PEARSON

Mobile Computing

Theory and Practice

Kumkum Garg

Mobile Computing

This page intentionally left blank

Mobile Computing
Theory and Practice

Kumkum Garg
Department of Electronics and Computer Engineering
Indian Institute of Technology Roorkee

PEARSON
Delhi • Chennai • Chandigarh

Assistant Acquisitions Editor: Pradeep Banerjee
Assistant Production Editor: Amrita Naskar
Composition: Aptara®, Inc.

ISBN 978-81-317-3166-6

10 9 8 7 6 5 4 3 2 1

Published by Dorling Kindersley (India) Pvt. Ltd, licensees of Pearson Education in South Asia.

Head Office: 7th Floor, Knowledge Boulevard, A-8(A), Sector – 62, Noida, UP 201309, India.
Registered Office: 11 Community Centre, Panchsheel Park, New Delhi 110017, India.

Contents

This page intentionally left blank

Preface

Mobile computing or computing-on-the-go is proving to be one of the most promising technological advances in computer science and engineering to date. With the advent and proliferation of portable, handheld hardware devices, equipped with wireless communication interfaces and carrying innovative applications and systems software, computing has now become truly 'pervasive' or 'ubiquitous'. It is now commonplace to see people sitting in airport and hotel lounges, meeting rooms and even open spaces, keying away at their PDAs or laptops, checking e-mails and appointments, making to-do lists or just chatting with their friends. We are also looking at 'smart dust', in which thousands of miniature processing devices can be literally scattered in a battlefield or natural calamity areas to form a network and monitor the various activities therein, like movement of the enemy, management of bushfires, relief supplies and rehabilitation work, etc.

Technological advances create newer and more innovative applications everyday, which in turn fuel the demand for new technology. This has become a not-so-vicious circle, keeping researchers and developers on their toes all the time. The beneficiary is of course the layman on the street, literally so in the case of mobile computing.

It is important to note that mobile computing is not just mobile or wireless communication, as some would believe. There is much more to mobile computing, and it is to remove this confusion that this book has been written. Of course, provision of higher and more wireless bandwidth is the driving force for mobile computing. But what is more important and challenging is the design of various application protocols and algorithms, the small-footprint operating systems, efficient usage of the small-sized user interfaces and, above all, providing security of systems and applications.

This book provides a focussed look at all the issues mentioned above and gives an insight into the large number of technologies available in these areas to the user today. Apart from the theory, which is presented in an easy-to-understand form, we have provided many examples and suggestions for hands-on programming to help understand better the underlying technologies. These have been actually undertaken by senior undergraduate and postgraduate students of computer science at IIT Roorkee. To assist the reader in programming applications, an appendix has been included which deals with some important aspects of Java network programming.

This book is intended for both professionals and students of senior undergraduate- and postgraduate-level engineering courses in electrical, electronics and computer science who have a background in computer networks and Java programming. It can be used for a one-semester or a one-quarter course. It can also be adopted for short-term training courses for new employees or trainees. To make the new concepts easy to understand, each chapter ends with multiple-choice

review questions. Other research-oriented and programming-type questions which exercise the readers' mind are also included.

Book organization

This book has been organized into 12 chapters, covering the entire gamut of technologies relevant to mobile computing. These include wireless and cellular communication, wireless local area networks (WLANs), logical mobility consisting of process migration and mobile agents, handheld devices and their operating systems, physical mobility, mobile ad hoc networks, wireless sensor networks, wireless application protocol and the mobile Internet, security issues in mobile applications, etc. The last chapter gives a brief idea of some design projects that can be undertaken to better understand the theory. An appendix is also included for explaining the basics of Java network programming. The material is just right for a four-month, one-semester course.

For a short-term course for students who are familiar with the basics of wireless communication, Chapters 2, 4, 10 and 11, which deal with wireless communication, migrating processes, mobile agents and security, respectively, can be omitted.

The concepts discussed in this book can also be used for research in this fast-growing field, since most of the technologies that are used and are applicable today may not be relevant tomorrow as requirements for newer applications arise.

Acknowledgements

Over the entire duration of the writing and compiling of this book, many people have helped me; without them, this book would not have been possible.

First and foremost, I would like to thank the many experts who reviewed drafts of this book. Their suggestions have certainly helped to improve the content and presentation of the book.

I am grateful to my Ph.D. student R.B. Patel, who first suggested that I introduce a course on mobile computing at IIT Roorkee in 2003 in the postgraduate curriculum and write a book on this important topic.

My heartfelt thanks to all my postgraduate and senior undergraduate students at IIT Roorkee, who designed and developed various projects related to mobile computing. These projects provided the content for the last chapter of the book and helped tremendously in adding to the 'practice' part of the title of the book.

