PART III

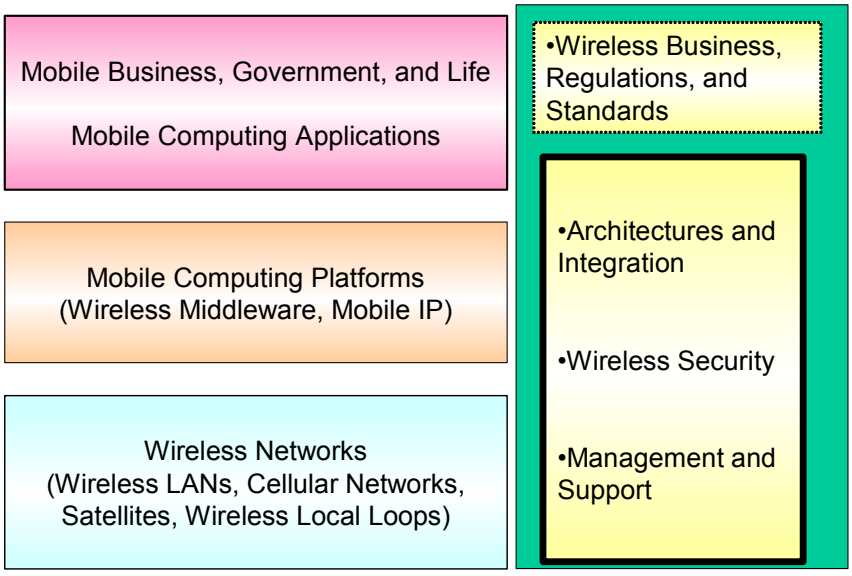
Architectures, Security, and Management

This part concludes our discussion by examining the issues that span *all* layers of wireless systems – from m-business to network adapter cards. The focus is on synthesis of the concepts and technologies discussed so far to build and manage working solutions. Topics include architectures, integration, wireless security analysis and design, strategic planning, and management platforms (box with dark borders in the framework shown below).

Chapter 11: Integrated Architectures for Wireless

Chapter 12: Wireless Security

Chapter 13: Management and Support Issues



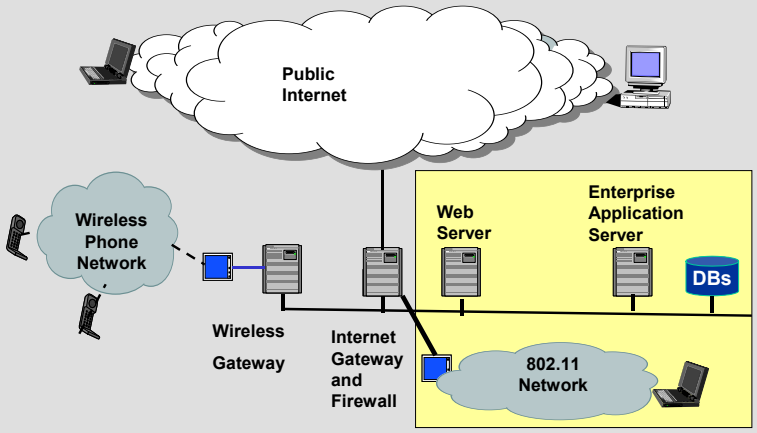
11 Integrated Architectures for Wireless

11.1 INTRODUCTION 11-3
11.2 CONCEPTS AND A FRAMEWORK FOR DISCUSSION 11-5
11.2.1 What is an Integrated Architecture? *11-5*
11.2.2 A Framework for Integrated Architectures *11-9*
11.2.3 Horizontal versus Vertical Integrations *11-11*
11.3 WIRELESS VISION AND EXAMPLES 11-13
11.3.1 An Integrated Architecture Vision *11-13*
11.3.2 The Internet as an Example of Good Architectural Vision *11-15*
11.3.3 Example of a General Integrated Architecture *11-16*
11.4 INTEGRATED NETWORK ARCHITECTURES FOR WIRELESS – LAYER 1 TO 2 ISSUES 11-17
11.4.1 Adapter Cards for Detecting Different Wireless Systems *11-18*
11.4.2 Network Interconnectivity Devices and Protocol Converters *11-18*
11.4.3 Network Integration Examples *11-19*
11.5 ROAMING AND MOBILE IP FOR WIRELESS INTEGRATION 11-21
11.5.1 Roaming Revisited *11-21*
11.5.2 Mobile IP as a Basis for Integration *11-21*
11.5.3 Examples of Mobile IP *11-21*
11.6 MIDDLEWARE AND MOBILE COMPUTING PLATFORMS FOR INTEGRATED ARCHITECTURES . 11-22
11.6.1 Overview *11-22*
11.6.2 Wireless Middleware Services *11-23*
11.6.3 Mobile Computing Platforms (Mobile Application Servers) *11-24*
11.6.4 Example of a Mobile Application Server – Oracle9iAS Application Server for Wireless 11-25
11.7 INTEGRATED APPLICATION ARCHITECTURES FOR MOBILE SERVICES 11-25
11.7.1 Overview of Mobile Application Architectures *11-25*
11.7.2 Integrated Application Architecture Examples *11-27*
11.8 TECHNOLOGY TREND – COMPONENT-BASED INTEGRATED ARCHITECTURES 11-30
11.8.1 What is a Component? *11-30*
11.8.2 What is a Component-based Architecture? *11-32*
11.8.3 .NET and J2EE as Component-based Platforms *11-35*
11.8.4 Web Services – The Cornerstone of Component-based Platforms *11-36*
11.8.5 Mobile Web Services *11-40*
11.9 CONCLUDING COMMENTS 11-41
11.10 REVIEW QUESTIONS AND EXERCISES 11-42
11.11 REFERENCES 11-42

Case Study: Architecture of a Wireless System

An small insurance company wanted to develop an architecture for its current and future applications. The company was interested in providing access, for its authorized users, to its enterprise applications and databases from the public Internet. In addition, salesmen in the field needed to look at the same information over a cellular network. The company had also developed an internal 802.11 network for its employees. The architecture had to be flexible, secure, and integrated.

The following figure shows the resultant architecture based on a hybrid network. Web access over the public Internet to enterprise applications and databases is controlled by an Internet gateway that also includes firewall software. A wireless gateway is introduced to handle cellular users. For security purposes, the wireless gateway is connected to the firewall. The internal 802.11 network is also connected to the external firewall for security purposes (this is commonly done to enforce security in internal wireless LANs). The architecture uses specialized servers and gateways for flexibility As new services are needed, more or new servers can be added to this architecture.



11.1 Introduction

So far we have discussed different components of wireless systems,[1] i.e., the different types of wireless networks (wireless LANs, cellular networks, wireless local loops, satellites), wireless Internet and Mobile IP, mobile computing platforms, and mobile computing applications for m-business. The natural question is: how do you integrate the various components of a wireless system into a working solution that satisfies the customer and corporate needs? This question guides the discussion in this chapter.

Building solutions that work in the turbulent and highly fluctuating wireless landscape is a daunting task owing to changing customer needs, the large number of technical options, competing standards, and a multitude of vendor products. Add to this the potential risks and pitfalls associated with any new technology, and you get the picture. The purpose of this

[1] The term "wireless system" is used here to represent the entire stack (physical wireless network to mobile computing application).

chapter is to discuss how the various pieces can fit into a functioning architecture that can survive the changes in technologies, standards, and market needs.

We start with a conceptual overview of architectural issues and establish a framework for discussion (Section 11.2) that is used throughout this chapter. Section 11.3 presents an architectural vision of a hybrid wired-wireless network designed to support various mobile users seamlessly. This vision is illustrated through a few examples. The balance of this chapter essentially explains the different aspects of this vision, starting from integration at wireless network levels and then proceeding to integrations at the platform and application levels. Discussions highlight the role of multi-network adapter cards, Mobile IP, wireless middleware, mobile application servers, and different computing applications such as m-commerce, m-portals, and SMS. The chapter concludes with a discussion of the emerging trend of using components, component-based platforms, and Mobile Web Services in building the flexible and integrated systems of the future.

Chapter Highlights

- Integrated architectures for wireless are based on a hybrid network that integrates high-speed 802.11 hot spots and wireless/wired business LANs with lower-speed cellular networks. Integration requires capabilities at several levels that go from lower network issues to application components.
- Physical communication level (layer 1 and 2) integration requires adapter cards that can detect if a user is in a different coverage area. In addition, network protocol converters and gateways are needed between different types of networks.
- Handoffs and roaming support between multiple networks require Mobile IP or some other capability to support handoffs as the mobile devices roam from one network to another.
- Wireless middleware services such as WAP and i-mode can shield the application's developers from the underlying wireless network details.
- Uniform application and user interfaces are needed to provide seamless access to back-end systems from wired as well as wired networks.
- Components, component-based platforms, and Mobile Web Services represent a significant trend in building the flexible and integrated systems of the future.

The Agenda
- Concepts and Vision
- Levels of Integrated Architectures
- Component-based Architectures

11.2 Concepts and a Framework for Discussion

11.2.1 What is an Integrated Architecture?

"Integrated architecture" combines two heavily loaded terms: architecture and integration. So let us first define these two terms. Simply stated, an architecture shows how individual components tie together to satisfy the overall system requirements. Many views on architectures exist at present (see, for example, Askit [2001], Bruegge [2000], Clemens [2002], and Herzum [2000]). It is not our objective to give a comprehensive discussion and comparison of architecture definitions (see the sidebar, "Architectures: Glossary and Definitions"). Instead, we adopt the following simple but highly operational definition, based on Webster, for the purpose of discussion:

Definition: An architecture of a system is a structure that describes three things:

- Components of the system (what are the pieces of a system?)
- Functions performed by the components (what do they do?)
- Interfaces/interactions between the components (how do they work with each other?)

This definition is consistent with the IEEE 610.2 definition of an architecture: "The structure of the components, their properties, relationships, and the principles and guidelines governing their design and evolution over time."

A wireless system architecture, from this point of view, consists of the components (network components such as wireless access points, platform components such as WAP gateways, and application components such as mobile databases), what they do (connect with the user, translate content, provide needed data), and how they interface/interact with each other. For example, the view of GSM network architecture shown in Figure 11-1 identifies all the components of the GSM network, including the mobile station (MS), base station subsystem (BSS), mobile switching center (MSC), home location register (HLR), and visitor location register (VLR). This architecture also specifies the functional roles of the GSM components, and the detailed interfaces between the various components.

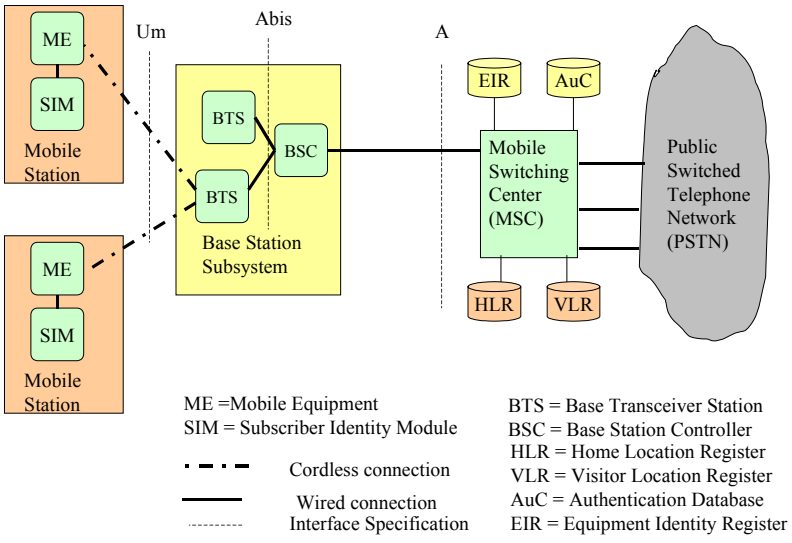


Figure 11-1: GSM Network Architecture

We have seen similar architectures of IEEE 802.11, Bluetooth, IEEE 802.16 and others in the previous chapters. Most architecture specifications can be quite detailed (Bluetooth specifications are, for example, more than 1500 pages long). For this reason, high-level architectures are created to capture the key ideas. Also some architectures only concentrate on certain aspects of a system (for example, wireless architectures concentrate on layer 1 and 2) while others cover almost all layers but with a narrow focus (for example, WAP 1.0 covered many layers of the OSI model for cellular networks).

Given that anyone can patch up a few components to build a system, are there some guidelines that dictate a good architecture? Basically a good architecture should be able to tolerate technology changes and be flexible and scalable to handle different and new users (see the sidebar, "What are Good Architectures?"). Our interest is in ***"integrated architectures"*** that consist of components that are *seamlessly* combined to support similar conventions or styles. As expected, there are many different views and definitions of integration, especially at the enterprise application level (see the sidebar, "Different Views and Definitions of Application Integration"). For our purposes, integration refers to the ease with which systems can be used [Nutt 1992] – we primarily concentrate on the user benefits. *Integrated systems basically minimize the effort needed to the user of the system – the user may be a human being or another automated system.* This implies that two systems, S1 and S2, are integrated if they:

- Share and exchange information without external intervention
- Are seamless in terms of operations
- Show consistency of behavior and presentation

Thus an integrated wireless architecture would provide seamless access to a diverse array of resources across a hybrid network of wireless LANs, cellular networks (3G, 2.5G, GSM), Bluetooth WPANs, and a variety of public shared "hotspot" wireless LANs in hotels, restaurants, airports, plazas, gas stations or other business centers. We will present such an architectural vision in Section 11.3.

Architectures: Glossary and Definitions

Building Construction Architectures: The structural abstractions (e.g., blueprint) and styles (families of related common variations) that define a class of structure (e.g., a cathedral) or a particular structure (e.g., my house). Architecture usually focuses on the big picture and not the details of what color my rug is or specific pictures on my wall, though such details can be viewed as architectural since they could be consonant or dissonant with the architecture's theme. There is no clear dividing line.

Application Architectures*:* Application architectures are broad architectures of the domain/application of interest – they show the application components and their interfaces/interactions with each other and the external environment.

Software Architecture: A static framework or skeleton (structure or set of conventions) that provides the form of a software system and the conventions, policies, and mechanisms for composing itself with subsystems, or component parts, that can populate the architecture. The architecture defines how the parts relate to each other, including constraints governing how they can relate. An abstract framework is one that has not been instantiated with specific subsystems.

Internal Architectures: Subsystems may have internal architectures (an O/S calls a compiler or a DBMS). In general, one man's floor is another man's ceiling. That is, there may not be a good way to distinguish architecture from the rest of design, though one can still describe architectural abstractions and prove their properties.

Other Glossaries:

- Gio Wiederhold's glossary of I3 architecture terms (www-db.stanford.edu/gio/1996/glossary.ps).
- DoD DISA terminology (www.disa.mil/disasub.html).
- NIIIP Reference Architecture: Concepts and Guidelines – Glossary (www.niiip.org/public-forum/ntr86-010html/Rashort-8.html).

