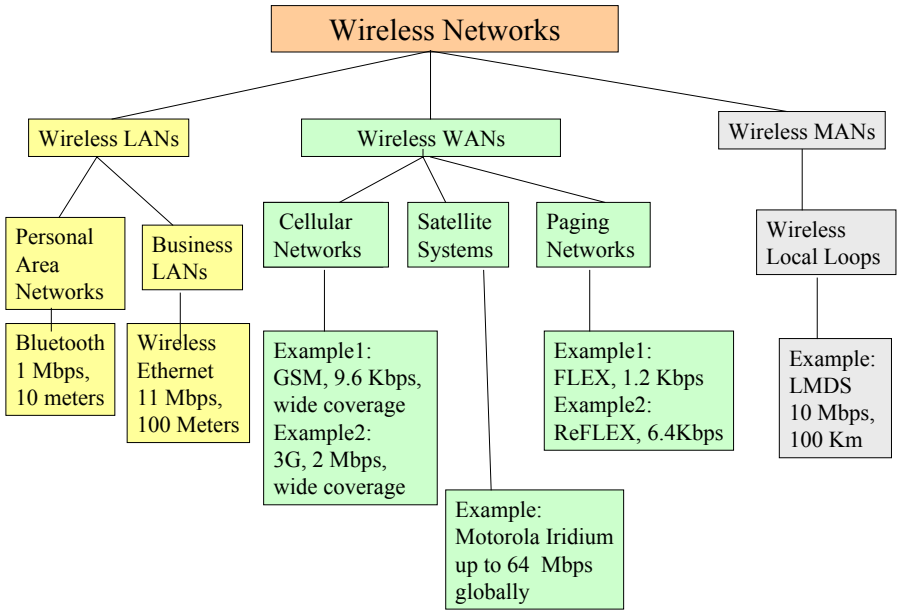

Figure 4-5: A View of Wireless Network Landscape

4.3.4 IP Networks and the Internet

Internet is of particular interest to e-business, since more than 80% of the e-business activities are expected to be conducted over the Internet. The origin of Internet is the ARPANET (Advanced Research Projects

Agency Network) that was initiated in 1969 to support researchers on DOD (Department of Defense) projects. For many years, Internet was used mainly by scientists and programmers to transfer files and send/receive electronic mail. The users of Internet relied on text-based user interfaces and tedious commands to access remote computing resources. In 1989, this changed with the introduction of World Wide Web (WWW), commonly referred to as the Web. Technically speaking, Internet is a large collection of IP (Internet Protocol)-based networks that are interconnected through a wide range of interconnection devices. IP networks reside on a variety of physical network elements (e.g ., ATMs, DSL, wireless networks) to support Web technologies. The Web has been a major contributor in turning the Internet, once an obscure tool, into a household word. The Web allows users to access, navigate and share information around the globe through GUI clients ("Web browsers") that are available on almost all computing platforms. The Web browsers allow users to access information that is linked through hypermedia links. Thus a user transparently browses around, or "surfs" around, different pieces of information that is located on different computers in different cities and even in different countries. In addition to Web, the IP-based networks are being used for a wide range of applications such as Internet Telephony, video conferencing, and corporate computing.

Technically speaking, ***Internet*** is a network based on the TCP/IP protocol stack. At present, the term Internet is used to refer to a large collection of TCP/IP networks that are tied together through network interconnectivity devices such as routers and gateways . The TCP/IP (Transmission Control Protocol/Internet Protocol) was developed in the late 1960s and early 1970s by the Defense Advanced Research Projects Agency (DARPA). TCP/IP was developed for interconnecting many computers in the ARPANET (Advanced Research Projects Agency Network). ARPANET initially consisted of five protocols (indicated with * in the following list) that have been augmented with other key protocols (see Figure 4-6):

- *Internet Protocol (IP) for interconnecting and routing messages to a large number of physical networks
- *Transmission Control Protocol (TCP) for reliable information transfer
- User Datagram Program (UDP) for fast, but unreliable, information transfer
- *File Transfer Protocol (FTP) for file transfer
- *Simplified Mail Transfer Protocol (SMTP) for email
- *Terminal emulator (Telnet) for terminal emulation
- Domain Name Services (DNS) to translate an address such as www.mycorp.com to a physical IP address such as 192.28.200.32
- Hypertext Transfer Protocol (HTTP) for Web applications
- Real Time Protocol (RTP) for audio and video applications

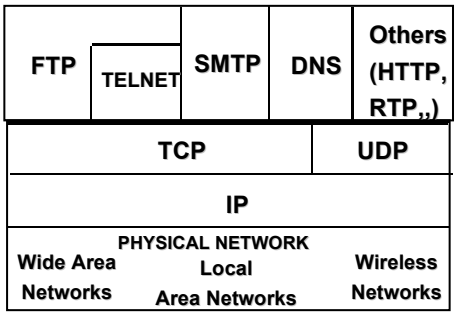

Figure 4-6: The IP Stack – Foundation of Internet

Although, the Internet at present uses TCP (i.e., higher level protocols and applications are based on TCP), this may not be true in the future since some future (especially real time) applications may be built directly on IP or newer alternatives to TCP. The main strength of IP is that it runs on top of a very diverse array of physical networks (wide area, local area, wireless) . In fact, IP supports almost all current physical network technologies and is expected to support most of the future high speed networks. We thus will use the following simple definition of the Internet:

Definition: Internet is a network of networks that is supported by the Internet Protocol (IP).

What does this mean? Basically it says that you need to have an IP network (or a gateway that translates to IP) to join the Internet. Once you have an IP network, then you can run almost any physical network under it and take advantage of voice, data, or video applications for your e-business that run on top of IP. At present, the term Internet is used to symbolize a Public Internet that is not owned by any single entity -- it consists of many independent IP networks that are tied together loosely.

Initially, the public Internet was used to tie different university networks together. With time, several commercial and private networks have joined the public Internet. The computers on the public Internet have publicly known Internet Protocol (IP) addresses that are used to exchange information over the public Internet (we will discuss IP addresses later). The public Internet at present consists of millions of computers (PCs, Macs, Sun workstations, HP systems, IBM mainframes) that are interconnected through thousands of networks that use different underlying network technologies (ATMs, frame relays, Ethernet LANs, and wireless networks) in different parts of the world. All these computers and networks are tied together through a global IP network (see Figure 4-7).

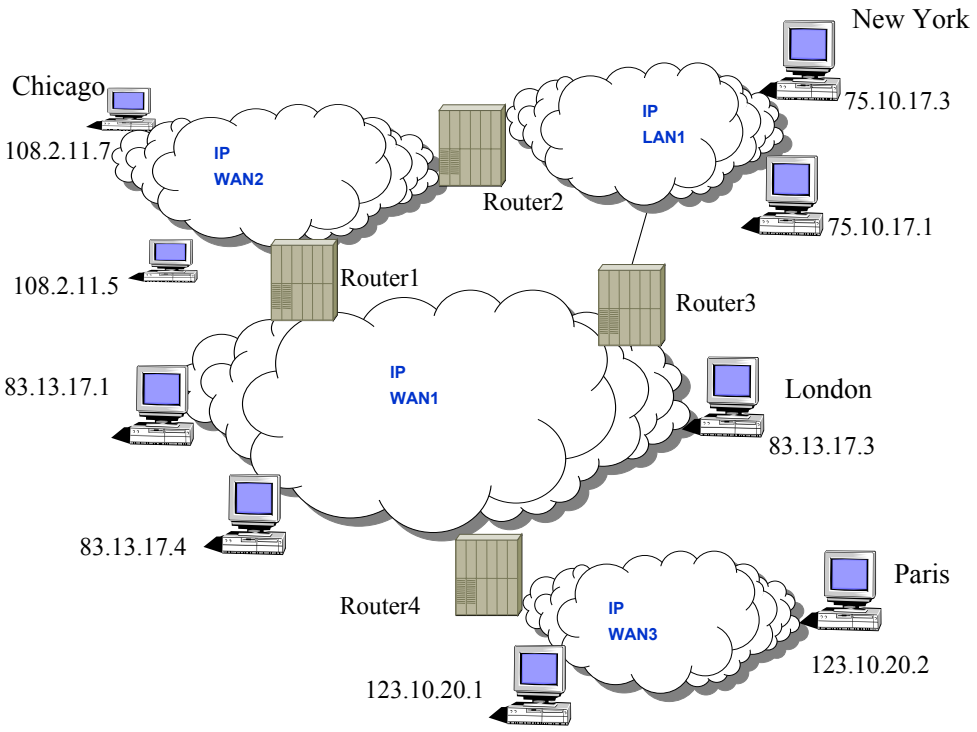

Figure 4-7: Public Internet – A Global IP Network

4.3.5 Private Value Added Networks (VANs).

In E-commerce (EC), network services have traditionally been the realm of VANs vendors such as GEIS, AT&T, and Harbinger. These VANs have supported EC activities through very reliable, secure and well managed private data networks and proprietary technologies. In particular, these networks support purchasing and billing quite well (i.e., purchasing by sending purchase orders and billing by supporting invoices). The well known EDI standards X850 (for purchase orders) and X810 (for invoices) are evidence of this success. However, these networks were never designed for general public advertising and browsing. In addition, VANs are too expensive and require too many proprietary interfaces and software modules to support advertising and browsing services for a large number of trading partners.

4.3.6 Extranets and Virtual Private Networks (VPNs)

"Extra-net" or "enterprise intra-nets" are semi-private IP networks which are used to communicate within a group of interdependent communities of enterprises or trading partners. Examples of such a group of interdependent community would be the automotive industry (including parts suppliers, manufacturers, retailers, and insurers), the health care industry (including physicians, pharmacists, hospitals, labs, and health insurers), or the real estate industry (including brokers, lending agencies, insurers, lawyers, and inspectors). To succeed, Extranets need to support high quality EC services (e.g., advertising, browsing/selection, purchasing, billing, and payments) coupled with security and management considerations.

An *Extranet* consists of a collection of Internet segments (intranets), each protected by firewalls, which are interconnected using secure leased lines across the remote locations. This solution provides security and guaranteed bandwidth, at the cost of leasing lines from telecomm providers. In contrast, *Virtual Private Networks* (VPNs) achieve a similar goal (that is, securely connecting remote locations, branch offices, field workers, telecommuters, vendors, customers, and suppliers) using the public Internet instead of leased lines. Specifically, VPNs encrypt the messages to provide security.

Time to Take a Break

- Network Overview
- Computing Platforms Overview
- Middleware Services
- 3G Distributed Computing Platforms

Suggested Review Questions Before Proceeding

- What role do networks play in the modern enterprises. L:ist three examples.
- What is the OSI model and how it is used to describe interconnectivity
- What are the tradeoffs between wireless and wired networks? How do broadband networks fit into this picture?
- What is the technical definition of the Internet and what are the key protocols used in the Internet?

4.4 Computing Platforms – Operating Systems and Local Systems Software

Computing platforms, as shown in Figure 4-8, provide the basic computing hardware, operating systems, and local systems software such as database and transaction managers to support applications. We do not discuss computing hardware, but operating systems and local system software packages are reviewed briefly.

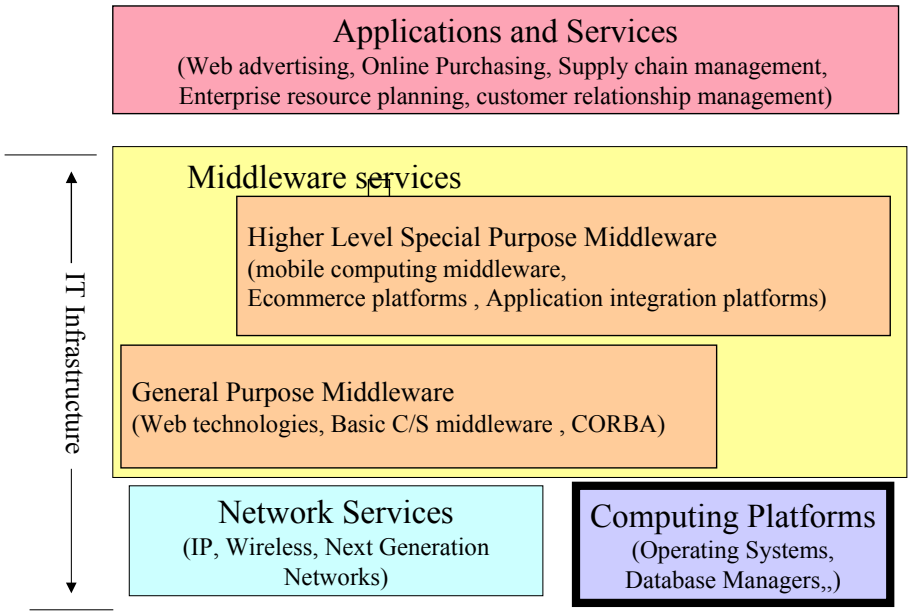

Figure 4-8: Computing Platforms as part of the IT Infrastructure

4.4.1 Operating Systems

Simply stated, an operating system is the computer system's chief manager. As shown in Figure 4-9, the operating system surrounds the computing hardware and all other software (local system software such as database managers, middleware, application software, and the user interfaces) rely on the operating system. It decides which hardware resources will be used, which programs will be run, and the order in which activities will take place. Technically, an operating system is a program, or a collection of programs, which allocates computer resources (memory, CPU, I/O devices, files, etc.) to processes (user commands, application software, middleware services, database managers, other operating systems). An operating system also gives the users a command language to invoke the operating system facilities and to access the computing resources. This command language, also called control language in some systems, is used to access editors, compilers, utilities and other operating system resources.

Many operating systems have been developed over the last thirty years. Examples of state of the market operating systems are Windows and its variants (NT, 2000, XP), Unix, Linux, and MVS. Typical functions performed by an operating system are:

- **Handle User Commands**. The operating system receives, parses and interprets the user commands. It also displays results of the user session with information about resource utilization, etc.
- **Allocation and Assignment.** The operating system allocates resources needed to execute the user commands and jobs, if the user is allowed to access the resources. It provides locations in primary memory for data and programs and controls the input and output devices such as printers, terminals, and telecommunication links.
- **Scheduling.** The operating system decides when to schedule the jobs and user requests that have been submitted. It also coordinates the scheduling in various areas of the computer so that as many things can be done in parallel as possible. The operating system must schedule jobs according to organizational priorities. On-line order processing may have priority over a job to generate mailing labels.
- **Monitoring.** The operating system monitors the activities of the computer system and the processes in the system. It keeps track of each computer job and may also keep track of who is using the system, of what programs have been run, and generate any error messages.

- **Security.** The operating system authenticates the user request for proper authority and keeps track of any unauthorized attempts to access the system. Information system security is discussed in detail in the Management Module.