I also thank IIT Roorkee and MIT Manipal for providing the working environment that made this book possible.

Last but not the least, I thank my family members and friends whose support and constant encouragement during the three years of writing the book made this effort worthwhile and without their support, this book could never have been finished.

KUMKUM GARG

Introduction to Mobility 1

Mobility has been the hallmark of all animate and living entities in nature. Animals move from place to place, migrating to find food and shelter. Similarly, early humans migrated from their natural habitats in search for food. Today, humans move in search of better employment, entertainment, travel, etc. Thus, mobility stems from a desire to move towards resources and away from scarcity.

As in nature, so also in the field of computer science, mobility is becoming important and necessary. Today, both physical and logical entities can move. Physical entities are small, mobile computers that can change their actual location, unlike early systems, which were bulky in size and therefore immobile. Logical entities may be either the running user applications (processes) that migrate within a local cluster of computers or mobile agents, which are network applications that migrate in a network and execute on behalf of their owners anywhere in the Internet.

The concept of mobility in the field of computer science has thus been chronologically provided in process migration since the 1970s, in mobile computers since the 1980s and in mobile agents from the 1990s. In this chapter, we shall briefly discuss these concepts and their benefits and challenges for deployment. We shall come back to visit them in detail in subsequent chapters.

1.1 Process migration

Process migration is the act of transferring a process between two computers connected through a wired or wireless medium. A process is an operating system abstraction and has code, data and state, besides a unique identity in the system. Traditionally, process migration was used to achieve load distribution in a multiprocessor system like a cluster or network of computers, or it was resorted to for providing fault tolerance in such systems.

Many research operating systems have implemented full-blown process migration mechanisms, as shown in Accent (Zayas 1987), Chorus (Rozier and Legatheaux 1986), Mach (Acetta et al. 1986) and VKernel (Cheriton 1984). On the other hand, commercial migration-related products provide a higher-level, checkpoint-like restart version of migration, as seen in Condor (Litzkow, Linvy, and Mutka 1988).

The main benefits of process migration are that a process might move towards an underloaded computer, a specific database, or some rare hardware device. Furthermore, it enables movement of the programming environment and application to a desired location. For example,

if a computer has a partial failure or is about to shut down, a running process can migrate to another computer and continue execution there. The resulting flexibility and reliability are important and necessary.

1.2 Mobile computing

Mobile computing is computing that allows continuous access to remote resources, even to small computing devices such as laptops, palmtops and other handheld devices like personal digital assistants (PDAs) and digital cell phones. Mobile computing has become possible with the rapid advances in very-large-scale integration (VLSI) and digital/wireless communication technologies. There are basically three issues of concern in physical mobility. These are given below and have been dealt with in various ways by various researchers. We shall introduce these issues in this chapter but discuss details in subsequent chapters.

1. Weak connectivity: It is a well-known fact that wireless communication suffers from frequent disconnection and slow speeds, as compared with wired communication. The challenge is how a computer can operate when disconnected from the network or intermittently connected or connected over very slow communication links. This issue has been taken up in the CODA file system, which will be discussed in detail in Chapter 5.

2. Wireless connectivity: When a computer moves between cells in a wireless network or from one computer network to another, it is required to continue operating without having to re-register in the new location. In other words, the handoff should be smooth. This issue has been dealt with admirably by the development of two protocols—mobile Internet protocol (IP) and cellular IP, both of which are discussed in detail in Chapter 5.

3. Ubiquitous computing: This is the term coined by Mark Weiser and refers to the scenario when computers are present everywhere around us but have been rendered so small and cheap that they fade into the background. This is also called pervasive computing. Wireless sensor networks (WSNs) are examples of such ubiquitous or pervasive computing, and are discussed in detail in Chapter 7.

Thus, mobility of physical devices can be viewed at three different levels of granularity. These are as follows:

1. **Macro-mobility:** This is mobility through a global network. While moving in such a network, it should be possible to communicate without breaking the existing access. In Chapter 5, we shall read about mobile IP, which is the protocol that takes care of macro-mobility.
2. **Micro-mobility:** This is mobility of a device in one single administrative domain of the global network. For cellular networks, this is the lowest level of mobility. Cellular IP is the protocol designed to take care of micro-mobility, and this will also be discussed in Chapter 5.
3. **Ad hoc mobility:** This is mobility within a mobile ad hoc network (MANET), caused by device mobility constantly changing the network topology. We shall study MANETs in Chapter 7 and visit several ad hoc routing protocols therein.

Whatever the type of mobility, the benefits of mobile computing are obvious, since there is physical movement towards a desired resource. Here, both the owner and the computer move to

provide both qualitative and quantitative benefits. Since it is possible to use computer resources while moving, users can take the computer away from its usual workplace and still be productive. Thus, mobile computing, like process migration, enables movement of the programming environment and application. If a wireless phone cannot connect from a specific area, moving to a new area can overcome natural obstacles.