Source: Craig Thompson and Frank Manola, "Component Software Glossary," www.objs.com. Glossary sponsored by the Defense Advanced Research Projects Agency and managed by the U.S. Army Research Laboratory under contract DAAL01-95-C-0112, 1997

What are Good Architectures?

First, a few nice quotes:

"We shape our buildings and afterwards our buildings shape us." – Winston Churchill

"While any single product is apt to become quickly outdated, a well-designed and open-ended architecture can evolve along with critical technologies, providing a fixed point of stability for customers and serving as the platform for a radiating and long-lived product family." – C. Morris and C. Ferguson, "How Architecture Wins Technology Wars," *HBR* (March-April 1993).

Good architectures, in general,

- **Survive changes in technologies**. Good architectures attempt to separate and hide lower-level network details from higher-level applications issues. This is the reason why OSI-style layered architectures are used so often.
- **Survive changes in business processes**. Business processes and business flows change frequently due to conditions such as new markets, new customers, product alignment, mergers and acquisitions. It is desirable, as much as possible, to design a solution where the business rules are kept in one component that can be modified if needed. An example of such an architectural approach can be found in the Telecommunications Forum's (www.tmforum.org) Catalyst Projects, where the business workflow is externalized into a separate component.
- **Survive attacks and intrusions**. Systems at present are subject to intrusions due to security and hardware/software failures. Good architectures must be able to survive intentional or un-intentional intrusions, attacks, and assaults. Thus good security and fault-tolerant features must be designed into an overall system architecture.
- **Scalable for additional workloads**. Successful systems become more popular (naturally) and thus add more users. The architectures must be developed with scalability in mind so that the system does not fall apart if it goes from 100 to 1000 users.
- **Flexible for new features.** Successful systems also need to continuously add new features to handle the growing needs of existing users and the new needs of future users. Thus a modular, component-based architecture is highly desirable, where each

component can embed a new feature.

Different Views and Definitions of Application Integration[2]

1. Enterprise Integration = a set of services and solutions for bringing together disparate applications and business processes as needed to meet the diverse information requirements of your customers, partners, suppliers, and employees (Source: "Enterprise Integration" – HP Services website).

2. EAI: Acronym for ***e****nterprise* ***a****pplication* ***i****ntegration*. EAI is the unrestricted sharing of data and business processes throughout the networked applications or data sources in an organization. Early software programs in areas such as inventory control, human resources, sales automation and database management were designed to run independently, with no interaction between the systems. They were custom built in the technology of the day for a specific need being addressed and were often proprietary systems. As enterprises grow and recognize the need for their information and applications to have the ability to be transferred across and shared between systems, companies are investing in EAI in order to streamline processes and keep all the elements of the enterprise interconnected. There are four major categories of EAI:

- Database linking: databases share information and duplicate information as needed.
- Application linking: the enterprise shares business processes and data between two or more applications.
- Data warehousing: data is extracted from a variety of data sources and channeled into a specific database for analysis.
- Common virtual system: the pinnacle of EAI; all aspects of enterprise computing are tied together so that they appear as a unified application.

(Source: webopedia.com).

3. EAI refers to the plans, methods, and tools aimed at modernizing, consolidating, and coordinating the computer applications in an enterprise. Typically, an enterprise has existing legacy applications and databases and wants to continue to use them while adding or migrating to a new set of applications that exploit the Internet, e-commerce, extranet, and other new technologies. EAI may involve developing a new total view of an enterprise's business and its applications, seeing how existing applications fit into the new model, and then devising ways to efficiently reuse what already exists while adding new applications and data. Source: EAI (iway software: http://www.iwaysoftware.com/definition/eai-enterprise-application-integration.html).

4. EAI refers to the plans, methods, and tools aimed at modernizing, consolidating, and coordinating the computer applications in an enterprise. Typically, an enterprise has existing legacy applications and databases and wants to continue to use them while adding or migrating to a new set of applications that exploit the Internet, e-commerce, extranet, and other new technologies. EAI may involve developing a new total view of an enterprise's business and its applications, seeing how existing applications fit into the new model, and then devising ways to efficiently reuse what already exists while adding new applications and data. Source: http://www.iwaysoftware.com/definition/eai-enterprise-application-integration.html

[2] These definitions have been extracted from the definitions that were collected by my students in the "Enterprise Integration" class at Fordham Graduate School of Business during the spring of 2003.

5. "EAI is the business strategies, processes, and technologies intended to provide seamless and uniform development, extension, perception, use, and management of the means to execute business functions." – Alternative Technologies. EAI is the convergence of low-level application integration solutions such as message-oriented middleware and high-level process management technology. Combining the speed and application connectivity of message-oriented middleware, with the business-focused abstraction, control, visibility and measurement of process management technologies, EAI provides organizations with the ability to rapidly implement reusable business-focused integration solutions. These solutions will meet the needs of the customers and the business today and are flexible for the ever-changing business landscape in the future. Source: HP Consulting Services, "EAI Overview" – http://www.hp.com/hps/briefs/application_inter.pdf

6. Enterprise integration offers numerous benefits. It allows companies to leverage best-of-breed software and develop cutting-edge e-commerce solutions, while they extend the life of their legacy systems and other existing information technology investments. It facilitates fast, seamless communication among an organization's systems, as well as those of its suppliers, partners and customers. And, enterprise integration helps exploit strategic sourcing opportunities (including application service providers and business service providers) to create or connect to portals, exchanges and other emarkets. Enterprise integration can also help reduce costs, increase operational efficiencies, expedite time to market, and improve return on information technology investments. These benefits are necessities in today's marketplace, not luxuries. Customers do not wish for the ability to place their own orders, check an order's status or perform other service transactions, they demand it. However, without enterprise integration, your infrastructure will lack the robustness and flexibility to grow in today's ever-changing economy. Source: http://www.accenture.com/xd/xd.asp?it=enweb&xd=services\se\ebs_capa_whyeai.xml

7. "Enterprise integration is a vital strategic enabler, not just a tactical solution. In addition to integration efficiencies and the ability to manage change, it helps companies grow their businesses in new and innovative directions." – Karsten Alva-Jorgensen, associate partner, Enterprise Integration, Accenture. Source: http://www.accenture.com/xd/xd.asp?it=enweb&xd=services\se\ebs_capa_approach.xml

11.2.2 A Framework for Integrated Architectures

Integrated architectures are based on a continuum of services that go from low-level network interconnection technologies to business applications and processes. Specifically, integration in wireless systems requires capabilities at the following levels (see Figure 11-3).

- **Physical communication level** (Layer 1 and 2). At the very basic level, adapter cards are needed that can detect if a user is in a different coverage area. For example, a card that can recognize GSM, GPRS, and 802.11 signals is needed in mobile devices to operate in a hybrid wireless network. In addition, network protocol converters and gateways are needed between different types of networks. See Section 11.4.
- **Handoffs and roaming support between multiple networks**. Mobile IP is a major player in supporting handoffs as the mobile devices roam from one network to another and as the IP addresses change due to the roaming. See Section 11.5.
- **Mobile computing platforms for integration.** At higher levels, mobile computing platforms provide the middleware services such as WAP and i-mode to shield the application developers from the underlying network details. Mobile application servers,

such as the Oracle9iAS-wireless server, combine several middleware services into a single platform. See Section 11.6.

- **Application and user interfaces.** At the application level, consistent user interfaces are needed for seamless operations. Microbrowsers and specialized markup languages such as WML (Wireless Markup Language) support these facilities and also provide access to back-end systems. See Section 11.7.

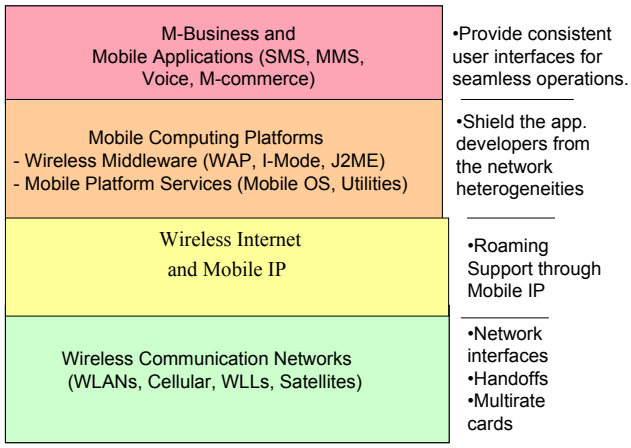


Figure 11-2: A Framework for Integration

By using many of these different technologies, mobile users can move from wired to wireless networks in a seamless manner. The mobile devices need a network adapter card that will tune to a wireless LAN or WAN as the users roam between the coverage areas of LANs or WANs. For example, the mobile device will tune into a hotspot LAN when you enter coverage area of that LAN. The software in the mobile devices will dynamically load drivers to connect to the right type of network. The middleware and application interfaces maintain the logical connection while switching from one network to the other.

In modern enterprises, the issues of integration go beyond applications and include business processes at the enterprise as well as inter-enterprise (B2B) levels. Figure 11-3 shows this broader view that spans network interconnectivity to cross-enterprise integration for B2B trade. At each level, a set of technologies is employed. As shown in Figure 11-3, the technologies range from network transport to B2B integration. Also note that the value of integration goes up as you go to higher levels of integration – integrating several businesses together has much higher business value than just hooking two networks together. Also note that, at each layer, a protocol/data converter (gateway) may be needed if the protocols of the senders and receivers do not match. Thus a network gateway may be needed between a 3G network and an 802.11 LAN; an i-mode-to-WAP converter may be needed at the application connectivity layer; an EDI-to-XML translator may be needed at the information transformation level, and a workflow translator may be needed at the process management level. This means that integration is both challenging and important (consider integrating 50 applications that all use very different technologies).

A technical solution to the integration problem should meet several design goals such as insulating the business from changes in the technologies, and helping with combining systems from acquired companies. The systems which attempt to support these goals are the ***Enterprise Application Integration*** (EAI, also known as eAI, e-Business Application Integration), platforms. The purpose of these integration platforms is to package all the converters and translators into a single "bus" that can be used by multiple applications to communicate. Due to their focus on applications, the EAI/eAI platforms do not typically

include network connectivity issues – they go up from the application connectivity layer. The sidebar, "An EAI Solution for Wireless" briefly describes an EAI platform for wireless. See Umar [2004][3] for a detailed discussion of enterprise application integration and EAI/eAI platforms.

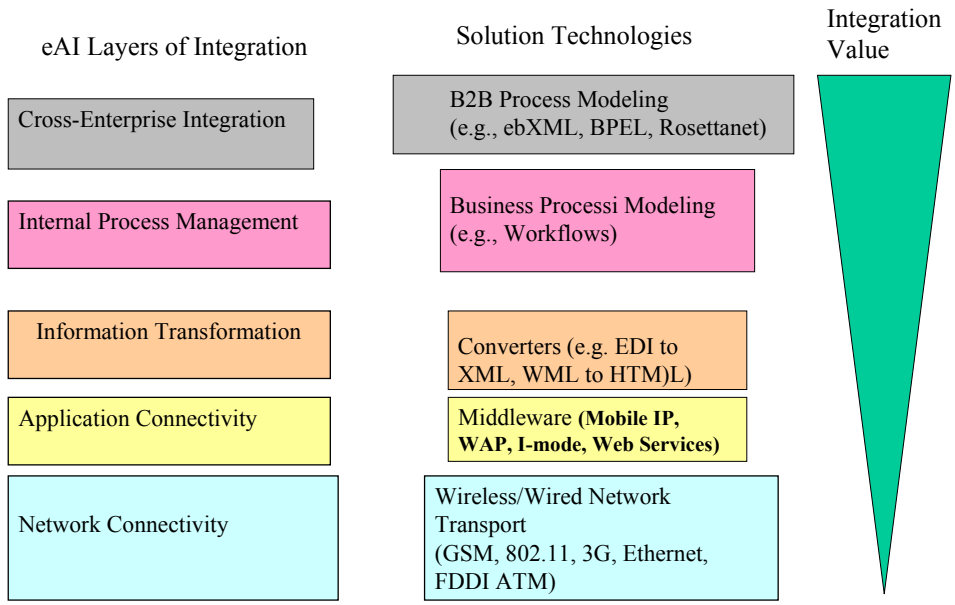


Figure 11-3: Levels of Integration and Enabling Technologies

An EAI Solution for Wireless

Many EAI platforms are commercially available from suppliers such as IBM, Tibco, Vitria, and WebMethods. Some have also added features for wireless integration. An example is the Xora Platform 3.0 from Xora, Inc. This software uses a variety of prebuilt connectors to applications such as Clarify, Remedy and Siebel sales force and customer relationship management systems, Oracle ERP and CRM applications, i2 supply chain applications, and Vitria's EAI platform. It also uses a Java tool kit to create connectors to other applications, and includes canned code that automatically adapts content to various devices. The platform simplifies the conversion of enterprise applications for wireless deployment.

11.2.3 Horizontal versus Vertical Integrations

Given the multiple layers of integration discussed previously, two systems can be integrated by using the following two approaches:

- **Vertical integration:** Each system is integrated with layers above and below. For example, a supplier can build an m-commerce application that is integrated with its own proprietary technology stack. DoCoMo applications that are built by using i-mode to run on NTT DoCoMo networks provide a good illustrative example of vertical integration.

[3] A. Umar, *e-Business and Distributed Systems Handbook: Integration Module*, 2nd ed., NGE Solutions, 2004.

- **Horizontal integration:** Different systems in the same layers are integrated either by using a common protocol and associated technologies, or by using protocol converters and gateways. For example, Mobile IP can provide horizontal integration at the IP layer to applications that use IP over different wireless networks.

An architect can choose a mixture of these approaches to build an integrated system. For example, lower-level networks can be horizontally integrated but the applications are vertically integrated, and vice versa. Figure 11-4 illustrates the various options. For the sake of illustration, the network in this figure indicates the physical network (layer 1 and 2). The TCP/IP layers are not shown explicitly –they are implicitly included in the platform. Let us discuss these options by using two m-commerce applications (app1 and app2), where app1 runs on a GSM phone supported by WAP while the other runs on an 802.11 LAN by using the Wireless Java technologies.

Figure 11-4a shows the completely vertical application integration in which the two applications are completely standalone. No integration exists between the two applications at any level. So, app1 uses a different user presentation, different handset and different support technologies than app2. In other words, to use app2, the user of app1 has to buy a new device, subscribe to a different network, and learn new commands – not an entertaining thought.

Figure 11-4b shows horizontal integration at the physical network level (layer 1 and 2). This could mean that either the two networks are converted into one (GSM, say, has been abandoned in favor of 802.11), or adapters/protocol converters are employed to translate between the two network protocols. This also implies that roaming support between the two networks has also been worked out. But beyond the low-level network integration, the two applications still use different devices and presentations because they are vertically integrated at higher layers.