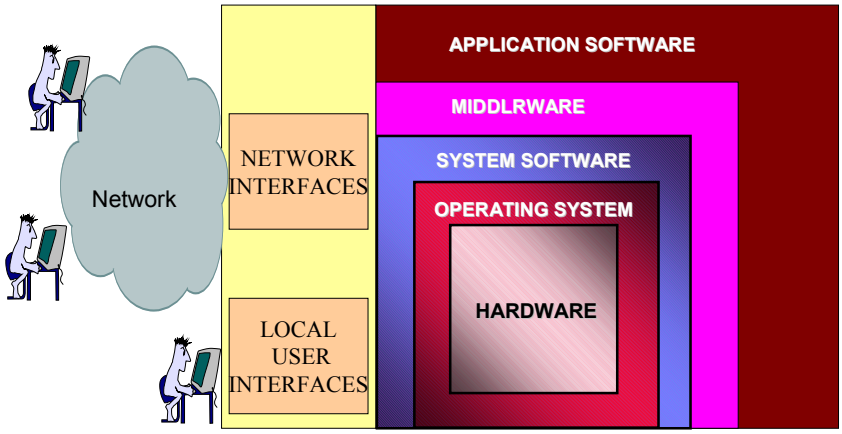

Figure 4-9: Interrelationships Between Various IT Components

Figure 4-10 shows a functional model of operating systems. The user command manager parses and executes the user commands. The memory manager allocates the main memory and the hardware registers to various processes. Most memory managers include the capabilities to manage virtual memory and translate between virtual to real memory. The CPU (central processing unit) manager allocates the CPU to the processes. A variety of CPU scheduling schemes (time slicing, interrupt driven) have been employed in the operating systems. The file managers manage the data resources in the system. This normally includes catalogs, file sharing, etc. The I/O (input/output) managers steward all the local and remote I/O activities. The network communication facilities may be included in some operating systems as I/O facilities.

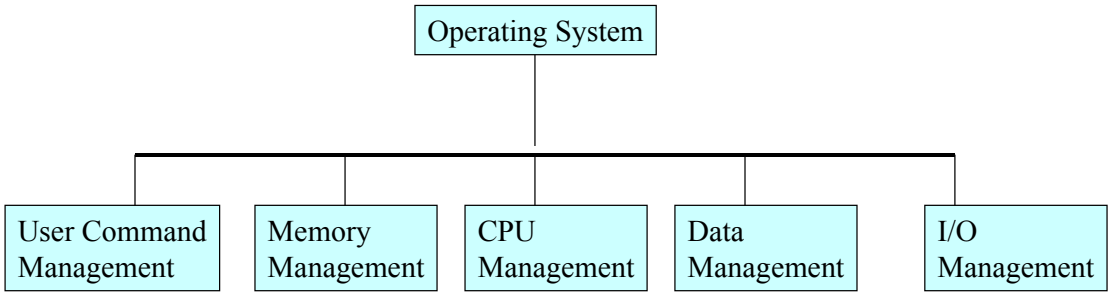

Figure 4-10: Centralized Operating System - Functional View

Examples of PC and Workstation Operating Systems

Microsoft's Windows 98 is a 32-bit operating system that can address data in 32-bit chunks and run programs that take up more than 640 kilobytes of memory. It features multitasking, multithreading, and powerful networking capabilities, including the capability to integrate fax, e-mail, and scheduling programs. Windows 95 was an earlier version of this operating system. It also has support for additional hardware technologies such as MMX, digital video disk (DVD), videoconferencing cameras, scanners, TV tuner-adapter cards, and joysticks. It also includes a group collaboration tool called NetMeeting and FrontPage Express, a tool for creating and storing Web pages.

Windows Millennium Edition (Windows Me) is an improved version of Windows 98 and features tools to let users edit video recordings and put them up on the Web and tools to simplify home networking of two or more PCs. A media player bundled with Windows Me can record, store, and play CDs, digital music downloaded from the Internet, and videos. Windows Me users can also import, store, and share photos. It also has improved capabilities for safeguarding critical files.

Windows 2000 is another 32-bit operating system with features especially for applications in large networked organizations. Windows 2000 is used as an operating system for high-performance desktop and laptop computers and network servers. There are two basic versions of Windows 2000—a Professional version for users of stand-alone or client desktop and laptop computers and several server versions designed to run on network servers and provide network management functions, including tools for creating and operating Web sites and other Internet services. Windows 2000 shares the same graphical user interface as the other Windows operating systems, but it has more powerful networking, multitasking, and memory-management capabilities. Windows 2000 can support software written for Windows and it can provide mainframe-like computing power for new applications with massive memory and file requirements. It can even support multiprocessing with multiple CPUs.

Windows XP (for eXPerience) combines strong features of Windows 2000 and Windows 98/Me. The Windows XP Home Edition is for home users and the Windows XP Professional Edition targets mobile and business users.

Windows CE is a scaled down version of Windows to run on small handheld computers, personal digital assistants, or wireless communication devices such as pagers and cellular phones. It is a portable and compact operating system requiring very little memory. Information appliances and consumer devices can use this operating system to share information with Windows-based PCs and to connect to the Internet.

Mac OS, the operating system for the Apple Macintosh computer, provides multitasking, powerful multimedia and networking capabilities, and a mouse-driven graphical user interface. New features of this operating system allow users to connect to, explore, and publish on the Internet and World Wide Web. It also allows users to use Java software and load different language fonts in Chinese, Japanese, Korean, Indian, Hebrew, and Arabic for use in Web browser software. Mac OS X, a newer Apple operating system, has a Unix-based foundation for additional features of reliability, graphics, and open-source features.

Unix is an interactive, multiuser, multitasking operating system developed by Bell Laboratories in 1969 to help scientific researchers share data. Unix was developed to network various machines together with built-in support for communications and networking. Unix is commonly used on workstations and minicomputers from Sun, HP, and IBM. Unix provides the reliability and scalability for running large systems on high-end machines. Unix can run on many different kinds of computers because it is written in C. In addition, application programs that run under Unix can be ported from one computer to run on a different computer with little modification. Unix is powerful but very complex, with several intricate commands with GUI support. Vendors have developed different versions of Unix that are incompatible, thereby limiting software portability.

Linux is a Unix-like operating system that runs on Intel, Motorola, Mips, and other processors. Linux is a competitor to Windows operating systems and can be downloaded from the Internet free of charge or purchased for a small fee from companies that provide additional tools for the software. The source code for Linux is available along with the operating system software so that it can be modified by software developers to fit their particular needs. Linux is an example of open-source software, which provides all computer users with free access to its source code so that they can modify the code to fix errors or to make improvements. Open-source software such as Linux is not owned by any company or individual. Because it is free and reliable, it has become popular during the past few years among sophisticated computer users and businesses as an alternative to Unix and Windows 2000. Major application software vendors are starting to provide versions that can run on Linux.

4.4.2 Local Systems Software

The local software in a distributed environment provides access and manipulation of local data and processes located on networked machines. Examples of the local software are:

- Database managers
- Transaction managers
- File managers
- Print managers
- Language translators
- Utility programs

Database managers, also known as database management systems (DBMSs), provide access to databases for on-line and batch users. In a typical database environment, different users can view, access and manipulate the data in a database. A DBMS is designed to a) manage logical views of data so that different users can access and manipulate the data without having to know the physical representation of data, b) manage concurrent access to data by multiple users, enforcing logical isolation of transactions, c) enforce security to allow access to authorized users only, and provide integrity controls and backup/recovery of a database. Relational database managers such as DB2, Oracle, Sybase, and/or Informix are typically used in many contemporary applications. Older systems use hierarchical database managers such as IMS. Object oriented databases are still in their infancy (see the tutorial on "Databases and SQL" in the Tutorials Module). An important issue in EB environments is to access remotely located databases from a variety of customer devices.

Transaction managers (TMs), also known as transaction processing monitor (TP monitor), monitor the execution of transactions (sequence of statements that must be executed as a unit). TMs specialize in managing transactions from their point of origin to their termination (planned or unplanned). Some TM facilities are integrated with the DBMS facilities to allow database queries from different transactions to access/update one or several data items (IBM's IMS DB/DC is an example). However, some TMs only specialize in handling transactions only (CICS, Tuxedo and Encina are examples). The key issue in distributed environments is how to extend the scope of local transaction management to managing the execution of transactions across multiple sites.

File managers are responsible for providing access and manipulation of text documents, diagrams, charts, images, and indexed files. A very wide range of file managers have been developed since the 1960s. An important issue in distributed environments is to provide access to files that are dispersed around a network. This issue has been addressed by middleware such as the SUN Network File Server (NFS).

Print managers are responsible for printing operations. Obviously, different types of print managers are available for different types of print devices. Almost all LANs at present provide access to print managers on the LAN.

Language translators (compilers) translate high-level language programs written in programming languages such as Java, C, C++, COBOL, FORTRAN, or C into machine language that the computer can execute. The program in the high-level language before translation into machine language is called source code. A compiler translates source code into machine code called object code. Just before execution by the computer, the object code modules are joined with other object code modules in a process called linkage editing. The resulting load module is what is actually executed by the computer. Figure 4-11 illustrates the language translation process. Some programming languages such as BASIC do not use a compiler but an interpreter, which translates each source code statement one at a time into machine code during execution and executes it. Interpreter languages are very slow to execute because they are translated one statement at a time. Language translators also include assemblers that are similar to compilers, but are used to translate only assembly language (low level programs) into machine code.

Utility Programs. Local system software includes utility programs for routine, repetitive tasks, such as copying, clearing primary storage, checking for viruses, calendaring, calculators, clocks, or sorting. These utilities perform such functions as setting up new files, deleting old files, or formatting diskettes. Utility programs are prewritten programs that are stored so that they can be shared by all users of a computer system and can be used rapidly in many different information system applications when requested. Examples of utility programs are Norton Utilities, and Microsoft Accessories that consist of Notepad, Clock, and a Calculator, among other things.

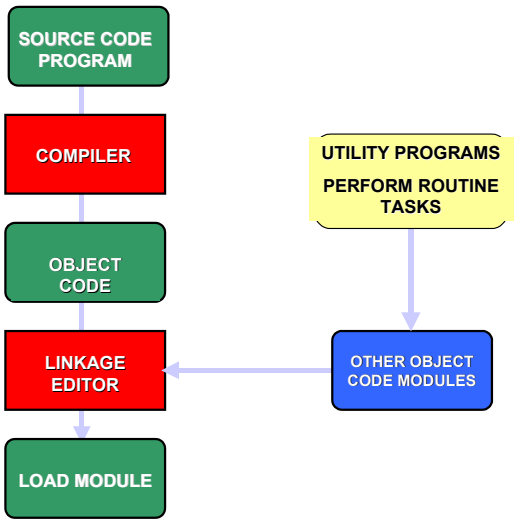

Figure 4-11: Language Translation Process

Time to Take a Break

- ✓ Network Overview
- ✓ Computing Platforms Overview
- Middleware Services
- 3G Distributed Computing Platforms

Suggested Review Questions Before Proceeding

- What is an operating system and what functions does it perform. List names of two operating systems.
- What is local system software and what does it do? List some examples of local system software that you are familiar with.

4.5 General Purpose Middleware

4.5.1 Overview

Middleware is a crucial component of modern IT infrastructure. Basically, middleware is a set of common business-unaware services that enable applications and end users to interact with each other across a network. In essence, middleware is the software that resides above the network and below the business-aware application software (see Figure 4-12). **General purpose middleware** is used to support general distributed applications such as web and distributed object applications and is not restricted to specialized applications such as e-commerce and mobile apps. A common example of such middleware is email because it provides business unaware services that reside above networks and interconnect users (in several cases applications also). Module "General Purpose Middleware" of this book discuss this type of middleware in great detail. Here is a brief overview.

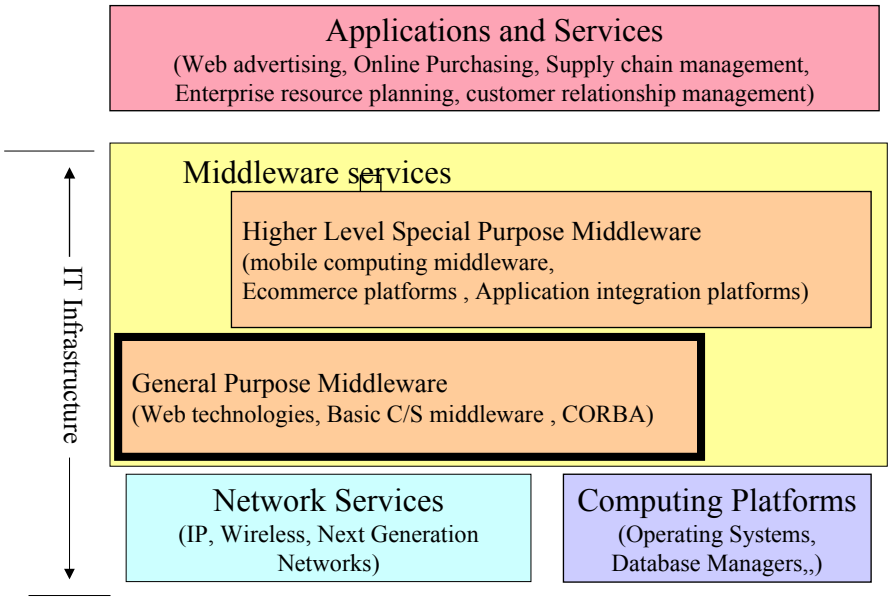

Figure 4-12: General Purpose Middleware as part of IT Infrastructure

4.5.2 Basic Client/Server Middleware

We will use the following definition in this book:

> **Definition of Middleware:** Middleware is a set of common business-unaware services that enable applications and end users to interact with each other across a network. In essence, middleware is the software that resides above the network and below the business-aware application software and denote a reusable, expandable set of services and functions that benefit many applications in a networked environment.

According to this definition, the key ingredients of middleware are (see Figure 4-13):

- It provides common business-unaware services
- It enables applications and end users to interact with each other across a network
- It resides above the network and below the business-aware application software

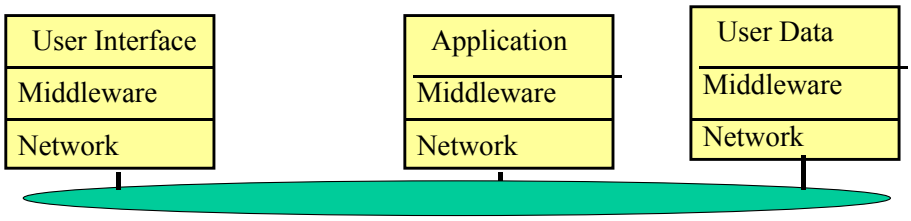

Figure 4-13: Middleware Overview

Middleware is business-unaware (i.e., it does not have any business logic) and is available as a common set of routines (see Figure 4-14). The services provided by these routines are available to the applications through application programming interfaces (APIs) and to the human users through commands and/or graphical user interfaces (GUIs). The commonality implies that these routines are available to multiple applications and users. Ideally, middleware should be transparent to end-users but necessary -- the end-users should be unaware when it is there and aware only when it is not. According to our definition, the following software qualifies as middleware (if you do not know about these, do not worry; the rest of the book is filled with information about many of these software packages):

- Email (it is business unaware and interconnects users at different sites)
- Terminal emulators and file transfer packages (these business unaware services connect users to resources)
- Web browsers (they are business unaware and support user access to resources on the Internet. As a matter of fact, as we will see in a later chapter, the World Wide Web is a collection of middleware services)
- Database drivers and gateways such as ODBC drivers and Sybase Omni Server (they provide access to remotely located databases)
- Object Management Group's CORBA (Common Object Request Broker Architecture) because it provides services for distributed object applications.

Here is a list of software that do not qualify as middleware, according to our definition:

- Operating systems such as Unix (it operates only on local resources)
- Airline reservation system (it is business aware)
- Network routers (they do not reside above the network -- they are part of network services)

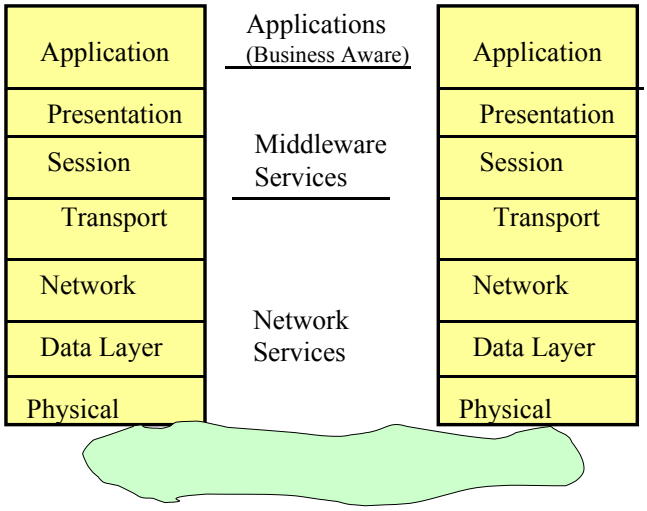

Figure 4-14: Middleware - A Look In Terms of the OSI Layers

Client/server (C/S) architecture provides the fundamental framework that allows many technologies to plug in for the applications of 1990s and beyond. Commercially available middleware services enable C/S interactions. Initial implementations of client/server architecture were based on the "two-tiered" architectures shown in Figure 4-15 (these architectural configurations The first architecture (Figure 4-15b) are used in many presentation intensive applications (e.g., Web applications) and to provide a "face lift" to legacy applications by building a GUI interface that invokes the older text-based user interfaces of legacy applications. Figure 4-15c represents the distributed application program architecture in which the application programs are split between the client and server machines and they communicate with each other through the remote procedure call (RPC) middleware. Figure 4-15d represents the remote data architecture in which the remote data is typically stored in an "SQL server" and is accessed through ad hoc SQL statements sent over the network. Figure 4-15e represents the case where the data exists at client as well as server machines (distributed data architecture).

Although a given C/S application can be architected in any of these configurations, the remote data and distributed program configurations are quite popular at present. The remote data configuration at present is very popular for departmental applications and is heavily supported by numerous database vendors (as a matter of fact this configuration is used to represent typical two-tiered architectures that rely on remote SQL). Most data warehouses also use a remote data configuration because the data warehouse tools can reside on user workstations and issue remote SQL calls to the data warehouse. However, the distributed programs configuration is very useful for enterprise-wide applications because the application programs on both sides can exchange information through messages.

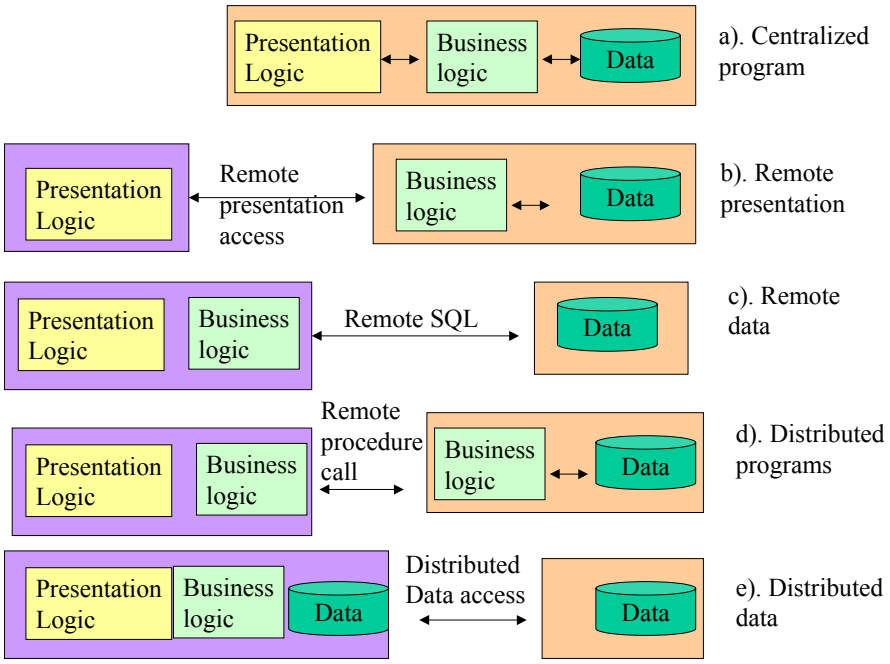

Figure 4-15: Traditional Client/Server Architectures

4.5.3 Web and XML

4.5.3.1 Web Overview

World Wide Web (WWW) provides a GUI access to the widespread Internet resources. Technically speaking, WWW is a collection of middleware that operates on top of IP networks (i.e., the Internet). Figure 4-16 shows this layered view. The purpose of the WWW middleware is to support the growing number of users and applications ranging from entertainment to corporate information systems.

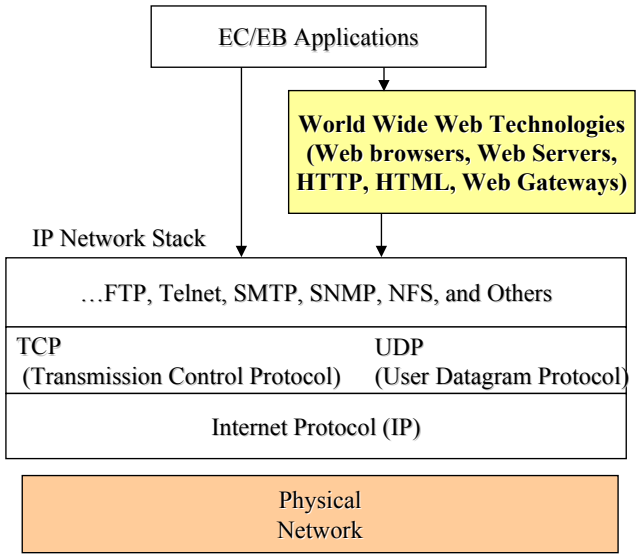

Figure 4-16: Technical View of Internet and World Wide Web

Like many other (successful) Internet technologies, the WWW middleware is based on a few simple concepts and technologies such as the following (see: Figure 4-17):

- Web servers
- Web browsers
- Uniform Resource Locator (URL)
- Hypertext Transfer protocol (HTTP)
- Hypertext Markup Language (HTML)
- Web navigation and search tools
- Gateways to non-Web resources

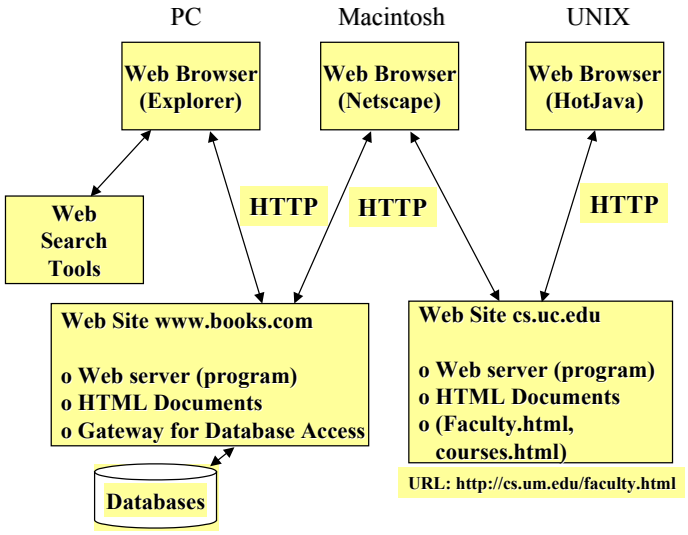

Figure 4-17: Conceptual View of World Wide Web

Let us briefly review these components and show how they tie with each other through an example.

Web sites provide the content that is accessed by Web users. Web sites are populated and in many cases managed by the content providers. For example, Web sites provide the commercial presence for each of the content providers doing business over the Internet. Conceptually, a Web site is a catalog of information for

each content provider over the Web. In reality, a Web site consists of three types of components: a Web server (a program), content files ("Web pages") and/or gateways (programs that access non-Web content). A Web server is a program (technically a server process) that receives calls from Web clients and retrieves Web pages and/or receives information from gateways (we will discuss gateways later). Once again, a Web user views a Web site as a collection of files on a computer, usually a UNIX or Windows NT machine. In many cases, a machine is dedicated/designated as a Web site on which Web accessible contents are stored. As a matter of convention, the entry point to a Web site is a "home page" which advertises the company business. Very much like storefront signs in a shopping mall, the home pages include company logo, fancy artwork for attention, special deals, overviews, pointers to additional information, etc. The large number of Web sites containing a wide range of information that can be navigated and searched transparently by Web users is the main strength of WWW. Figure 4-17 shows two Web sites -- one for a shoe shop (www.shoes.com) and the other for a computer science department for a university (cs.um.edu).

Web browsers are the clients that typically use graphical user interfaces to wander through the Web sites. The first GUI browser, Mosaic, was developed at the National Center for Supercomputer Applications at the University of Illinois. Mosaic runs on PC Windows, Macintosh, UNIX and Xterminals. At present, Web browsers are commercially available from Netscape, Microsoft and many other software/freeware providers. These Web browsers provide an intuitive view of information where hyperlinks (links to other text information) appear as underlined items or highlighted text/images. If a user points and clicks on the highlighted text/images, then the Web browser uses HTTP to fetch the requested document from an appropriate Web site. Web browsers are designed to display information prepared in a markup language, known as HTML. We will discuss HTTP and HTML later. Three different browsers are shown in Figure 4-17. Even though these are different browsers residing on different machines, they all use the same protocol (HTTP) to communicate with the Web servers (HTTP compliance is a basic requirement for Web browsers).

Most browsers at present are relatively dumb (i.e., they just pass user requests to Web servers and display the results). However, this is changing very quickly because of Java, a programming language developed by Sun Microsystems. Java programs, known as Java applets, can run on Java compatible browsers. This is creating many interesting possibilities where Java applets are downloaded to the Java enabled browsers where they run producing graphs/charts, invoking multimedia applications, and accessing remote databases.

Uniform Resource Locator (URL) is the basis for locating resources in WWW. A URL consists of a string of characters that uniquely identifies a resource. A user can connect to resources by typing the URL in a browser window or by clicking on a hyperlink that implicitly invokes a URL. Perhaps the best way to explain URLs is through an example. Let us look at the URL "http://cs.um.edu/faculty.html" shown in Figure 4-17. The "http" in the URL tells the server that an HTTP request is being initiated (if you substitute http with ftp, then an FTP session is initiated). The "cs.um.edu" is the name of the machine running the Web server (this is actually the domain name used by the Internet to locate machines on the Internet). The "/faculty.html" is the name of a file on the machine cs.um.edu. The "html" suffix indicates that this is an HTML file. When this URL is clicked or typed, the browser inititiates a connection to the "cs.um.edu" machine and initiates a "Get" request for the "faculty.html" file. Depending on the type of browser you are using, you can see these requests flying around in an appropriate window spot. Eventually, this document is fetched, transferred to and displayed at the Web browser. You can access any information through the Web by issuing a URL (directly or indirectly). As we will see later, the Web search tools basically return a bunch of URLs in response to a search query. The general format of URL is:

protocol://host:port/path

where

- protocol represents the protocol to retrieve or send information. Examples of valid protocols are HTTP, FTP, Telnet, Gopher, and NNTP (Network News Transfer Protocol).
- host is the computer host on which the resource resides.
- port is an optional port number (this is not needed unless you want to override the HTTP default port, port 80).
- path is an identification, typically a file name, on the computer host.

Hypertext Markup Language (HTML) is an easy to use language that tags the text files for display at Web browsers. HTML also helps in creation of hypertext links, usually called hyperlinks, that provide a path from one document to another. The hyperlinks contain URLs for the needed resources. The main purpose of HTML is to allow users to flip through Web documents in a manner similar to flipping through a book, magazine or a catalog. The Web site "cs.um.edu" shown in Figure 4-17 contains two HTML documents: "faculty.html" and "courses.html". HTML documents can imbed text, images, audio, and video.

Hypertext Transfer Protocol (HTTP) is an application-level protocol designed for Web users. It is intended for collaborative, distributed, hypermedia information systems. HTTP uses an extremely simple request/response model that establishes connection with the Web server specified in the URL, retrieves the needed document, and closes the connection. Once the document has been transferred to your Web browser, then the browser takes over. Keep in mind that every time you click on a hyperlink, you are initiating an HTTP session to transfer the needed information to your browser. The Web users shown in Figure 4-17 access the information stored in the two servers by using the HTTP protocol.

Web navigation and search services are used to search and surf the vast resources available over the "cyberspace". The term cyberspace, as stated previously, was first introduced through a science fiction book by [Gibson 1984] but currently refers to the computer-mediated experiences for visualization, communication, and browsel/decision support. The general search paradigm used is that each search service contains an index of information available on Web sites. This index is almost always created and updated by "spiders" that crawl around the Web sites chasing hyperlinks for different pieces of information. Search engines support key-word and/or subject-oriented browsing through the index. The result of this browsing is a "hit list" of hyperlinks (URLs) that the user can click on to access the needed information. For example, the Web users in Figure 4-17 can issue a keyword search, say by using a search service for shoe stores in Chicago. This will return a hit list of potential shoe stores that are Web content providers. You then point and click till you find a shoe store of your choice. Many search services are currently available on the Web. Examples are Yahoo, Google, Lycos and Alta Vista. At present, many of these tools are being integrated with Web pages and Web browsers. For example, the Netscape Browser automatically invokes the Netscape home page that displays search tools that you can invoke by just pointing and clicking. It is beyond the scope of this book to describe the various Web navigation and search tools. Many books on Internet describe these search tools quite well.

Gateways to non-Web resources are used to bridge the gap between Web browsers and the corporate applications and databases. Web gateways are used for accessing information from heterogeneous data sources (e.g., relational databases, indexed files and legacy information sources) and can be used to handle almost anything that is not designed with an HTML interface. The basic issue is that the Web browsers can display HTML information. These gateways are used to access non-HTML information and convert it to HTML format for display at a Web browser. The gateway programs typically run on Web sites and are invoked by the Web servers. At present, Common Gateway Interface (CGI) is used frequently. "Relational gateways" that provide access to relational databases from Web browsers are an area of active work.

4.5.3.2 A Simple Example

Figure 4-18 illustrates how the Web components can be used for a department store "Clothes-XYZ". This store wants to advertise its products on the Web. (i.e., wants to be a Web content provider). The store first designates a machine, or buys services on a machine, called "clothes.com" as a Web site. It then creates an overview document "overview.html" that tells the potential customers of the product highlights (think of this as the first few pages of a catalog). In addition, several HTML documents on the Web site for different types of clothes (men.html, women. html, kids.html) are created with pictures of clothes, size information etc. (once again think of this as a catalog). We can assume that the overview page has hyperlinks to the other documents (as a matter of fact, it could have hyperlinks to other branches of Clothes-XYZ). In reality, design of the Web pages would require a richer, deeper tree structure design as well as sequential links for alphabetical and keyword searches needed to support the "flipping through" the catalog behavior.