Figure 11-4c shows horizontal integration at the IP and platform level (layers 3 to 6). In this case, the underlying physical networks are not integrated. Instead, Mobile IP is used to maintain connectivity as the users move from one network to the other.

Figure 11-4d shows horizontal integration at the application level (layer 7). Here the user sees the same application running on two different devices with the same look and feel. However, the lower layers are not integrated. In this case, the application code hides all the heterogeneities of the platform and the network.

Besides these four configurations, other integration approaches can be devised. For example, more than one layer could be horizontally integrated. Theoretically, all layers could be horizontally integrated so that all applications use the same networks, platforms, and presentations. This, however, implies too much cost and effort, given the diverse array of technologies currently available.

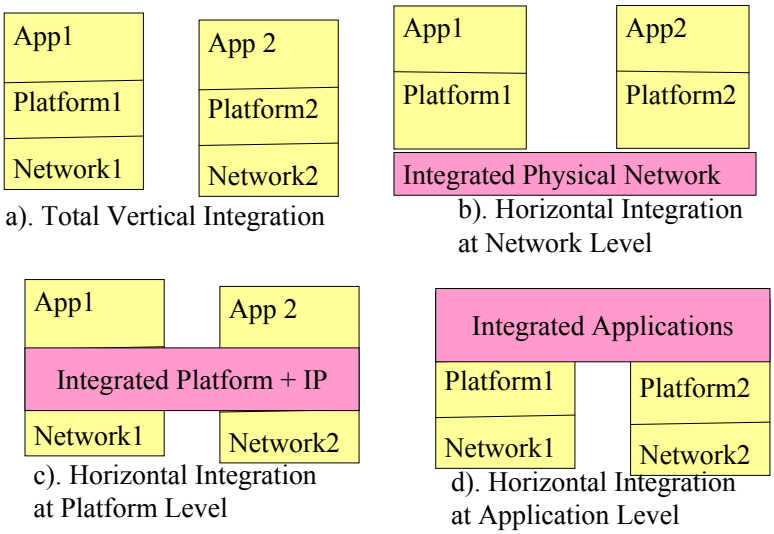


Figure 11-4: Vertical versus Horizontal integration

Note: The network in this figure indicates the physical network (layers 1 and 2). The TCP/IP layers are not shown explicitly, for the sake of simplicity. They are implicitly included in the platforms.

11.3 Wireless Vision and Examples

11.3.1 An Integrated Architecture Vision

Figure 11-5 suggests an integrated architectural vision that serves the needs of mobile workers and enterprises through a hybrid network consisting of public or private wireless LANs and wireless wide area networks.[4] The architecture presents a mixture of wireless LANs, cellular networks (3G, 2.5G, GSM), Bluetooth WPANs, and a variety of public shared "hotspot" wireless LANs in hotels, restaurants, airports, plazas, gas stations or other business centers. The growth of hotspots is an interesting development because they are filling the void as the 3G networks are being delayed. The main idea is that a wide range of individual wireless networks and fixed wireless LANs are interconnected through a mixture of wireless or wired networks to provide seamless services over the Internet.

[4] This discussion is based on views presented by C. Dharwan on www.mobileinfo.com (2002).

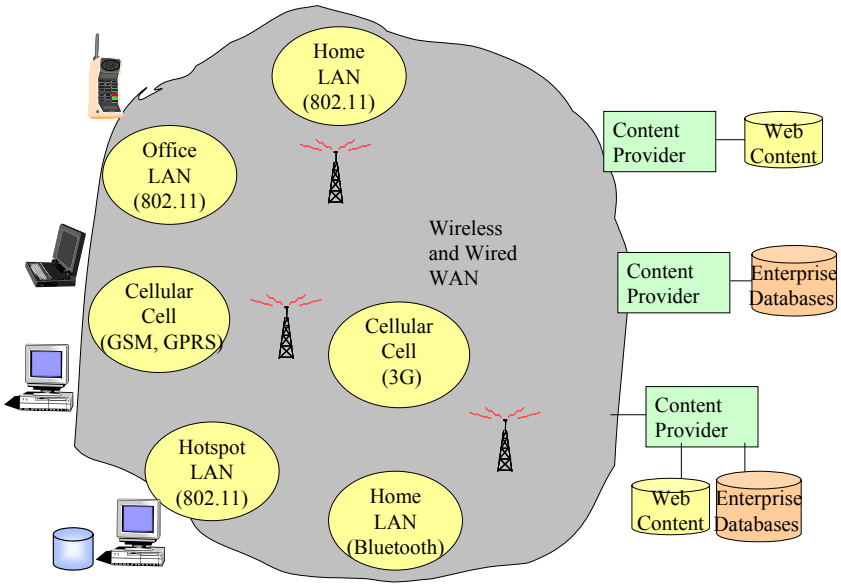


Figure 11-5: A Wireless Architectural Vision

This architecture provides high-speed (IEEE 802.11b LAN at 11 Mbps or 802.11g up to 54 Mbps) access within the hotspots and business LANs, and lower speed (56 Kbps to 384 Kbps) when outside the LANs by utilizing GSM or 2.5G GPRS cellular networks. Bluetooth technology, although not as popular as 802.11, has found a niche in the wireless personal area networks (WPANs). Integration, as discussed in the integration framework (Section 11.2.2), requires facilities at several levels:

- Physical communication level (layers 1 and 2) with network adapter cards and network protocol converters and gateways
- Handoffs and roaming support between multiple networks through Mobile IP and other approaches
- Middleware and mobile computing platforms that shield the application developers from the underlying network heterogeneities
- Application and user interfaces to present a unified user view

As explained earlier, these different technologies allow mobile users to move from wired to wireless networks in a seamless manner. The network adapter cards tune into a 802.11 hotspot LAN or a GSM cellular network as a user moves from one coverage area to the other. The middleware and application interfaces maintain the logical connection and consistent user interfaces while switching from one network to the other.

This vision also allows users to perform different activities as they move from one location to another. At a hotspot, for example, the user could do Web surfing and other Internet information-related work, and receive SMS messages while walking around or driving around in a car. The user can stay logged on and stay in touch, for example, even though 802.11 LAN signals fade and the GPRS wireless WAN takes over. Thus different models of human interactions can be supported – relaxed and intense work by a user while stationary at different sites, but occasional short emails while in motion – without the user having to logon/logoff [Dharwan 2000].

This integrated architectural vision has several benefits. First, it favors coexistence of different technologies (802.11, Bluetooth, 3G, 2.5G, GSM) instead of universal dominance of one. Second, it promotes development of individual technologies and integration of these technologies into a seamless hybrid location-aware network. Finally, this architecture

addresses the spectrum shortage problem because in densely-populated areas, wireless LAN-based hotspots can carry the traffic, freeing the frequency bands for truly mobile workers in their cars and trains. On the downside, many technical as well as business issues need to be resolved. For example, development of appropriate adapter cards and roaming support remains a challenge. Perhaps the most difficult obstacles lie in the business arena, as the opposing self-interests of wireless cellular network vendors, wireless LAN vendors, and the various regulatory bodies may impede cooperation. However, progress is being made in this area, as evidenced by the acquisition of and investment in 802.11-based hotspots by the large wireless providers.

11.3.2 The Internet as an Example of Good Architectural Vision

Are there some well-known examples of good architectures that satisfy most of the considerations discussed in the sidebar, "What are Good Architectures?" In other words, are there architectures that can tolerate technology changes and be flexible and scalable to handle different and new users over time? An example of such an architecture is definitely the TCP/IP stack that is at the core of the Internet. This layered architecture was introduced in 1969 to support file transfers and emails between less than 100 DARPA researchers on very slow networks (300 bps or less) that existed circa 1970. However, the same architecture at present supports millions of users, with applications that range from Web browsing to IP telephony, over broadband and wireless networks that are thousands of times faster than the networks of the 1970s.

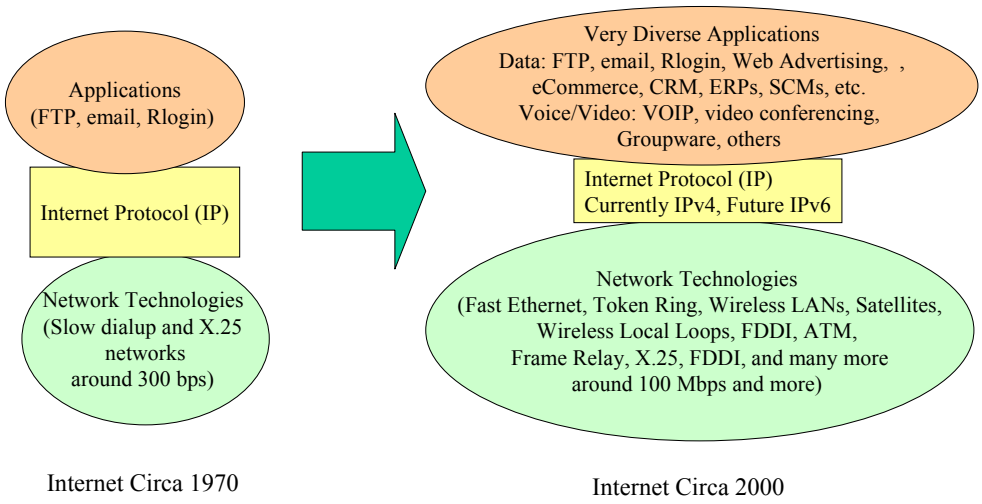


Figure 11-6: Evolution of the TCP/IP (Internet) Stack

The designers who built the TCP/IP stack developed a layered approach that separated network technologies from applications and thus served as the foundation for the survival of the TCP/IP stack through numerous changes and growth over more than 3 decades (Figure 11-6). Several other examples of well-thought-out architectures also exist in computing, such as IBM's 370 architecture that was introduced in the 1970s for large scale IBM mainframes and still is the foundation of IBM mainframe systems. The currently popular component-based architectural frameworks based on Web services, discussed later in this chapter, are also designed to make it easier to develop survivable, flexible, and scaleable systems.

11.3.3 Example of a General Integrated Architecture

Figure 11-7 shows an integrated architecture that includes a hybrid network for access to mobile, positional and voice applications, in addition to the traditional Web access. This represents an actual running system that initially provided only Web access over the public Internet to enterprise applications and partner databases/applications, shown on the right side of the diagram. With time, the same information was needed by the users through cellular networks and 802.11 LANs. In addition, the new users needed voice and positional support. The main requirement was that the existing system was to stay intact – the new users and applications had to be integrated into the existing system.

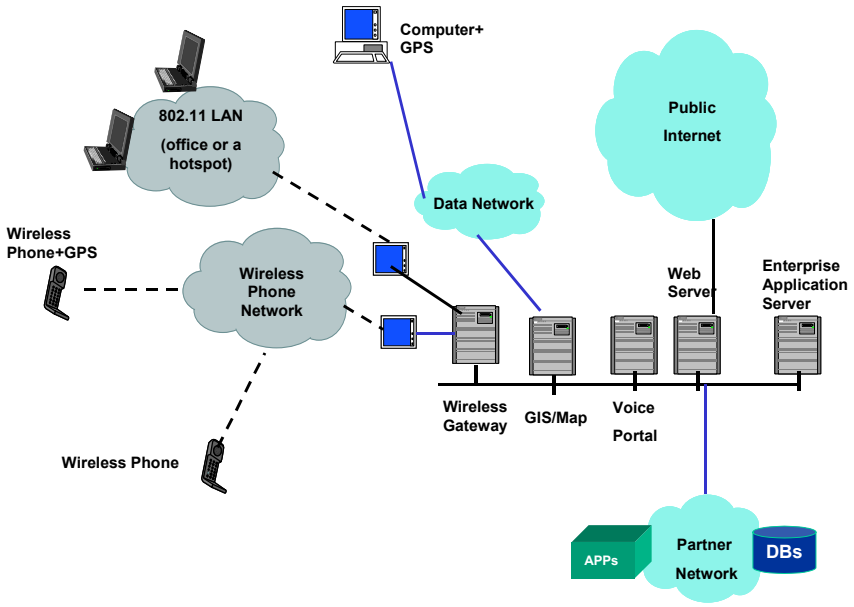


Figure 11-7: Positional and Voice Commerce

To satisfy these requirements, two additional servers were introduced: a GIS (Geographical Information System) map server and a Voice Portal. In addition, a wireless gateway was introduced to handle mobile users. All wireless devices from the cellular and 802.11 LANs (with and without GPS support) are connected to the wireless gateway, which consults a GIS map for GPS support. The wireless gateway serves as the main integration point for all wireless devices and also provides security services for mobile users. Some wired devices with GPS support are directly connected to the GIS map sever. The wireless gateway also does interactive voice response, voice recognition and speech generation. The Voice Portal provides voice menus or directories for users to select or traverse services. Based on this information, the back-end applications are accessed. These applications may reside in the Enterprise Application Server that provides various transaction services (e.g. shopping carts, form requests) or may be part of a trader network.

The main idea of this architecture is that different servers are developed for specialized and new services. As new services are needed, more or new servers can be added to this architecture. Thus, the servers and wireless gateway become integration points and satisfy the survivability, flexibility, and scalability requirements of the enterprise.

Time to Take a Break
- Concepts and Vision
- Levels of Integrated Architectures
- Component-based Architectures

11.4 Integrated Network Architectures for Wireless – Layer 1 to 2 Issues

Figure 11-8 shows a layered view of the network technologies that form the hybrid "Next Generation Networks." At the lowest level (layer 1 and 2 of the OSI Model) are the wired and wireless networks that consist of home access networks, fast packet switching systems that reside on fiber optics, and wireless networks. The key point about the lower-layer network technology developments is that the IP stack runs on top of all the physical networks (wired or wireless). Thus the IP user does not need to know the underlying physical network. This helps us to provide seamless integration at the IP level through Mobile IP (see next section). However, network integration at lower layers is still needed. Specifically:

- Adapter cards are needed to detect and connect to the different type of networks.
- Network interconnectivity devices, especially protocol converters and gateways, are needed to glue together networks from different vendors, featuring different protocol stacks and utilizing different communication technologies.