Once HTML documents have been created on the Web server, then an Internet user can browse through them as if he/she is flipping through a catalog. The customers typically supply the URL, directly or indirectly, for the overview (http://clothes.com/overview.html) and then use the hyperlinks to look at different types of clothes. Experienced customers may directly go to the type of clothes needed (e.g., men may directly go to "men.html" document). As shown in Figure 4-18, the URL consists of three components: the protocol (http), the Web server name (clothes.com), and the needed document (overview.html). HTTP provides the transfer of information between the Web users (the clients) and the Web Servers.

At first, Clothes-XYZ is only using Web to store an electronic catalog. After a customer has browsed through the catalog and has selected an item, he/she calls the store and places an order. Let us say that Clothes-XYZ wants to be more forward looking and wants the customers to purchase the items over the Internet. In this case, a "Purchasing Gateway" software is developed and installed at the Web site. This gateway program gets into action when a user clicks on the "purchase" button on his screen. It prompts the user with a form (HTML supports forms) that the user fills out. The gateway program uses this form information to interact with a purchasing system that processes the purchase (see Figure 4-18). The purchasing system can be an existing system that is used for traditional purchasing. The role of the gateway is to provide a Web interface to the purchasing system.

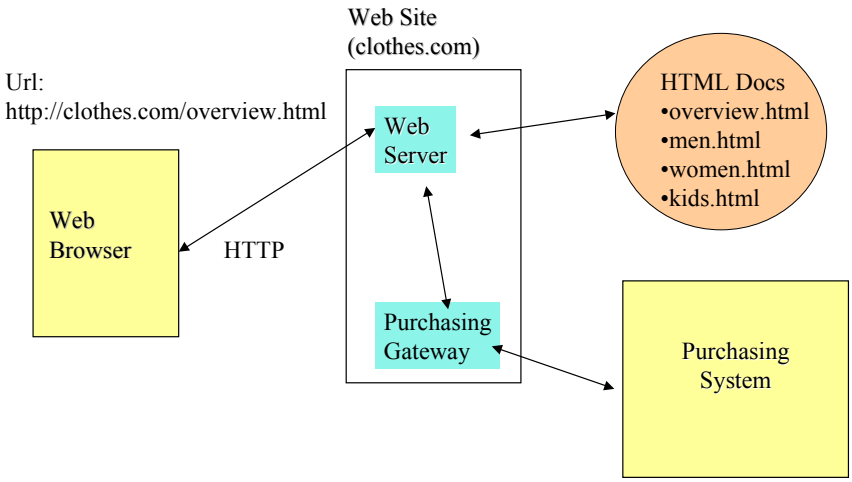

Figure 4-18: A Simple Web Example

4.5.3.3 XML Overview

XML is a markup language, similar to HTML, for documents containing structured information. The main limitation of HTML is that it only concentrates on presentation (i.e., headers, highlights, etc) . If you need to specify, for example, the structure of information (i.e., represent customer name and address), then HTML cannot help. In fact, XML is a simplified version of SGML (Standard Generalized Markup Language) – the first standardized markup language. Unfortunately, SGML is too overbearing to use in tandem with the Web. Thus XML is a simplified version of SGML and is similar to HTML (in fact, HTML and XML are both simplifications of SGML).

Before going into details, let us quickly show a very simple example of XML. The following statements represents an XML document that contains customer name and address:

```
<CUSTOMER>
        <NAME> Joe </NAME>
        <ADDRESS> NY </ADDRESS>
```

```
        <AGE> 33 </AGE>
    </CUSTOMER>
```

You can see the striking resemblance between HTML and XML, at least at this simple level (i.e., tags in the format <tag> that are terminated by </tag>). As we will see, you can develop XML documents that represent customers, orders, bills, airline schedules, TV programs, bank statements, catalogs, etc. by just creating new tags. XML is very popular at present with applications ranging from e-commerce to music.

The term "document" in XML-context refers not only to traditional documents, like articles and books , but also to other XML "data formats" such as e-commerce transactions, mathematical equations, and graphics. XML provides a facility to define tags and the structural relationships between them for documents. There is no preconceived semantics (i.e., meaning) associated with XML because there is no predefined tag set. All of the semantics of an XML document are either defined by the applications that process them or by stylesheets. The XML specification by W3C sets out a set of goals for XML that include ease of use, flexibility, and use over the Internet (see the sidebar "XML Goals"). A great deal of XML activity exists at present in various areas such as Web (new Web browsers support XML), electronic commerce (XML is being considered as a possible replacement for EDI), data management (e.g., XMI , XML Metadata Interchange, used to exchange models between vendor tools, and CWM, Common Warehouse Metadata, for Oracle and other data warehouses), and publishing (XML is often used in place of SGML because of its "lightness").

According to the W3C, XML will

- Enable internationalized media-independent electronic publishing
- Allow industries to define platform-independent protocols for the exchange of data, especially the data of electronic commerce
- Deliver information to user agents in a form that allows automatic processing after receipt
- Make it easier to develop software to handle specialized information distributed over the Web
- Make it easy for people to process data using inexpensive software
- Allow people to display information the way they want it, under style sheet control
- Make it easier to provide metadata -- data *about* information -- that will help people find information and help information producers and consumers find each other

The XML Activity (phase 1), was started by the W3C in June 1996. It culminated in the W3C XML 1.0 Specification (issued February 1998, revised Oct 2000). In the second phase, work proceeded in a number of working groups in parallel. In September 1999, W3C began the third phase, continuing the unfinished work from the second phase and introducing a Working Group on XML Query. Since 1996, the work of several W3C working groups and other standards/industrial bodies has resulted in a "family" of XML standards that include (see Figure 4-19):

- Document Type Declarations (DTD) to specify the set of rules for the structure of an XML document. DTDs are used to verify and validate XML documents and are thus central to XML-based e-business exchanges.
- XSL (eXtensible Stylesheet Language) to display information. XSL supports a basic premise of XML – separation of content from presentation. XSL is also useful to translate one XML description to another.
- XML query language to support querying of XML data.
- XML schema to specify the format (e.g., the character lengths) and relationships between XML elements.
- Other developments such as XML Link, XML signature, and XML Path.
- Variants of XML for different application areas. Examples are the Wireless Markup Language (WML), Voice Markup Language (VML), Mathematical Markup Languge (MathML), Chemical Markup Language (CML) and others.

Due to the popularity and growth of XML, W3C has started an XML Coordination Group to coordinate the workflow and dependencies between various working Groups. The Group also maintains liaison inside and outside the W3C and gathers and forwards requests for additional requirements to the appropriate WG(s).

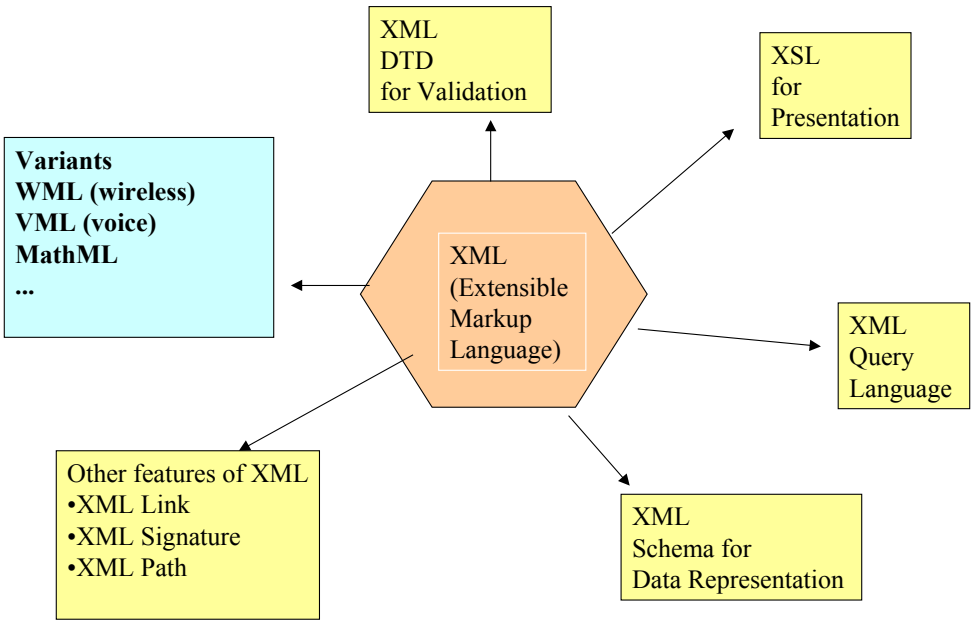

Figure 4-19: The XML Family

4.5.4 Distributed Objects for Enterprise Wide Applications

4.5.4.1 Overview

The trend at present is to extend the OO concepts to enterprise-wide distributed applications. Simply stated, distributed objects are objects that can be dispersed across the network and can be accessed by users/applications across the network. Conceptually, enterprise-wide applications are decomposed into objects that can roam around the network. An object on one machine can send messages to objects on other machines thus viewing the entire enterprise as a collection of objects. This concept naturally extends the notions of object frameworks, business objects, and component software to distributed systems. Distributed objects present a very powerful technology that has the potential of addressing many problems facing the IT community today (i.e., reuse, portability and interoperability). This is because applications can be constructed by using reusable components that encapsulate many internal details and can inter-operate across multiple networks and platforms.

Technically, distributed applications can be viewed as a collection of objects (user interfaces, databases, application modules, customers). Each object has its own attributes, and has some methods which define the behavior of the objects (e.g., an order can be viewed in terms of its data and the methods which create, delete and update the order object). Interactions between the components of an application can be modeled through "messages" which invoke appropriate methods. In particular, classes and inheritance are extremely useful in modelling distributed applications because these concepts lead to reuse and encapsulation - critical to managing the complexity of distributed systems. For example:

- A customer can be defined as a class from which other business classes that define different types of customers can inherit properties.
- An inventory can be defined as a class from which other properties of specific inventory items can be inherited.
- An entire legacy application can be viewed as an object (or a class) by using object wrappers that mediate between legacy systems and OO users (we will discuss object wrappers in later chapters).

Figure 4-20 shows a conceptual view of a distributed object model:

- Objects are data surrounded by code with properties such as inheritance, polymorphism, encapsulation, etc. Objects can be clients, servers, or both.
- Object brokers allow objects to dynamically find each other in a distributed environment and interact with each other over a network. Object brokers are the backbone of distributed object-oriented systems.
- Object services allow the users to create, name, move, copy, store, delete, restore, and manage objects.

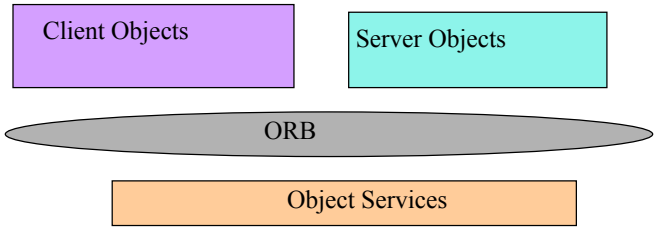

Figure 4-20: The Basic Distributed Objects Model

How do remote objects make their services (operations) available to their clients? This is done through two concepts: interfaces and Interface Definition Language (IDL). An interface represents the external view of an object and IDL is used to code and store this view in an interface repository. Interfaces and IDL provide the basic glue for distributed object computing. IDL is used not only to define new services provided by objects, but also to "wrap" existing and legacy systems so that they behave externally as objects. For example, a legacy application written in Cobol could behave as a server object as long as it has an IDL and it provides the operations defined by the IDL. Thus the "IDL-ized" programs run on top of an ORB without revealing their internal details (see Figure 4-21).

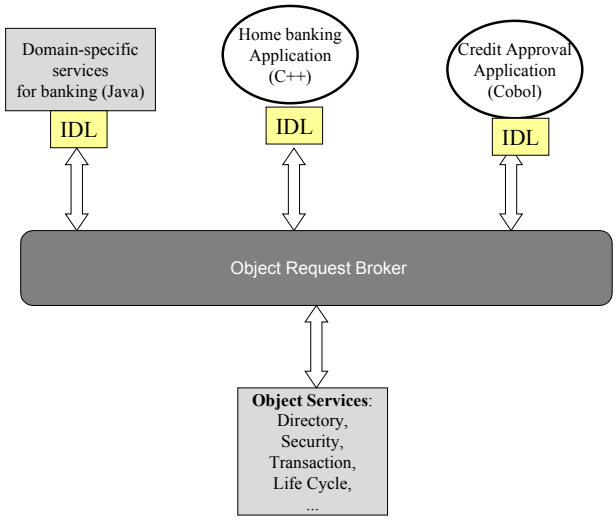

Figure 4-21: Interface Definition Language in Action

4.5.4.2 Component Software

Components are high-level "plug and play" software modules that perform a limited set of tasks within an application. Components are essentially small applications, also known as "applets", that are recognizable by the users (i.e., they do not perform internal programming tasks such as initialize internal memory locations). For example, Microsoft Draw is an applet within Microsoft applications. This particular component is high

level enough to perform end-user type functions (it draws boxes, arrows, circles, etc.). However, it is not a stand-alone application, it only works with other applications components.

Does everybody agree what a component is? Of course, not (that would be a miracle!). However, most agree that a) it is an independently delivered software and b) it has an interface. A component may not have all the features of an object (i.e., inheritance and polymorphism) but it may be constructed from objects ("Components are molecules, objects are atoms", [Frye 1997]). Components are very much like objects, however, the emphasis is on recognition by users (many objects are oriented towards programming tasks).

Why components are important? The reason is simple -- they can be used as plug and play to build complete applications. It is possible that components could finally realize the long term dream of many pundits, i.e., assemble software as you assemble cars instead of custom building each application.

At present, component-based software architectures are at the center of industrial activity. For example, the Sun J2EE and Microsoft Dot Net architectures both support component-based software.

4.5.4.3 CORBA and ActiveX -- Middleware For Distributed Objects

Support of distributed object-based applications require special purpose middleware that allows remotely located objects to communicate with each other. Examples of middleware for distributed objects includes Object Management Group's (OMG's) CORBA (Common Object Request Broker Architecture) and Micrososft's ActiveX. Both of these middleware packages use the distributed object model based on the object request broker (ORB) that receives an object invocation and delivers the message to an appropriate remote object (see Figure 4-22). .

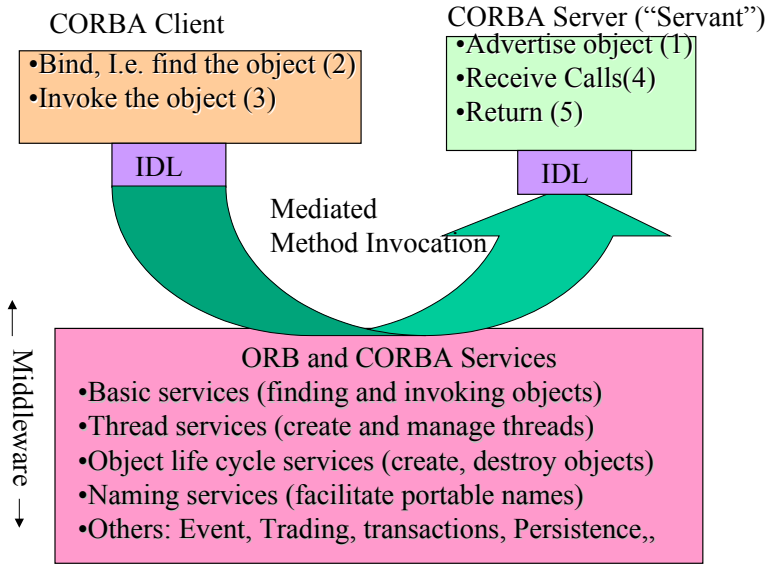

Figure 4-22: CORBA Conceptual View

CORBA was introduced in 1991 by OMG to go a step beyond OMA to specify the technology for interoperable distributed OO systems. CORBA specifications represent the ORB technology adopted by OMG and are published as OMG documents. The key concepts of CORBA are (see Figure 4-22):

- CORBA essentially specifies the middleware services that will be used by the application objects.
- Any object (application) can be a client, server or both. For purpose of description, CORBA uses the C/S model where clients issue requests to objects (service providers).
- Any interaction between objects is through requests. The information associated with a request is an operation to be performed, a target object, zero or more parameters, etc.
- CORBA supports static as well as dynamic binding. Static binding is used to identify objects at compile time while dynamic binding between objects uses run time identification of objects and parameters.

- An interface represents contracts between client and server applications. A typical interface definition shows the parameters being passed and a unique interface identifier. An Interface Definition Language (IDL) has been defined specifically for CORBA. Program stubs and skeletons are produced as part of the IDL compiling.
- CORBA objects do not know the underlying implementation details - an object adapter maps generic model to implementation and is the primary way that an object implementation accesses services provided by the ORB.

ActiveX was introduced by Microsoft in March 1996 as its main strategy for distributed objects and Web. Microsoft is positioning ActiveX as a complete environment for components and distributed objects. Almost everything coming out of Microsoft at the time of this writing is being based on ActiveX. Although ActiveX provides many capabilities, from a distributed objects point of view, the following features are significant (see Figure 2.11):

- All ActiveX components communicate with each other by using DCOM. So a Java applet (an ActiveX component) can call a remotely located Microsoft Word document (another ActiveX component) over DCOM. DCOM is the ORB in the ActiveX environment.
- The Web browser can behave as a container. For example, the Microsoft Internet Explorer can contain components such as Word documents, Java applets, C code, and Excell spreadsheets.
- Web technologies (browsers, HTML pages Java applets) can be intermixed with desktop tools (spreadsheets, word processors) for distributed applications.
- Serve facilities such as SQL servers and legacy access gateways can be invoked from ActiveX clients.

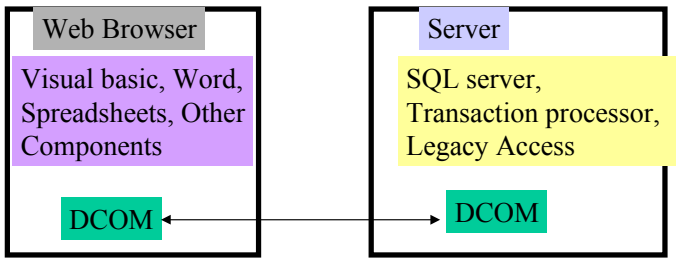

Figure 4-23: ActiveX Conceptual View

4.5.4.4 Enterprise JavaBeans™ and CORBA Components

Enterprise JavaBeans™ (EJB) technology defines a model for the development and deployment of reusable Java server components. Components are pre-developed pieces of application code that can be assembled into working application systems. Java technology currently has a component model called JavaBeans, which supports reusable development components. The EJB architecture logically extends the JavaBeans component model to support server components.

The CORBA Component architecture consists of several interlocking conceptual pieces that enable a complete distributed enterprise server computing architecture. These include an Abstract Component Model, a Packaging and Deployment Model, a Container Model, a mapping to EJB and an Integration Model for Persistence and Transactions.

In order to build enterprise-scale applications, developers need to integrate their business logic in a distributed architecture which includes (at a minimum) transactions, persistence, events and naming. They also need to be able to tune their application and have flexible deployment models. Modeling, designing, and implementing such applications is quite complex. CORBA's flexibility gives the developer a myriad of choices, and requires a vast number of details to be specified. The complexity is simply too high to be able to do so efficiently and quickly.

4.5.5 Distributed Transaction Support

Business transactions are at the core of electronic commerce. Examples of typical EC transactions are purchasing, claim processing, and billing/payment. For business to business activities in ECs, the importance of supporting highly reliable and secure business transactions is quite obvious. Formally, a *transaction* is a collection of operations on a database which has the so-called ACID properties:

- **Atomicity**: All of the operations in the transaction must take place, or none must take place. In practice, if any of the elementary steps that are part of the transaction action fails, then all the steps must be undone;
- **Consistency**: The result of performing all the operations in the transaction is to take the database from one consistent state to another consistent state;
- **Isolation**: Other users of the database are isolated from any intermediate states of the transaction, i.e., they may see the state of the database before the transaction begins or after it completed, but not any state in the middle;
- **Durability:** Once all the actions in the collection have completed, the effects endure even in the event of system crashes.

EC is a mixture of decision support and transaction processing activities. Normally, only a portion of the core EC activities are transactional. Two main types of EC transactions are relevant:

- EC transactions between trusted business partners (e.g., suppliers and corporations that enter business agreements and contracts to buy and sell products). These transactions typically are large in volume (large amounts of money and goods), introduce medium traffic, and require rigorous security. These transactions are perfect candidates for Extranets.
- EC transactions between suppliers and the general public (e.g., Internet shopping malls)

Transactional support is implemented differently in different types of systems. For centralized mainframe systems, on-line transaction processing (OLTP), has been built for ACID transactions and has been a backbone of commercial data processing since the early 1970s. Mainframe-based transaction managers (TMs) such as CICS and IMS-DC/IMS-TM have matured over the years to provide high performance and reliable services The situation is dramatically different in distributed environments that characterize EC:

- TP-Less, i.e., do not use any transaction management facilities;
- TP-Lite, i.e., use database procedures to handle updates;
- TP-Heavy, i.e., use a distributed transaction manager to handle updates.

Which of these approaches works depends on the type of EC activities being considered. It appears that each one of these approaches have certain pluses and minuses for EC. The following questions should be asked before deciding on the approach:

- In what format is the data stored (databases, flat files)? If the data is stored in multiple databases and flat files, then TP-Lite is not suitable (database procedures only work in RDBMS environments);
- How many SQL servers does the data reside on? If the application needs to update and commit data that is stored on multiple servers, then TP-Heavy should be used (database procedures cannot participate with other database procedures in a distributed transaction).
- What is the requirement for data synchronization? If the data synchronization interval is periodic, then a TP-Lite solution combined with a data replication server may be useful to handle updates against replicated data.
- What are the requirements for performance and load balancing?

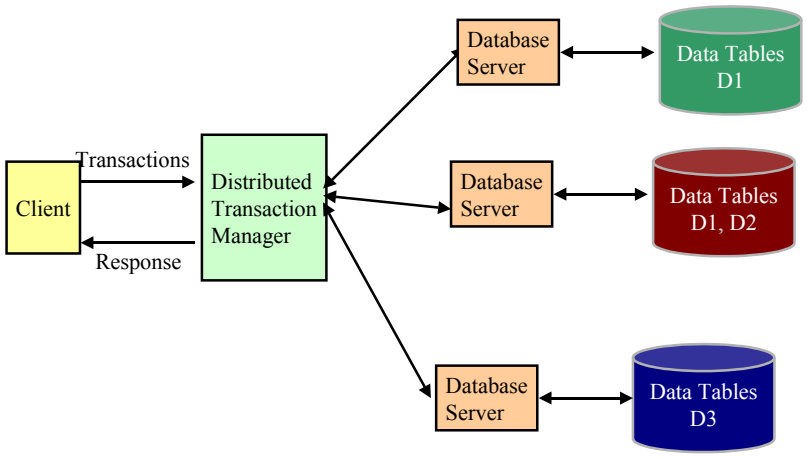

Figure 4-24: Models of Distributed Transactions

TP-Less works well when you do not need any transaction processing capabilities. TP-Lite solutions with database procedures are much faster, on the surface, than the TP-Heavy solutions that require synchronization between sites. But TP-Heavy solutions provide many sophisticated procedures for dynamic load balancing, priority scheduling, process restarts, and pre-started servers that are especially useful for large scale production environment. These features are the main strength of TP-Heavy products because many of these products have been used over the years to handle thousands of transactions in production OLTP (on-line transaction processing) environments. Figure 4-24 shows a basic model of a distributed transaction processor that may use TP-Heavy in distributed environments. The protocol used in this case is two phase commit.

While the debate between the TP-Lite and TP-Heavy proponents continues, most EC projects are completely ignoring this whole issue by focusing primarily on EDI and in some sense re-inventing the wheel. EDI, at best, is a TP Less approach. Some EC applications are being deployed by using TP-Lite while large mission critical EC applications, if any, use TP-Heavy only at back-end mainframe systems. In the meantime, it seems that many small EC applications are quite happy with TP-Less.

.

Suggestion: This may be a good time to take a break.

4.6 Higher Level Middleware and Application Support Platforms

This type of middleware provides value added features needed by the mobile e-business applications. Examples of these services include middleware to support wireless and mobile computing devices, catalogs services, EDI, enterprise integration software, emarket suport, and XML at C2B as well as B2B levels (see Figure 4-25). This middleware is discussed extensively in the Platforms Module of this book. This section gives an overview of the key ideas.

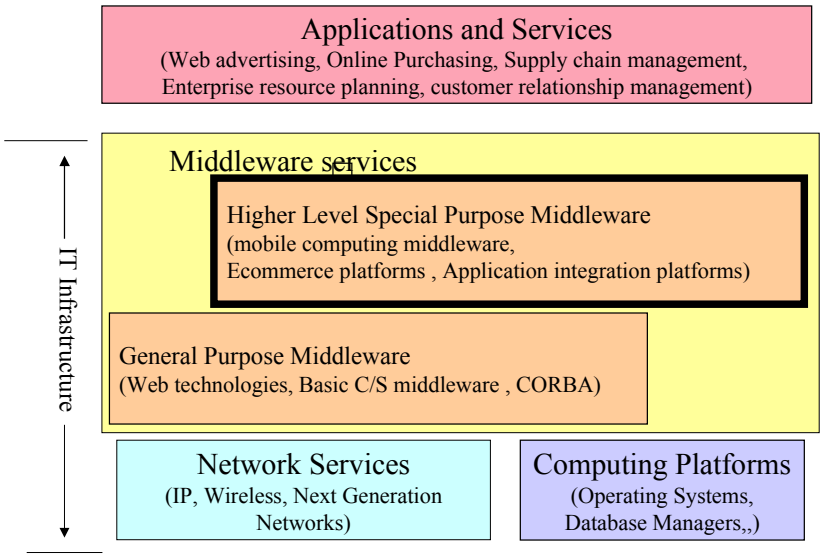

Figure 4-25: Higher Level Middleware as part of IT Infrastructure

4.6.1 Middleware for Mobile Computing and Wireless Networks

4.6.1.1 Overview

Middleware for mobile computing applications appears to follow two main approaches:

- Information hiding;
- Information providing.

Information hiding wireless middleware attempts to smooth over the mobile computing issues, as much as possible, so that the same applications can run on wired as well as wireless networks. This is a conceptually desirable goal because application developers should not have to be aware of the underlying network characteristics. This goal is met by the wireless middleware typically through specialized APIs that provide functionality such as the following:

- Monitoring to determine whether a mobile device is powered on and within the range of wireless WAN coverage. The middleware provides this information to the applications as a status indicator and/or end users as a screen icon.
- Optimization techniques to improve throughput by using a combination of data compression, intelligent restarts (i.e., restart at the point of disconnection and not from the beginning of session), pre-fetching, and data caching (keep data at local sites in case it needs to be accessed again) techniques. This middleware may also provide bundling of smaller packets to larger packets to save communication costs (most wireless systems charge by number of packets sent, thus reducing end user costs).
- Monitor and limit the number of simultaneous wireless connections to be maintained by the applications.
- Provide agents that reside on the wireless servers. Agents can perform a variety of services such as bandwidth optimization, asynchronous clients notification when a special situation arises (e.g., a new customer arrives), automatic session setup, wakeup of "dormant" users, and ad hoc queries handling.

This type of middleware is suitable for stand-alone and simple C/S applications. As an example of "information hiding" wireless middleware, IBM's ARTour allows TCP/IP applications to run, unchanged, over wireless networks. ARTour runs on top of CDPD and several other services. While it is theoretically possible to run TCP/IP applications directly over CDPD, you do need CDPD running everywhere to support such applications (CDPD may not be provided by several cellular operators). The advantage of using

packages such as ARTour is that they can run on top of CDPD and non-CDPD networks. Middleware packages like ARTour also use compression and reduce the IP header information (IP headers are 40 character long) to maximize throughput.

Information providing wireless middleware provides as much information about the underlying environment to the application as possible. In fact, this class of middleware exploits the network quality of service, cost, and location information for optimum performance. In particular:

- Network QoS can be used to modify application behavior. For example, the MOST project at Lancaster University uses a database application that determines the response to be sent to the user based on number of matches and the network QoS. Thus, if the application has many matches (i.e., long response to a query) but the network is slow, then the application dynamically sends only few selected matches. Similarly, different compression techniques can be used for different network QoSs (i.e., use more compression for slower networks).
- Cost information can be used by the middleware to modify application behavior. For example, if a user is being charged per second, then messages can be batched up before transmission.
- Location information can be used by the middleware to direct application behavior. For example, location information can be used to route information to nearest printers or computers.
- End-system characteristics can also be exploited by the middleware. For example, specialized user-interfaces can be displayed for mobile users and the laptop can be put in a "doze" mode to conserve batteries while it is waiting for chunks of data remote sites.

The basic philosophy of this class of middleware is to detect changes in the mobile environments and supply the change information to the applications so that they can modify their behavior changes in the mobile environments. This type of middleware, currently not state of the market, is essential for advanced mobile applications.

4.6.1.2 The Wireless Application Protocol (WAP)

The WAP forum is an industry group aimed towards providing a set of protocols to enable the presentation and delivery of wireless information and telephony services on mobile phones and other wireless terminals. Two main constraints make this market to be different from the wireline market. Firstly, the wireless links are typically constrained by low bandwidth, high latency as well as high error rates. Secondly, the wireless devices are constrained due to limited CPU power, limited memory and battery life as well as the need for a simple user interface.

WAP specifications address these issues by using the existing standards where possible with or without modifications and also by developing new standards that are optimized for the wireless environment where needed. The WAP specification has been designed such that it is independent of both the air interface used as well as independent of any particular device.

The key elements of WAP specification include:

- A definition of the WAP Programming model shown in Figure 4-26. This is based heavily on the existing WWW programming model.