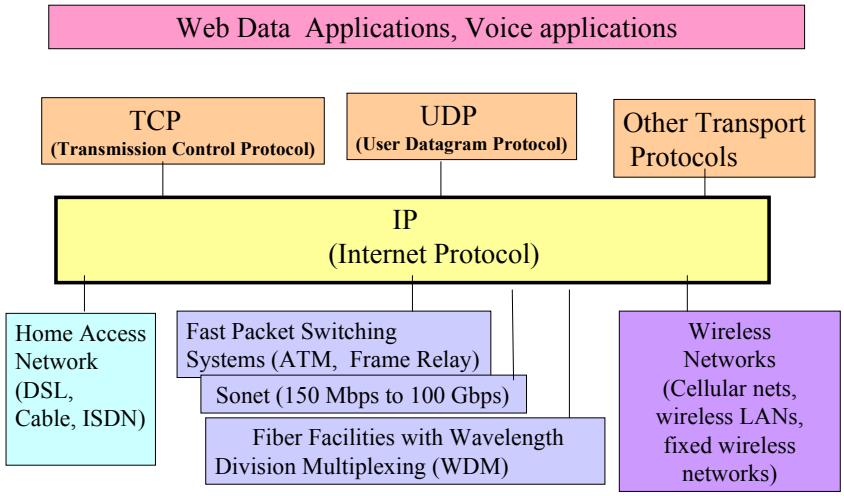


Figure 11-8: Network Technology Layers for Next Generation Networks

11.4.1 Adapter Cards for Detecting Different Wireless Systems

At the lowest level, suitable adapter cards are needed that can detect if a user is in a different coverage area. One of the most popular examples of these cards are the "multi-band" cards for GSM that recognize the following different GSM frequencies and work accordingly:

- **GSM 900** (simply known as GSM) – operates in the 900 MHz frequency range and is the most common in Europe and the rest of the world.
- **GSM 1900** (also called PCS-1900, or DCS-1900) – is the only frequency used in North America for GSM.
- **GSM 1800** (also called PCN or DCS-1800) – operates in the 1800 MHz frequency range, and is found in a rapidly increasing number of countries including France, Germany, Switzerland, the UK, and Russia.

Dual-band (900-1800 and 900-1900) phones and tri-band (900-1800-1900) are commercially available to provide support everywhere. These cards allow a customer to use the same GSM phone that works in Europe as well as the US – a miracle! Adapter cards are also available that can recognize GSM as well as CDMA cell frequencies and switch between the two. The challenge is to build adapter cards that can recognize cellular as well as 802.11 signals and switch between the two as a user moves from an 802.11 network to a cellular coverage area. The technical problem is to build cards that can recognize signals from the 1 GHZ to 5 GHz range (recall that cellular phones operate in the 1 GHz range while the 802.11 LANs operate in the 2.4 to 5GHz range).

More research and development is needed to develop the radio-controlled multi-network adapters that can operate in very diverse wireless environments. Companies such as Padcom, Motorola, Cisco, and Ericsson have demonstrated the technical feasibility of such cards. Rapid developments in this area are expected as several companies develop multi-radio, multi-network chips.

11.4.2 Network Interconnectivity Devices and Protocol Converters

Network interconnectivity devices such as hubs and switches are used commonly to connect different network elements of a network. For example, an Ethernet hub or switch is used to connect Ethernet devices together. See Appendix A for a review of these devices. The main challenge is hybrid networks where different subnetworks need to be interconnected. In hybrid networks, protocols of one subnetwork need to be converted to protocols of other subnetworks for end-to-end communications. A network gateway connects two dissimilar network architectures and is essentially a protocol converter. A network gateway can convert protocols at any layer to achieve interoperability between two different networks. The issues to be discussed include the following:

- How does an 802.11 wireless access point communicate with an IP router or an 802.4 hub or switch?
- How is GPRS added to GSM networks?
- How do wireless protocol converters work?

Consider, for example, how an 802.11a device communicates with an 802.11b access point (AP). Suppose you have a few 802.11a devices in a predominantly 802.11b network. You have the following choices:

- Use an 802.11a-to-802.11b protocol converter to convert frequencies, protocols, etc.
- Let the 802.11a devices talk to an 802.11a AP, and the 802.11b to an 802.11b AP. Now a multi-protocol router could handle the traffic.
- Run IO on top of both and then let IP hide the underlying differences.

Network gateways convert protocols between different network protocol stacks. However, a gateway may convert any type of protocol. For instance, email gateways allow different email packages to exchange email by converting one format to another. An example is the Softswitch Mail Gateway which converts many email protocols for email exchange. Figure 11-9 shows a gateway that converts Net1 protocols to Net2. Keep in mind that a network gateway is a function and not a device. This function may be performed by a special purpose dedicated computer, an application-specific integrated circuit (ASIC) chip, a workstation with associated software (e.g., a Sun Sparc with gateway software), or a software module which runs as a task in a general purpose computer.

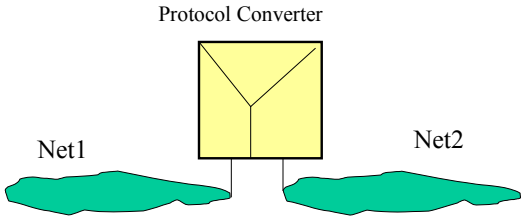


Figure 11-9: Protocol Converter

11.4.3 Network Integration Examples

11.4.3.1 Example: City of Oakland Police Department Deploys LAN/WAN Mobile Data Solution

The City of Oakland, California, provided its law enforcement officers with high-speed access to large data files from their patrol vehicles. This high-speed access was viewed as a way of boosting the efficiency of police officers, because previously they had to wait for hours for mug shots or detailed crime reports over slow cellular networks. The City's police force is now equipped with over 220 squad car computers enabling officers to receive large files via a wireless LAN available to them in parking-lot sites ("parking hot spots") citywide.

The solution is based on Padcom's TotalRoam Ellipse™ software platform, which seamlessly supports the data exchange between wireless LANs and WANs. TotalRoam's Ellipse combines the 802.11 Wireless LAN capabilities with the Motorola RD-LAP network interfaces for WAN transmission. When a patrol vehicle enters any of the 802.11 coverage areas, officers are able to begin large file downloads. The officers can interrupt the process if they are called to a crime scene, and later resume the process when they return to any one of the LAN-served parking lots. This hybrid network implementation by the City of Oakland is a good example of wireless network architectures that utilize a wireless LAN in a local area environment and a wireless WAN only outside the LAN coverage area.

Source: http://www.mobileinfo.com/news_2001/issue22/oakland_police.htm

11.4.3.2 Example of an Integrated Network Architecture

A medium-sized manufacturing company, with headquarters in Ohio, wanted to develop high-bandwidth and wireless networks for future applications. The management was really interested in exploring the use of new and innovative applications in voice, data and images for office automation, factory floors and engineering design work. There was a particular interest in using wireless LANs in a corporate setting. A network architecture task force was established to:

- Develop a physical communication layout for broadband and wireless networks

- Establish a network interconnectivity plan that interconnects various wireless LANs to a private ATM network and the public Internet

The task force proposed the integrated network architecture shown in Figure 11-10. This hybrid architecture includes wireless as well as broadband (ATM) networks in a corporate intranet and shows a long range conceptual view. At the core of this network is a corporate backbone (a Fast Ethernet LAN – an FDDI LAN could be used here also). This backbone is connected to the wireless access points and the ATM switch. An Internet gateway/firewall is also connected to this backbone to provide access to the public Internet. In architectures of this nature, the corporate backbone serves as the "integration bus."

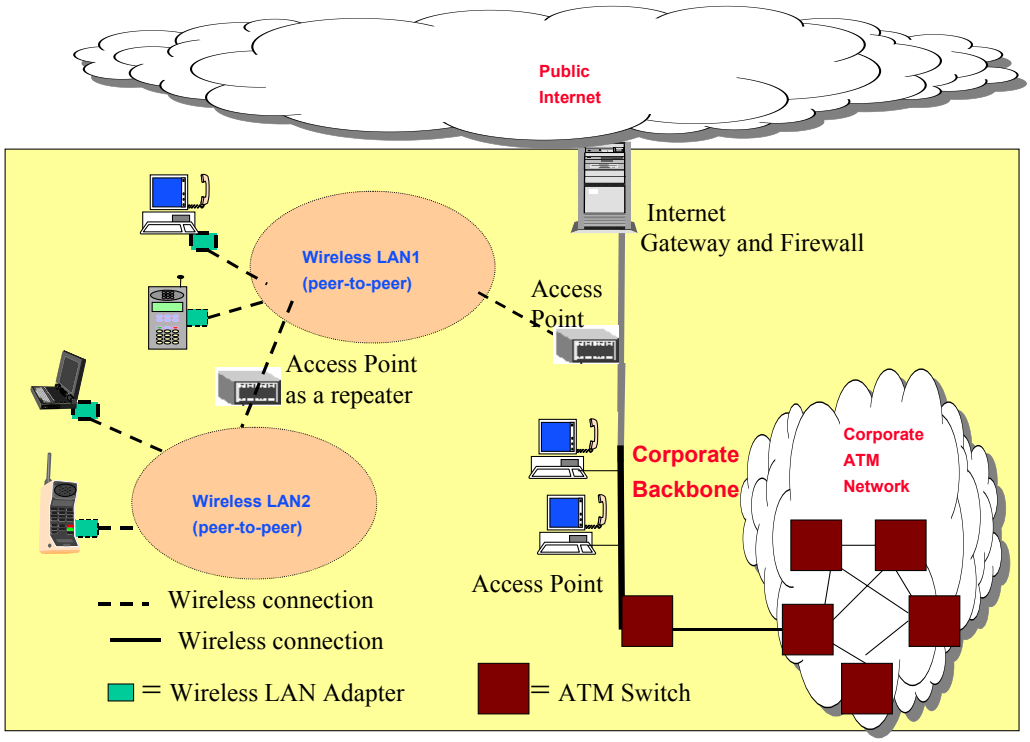


Figure 11-10: A Possible Configuration

11.4.3.3 Example of Network Integration Engineering and Consulting Services

Integration of wireless and wired network elements (NEs) into working networks is a complex task that requires a great deal of specialized knowledge. Due to market demands for continual network integrations, several companies such as Lucent Technologies, Nortel and Cisco provide wireless network integration services. Examples of these services include:

- Configuration, installation, testing and troubleshooting of the various NEs and their interconnections to verify that all NEs work together with each other and other interfacing equipment
- Engineering/re-engineering of MSCs (Mobile Switching Centers) to implement the smooth deployment of new 3G MSCs
- Mobility Manager HLR/VLR migration services for the migration of HLR/VLR functionality from one system to another
- Integration and integration of CDMA services with TDMA
- Integration of legacy wireless (e.g., 1G) with 3G

11.5 Roaming and Mobile IP for Wireless Integration

11.5.1 Roaming Revisited

A critical function in wireless networks is roaming, which allows wireless users to move from cell to cell. Roaming support is well known in cellular networks, especially GSM networks which provide universal roaming. But the situation is different in WLANs. For example, the 802.11 standard does not specify roaming. Thus roaming in 802.11 LANs is left to the LAN vendors.

Companies that manufacture wireless access points (APs) have implemented their own flavors of roaming and have partnerships to provide broader roaming support. For example, if you have an AP from one company, then you can roam if the other APs in your network are from the same company or its partners. An ***Inter-Access Point Protocol (IAPP)*** was developed by several companies, led by Lucent. Over time, IAPP has become a de facto standard protocol for roaming between WLANs. While IAPP partially addresses roaming between WLANs, roaming between WLANs and cellular networks is a different issue and is addressed by Mobile IP.

11.5.2 Mobile IP as a Basis for Integration

Mobile IP, discussed in a previous chapter, is needed to maintain connectivity as the users move from one type of network to another. The main idea of Mobile IP is that the same Internet connection address is maintained as you move your mobile device from one location to another. As discussed previously, the traffic is forwarded by using a "care of" address (see Figure 11-11). Let us assume that a laptop computer C1 is assigned to an office WLAN, called the home network for C1. The IP address on the home network is called the home address for C1. Suppose that the mobile user owning C1 is taking a flight and has moved to a hotspot WLAN at an airport, known as a foreign network. Mobile will maintain the connection by first letting C1 register with a network node on the foreign network. This node is called a foreign agent. The foreign agent takes responsibility for C1 and gives its address as a "care of" address to an agent on the home network, called the ***home agent.*** As illustrated in Figure 11-11, the home agent gets all the incoming traffic for C1 (step 1); the traffic is routed to the "care of" address – the foreign agent (step 2); the foreign agent routes the traffic to C1 in the foreign network (step 3); and the return traffic is routed back to the IP server without having to go through the home agent (steps 4 and 5).

11.5.3 Examples of Mobile IP

Mobile IP is used when a user roams to an IP address that is different than the one that has been loaded into his/her device. This implies that the Mobile IP does not work to handle roaming where some devices do not have IP addresses. For example, if a cellular network only supports voice users over GSM, then Mobile IP cannot help us. Mobile IP also differs from dynamic addresses. See the sidebar "Dynamic IP Addressing". Mobile IP is used commonly in wireless networks where a user may move from one network to another without causing a disconnection. For example, suppose you are doing a large file download that may

last 20 minutes. Then, if you move from one LAN cell to another, Mobile IP will keep you connected and will not interrupt your file transfer.

Mobile IP has not been deployed widely at present; however, many experiments at universities to evaluate the performance of Mobile IP have been conducted. An example is the FSA (Full Stack Adaptation) project at the University of Florida [Helal 2000]. This project built a testbed that showed how Mobile IP works in a hybrid network (802.11, fast Ethernet, and iDEN) environment. The tests concentrated on measuring handoff performance of Mobile IP. Another example is the wireless integration experiments using Mobile IPv6 conducted at Cambridge University [Chakravorty 2003]. This project built a WLAN-GPRS testbed and evaluated handoff performance of Mobile IPv6 for TCP connections.

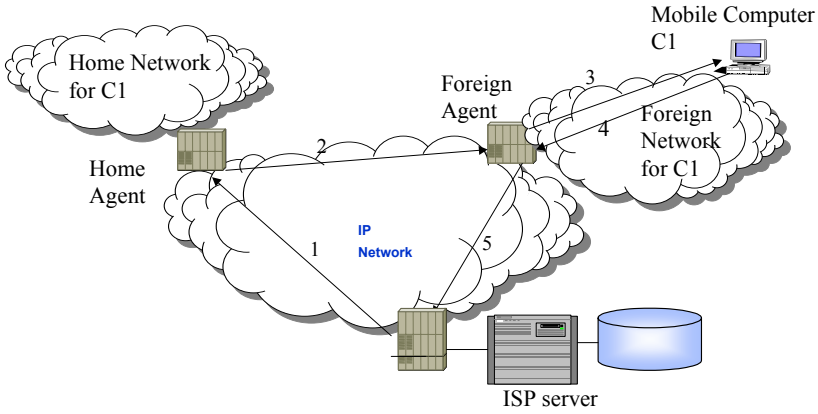


Figure 11-11: A Mobile IP Scenario

Dynamic IP Addressing

Dynamic IP addresses are supported by DHCP (Dynamic Host Configuration Protocol). DHCP automatically assigns an IP address when a device is turned on. But once a user moves out of the DHCP range, for example the building where DHCP has been installed, then you have to configure a new IP address. Consider the scenario where John works in an office with DHCP support. When John arrives to the office in the morning, he connects his laptop to the Ethernet cable and gets an address through DHCP. But John also teaches at a local university, and when he goes to his university office, he connects his laptop to the university network and gets another IP address from the university DHCP. But if John goes to a hotel that does not have DHCP support, then he cannot connect without using the IP address assigned by the hotel.