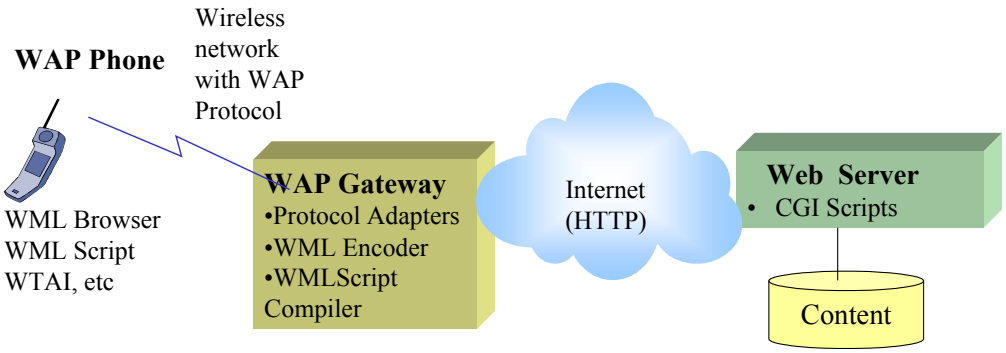

Figure 4-26. WAP architecture.

- A markup language called Wireless Markup Language (WML) similar to XML but optimized for the domain of wireless links and devices;
- Specification of a microbrowser in the wireless terminal. This is analogous to the standard Web browser;
- A protocol stack designed specifically for the wireless environment, as shown in Figure 4-27.

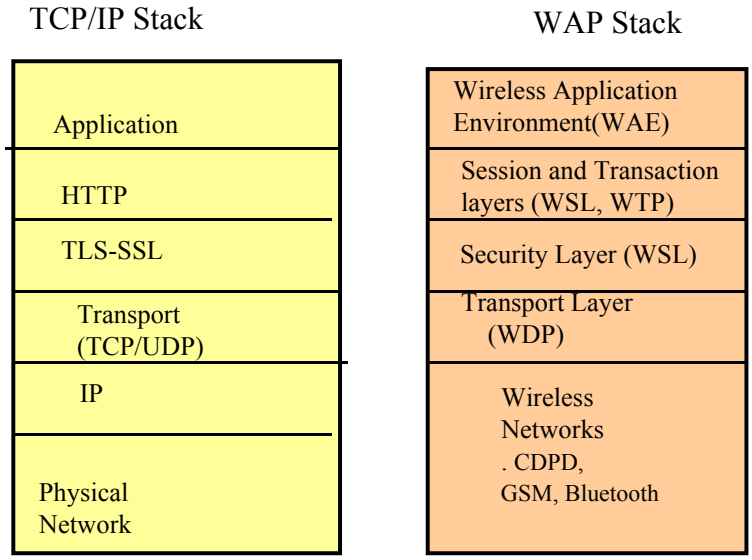

Figure 4-27. WAP stack and the Web

A framework to allow access to telephony services such as call control, messaging etc from within the WMLScript applets. This is specified under the Wireless Telephony Applications (WTA) specification.

Clearly, some of the benefits that follow from WAP is the development of a user interface appropriate for handheld devices as well as a specification tailored to the wireless environment.

4.6.2 Enterprise Application Integrators (EAIs) and Message Brokers

Simply stated, an EAI is a collection of technologies (middleware such as CORBA, adapters/gateways for protocol conversion, data transformers, transaction managers, and work/process flow systems) that allow diverse applications to talk to each other. For example, an EAI package can allow new EC applications written for Web users to seamlessly communicate with back-end mainframe-based ERP systems and databases. At their best, EAI systems hide all the complexity needed to enable interactions between applications that were developed at different times on different platforms, and using different technologies.

Thus EAI is not a new technology – rather, it is a design philosophy or architecture resulting from the combination of "well-known" technologies.

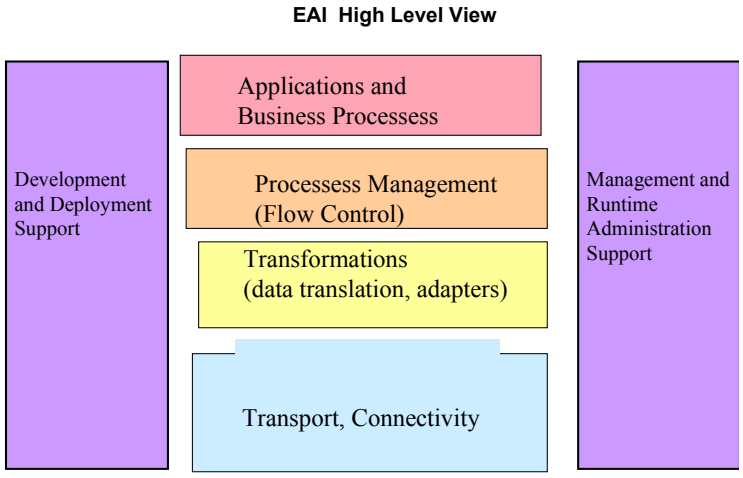

Figure 4-28. EAI

EAI systems are typically organized around a (often-logical) hub and spoke architecture. Applications are interconnected through the hub. For instance, below is a diagram from the Active EAI system. Basically, a hub provides a set of shared services that are used to support and mediate the interaction between the integrated applications. Simply organizing application integration around a hub is useful as it cuts down on the number of interfaces that need to be supported between the systems.

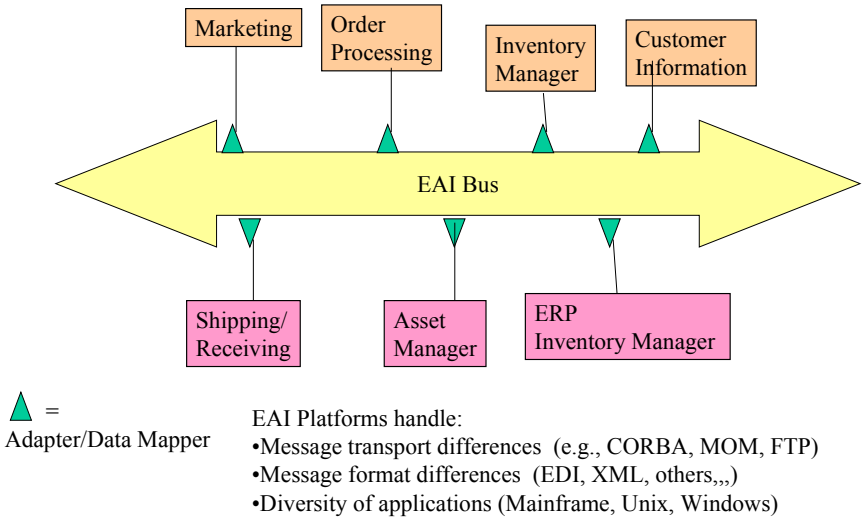

Figure 4-29. EAI Platforms in Action

Most EAI systems are based on **Message Brokers**. Message brokers are systems that combine three approaches to system integration:

- Hub based systems: Message brokers are organized around hubs rather than point-to-point. The hubs are typically *virtual* (to prevent single points of failure) and provide a set of services, e.g., transactional delivery of messages, routing, data mapping, system management, etc.;

- Asynchronous interactions: While often supplying other types of transactional interactions (such as two-phase commit) message brokers tend to rely heavily on asynchronous messaging;
- Messages (or events): Message brokers rest inter-application integration on messages or events, which are short lived objects or data structures which convey "descriptions" of events occurring in the distributed system (such as the acquisition of a new customer) to other entities in the architecture.

4.6.3 Middleware to Support E-commerce

Simply stated, electronic commerce (EC) is buying and selling over the network. At a very basic level, the following core EC functionality needs to be supported:

- Advertising
- Items Browsing, selection, items purchase cart management
- Purchasing
- Billing /invoicing
- Payments

The infrastructure supporting these functionalities is subject to stringent security, speed and reliability requirements. A wide range of IT infrastructure components are needed to support these and other EC activities. Let us briefly review the key players.

4.6.3.1 Billing and Electronic Payment Systems

Billing and payment systems are naturally an important part of EC. A variety of billing systems for users of EC over the Internet are evolving. Examples of such systems are Cybercash, GCTech, Verifone, Datex, Kenan System, and Telcordia Technologies' Interactive Billing System.

Electronic payment systems are central to EC given that on-line consumers must pay for products and services. In particular, payments and settlements must be resolved between all partners (customers, merchants, banks, brokers, etc.) quickly and smoothly otherwise the whole business chain is disrupted. On-line sellers need to decide how best to support payments for online purchases, what type of currency to use, and what type of electronic fund transfer (EFT) system to use. The payment is typically handled by a **Payment Server.** Current approaches to payment fall into the following broad categories:

- Retailing payments such as credit cards (e.g., VISA or MasterCard), charge cards (e.g., American Express), or private label cards (e.g., Sears cards)
- Banking and financial payments such as large scale or wholesale payments (e.g., bank-to-bank transfer), retail or small scale payments (e.g., cash, ATM cards, checks)
- Digital token-based systems that include electronic cash, electronic checks, and smart cards

The first two categories (retailing and banking systems) are not completely adequate for large scale EC -- they assume that the parties will be in physical presence and enough delays will be built into the system for frauds and overdrafts to be detected. Most of the current work in EC payments concentrates on different types of token-based systems such as e-cash, electronic checks, smart cards, and the like.

4.6.3.2 Catalog management systems.

The Internet enables universal connectivity, mass customization, and improved customer relationships for businesses. The universal connectivity provided by the Internet enables companies with nothing more than a virtual presence to offer their goods and services to millions of customers. The ability to tailor the selling of goods or services to each individual customer anywhere in the world is known as Mass Customization. The Internet provides a medium for greater flow of information between buyers and sellers so that the entire

shopping experience can be customized to meet the needs of the customer. Among the first to recognize and exploit the advantages of the Internet in order to reshape their respective markets were consumer-focused (business to consumer, B2C) companies like Amazon, Dell and eBay. The next wave of change will be exploited by business-focused companies, where the impact will quickly dwarf the impact of consumer-oriented markets (business to business, B2B). According to industry analyst Forrester Research, B2B electronic commerce will account for $1.3 trillion in 2003, compared to B2C electronic commerce which will reach $108 billion by 2003.

4.6.3.3 Electronic Data Interchange (EDI) and XML

The EDI standard specifies the encoding of messages to be used for transactions between business partners. The early practice was for most EDI applications to run on mainframes and to communicate via asynchronous or bisynchronous point-to-point network protocols on proprietary networks. The current trend is toward PC or workstation EDI applications communicating via store-and-forward message protocols over IP networks. A typical business interaction may involve a sequence of EDI messages flowing back and forth between the business partners, with the role of client and server reversing from time to time. EDI does not provide any mechanisms for control of the entire process flow. That depends on appropriate mechanisms operating internally within the domain of each of the business partners.

Electronic Data Interchange (EDI) required a unique and expensive effort to implement EDI-based solutions between *each pair* of companies. This one-to-one approach ignores universal connectivity, one of the most powerful principles of the Internet. The introduction of the eXtensible Markup Language (XML), allowed the easy creation of one-to-many links between businesses to become a reality. Enterprise software vendors are now rapidly supporting XML, and standards are being defined to further simplify the interchange of data using XML. As discussed previously, XML is a markup language, similar to HTML, for documents containing structured information. XML is being considered as a possible replacement for EDI because it can be used to represent information to be exchanged between traders.

Companies need a common language through which to exchange structured information to conduct EC. This responsibility was earlier passed on to EDI, but EDI is having to compete with XML at present. XML consists of text delimited by tags; so it is easily conveyed over the Internet. In XML the tags can define the meaning and structure of the information for order processing, invoices, billing, and payments. XML has been embraced enthusiastically by the major IT suppliers and user groups to specify information between trading partners. In particular, industry giants such as IBM, Microsoft, Sun, and Oracle all support XML and are developing major XML-based products and collaborations. XML is becoming the world standard platform for EC transactions. However, in any business application, XML itself is not the answer. It is only a standard foundation on which answers can be built.

4.6.3.4 Basic security services and SSL

A secured server will use Secure Sockets Layer (SSL) technology to provide a safe way to transmit sensitive information, such as credit card numbers, online banking, email messages, surveys and other personal information.

The SSL protocol provides data encryption, server authentication, message integrity, and optional client authentication for a TCP/IP connection, allowing for the secure transfer of sensitive information over the Internet. SSL consists of software installed in browsers and on servers and can be obtained by subscribing to a Secured Service Provider such as ssl.com or by obtaining a Server Certificate from ssl.com and installing it on an existing secured server.

All major browsers and servers today are "SSL capable". SSL technology was developed by Netscape Communications Corporation and has become the industry-standard method for protecting web communications.

Let us look at SSL briefly (details can wait). SSL uses public key encryption to provide security at the packet level. At the receiving end, SSL provides Server Authentication Message Integrity checks.SSL comes in two strengths, 40-bit and 128-bit, which refer to the length of the "session key" generated by every encrypted

transaction. The longer the key, the more difficult it is to break the encryption code. Most browsers support 40-bit SSL sessions, and the latest browsers, including Netscape Communicator 4.0, enable users to encrypt transactions in 128-bit sessions - trillions of times stronger than 40-bit sessions. Global companies that require international transactions over the web can use a global server certificates program to offer strong encryption to their customers. A great deal of information about SSL can be found at the SSL website (www.ssl.com).

4.6.4 Electronic Commerce Platforms: Packaging EC Middleware

The IT infrastructure services are being packaged together as **electronic commerce platforms**. These platforms provide a combination of infrastructure services such as networking, operating systems, middleware, and management for EC applications that may exist in the business-to-business and/or customer to business contexts. Many vendors, as we will see shortly, such as IBM, Microsoft, Oracle, Netscape, and Sun are providing such integrated EC platforms that can provide almost all services (both transactional and non-transactional) needed for EC. At a conceptual level, the EC platforms consist of the following building blocks:

- Network and operating system services
- Core middleware (e.g., web)
- EC specific middleware
- Management and support services

Electronic commerce over the Internet is in its infancy at present and will evolve through various stages. A possible scenario of evolution is presented in Table 4-1. In the earlier stages, enterprises utilize services such as email and EDI. As more business processes are automated, and more business information is generated and processed in electronic form, additional services such as decision support and workflow between organizations is needed. Each stage will introduce new challenges in the infrastructure (i.e., middleware, networks, management and support services). Most business communities are currently in stage 1 of EC, although some have started experimenting with stage 2 applications. It can be seen that most of the growth will be in the middleware, especially on transaction and flow management between organizations.

To illustrate how E-commerce platforms look like, let us briefly review IBM Net.Commerce and companion products, currently known as **WebSphere.** This platforms provides the tools to develop, host and operate informational and transactional Web sites on the Internet/Intranet. Net.Commerce supports a multitude of APIs for linking electronic commerce activities with other applications such as order processing, inventory, pricing, shipping, and tax calculations. Figure 4-30 shows a conceptual view of IBM Net.commerce. At the core of this architecture is the Net.Commerce Server that provides the set of APIs, command support, macros, daemons for dynamic Web page generation, and multihosting for accommodating multiple virtual stores on the same machine. The Net.Commerce Database houses the transactional (e.g., order information, shipping information) as well as non-transactional (e.g., tax calculation tables) catalogs, databases. These databases come with native support for DB2 and can be optionally stored in any other ODBC compliant databases. The Internet Connection Secure Server establishes secure sessions across the Internet between the end user's browser and the Net.Commerce applications. The Net.Commerce Administrator provides a suite of tools that are used to create and administer Net.Commerce sites and virtual storefronts. The Net.Commerce Utilities support import of data into catalog from external sources such as supplier stores, and legacy data interfaces for access to business applications("Merchant Legacy Systems") such as inventory and billing. Finally, the Net.Commerce Companion Products supports related products such as Secure Electronic Transaction (SET) payments, catalog workbench for creating and maintaining catalogs, data mining tools, and Lotus Notes Domino for document flow.