11.6 Middleware and Mobile Computing Platforms for Integrated Architectures

11.6.1 Overview

Mobile computing platforms, discussed in Chapter 4, can play a fundamentally important role in building integrated architectures of the future. These platforms support the development as

well as operation of mobile applications such as mobile messaging, m-commerce, m-business, and many others discussed in Chapter 2. Horizontal integration at the mobile computing platform level, discussed in Section 11.2.3, resides above the underlying heterogeneous networks and thus can hide the complexity of the underlying network from the applications. Broadly speaking, these platforms provide the following type of services:

- Local services that support the individual mobile devices. These services consist of operating systems needed to run the mobile devices, and local system software services such as database managers, transaction managers, and utilities for mobile devices. Common local services that can be deployed on multiple mobile devices make it easier to develop applications that can be deployed on multiple devices (build once, run everywhere). These services may, for example, support large display screens and may also include security features such as SSL (Secure Sockets Layer) support.
- Wireless middleware services, also known as interconnectivity software, that interconnect mobile users, databases and applications with each other. Middleware components are *business/industry-unaware* software modules that provide a variety of interconnectivity services such as remote access to a corporate database from a mobile phone. These services, as we will see later, perform conversions between different formats (among other things) to present an integrated view.
- Mobile computing platforms, also known as "mobile application servers," package a variety of wired/wireless middleware services into a single framework that can support the current and future breed of mobile applications. Examples of these platforms are Nokia WAP (Wireless Application Protocol) Sever, Sun J2EE/J2ME (Java2 Micro Edition), Oracl9iAS, and IBM Webshpere.

11.6.2 Wireless Middleware Services

Middleware services for mobile computing applications, as discussed in chapter 4, have a vital role in wireless integration. Unique middleware services are needed because wireless devices, especially handhelds, are resource-constrained, have battery restrictions, and communicate over slow, unreliable networks. The 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 know the underlying network characteristics. Because middleware provides standard interfaces to operating systems and applications, it is a good basis for integrating wired and wireless systems. Over the years, a variety of middleware services have been introduced. We have discussed these services at length in Chapter 4. Here is a quick recap.

The following common features of wireless middleware products are important for integrated architectures [Vichr 2001]:

- **Transformation**: The middleware transforms data from one format to another (e.g., HTML to WML). The transformation may be intelligent enough to transform different types of data to different types of devices. For example, it can produce VXML or WML depending on the type of device.
- **Detection and storage**: Wireless middleware products can detect and store mobile device characteristics in a database. Upon detecting the type of mobile device or channel being used (e.g., GSM or 802.11 frequency range), the middleware can optimize the wireless data output according to device attributes.
- **Optimization**: Middleware products can compress data to minimize the amount of data being sent over a slow cellular wireless link.

- **Message delivery**: Middleware can store and forward messages if the user is disconnected from the network.
- **Security**: Security features can be imbedded in wireless middleware to ensure end-to-end security. For example, digital certificates for handheld devices can be managed by a middleware service. We will discuss wireless security in Chapter 12.
- **Operation support**: Middleware can offer network and systems management utilities and tools to allow monitoring and troubleshooting wireless devices and networks. We will discuss this topic in the "Management Platforms" section of Chapter 13.

A variety of general-purpose middleware services have emerged over the years (see Umar [2004]).[5] Examples are CORBA, DCOM, MOMs (message-oriented middleware), and others. Many of these have been extended and specialized for wireless. For example, Wireless CORBA was specified by OMG as a specialization of CORBA. Similarly, the currently popular Web Services have been specialized into Mobile Web Services (we will look at these later). In the same vein, Sun's J2ME (Java 2 Micro Edition) has been specified as a family member of the Sun J2 EE (Java 2 Micro Edition) product line.

11.6.3 Mobile Computing Platforms (Mobile Application Servers)

At present, various middleware services are being packaged together into mobile computing platforms, also known as mobile application servers. For example, typical mobile computing platforms include support for cellular networks, location-based services, voice support, and transformation of content to/from different formats. These platforms can play a key role in building integrated applications because they include wireless and wired middleware services. Figure 11-12 shows the conceptual components of a Mobile Application Server (MAS), introduced in Chapter 4. A MAS is part of a multi-tier (mostly three-tier) architecture that typically consists of a thin client tier residing on handheld devices; a middle tier consisting of business applications and a set of middleware and network services; and a back-end tier consisting of mission-critical databases and applications.

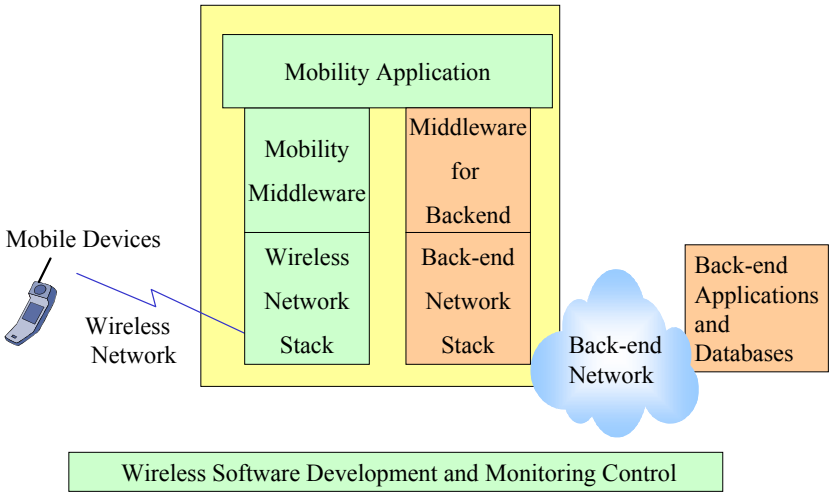


Figure 11-12: Mobile Application Server – Conceptual View

In general, mobile application servers act as an architectural extension to functions that a particular middleware is unable to provide – they compensate for the features missing in

[5] Umar, *e-Business and Distributed Systems Handbook: Middleware Module*, 2nd ed., NGE Solutions, 2004.

either direction. For example, they add location-based services to conventional applications and provide message queuing and transaction processing for wireless applications. This allows wireless devices to support financial transactions and engage in m-commerce activities. In addition, an application server might participate in the intelligent transformation of Web-based application interfaces.

11.6.4 Example of a Mobile Application Server – Oracle9iAS Application Server for Wireless

A large number of application servers, initially known as middleware platforms, are commercially available that combine mobility, e-commerce, and B2B trade. See Umar [2004][6] for details. We reviewed several examples of mobile application servers in Chapter 4. Let us review, for illustrative purposes, the Oracle9iAS Application Server as an example of a MAS. Oracle9iAS relies on the partner community to supply various technology components. It provides a Wireless Integration Kit for partners to easily integrate the wireless services of the Oracle9i Application Server. In particular, it provides the following features (see the Oracle site for additional details):

- **Integration with Location-Based Services (LBS).** This allows the developers to integrate location services such as driving directions, mapping, routing, alerting and advisories to a mobile user. Oracle9iAS Wireless does not necessarily perform the location services itself, but instead it relies on external location service providers for Geocoding, Mapping, Routing, Traffic Services, and Yellow Pages.
- **Messaging and Alerting Integration**. This involves details on how to develop drivers for different messaging paradigms such as SMS, Fax, MMS, etc. Oracle9iAS Wireless contains a messaging subsystem that handles the sending, receiving, and routing of messages. The messaging subsystem uses a collection of drivers to handle the device-specific or communication protocol-specific processing. Drivers are available for SMS gateways, email, voice, and other messaging systems.
- **Multi-Channel Server Integration**. These provide server APIs that can be used as technology providers and system integrators to develop new features. These APIs can be used to provide services such as billing support, data mining and provisioning.
- **Voice gateway integration** – Oracle9iAS generates W3C-compliant VoiceXML code. It also specifies the process for writing grammar drivers that can be used in VoiceXML applications. The server also facilitates development of new voice applications.

11.7 Integrated Application Architectures for Mobile Services

11.7.1 Overview of Mobile Application Architectures

Horizontal integration at the mobile applications level, as discussed in Section 11.2.3, has the benefit that the user is completely unaware of the network as well as platform heterogeneities. For example, if a user can access a wide range of information located on a hybrid network from a single application from the same device, then the user gets an integrated view even though the underlying systems are not integrated. Mobile applications, broadly speaking, can be designed in at least three different ways to meet the needs of different types of users:

[6] Umar, *e-Business and Distributed Systems Handbook: Platforms Module*, NGE Solutions, 2nd ed., 2004.

- **"Standalone" Mobile Applications.** These applications are standalone because they are self-contained in a wireless network and do not access any back-end or remotely located systems. Many of these applications operate in mobile ad-hoc network environments such as Bluetooth. For example, many Bluetooth applications replace cables and allow Bluetooth-enabled printers, keyboards, modems, and computers to exploit device proximity and discovery protocols to connect to each other. Similar applications exist in wireless sensor networks and "zigbees." Typical applications of this type run on self-contained clusters of wireless devices without connecting to an external infrastructure.
- **Ubiquitous Access to Existing Applications.** These applications exploit the MPTV (mobile, positional, TV, and voice) capabilities of wireless devices to broaden access to the existing applications and databases on fixed networks. The wireless devices interact with the user and present information in a variety of formats after communicating with the existing applications over a wireless network. In many cases, mobility is another feature added to existing applications. Examples are the m-commerce applications and MEBAs (Mobile Ebusiness Applications) as described in Chapter 2.
- **Mixture.** Many applications combine the standalone applications with existing applications to provide powerful services. For example, a mobile ad-hoc network can include a Bluetooth-enabled gateway that accesses a supply chain management system.

Figure 11-13 shows a conceptual view of a generic architecture that can be customized to represent these three architectures. The main differentiating feature of this architecture is the front-end integration layer. This layer must perform several functions: a) coordination between mobile devices, b) view integration for various devices, and c) mobility-specific processing. Coordination between mobile devices can be used for standalone applications. View integration takes into account the various types of mobile devices that need to access back-end applications. For example, the mobile Web browsers, telephones, PDAs, and palm pilots must be supported so that they can access the same type of back-end applications and databases as the wired ones. Mobility-specific software is responsible for "roaming support" (e.g., the GIS Map for GPS) and provides, if needed, uniform access of CRM, ERP and proprietary or custom-developed business/commerce applications. This can be used in all three architectures and should include the following:

- Data synchronization between PDA devices, laptops, and server-based systems;
- File distribution, including the ability to refresh commonly used or accessed data files with updated information;
- Software updates and software distribution, particularly important for client-server-based enterprise applications that require periodic upgrades; and
- Portal-based entry into data repositories, which will provide a central point of presence for both internal and external users to identify, retrieve, and synchronize information.

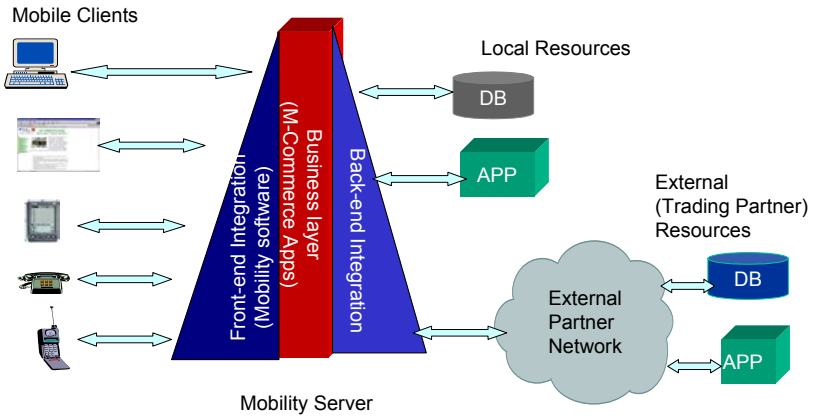


Figure 11-13: Conceptual Architecture for Mobile Applications

Figure 11-14 shows a closeup of the front-end integration layer. In this view, two gateways convert the WAP and Voice XML content to standard HTML/XML before it is sent to the Web server. Thus the Web server, and consequently the back-end applications, do not know whether the users are communicating over wireless or wired networks. These gateways serve as the wrappers or adapters that hide the wired versus wireless communications from the back-end applications.

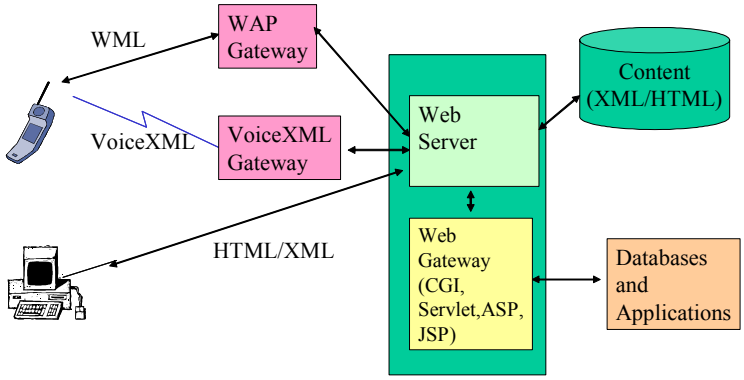


Figure 11-14: A Closer Look at Front-End Integration

11.7.2 Integrated Application Architecture Examples

Numerous examples of applications that integrate wired with wireless networks have been built and deployed. An example is a custom sales management system that runs on hand-held computers by linking its applications with its back-end sales system. Another example is a wireless solution that wraps wireless access around existing business applications (accounting, inventory management or customer relationship management). This reduces the need to revamp existing systems. Yet another example is a custom sales management system that runs on hand-held computers and is linked with back-end applications that run on an internal AS/400 system. The company, Atomic, used Dharma Systems' eUnify to achieve this. Here are some additional examples.

11.7.2.1 Short Messaging Services

SMS architecture, shown in Figure 11-15, integrates several technologies to provide a seamless service to the end-users. A Short Message Service Center (SMSC) directs all short

messages to and from the mobile phone by using a store and forward model. The SMSC finds the roaming customer by consulting a "home location register" (HLR) and transfers the message to the mobile device if the device is active. The SMSC software resides in the operator's network and manages the billing services. Many operators offer Web-based interfaces to their SMSC so that the users can send short messages to any mobile phone from the Web.

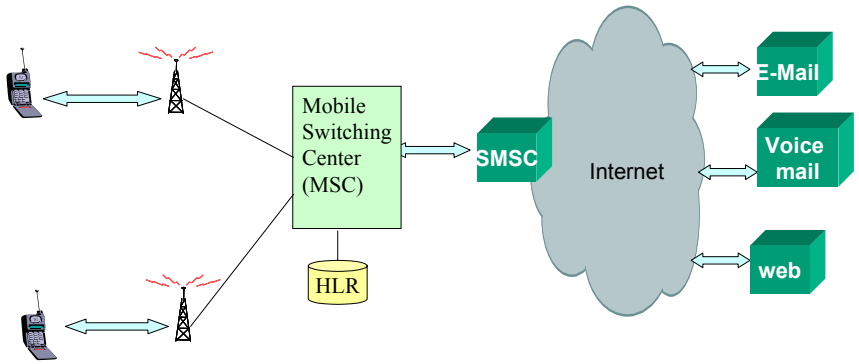


Figure 11-15: SMS Architecture

11.7.2.2 Mobile Purchasing

Figure 11-16 shows a simplified view of mobile purchasing that shows how an online purchasing system can be invoked from a mobile device. The wireless gateway translates the wireless protocols and presentation (e.g., WML over GSM) to the standard HTML over HTTP. Within this overall architecture, several configurations are possible. For example, the seller may only open up wireless access to the front-end applications (online-order processing application residing on the Web server) – all back-end applications are not accessed from the wireless devices. This has security benefits. The seller may also include location-based services to provide special advertisements for special geographical areas. Notice that the external suppliers are not directly exposed to the mobile users.

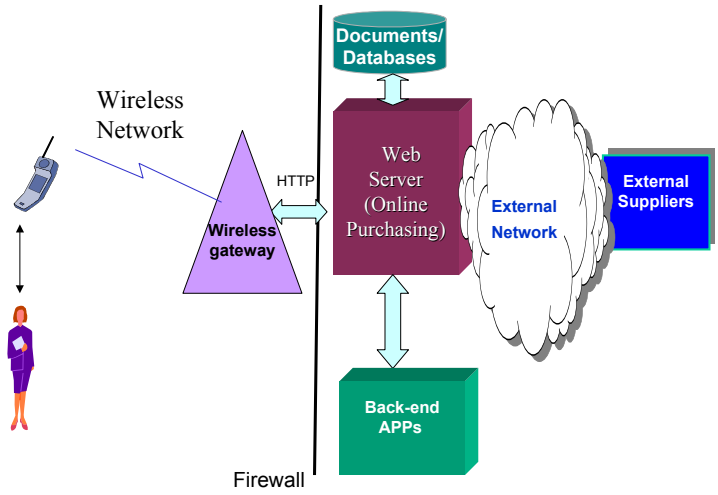


Figure 11-16: A Simple Internet-based Purchasing System

11.7.2.3 Mobile Portal Architectures

A mobile portal, as discussed in Chapter 2, is a consolidated information channel for the mobile customers. We discussed examples of several mobile portals in Chapter 2, including

the portals by DoCoMo, GM, and American Express. From an architecture point of view, these portals usually consist of 3 tiers: a mobile device tier, a gateway tier, and a portal data tier.

An example of the multi-tiered application architectures for mobile portals is the prototype that Intergraph built for the US Air Force (www.intergraph.com/solutions). The portal, shown in Figure 11-17, was developed at Intergraph's secure lab facilities in Huntsville, Alabama, to verify the concept and determine appropriate hardware, software, and wireless products. IBM Websphere with wireless extensions, called WebSphere Everyplace, was chosen as the main platform. The prototype architecture involves three tiers – a device tier, device access tier, and application tier. The device tier includes handheld devices installed with Web browsers, and WebSphere Everyplace Wireless Clients to provide communications with the second tier. The device access tier includes the Wireless Gateway Server, which receives the encrypted wireless signals, interacts with the Air Force Portal, and transmits information back to the wireless devices. The application tier consists of the Air Force Portal and directory services. Due to the sensitivity of information, the prototype included several security features.

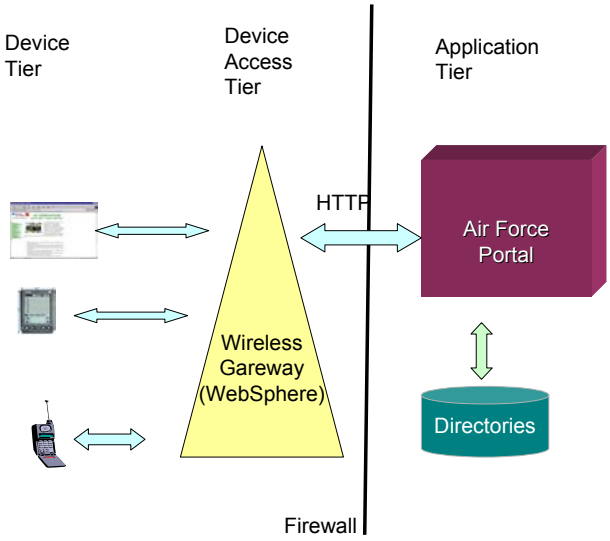


Figure 11-17: Air Force Portal

11.7.2.4 A Location-aware Personal Information Management System

Consider a system where the corporate network contains employee schedules and also information about its customers. In addition, a location-based service (LBS) is used to locate, through a mobile network operator, the proximity of sales staff. Due to privacy issues, mobile operators expose mobile phone location information only to people who have the permission of an individual for their location to be found. Let us assume that the LBS is powerful enough to support users as they roam through a hybrid network. This implies that multiple mobile operators have agreed to support roaming and to share the location information among them.

The personal information management (PIM) application that relies on universal LBS can be a very powerful tool to locate a highly mobile sales force and customer population. For example, a sales manager can detect that one his sales reps is in the same area where his major customer is. She can then arrange for a meeting between the two.

Privacy and security are the main issues in such systems, naturally. An extensive security system is needed to ensure that only mutually authorized and authenticated people can access this location information. For example, Bob specifies that his wife Pat and his boss Sally are the only ones who can view his location information. So when Sally tries to find where Bob is this afternoon, the PIM will query the security module and will authenticate Sally and verify if Bob has authorized her to access his information.

Time to Take a Break
- ✓ Concepts and Vision
- ✓ Levels of Integrated Architectures
- Component-based Architectures

11.8 Technology Trend – Component-based Integrated Architectures

No discussion about integrated architectures at the time of this writing is complete without recognizing the major technology trend of component-based systems and the platforms to develop such systems. These developments, in particular Web Services and Mobile Web Services (discussed later in this section), have the potential to provide reusable components that run on PDAs, laptops, or mainframes and can be invoked *globally* through HTTP, XML, and other Web-based standards. For example, a component for wireless-specific functionality such as location-based services could be developed and only included to support mobile devices. Similarly, components have been developed for speech recognition, GPS support, format conversions, and other commonly used features. If you could develop, for instance, a customer information system as a component, then this component could be reused in many mobile or fixed applications that need to access customer information. This section gives a quick overview of the main ideas.

11.8.1 What is a Component?

For the purpose of this book, we adopt the definition by Herzum and Sims [Herzum 2000] and define the term "software component," normally referred to as just a component:

> ***Definition:*** A ***component*** is a self-contained piece of software that has a well-defined interface and can be developed, delivered, installed, and run independently.

In addition, a component must have the following properties (see Figure 11-18):

- A component is built so that it can be easily composed of and combined with other components for enterprise needs. A standalone component that cannot be combined with others to build an application is of limited value.
- A component has a socket as a software interface with the infrastructure. For example, a component may have a socket for .NET or J2EE.

A good analogy to explain components is the way personal computers are built from a collection of standard components such as memory chips, CPUs, busses, keyboards, mice,

disk drives, monitors, etc. Because all of the interfaces between components are standardized, it is possible to mix components from different manufacturers in a single system. Similarly, the goal of component software is to standardize the interfaces between software components so that they too can work together seamlessly. Many desktop tools in the Microsoft Office Environment are currently available as components. Examples of typical desktop components are spell-checkers, format painters, calendars, print managers, etc. Conceptually, each icon on your toolbar can be a separate component. These components can be combined to build applications (you could find the same spell-checkers in MS Word as well as PowerPoint) and can plug into the infrastructure environment (Microsoft DCOM/COM+). In addition to Microsoft DCOM/COM+ components, several Java components, known as *Java Beans*, are also commercially available to display clocks, calendars, etc. The Java Beans are very popular in Unix desktop environments.

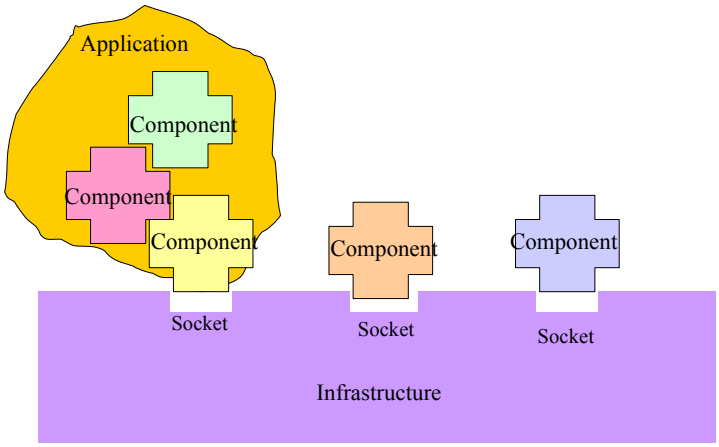


Figure 11-18: Conceptual View of Components

Components should not be thought of as small desktop tools only. As we will see shortly, *components can be large business modules ("business components") that can be used to build enterprise applications.* In fact, applications such as online purchasing can be thought of as components, if they have well-defined interfaces, and can be combined to build other applications. We can then think of a "mobility" component that can be combined with an online purchasing application to support m-commerce.

The components can exist at several levels. The following granularities, going from fine to coarse, will get us started (see Figure 11-19):

- **Primitive Component**: This is the lowest-level component that performs one specified function such as displaying a stock quote. This component resides on one machine, but is network-addressable (i.e., it can be accessed remotely). An example of a component for wireless is a WML-HTML transformer that converts WML to HTML and vice versa.
- **Business Component**: This component implements a single *business* concept such as purchasing, payment, billing, etc. A business component usually consists of one or more primitive components and can be thus distributed across machines to satisfy business needs. For example, a business component that implements an inventory business functionality may consist of several primitive components that reside on different sites of an organization.
- **Enterprise Component**: This component implements an enterprise-wide business functionality and consists of one or more business components. However, the main restriction is that this combination of business components by itself satisfies the requirements of well-defined interfaces and composability. For example, a Human

Resource (HR) System, such as PeopleSoft, could be thought of as an enterprise component with business components that handle payroll, vacations, time reporting, etc.

- **Inter-enterprise Components:** These components are very large-grained federated systems that handle, for example, B2B functionalities. Inter-enterprise components may consist of one or more enterprise components. An example of such a component is a supply chain management system that may combine order-processing and inventory-management enterprise components from different organizations.

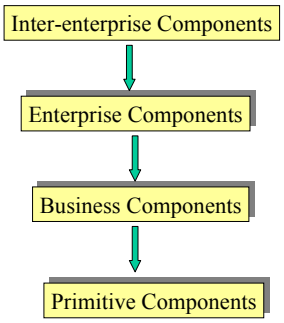


Figure 11-19: Granularity Levels of Components

Although components can be talked about at several levels, our main interest is in the primitive and business components and their role in building integrated mobile applications. Why? The primary reason is that, at the time of this writing, primitive and business components are a reality while higher-level components are not.

11.8.2 What is a Component-based Architecture?

The notion of using components to build mobile application architectures is appealing, but how can it be done? The main idea is to identify the individual components of the application, as well as the infrastructure needed to make this application operable in a hybrid wired/wireless environment. In essence, developing architectures of component-based mobile applications is like developing many mini-applications that need to interact with each other for corporate business goals. Figure 11-20 shows a simple view of component-based application architectures in terms of three logical tiers: the user tier that represents the user presentation logic, the business logic tier that represents the core business logic of the application, and the resource tier that represents internal as well as external databases and applications. It can be seen that components can fit in different tiers. In particular, one or more business components may implement the business tier, one or more primitive components may implement the user presentation logic, and some components may be used in the resource tier.

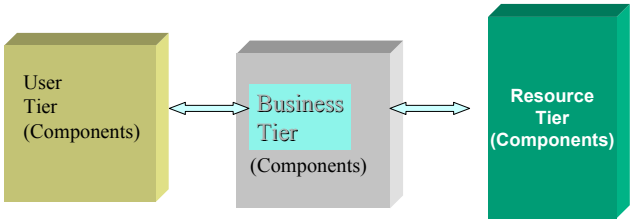


Figure 11-20: Simple View of Component-based Application Architectures

It may be argued that the notion of developing integrated architectures carefully is outdated because the work in component-based software will make it possible for us to assemble the

applications "on the fly." While this is an interesting notion and should be pursued diligently, we should note that component-based systems are not widely deployed at the time of this writing. In addition, we should be careful not to develop the generic wheel that can fit everything that moves (bicycles, cars, trains, and airplanes). A great deal of thinking is required to build the right components at the right granularity for the right type of applications. For additional information on this topic, the interested reader should consult the books by Askit [2001], Clemens [2002], Herzum [2000], and Umar [2004]. For the purpose of this book, we adopt the core idea of component-based software as a valuable effort in building mobile applications quickly. However, we recognize that component-based software will have to co-exist with older (in many cases, legacy) applications. Figure 11-21 shows a more detailed view of application architecture. Note that integration layers in the front end as well as the back end are needed for integration.

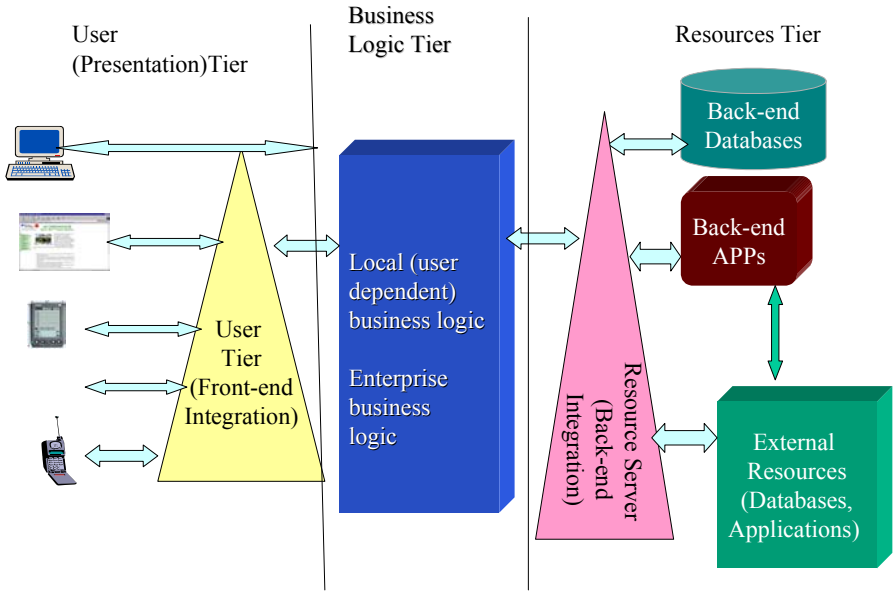


Figure 11-21: More Detailed View of Component-based Application Architecture

Components can be of different types and they can exist at several levels. Although our focus is on components that comprise business applications, components can also represent business-unaware services such as infrastructure services. Specifically, we can define the following infrastructure components (see Figure 11-22):

- **Middleware service components (MSCs)** that represent middleware services as components. For example, an e-commerce platform such as IBM's Websphere can be viewed as a middleware service component. Another example of MSC is a WAP gateway.
- **Network service components (NSC)** that represent network services as components. For example, Domain Network Service (DNS) could be represented as a component.