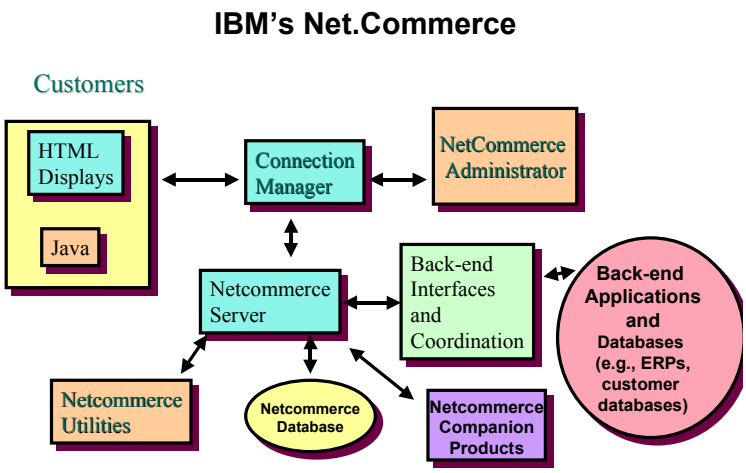

Figure 4-30. NetCommerce/WebShpere Architecture

	Typical Examples	Main Applications and Services	Network Services needed	Middleware Needed	Management and Support Services Needed
Simple E-commerce (Level 1)	Simple business to business commerce	• Mainly POs and invoices • Limited advertising and browsing	Internet + VAN	Email, EDI, Enhanced Fax, File transfer	Security
Professional E-commerce (Level 2)	Large electronic shopping malls, Military procurement systems	• Increased advertising and browsing • Integration within organizations	Extranet	Integration of workflows, enhanced EDI, email, legacy system integration, distributed objects	Directory Fault management
Advanced E-commerce (Level 3)	Electronic Commerce "Brokers"	• Seamless browsing and advertising • Integration and flow of work across organizations	Extranet	Distributed transaction processing, decision support, mobile computing, intelligent agents, real time multimedia	QoS

Table 4-1 Types of EC platforms

4.6.5 Platforms for Application Development and Deployment – The Application Servers

Simply stated, an application server (also known as app server) is a platform for development, deployment, and management/support of Web-based applications. The current and future versions of application servers include facilities for development, deployment, and management of web-XML applications. This includes facilities for EJB (Enterprise Java Bean) development, XML exchanges, load balancing, failure handling, and adapters for connecting to back-end applications.

Although application servers first appeared in client/server computing and on LANs, they really are associated with web-based development. The current genre of application servers are a result of several stages of web-based application evolution that has gone from static HTML to extensive development and management capabilities.

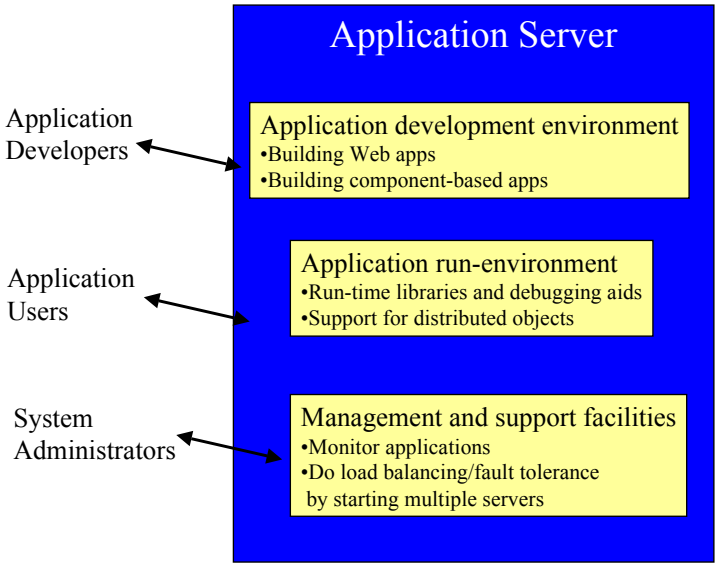

Figure 4-31: Typical Application Server Capabilities

The terms "Commerce Server" and "Application Server" are used commonly by e-commerce platform providers. Let us briefly discuss the differences between the two.

The term ***commerce server*** is often used loosely in the industry to describe the set of subsystems bridging the gap between a web server and a payment server. Common functionality of a CS includes shopping carts, catalog management, purchasing logic (e.g., customer verification), logs/audit trails to track sales activities, and interfacing with the back-end enterprise systems such as payment, order processing, and inventory control. Sometimes the term is used to include the web server and/or the payment server itself. When this occurs, it blurs/dilutes the meaning, making it difficult to understand what is actually contained in the commercially available commerce servers without analyzing their specifications.

Simply stated, an ***application server*** (also known as app server) is a platform for development, deployment, and management/support of Web-based applications. The current and future versions of application servers include facilities for development, deployment, and management of web-XML applications. This includes facilities for EJB (Enterprise Java Bean) component development, XML exchanges, load balancing, failure handling, and adapters for connecting to back-end applications.

In essence, a Commerce Server is an Application Server that specializes in e-commerce. An interesting illustrative example is the Netscape Application Server that combines web development capabilities with

enterprise applications that integrate with corporate data sources. At present, this server has evolved into Sun Iplanet that provides a complete set of e-commerce facilities.

A wide range of application servers are becoming available in the marketplace. For a current list of application servers, visit the serverwatch web site (http://serverwatch.internet.com/appservers.html). Examples of some of the app servers are (see the sidebar "List of Application Servers" for a more extensive list):

- BEA WebLogic Server (www.beasys.com)
- Borland AppServer (www.borland.com)
- ColdFusion (www.allaire.com)
- Lotus Domino (www.ibm.com)
- Oracle 9I App Server (www.oracle.com)
- WebSphere (www.ibm.com)

Time to Take a Break

- ✓ Network Overview
- ✓ Computing Platforms Overview
- ✓ Middleware Services
- 3G Distributed Computing Platforms

Suggested Review Questions Before Proceeding

- What is middleware and what role does it play in the IT infrastructure?
- How is the middleware different from the local system software?
- What are different types of middleware services and why are they needed?
- What are middleware platforms and application servers?
- What are the different types of application servers and what do they do? Why so many different application servers are emerging?

4.7 3G Distributed Computing Environments (Dot Net and J2EE)

The third distributed computing environments, also referred to as Object-oriented client/server Internet (OCSI) environments, combine the object-orientation, client/server and Internet concepts to deliver business functionality. In particular, these environments combine distributed objects and business components with Web and XML to support enterprise applications. In general, the OCSI environments support the multi-tiered applications, shown in Figure 4-32, that allow a wide range of users on a variety of devices to access back-end applications from the same or other (partner) enterprises. This conceptual model is the foundation of the two dominant distributed computing platforms (Sun's J2EE and Microsoft's Dot Net) at present. These

two platforms, in essence, package the wide range of enabling technologies discussed so far in this chapter under single umbrellas. We discuss these two platforms briefly.

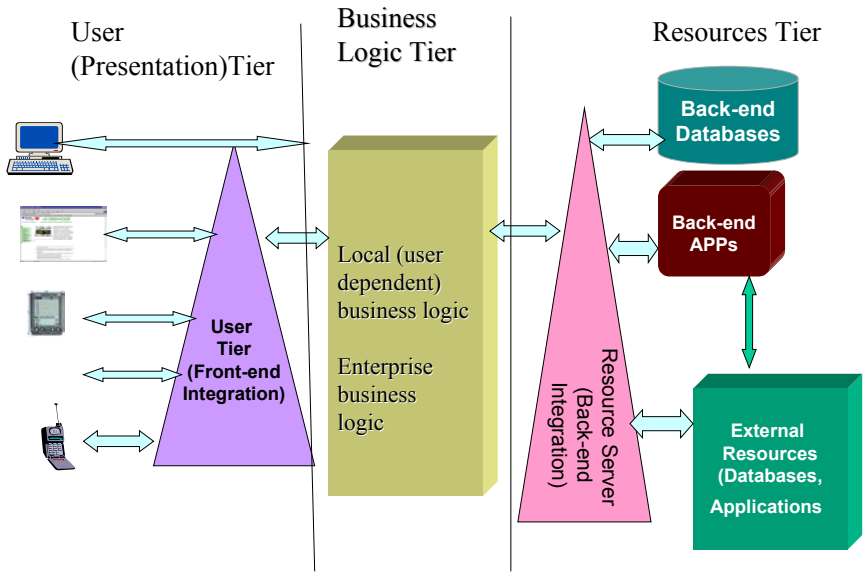

Figure 4-32: General Architecture of 3G Applications

4.7.1 Sun J2EE

The Java2 Enterprise Edition (J2EE) technology, proposed by Sun Microsystems, is a component-based approach to the design, development, assembly, and deployment of enterprise applications. The J2EE platform can be viewed as an application server that is based on reusable components, a unified security model, and transaction control. The Sun J2EE is also at the core of many other powerful app servers such as Oracle 9I Server and the BEA WebLogic App Server.

The J2EE is based on a multi-tiered application model and is not tied to the products and APIs of any one vendor. The overall J2EE architecture, shown in Figure 4-33, consists of the following components:

- Client tier components run on the client machine
- Web tier components run on the J2EE server
- Business tier components run on the J2EE server
- Enterprise information system (EIS) tier software runs on the the back-end systems

A J2EE application can be configured into two, three or four tiers, however, many J2EE multitiered applications are configured as three-tiered applications because they are distributed over three different locations: front-end machines where the clients reside, J2EE application server machines where the web server and the business logic resides, and the back-end machines where the enterprise database or legacy reside. Three-tiered applications are very flexible because a multithreaded application server resides between the client application and back-end information sources. This application server contains the business logic that is at the core of enterprise applications.

4.7.2 Microsoft's Dot Net

The Microsoft Dot NET platform includes a family of services and products, built around ***XML Web services.*** XML Web Services, also just known as Web Services, concentrate on software components that are accessible via standard Web protocols. The main idea is that most of the applications will be developed and delivered through the web, in particular, the components will exchange information through XML over

HTTP . Figure 4-34 shows the overall architecture of Dot Net. It shows how the XML Web Services, shown as Web Services, appear as components in different tiers of the Dot Net architecture.These components communicate with each other by using SOAP (Simplified Object Access Protocol) that can be used to transport XML documents over HTTP.

A great deal of information about various aspects of web services is available at present over the Web. An example of the source is the Microsoft site (http://msdn.microsoft.com). The book [Conard 2001] also covers the topic quite well. We will revisit Dot Net in later chapters of this module.

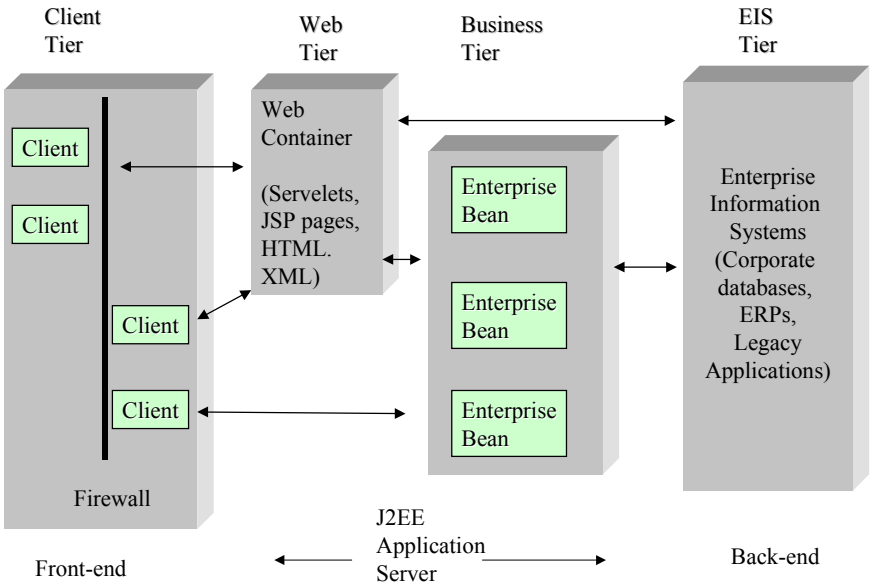

Figure 4-33: J2EE Environment

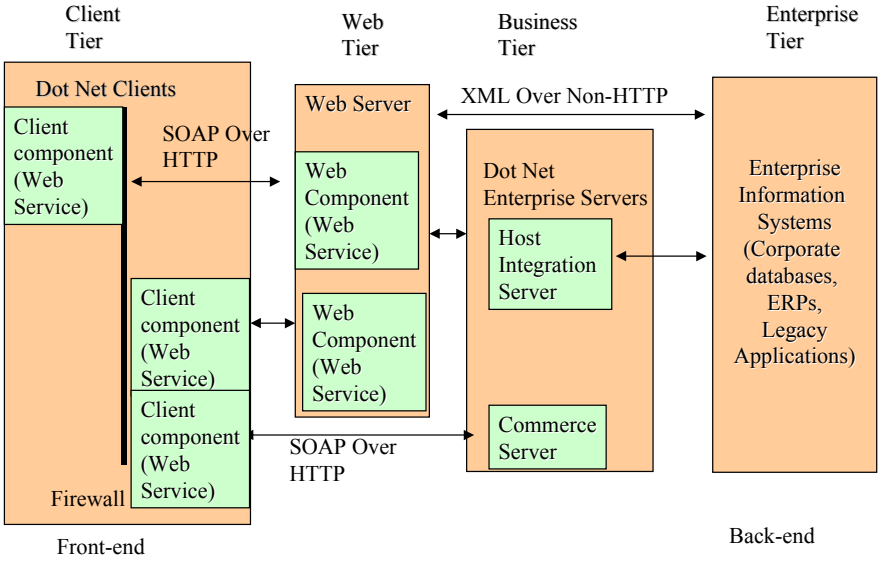

Figure 4-34: Microsoft Dot Net Architecture

4.8 Putting the Pieces Together – An Example

Let us quickly show how and where many of the technologies discussed so far fit into application architectures. Figure 4-35 shows the generic application architecture that we used in Chapter 2 of this Module to discuss various e-business applications such as CRM, emarkets, portals, etc. Table 4-2 shows how many of the enabling technologies discussed in this chapter can be used at the front-end/back-end integration and business layers for CRM, emarkets, and portals. This diagram does not show the network infrastructure explicitly . However, the users, layers, and the back-end applications can reside on multiple computers that are interconnected through a network of LANs, WANs, and MANs over wired and wireless facilities.