Notice that to qualify as a component, the infrastructure service must satisfy the following general component requirements; i.e. an infrastructure service is a component if it:

- Implements a well-defined, *business-unaware* concept and can be developed, delivered, installed, and run independently.
- Has a well-defined interface.
- Is self-contained.

- May rely on other components, but must not rely on implementation choices of those components.

What is the advantage of this component-mania? Well, ideally, now you can assemble an entire system (from applications to network services) from components supplied by different vendors. We are not there yet completely, but that is the general idea.

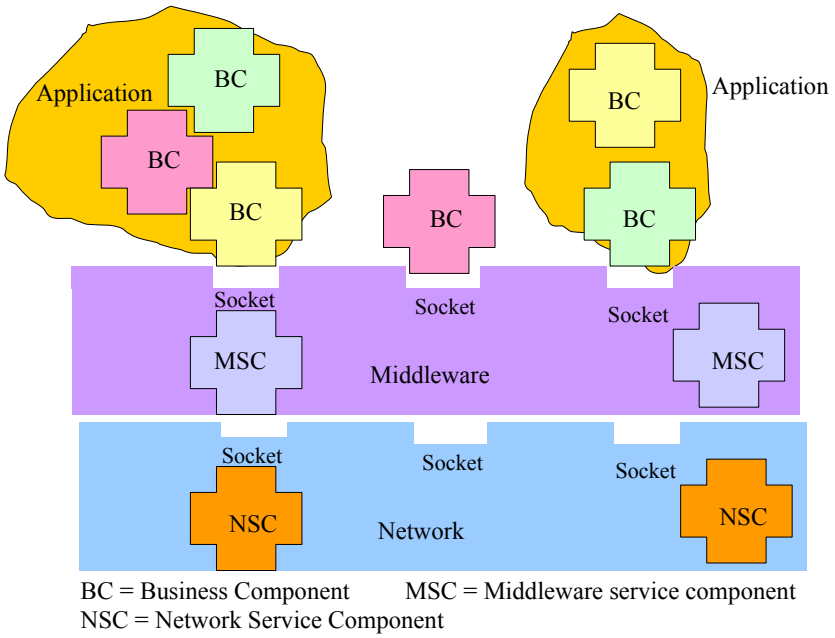


Figure 11-22: Components at Large

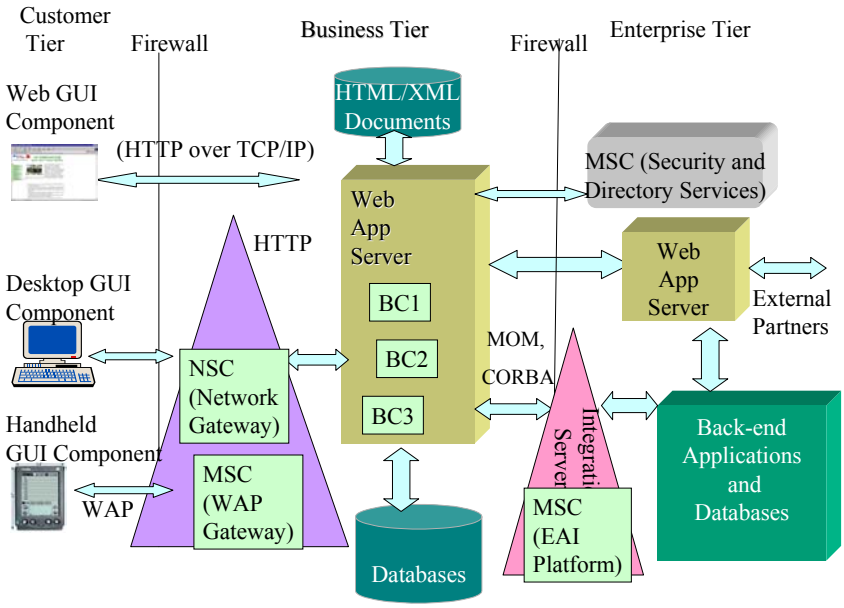


Figure 11-23: More Detailed View of Component-based Application Architecture

Figure 11-23 shows a physical view of the conceptual component model shown in Figure 11-22. The physical view represents a component-based application architecture that also shows some of the infrastructure services as components. In this case, the client tier consists of several components (desktop GUI component, Web GUI component, handheld GUI component). These components talk to the business tier that houses business components (BC1, BC2, and BC3). Note that the integration layers in the front end as well as the back end are populated with MSCs such as a WAP (Wireless Application Protocol) gateway and an EAI (Enterprise Application Integration) platform. The network gateway that translates non-HTTP to HTTP-TCP/IP can be viewed as a NSC.

11.8.3 .NET and J2EE as Component-based Platforms

One of the main reasons for attention to component-based architectures is that component-based platforms are commercially available at present. In particular, Web Services, also known as XML Web Services, are standards for component-based platforms. Sun J2EE and Microsoft .NET both use Web Services (for more information about J2EE and .NET, see the websites http://www.sun.com/ and http://msdn.microsoft.com/, respectively). Figure 11-24 shows a conceptual component-based architecture platform that is a generalization of the Sun J2EE, Web Services, and Microsoft .NET and is also very close to the architecture shown in Figure 11-21. Due to the popularity of these platforms, future mobile application architectures are increasingly relying on Web Services, and a major new initiative on Mobile Web Services is underway at the time of this writing. We will discuss Web Services and Mobile Web Services later in this section.

The component-based architecture is made up of the following components that can exist in different tiers:

- **Client-tier components** run on the client machine. For example, application clients (non-Web clients) and applets are client components. These components may or may not contain extensive business logic and so may be "primitive components."
- **Web-tier components** run on the Web server that acts as a "container." For example, Java Servlet, JavaServer Pages™ (JSP™), and ActiveServerPages (ASP) components are Web components that reside on the Web server. These components may contain business logic and may thus qualify as business components.
- **Business-tier components** run on the business tier under the control of enterprise component containers. The enterprise components that run on this tier are *definitely* business components. A well known example of these business components is the Enterprise JavaBeans™ (EJB™), currently supported by a variety of commercially available EJB containers from Sun, BEA and others.
- **Enterprise-system tier** software runs on the back-end systems. The enterprise information system tier handles enterprise information system software, and includes enterprise infrastructure systems such as enterprise resource planning (ERP), mainframe transaction processing, database systems, and other legacy information systems. These systems traditionally contain a great deal of business logic but they may not qualify as business components because they typically do not have well-defined interfaces and are highly interdependent on each other. However, you can convert these resources to BCs by adding "application gateways" that wrap them and enforce uniform interfaces.

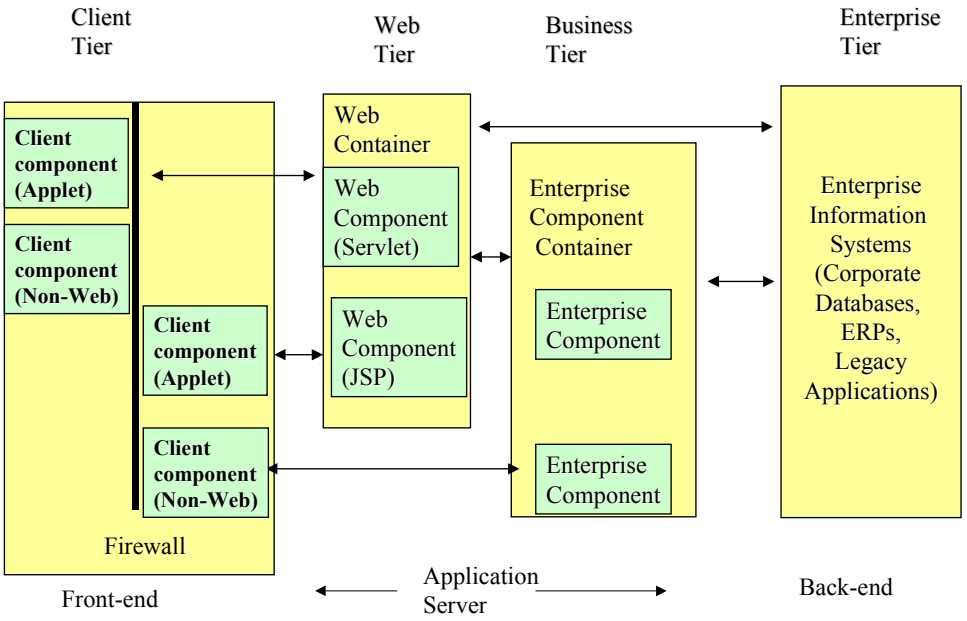


Figure 11-24: A Generalized Component-based Architecture

11.8.4 Web Services – The Cornerstone of Component-based Platforms

11.8.4.1 Web Services Principles

The main idea of Web Services (WS) is quite simple – combine Web and component-based systems into a single framework where the user-to-component as well as component-to-component interactions are conducted by using standard Web technologies. In other words, a WS component can be invoked from a browser or another WS component. XML is used for information exchange and messages are delivered by using HTTP. This provides a powerful paradigm for all types of platforms and hardware devices – from mobile phones and PDAs to laptops and back-end servers. By relying on the widely accepted Web and HTTP protocols as the basic communication fiber, Web Services can provide the glue between PC and mobile worlds. In essence, WS:

- provides application building blocks (business components) based on XML that connect to each other and other business components over the Internet
- supports mechanisms to describe the services provided by applications (business components) and to make the service available to others over the Internet
- relies on interface descriptions to invoke services from other applications without regard to how those applications were built, what platform they run on (handsets, desktops, or mainframes), and what devices are used to access them (PDAs or desktops)
- is designed to support application to application interactions over the Internet (unlike websites which are designed for human to application interaction)

Web Services promote a service-oriented architecture (SOA) that is currently at the foundation of Microsoft Dot Net (.NET) and Sun J2EE environments. In SOA, the needed services are published by service providers over the Internet. Once published with appropriate URLs, these services are accessible by service consumers via standard Web protocols such as HTTP. Since this is all done over HTTP, an application component (e.g., a payment system) in Atlanta can be combined with an order-processing component in Paris to handle a customer purchase request from Tokyo. URLs can be used to locate the application

components in a manner similar to locating Web pages. This can lead to globally distributed applications.

XML messages over standard Web protocols such as HTTP provide a lightweight and widely accepted communication mechanism that any programming language, middleware, or platform can participate in. Service providers register themselves in public or private business registries. The registered services fully describe themselves, including interface structure, business requirements, business processes, and terms and conditions for use. Visit http://www.xmethods.com/ for a listing of some interesting Web Services, and links to their accompanying documents. Consumers of the registered services read these descriptions to understand the abilities of these Web Services and then invoke them. Web Services consist of the following technologies:

- **XML and HTTP** are the core open Internet technologies that serve as the foundation of Web Services, The main source of information for XML and HTTP is www.w3.org.
- **WSDL** (Web Services Description Language) is an XML document that describes the location and interfaces a particular service supports. It is conceptually similar to IDL. The Web Services Description Language (WSDL) specification is available at http://www.w3.org/TR/wsdl.
- **SOAP** (Simple Object Access Protocol) is the message exchange protocol between Web Service consumers and providers. SOAP is conceptually similar to CORBA IIOP. SOAP is an XML-based protocol that has three major parts for defining a message exchanged between two components. The envelope defines a framework for describing message content and how to process it. The encoding rules define a serialization mechanism used to exchange application-defined data types. The remote procedure call (RPC) convention enables basic request/response interactions. The SOAP specification is available at http://www.w3.org/TR/SOAP/.
- **UDDI** (Universal Description, Discovery and Integration) is a directory of services on the Internet. This is conceptually similar to the CORBA/DCOM directory and naming services. UDDI contains XML-based service descriptions or business descriptions. It also defines an API for a service broker. More information regarding the UDDI initiative is available at http://www.uddi.org/.

It should be noted that only XML, HTTP, and WSDL are required technologies for WS. Once a service has been defined in WSDL, it can be sent as an email to a partner and the partner can invoke the service by using HTTP GET/PUT commands or SOAP. In this model, however, an email is needed each time there is a change to the WSDL. To avoid this, UDDI is used to show which services are available, where can you find them, and who supplies them. It shows the main features of Web Services.

Table 11-1: Main Features of Web Services

Features	Web Services
Components	Web Services Components
Interface Definition Language (IDL)	WSDL
Directory	UDDI
Brokering Services (location and binding)	Internet + UDDI
Invocation and Transport Protocol	HTTP, SOAP

11.8.4.2 A Quick Example of Web Services

Figure 11-25 shows the steps that take place to develop and use a phone service in a fully functioning WS application:

1. The provider creates, assembles, and deploys the phone service using the programming language and the software platform (.NET, for example) of the provider's own choice. It then defines the phone service in WSDL. A WSDL document exposes the Web Service to others.

2. The provider registers the service in UDDI registries. UDDI enables developers to publish Web Services and enables their software to search for services offered by others.

3. A prospective user finds the service by searching a UDDI registry. The consumer sends a request to UDDI (www.uddi.org) to locate where the loan service is located.

4. UDDI returns the phone service address (www.phone.com/WSDL) that in fact is a link to WSDL.

5. The consumer now issues a call to WSDL to learn how to invoke the phone service (i.e., what method to use, what are the input/output parameters).

6. The WSDL returns a definition of the phone service in WSDL format.

7. The consumer prepares a SOAP message, an XML document, that is sent to www.phone.com over HTTP.

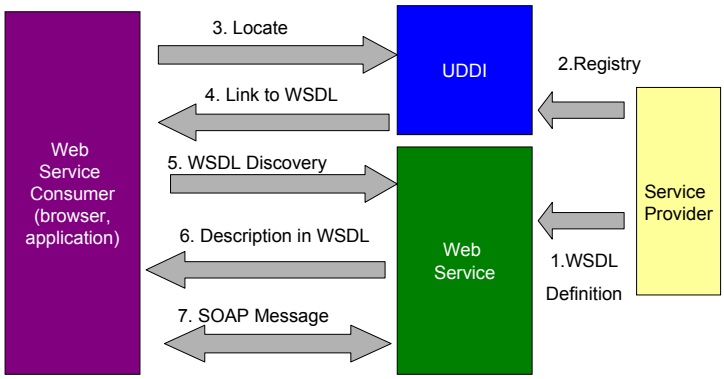


Figure 11-25: XML Web Services in Action

To satisfy this external flow, a number of things happen internally as shown in Figure 11-26. A Web Service contains a Listener that waits for requests. When a request is received, the Listener forwards it on to a component that implements the required business logic. This component might be designed to operate specifically as a Web Service, or it could be some other business component or COM/Java object that the Web Service wants to expose to the outside world. In the latter case, a developer will write some logic that acts as a façade for the Web Service and forwards the request on to the COM/Java object itself. The COM/Java object or other business logic may make use of a database or other data store, accessed using a Data Access layer. As expected, this looks like the classic N-tier architecture in which the Web Service can get its information from a database or another Web Service.

A great deal of information about various aspects of Web Services is available at Microsoft (http://msdn.microsoft.com), IBM (http://www-106.ibm.com/developerworks/Webservices/) and Sun (www.sunsoft.com) sites.

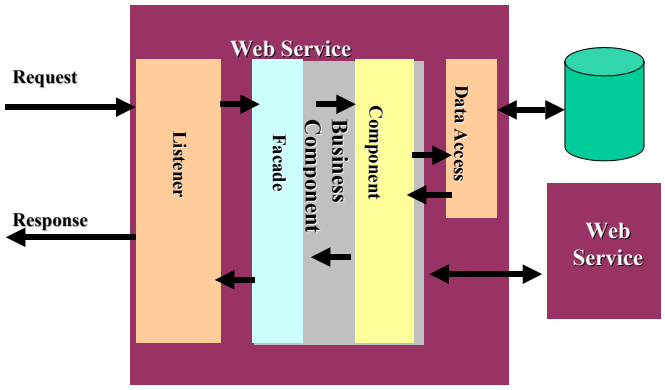


Figure 11-26: Internal Architecture of a Web Service

11.8.4.3 Developments in Web Services

To promote interoperability between different WS providers, **WS-I** (Web Services Interoperability Organization) has been established as an open organization dedicated to promoting and supporting Web Services interoperability across programming languages, platforms, and applications.

WS is moving beyond the basic framework of WSDL, UDDI, and SOAP for the description, publishing, and interaction between components. A more comprehensive stack, shown in Figure 11-27, is being developed to provide service composition, security, transaction support, coordination, reliable messaging, and other services. Several specifications have been proposed, and are being proposed at the time of this writing, to support different layers of this stack – from business processes to transport and coding. At the business process level, the Business Process Execution Language for WS (BPEL4WS), commonly known as BPEL, is noteworthy. Considerable work is also being done in WS transaction support and WS coordination. Another active area of work is WS security. In particular, a Security Assertion Markup Language (SAML) has been specified so that the authorization and authentication information between different WS components can be exchanged consistently. Several standards bodies such as W3C and Oasis, and industrial organizations such as IBM, Microsoft, and Sun are playing key roles in these developments. Detailed discussion and analysis of these specifications is beyond the scope of this book due to space limitations. See Curbena [2003][7] for a good overview of these developments and additional sources of information.

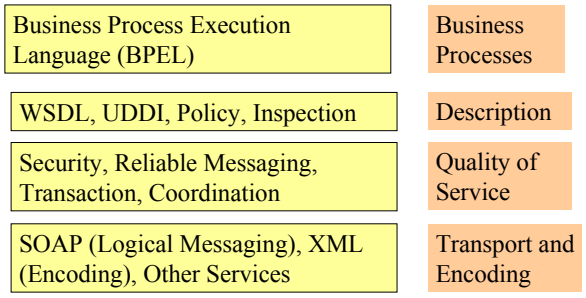


Figure 11-27: Evolving Web Services Stack

[7] F. Curbena et al., "The Next Step in Web Services," *CACM* (October 2003).

11.8.5 Mobile Web Services

Mobile Web Services (MWS) is a new specification that extends the core Web Services specifications to include the special capabilities and requirements of mobile computing. A framework for Mobile Web Services has been proposed to enable new services and products to be created for mobile computing. MWS is evolving at the time of this writing. The following have been identified:

- Application of WS-Security to mobile network security services. For example, a GSM-style SIM security device has been accepted for authentication.
- Development of a set of payment mechanisms within the Web Services architecture
- SMS services and MMS services
- Location-based services

Other areas for work within the activity have also been identified and are under review at the time of this writing. The two main areas of work are optimal transmission of WS requests over slow cellular networks and efficient processing of WS by mobile devices with processor limitations.

Figure 11-28 shows a conceptual view of the initially proposed three principal components of MWS and the interactions among them. The Subscriber is a service consumer (a client) that needs services of a Third-Party Service Provider (SP). One or more Client Applications are used by the subscriber – these applications run under the control of a Client Platform (e.g., a Symbian operating system) and use Security Credentials such as a SIM card. The SP offers a set of services that are defined through WSDL and possibly stored in a UDDI. For example, a bank could be an SP that provides account review, bill payment, and fund transfer as three possible services that have been defined through WSDL. A Mobile Network Operator is the third player in MWS – it provides authentication and payment services in addition to its own services known as Mobile Network Services. These Mobile Network Services are Web Services and represent SMS, MMS, Location or other proprietary services. The Authentication Service and the Payment Service are also Web Services operated by the Mobile Network Operator to perform authentication and payment authorization, respectively. The subscriber has a business relationship with a Mobile Network Operator and subscribes to one or more of the Operator services (e.g., SMS).

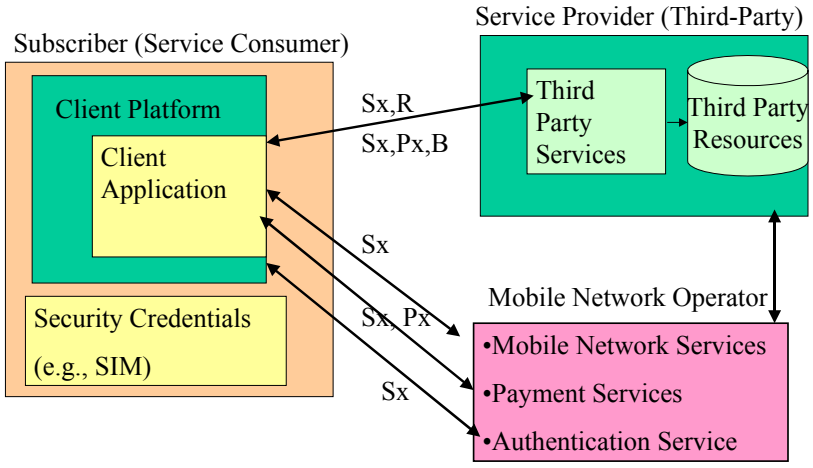


Figure 11-28: Mobile Web Services

Consider, for example, a scenario where Sue, a subscriber, has an account with bank1.com (an SP) that has defined services such as account review, bill payment, and fund transfer

through WSDL. Sue also has an account with T-Mobile, a Mobile Network Operator, that offers MMS as well as SMS as Mobile Network Services. Sue only subscribes to SMS. Suppose Sue wants to review her bank account. Then the following interactions will take place:[8]

- Sue obtains a Security Token (Sx) from the Authentication Service via her Client Platform.
- The Authentication Service authenticates the Subscriber by using the Security Credentials (SIM authentication) and issues a Security Token Sx.
- The Security Token Sx is sent to the SP along with an actual request R to review the account.

If Sue wanted to invoke "bill payment" service, then the Client Application obtains a Payment Token (Px) from the Payment Service once it has obtained the Security Token Sx. The Payment Service issues a Payment Token if it can successfully obtain payment authorization from the Subscriber by assuring that the Subscriber has agreed that a payment can be made for the service being made. Then the Security Token Sx is sent to the SP along with the Px and the actual request B to pay the bill.

Although Mobile Web Services are relatively new at the time of this writing, some tools for MWS are becoming available. An example is the Web Services Tool Kit for Mobile Devices by IBM. This tool kit supports J2ME, SOAP, and other capabilities. WS Security is supported for secure SOAP message exchanges using XML digital signatures, XML encryption and message authentication. The JVMs (Java Virtual Machines) on mobile devices support development of applications based on the J2ME specification, and IBM's WebSphere Studio Device Developer (WSDD).

Additional information about Mobile Web Services can be obtained from the IBM and Microsoft sites in this area.

11.9 Concluding Comments

This chapter has focused on the important question: how do you integrate the various components of a wireless system into a working solution that satisfies the customer and corporate needs? Instead of the details of individual components – we have done that in previous chapters – this chapter has attempted to explain how the various pieces can fit into a functioning architecture that can survive the changes in technologies, standards, and market needs. The discussion is based on a framework that starts with integrations of network issues at lower layers and then proceeds to platform and application levels. The emerging trend of using components, component-based platforms, and Mobile Web Services in building the flexible and integrated systems is highlighted.

Suggested Books on Wireless Architectures and Integration

- Dharwan, C. *Moble Computing – A Systems Integrator's Hanbook*. McGraw-Hill, 2000. This book is somewhat high-level, but it does cover various topics from an integration point of view.
- Geier, J. and Geier, J. *Wireless LANs.* Sams, 2001. The focus of this book is largely on 802.11 LANS, but there is a good chapter on LAN integration. A very easy-to-read

[8] This is a simplified version of the interactions. We have ignored the exchange of contexts for the sake of simplicity.

book.

- Kalakota, R. and Robinson, M. *M-Business: The Race to Mobility*. McGraw Hill, 2002. This book is written primarily for managers and lacks technical details but has some very interesting case studies and examples of m-business.
- Lin, Y. and Chlamtac, I. *Wireless and Mobile Network Architectures*. John Wiley & Sons; 2000. This is a very technical book with focus primarily on wireless networking issues (layers 1 to 4).

11.10 Review Questions and Exercises

1) What is an integrated architecture? Explain through an example from wireless.

2) Is the framework presented in Section 11.2.2 complete? Can you expand and refine it?

3) Expand the horizontal/vertical integration discussion in Section 11.2.3 to include more layers and different options.

4) Choose a case study of an integrated wireless system from the extant literature and analyze it in terms of intended use, the approach used, and architectural decisions.

5) Do you agree with the vision presented in Section 11.3? In your view, what are the plusses and minuses of this vision?

6) List the 3 main issues in network-level integration. Give an example of each.

7) What is the role of Mobile IP in wireless integration? What are its strengths and weaknesses?

8) How can wireless middleware help in wireless integration? Explain through an example.

9) Give additional examples of integrated mobile applications.

10) What are component-based architectures and why are they important for wireless?

11) What is the role of Web Services and Mobile Web Services in building future mobile computing applications?

11.11 References

Askit, M., ed. *Software Architectures and Component Technology*. Kluwer International Series in Engineering and Computer Science, 2001.

Bass, L., et al. *Software Architecture in Practice*. Addison-Wesley Pub, 1998.

Bosch, J. *Design and Use of Software Architectures*. Addison-Wesley, 2000.

Britton, C. *IT Architectures and Middleware: Strategies for Building Large, Integrated Systems*. Addison-Wesley, 2000

Bruegge, B. and Dutoit, A. *Object-Oriented Software Engineering*. Prentice Hall, 2000.

Caruso, F. and Umar, A. "Architectures to Survive Technological and Business Turbulances." *ISF Journal* (March 2004).

Chakravorty, R., et al. "Practical Experiences with Wireless Networks Integration using Mobile IPv6," *ACM SIGMOBILE Mobile Computing and Communications Review* 7, Issue 4 (October 2003).

Clemens, P. and Klein, K. *Evaluating Software Architectures*. Addison Wesley, 2002.

Dharwan, C. *Moble Computing – A Systems Integrator's Hanbook*. McGraw-Hill, 2000.

Ferdinandi, P. *A Requirements Pattern: Succeeding in the Internet Economy*. Addison-Wesley, January 2002.

Fowler, M., et al. *Patterns of Enterprise Application Architecture*. Addison Wesley Professional, 2002.

Geier, J. *Wireless LANs*. 2nd edition. Sams, 2002.

Helal, S., et al. "An Architecture for Wireless LAN/WAN Integration," IEEE Wireless Communications and Networking Conference (WCNC), 2000.

Herzum, P. and Sims, O. *Business Component Factory*. John Wiley, 2000.

Heineman, G., and Councill, W. *Component-Based Software Engineering*. John Wiley, 2001.

Hofmeister, C. *Applied Software Architecture*. Addison-Wesley, 1999.

Jell, T., ed. *Component-Based Software Engineering*. SIGS Books & Multimedia, 1998.

Kalakota, R. and Robinson, M. *M-Business: The Race to Mobility*. McGraw Hill, 2002.

Kuttan, S. "Distributed Computing." In *12th International Symposium*. Springer Lecture Notes in Computer Science, Dec. 1998.

Krakqwiak, S., ed. *Advances in Distributed Systems*. Springer Verlag Lecture Notes in Computer Science, 2000.

Lin, Y. and Chlamtac, I. *Wireless and Mobile Network Architectures*. John Wiley & Sons; 2000.

Linthicum, D. "Distributed Objects Get New Plumbing", *Internet Systems* (January 1997), pp. 4-5.

Melling, W. "Authoring an Enterprise Information Architecture: A Pragmatist's Approach to a Process." Proceedings of Gartner Group Confrerence, Feb. 22-24, 1995.

Nutt,G. *Open Systems*. Prentice Hall, 1992

Rimassa, G. "Wired-Wireless Integration: A Middleware Perspective." *Internet Computing* (September/October 2002).

Schmidt, D., et al. *Design Patterns*. Vol. 2. OMG Press, 2000.

Szyperski, C. *Component Software: Beyond Object-Oriented Programming*. Addison Wesley, 1998.

Umar, A. *e-Business and Distributed Systems Handbook: Integration Module*. 2nd ed. NGE Solutions, 2004.

Umar, A. *e-Business and Distributed Systems Handbook: Architecture Module*. 2nd ed. NGE Solutions, 2004.

Umar, A. *e-Business and Distributed Systems Handbook: Middleware Module*. 2nd ed. NGE Solutions, 2004.

Umar, A. *e-Business and Distributed Systems Handbook: Platform Module*. 2nd ed. NGE Solutions, 2004.

Umar, A. *Distributed Computing and Client/Server Systems*. Prentice Hall, 1993.

Umar, A. *Object-Oriented Client/Server Internet Environments*. Prentice Hall, 1997.

Vichr, R. and Malhotra, V. "Middleware Smooths the Bumpy Road to Wireless Integration." IBM Developer website: http://www-106.ibm.com/developerworks/wireless/library/wi-midarch/ (September 2001).

Web Links:

http://www.eacommunity.com. [Numerous articles, white papers, and resource links to enterprise architectures].

http://www.geocities.com/m_a_r_c_h/components.html . [This page lists various Web resources on Component-Based Software Development that may be of interest to researchers].