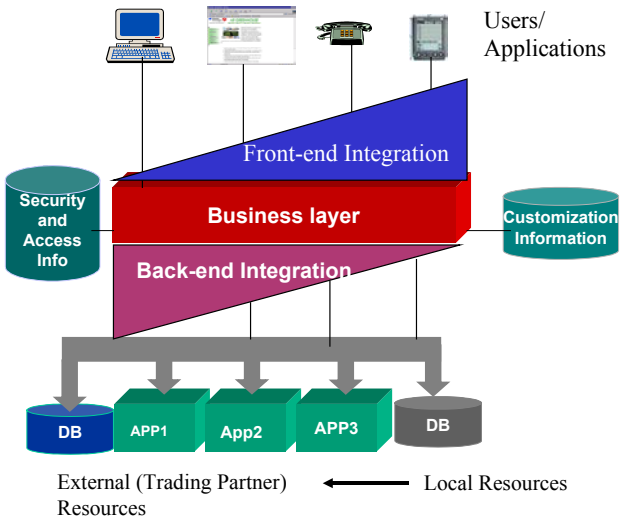

Figure 4-35: A Generic Application Architecture

Table 4-2: IT for Common e-business Applications

	E-CRM	ASP	Portals	Electronic Marketplaces
Front-end Integration	Web, XML, WAP, VOIP, VML	Web, XML	Web, XML, WAP, VOIP,VML	Web, XML
Business Layer Technologies	Web server, CORBA/ DCOM EJB, app servers	Web server, CORBA/ DCOM, EJBs, App servers	Web server, EJBs	Web server, Catalogs, e-payment
Back-end Integration	Remote access,legacy gateways, EAI, XML	Remote access,legacy gateways, EAI, XML	Remote access,legacy gateways, EAI, XML	Remote access,legacy gateways, EAI, XML

4.9 Case Study: Open Source-based Infrastructure

Open source software is software for which source code is available so that it can be modified by software developers to fit their particular needs. This idea is gaining popularity as compared to the "closed source" systems in which the vendor keeps the source code as a proprietary internal asset. Open source software provides all computer users with free access to its source code so that they can modify the code to fix errors or to make improvements. The software itself is not owned by any company or individual and is free. Because it is free and can benefit from the expertise of numerous software developers, it has become popular during the past few years among sophisticated computer users and businesses. The best known examples of open source software are the Linux operating system, the Apache Web server, and the Kerberos security system. Here is an example of how a small company is using open source software.

A retail furniture dealer, Raymour & Flanagan (R&F) upgraded the network from dumb-terminals to 486-based personal computers with Linux as the operating system. To view inventory, the PCs use the Netscape web browser and Apache Web server, also an open source product, was used for web users. The routers are managed with NetSaint (http://www.netsaint.org/), an open-source code program. The network supports 50 stores in the northeastern part of the United States over T1 connections to an IP-and-frame relay WAN. Advantages of this type of system include:

- Open source software for customizable applications
- Nominal, if any, licensing fees
- Nominal, if any, software update costs
- "Very" thin client configuration that consists of a Web browser residing on a customized kernel / OS on a low-end PC (Intel 40486 processor with 32 meg RAM or an Apple).

A disadvantage of this type of system may include the need to run a "dual shop-" Linux and Windows for some applications because some applications are not available on Linux. Overall, this appears to be practical albeit slightly more labor-intensive alternatives to the typical Microsoft applications for network deployment. While potentially slightly more labor intensive, R&F indicated that overall performance is quite satisfactory and the cost savings appear to be fairly substantial.

In almost any organization, costs, whether capital or expense, are a typical consideration. The following table attempts to compare the costs of a Microsoft based system to a Linux system at R&F. As can be seen, the savings based on using a Linux-based system could be an order of magnitude lower compared to a Microsoft platform. Since the cost savings are per machine, a Linux-based incremental expansion cost may be substantially less than a Microsoft-based network as the company expands its operation.

A correspondence with Brian Dewey, the network engineer for Raymour & Flanigan, who designed and implemented the network is given below to capture the main ideas. .

Table 4-3: Windows Vs Linux Network (Estimated Cost Comparison)

Assumptions: 50 Stores and 10 PCs/Store

	Windows		**Linux**	
Type	Quantity	Total	Quantity	Total Cost
PCs	500 Pentiums @ $1000ea	$500,000	500-80486 @ $400 ea.	$200,000
O/S	XP at $50	$25,000	Linux Kernel and Browser	$0
Server Support (1 Server /Store)	Windows 2000 @ $1199 / 10 Clients http://www.microsoft.com/windows2000/server/howtobuy/pricing/default.asp	$5,995	Red Hat $60 per year per server	$3,000

			2 Servers w/full support 2 $900 ea.	$1,800
Tech Support Call	One article indicated $55/call (Assume 1 call per week) http://www.bmug.org/news/articles/MSvsPF.html	$2,860	Red Hat Support	$0
Total Cost		**$533,855**		**$204,800**

Interview With Brian Dewey (R&F Network Engineer)

Brian Dewey is the Network Engineer for Raymour & Flanagan Furniture that built a largely open source network based on Linux. Below is a series of questions posed to Brian regarding the network.

Q: Would you use Linux again? If not, why not? Any "lessons learned?"

A: I would use Linux again for sure.... Linux is inexpensive, stable and very well supported through Redhat. I don't know if you ever have tried to get support from MS, but it's not easy. Redhat, I call one number and usually within 5 minutes I have a KNOWLEDGEABLE tech on the phone. I wouldn't use Linux in a production environment without support.

Lessons learned would be don't go into the project blind. Planning, planning, planning is the key. If you want to try Linux, install it at home and spend some time with it. The Distro's (Redhat, Mandrake) work very well and are very easy to use. There is ramp up time to use a new OS, but in the long run it is well worth it!

Q: How does the system compare to a non-Linux system in terms of performance, maintenance, and costs?

A: Performance: In my opinion Linux beats Windows hands down on performance. Even with newer SMP boxes Linux still rocks (up to 8 processors). Apache blows IIS away and Samba has been benchmark to be faster that NT at file and print services.

Maintenance: Redhat has a program you join called up2date (Redhat network) that registers your server and sends down patches and updates automatically or when you request it.

Costs. Linux is hard to price out. The OS is free, but we pay for support. So, the cost for fully supported Redhat Linux on a server is approx. $900 a year. There is also the subscription fee for the Redhat network of $60 a year per server. What I normally do is subscribe all my servers and then get the support for the main one or two. If you cost out Window 2000 server with the calls and the support you will see there is no comparison.

Other restrictions. On the Linux based thin terminals, we are restricted to non-MS software. We have to run Citrix (Windows Terminal Server) to run the access applications remotely, and they run Netscape browser and not Microsoft Internet Explorer (IE). We are beginning to see issues with Netscape and some web apps not working correctly. This is mostly due to the use of ActiveX and ASP by the web site designers and also coding for majority of web apps today is based on IE. We are looking at Mozilla (Netscape 6) and Opera 6 as replacement browses.

Q. How much outsourcing do you do and were there problems getting support since you're running Linux?

A. Web hosting. This was outsourced, we have just moved our web servers in-house. We have 2 full T-1's from WorldCom

Special Apps. The only problem was as I explained before with the Netscape browser on the Thin Terminals. Our in-house servers I've had no issues.

Q: I see you're supposed to be going with "Oracle-based applications for tasks like inventory." Is this Oracle's small business suite? Is it running on a UNIX box?

A. It's Oracle 8i running on a 4 way SMP Dell box running Redhat Linux. I don't believe it's the small business suite.

Q. Was R & F really running the inventory for 50 stores with MS-Access?

A. Inventory and all the store systems are run on a DataGeneral UNIX box running Oracle. Access was used for a lot of custom reporting and some data entry for commissions and parts ordering. Also they use Access 97 via Citrix to print out all the special tags and labels the store uses.

Q. Will the "Oracle-based apps" manage:“Back office” tasks such as A/P, A/R, Payroll, Inventory, etc? Can you use it to do business with other retailers and/or raw material / inventory suppliers?

A. Look at www.gers.com this is the store system we currently use. We use ADP for payroll and ADI for time tracking. ADP and ADI are NT server based apps so we are still a mixed shop.

Sources:

- Joe Campenla and James Wright, Project Report, Drexel University, June 2002 (edited by Amjad Umar)
- (http://networkcomputing.com/1305/1305centerfoldtext.html).

4.10 Hints about the XYZCorp Case Study

A rough sketch of the XYZCorp IT infrastructure architecture is shown in the following diagram. This sketch is old (was found in the back seat of an XYZCorp architect's car -- true story). There are two broad levels of interconnectivity:

- Network to network interconnectivity(i.e., bridges, routers)
- Applications to application interconnectivity (i.e., middleware)

Figure 4-36 shows primarily the network interconnectivity view. The reader should redraw this diagram and add the middleware components to this architecture. The sites where the servers (Web servers, Group servers, database servers, etc.) and the applications will be housed should be shown. This architecture should be updated to include wireless support and also cast into the Sun J2EE framework. If possible, the protocols to be used by each of the servers and clients should also be discussed for a complete picture.

If you do not like the following diagram, you may want to use Figure 4-4 instead (I won't blame you).

The architecture vision should be accompanied by a table that describes the various infrastructure components, what do they do, and why will they be needed for the corporation. It is also a good idea to work through a few scenarios that show how the information will flow between end-points (users on one side, applications on the other) of the infrastructure architecture.

4.11 Summary

The IT infrastructure (i.e., middleware, networks, operating systems, hardware) supports the modern distributed applications. We have reviewed the key concepts and building blocks of this enabling infrastructure. In summary:

- The client/server model is the most popular model for distributed applications at the time of this writing. This allows a user program at a workstation, typically, to act as a client issuing requests to programs and databases which may be located at many remote computing sites.
- The distributed object model extends the scope of the client/server model to include interactions between remotely located objects.
- The Web allows access to a multitude of information sources through Web browsers.
- Object-oriented client/server Internet (OCSI) environments combine the object-orientation, client/server and Internet concepts to deliver business functionality.
- OCSI applications represent the business logic as objects that may reside on different machines and can be invoked through Web services.

- Middleware supports and enables the OCSI applications. The middleware services have evolved, and continue to evolve since the early 1990s.
- Network services transport the information between remote computers. Networks are largely moving towards TCP/IP.

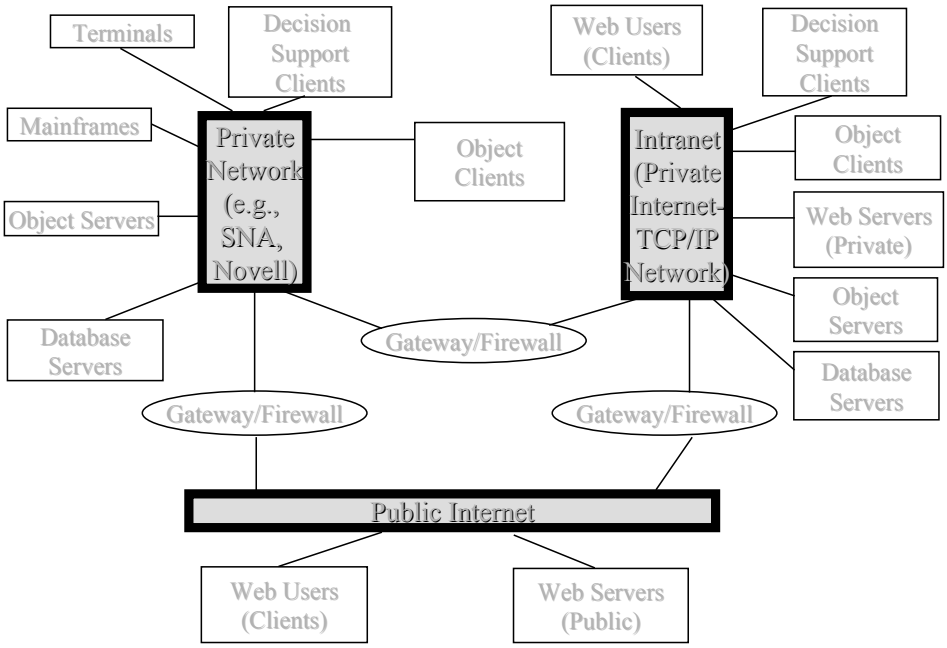

Figure 4-36: XYZCorp IT Infrastructure High Level View

4.12 Review Questions and Exercises

1. Present a different view of how client/server systems interrelate with distributed computing systems.
2. Give an example of a distributed application that does not use the client/server model.
3. Describe at least three different ways that you can invoke distributed object services from Web browsers, including applets.
4. Draw a conceptual diagram of IT infrastructure in an environment of your choice (e.g., business, finance, engineering, manufacturing).
5. List the factors you will use to evaluate off-the-shelf middleware to support OCSI applications.

4.13 Additional Information

Allen, P., "Realizing e-business with Components", Addison Wesley Professional; 2000.

Bell, T., Adam, J., and Lowe, S., "Communications", IEEE Spectrum 1996, Jan. 1996, pp. 30-41.

Birrel, A.D. and Nelson, B.J., "Implementing Remote Procedure Call", ACM Trans. on Computer Systems, Vol. 2, pp.39-59.

Booch, G., "Object Oriented Design with Applications", Benjamins Cummings, Second edition, 1994.

Comerford, R., "Computers", IEEE Software, Jan. 1996, pp. 42-25.

Comport, J., "Packaged Applications: Buy an Application, Inherit an Architecture", Gartner Group Briefing, San Diego, February 1995.

Conard, J., et al, "Introducing .NET", Mass Market, 2001.

Coulouris, G., and Dollimore, J., "Distributed Systems: Concepts and Design", Addison Wesley, 3rd edition, 2000.

Davydov, M., "Corporate Portals and e-business Integration", McGraw-Hill Professional Publishing; 2001.

Elsenpeter, R., and Velte, T., "e-business: A Beginner's Guide", McGraw-Hill Professional Publishing; 2000.

IDC, "The e-business Platform: Where Application Servers and Application Integration Meet", 2001.

Marcus, E., and Stern. H., "Blueprints for High Availability: Designing Resilient Distributed Systems", Wiley & Sons, 2000.

Sawhney, M., et al, "The Seven Steps to Nirvana: Strategic Insights into e-business Transformation", McGraw-Hill Professional Publishing, 2001.

Tannenbaum, A., "Modern Operating Systems", Prentice Hall, 1992.

Taylor, D., "Object Oriented Technology: A Manager's Guide", Addison-Wesley, 1994.

Trllica, C. "Software Applications", IEEE Spectrum, Jan. 1996, pp. 56-59.

Umar, A., "Object/Oriented Client/Server Internet Environments: Enabling Distributed Enterprises Through Middleware", Prentice Hall, 1997.

Umar, A., "Distributed Computing and Client/server Systems", Prentice Hall, revised edition, 1993.

Wilbur, S. and Bacarisse, B., "Building Distributed Systems with Remote Procedure Calls", Software Engineering Journal, Sept. 1987, pp. 148 - 159.

Whyte, W., and Whyte; B., "Enabling e-business - Integrating Technologies Architectures & Applications", John Wiley & Sons; 2001.

Wooding, T., "Business Frameworks", Object Magazine, January 1997, pp. 50-53.

Wu, J., "Distributed System Design", CRC Press; 1998.