A major benefit with mobile computing is that the use of computers is increased, not only for computer professionals, but also for the lay person. This is very important, because in this information age, having continuous access is imperative for everyone on the go.

1.3 Mobile agents

A mobile agent is a program that can move through a network and autonomously execute tasks on behalf of the users. An agent is different from a user application, as it represents and acts on the owner's behalf by inheriting the owner's authority. Unlike mobile code (applets), mobile agents carry data and thread of control. They require agent environments, acting like docking stations, to execute and are supported on top of a programming environment like a Java virtual machine (JVM).

Mobile agents are used to great advantage in applications like e-commerce, software distribution, information retrieval, system administration, network management, etc. They are well suited for slow and unreliable links and also provide fault tolerance. Many mobile agent systems have been developed and reported in the literature. Some of the more well-known systems are Aglets, Agent Tcl and PMADE (platform for mobile agent development and execution).

Since mobile agents also migrate towards a source of information or towards a computer that they manage, they provide great flexibility and can mean easier reconfiguration or improved reliability. Mobile agents may not have sufficient resources or connectivity from one host and may move to another host.

It can be seen from the above that there is much commonality between the three kinds of mobility discussed above. Researchers have, over the years, developed various means and mechanisms to deploy the above concepts to real-life situations. As a result, we have numerous technologies that can be used to advantage. We discuss some of these briefly below. Detailed discussions are given in subsequent chapters.

1. Java as a language offers many concepts that are directly useful for mobile systems. For example, remote method invocation (RMI), object serialization and mobile code are all very useful for process migration and mobile agents.
2. Similarly, wireless technologies provide support for mobile computing, with the development of many wireless protocols like Bluetooth, the Infrared Data Association (IrDA) standards, wireless access protocol (WAP), etc.
3. Infrastructure support for transparent movement of entities from one location to another on the Internet and for issues of performance, scalability and reliability have been provided by the presence of numerous mobile agent systems that have been developed in recent years.
4. Standardization has been provided in the form of CORBA (common object request broker architecture) and the MASIF (mobile agent system interoperability facility) standard, which allow for interoperable systems to be built and used worldwide.

1.4 Technical issues for mobility

Mobile systems, as we have seen from the above discussion, are being increasingly deployed worldwide. But there are many challenges and technical issues of concern here. These are as follows:

1. Security is the biggest challenge for mobility. Security includes user authentication, data integrity and privacy, prevention of denial of service and non-repudiation. It may be appreciated that it is easier to provide security for a stationary system as compared to a mobile one, since the former can be guarded by intrusion detection systems and firewalls. The latter provides more security holes that have to be plugged. These include problems like unauthorized access, data corruption, denial of access/service, spoofing, Trojan horses, replaying and eavesdropping, among others.

   The existing security infrastructure is designed only to protect stationary systems and thus needs to be adapted or modified for use in mobile systems. Security of mobile systems is the subject of Chapter 11.
2. Reliability, in terms of availability of resources, in the presence of disconnection, is also a major issue for mobile systems. In fact, it is both a technical issue and a benefit for mobility. Reliability can be improved by mobility but needs additional support in the form of caching and loading of state. Similarly, replication and check-pointing, file-hoarding, message-queuing and fault-tolerance tools need to be provided.
3. Naming and locating are common issues for all forms of mobility. Without locating a mobile object, communication with it or its control is not possible. Communication channels must be reconstructed after every movement. Naming is to be associated with authentication, and all recycling is to be done with great care. Controlling a mobile entity is necessary to check its status or to suspend, kill or recall it.

All three of the above issues and their implementation will be discussed in detail in subsequent chapters.

1.5 Personal communication sy stems

A personal communication system (PCS) is a generic name for a commercial system that offers several kinds of personal communication services and extended mobility. The Federal Communications Commission (FCC) defines PCS as a mobile and wireless service that can be integrated with different networks to provide a wide variety of mobile and wireless services to individuals and business organizations. It was deployed in the USA in 1996. A PCS employs a mobile station (MS), an inexpensive, lightweight and portable handset, to communicate with a PCS base station (BS). The common features of these systems are as follows:

1. They are based on a second-generation technology like GSM (global system for mobile communication), IS-136 or IS-95.
2. Since they use the higher 1900-MHz band, an MS needs more power. This is because higher frequencies have a shorter range than lower ones. Alternatively, it can be said that the BS and the MS need to be closer to each other; that is, use smaller cell sizes.
3. They offer a whole spectrum of communication services ranging from an ordinary cell phone, short message service (SMS), to cable TV and limited Internet access.

A typical PCS architecture is shown in Figure 1.1.

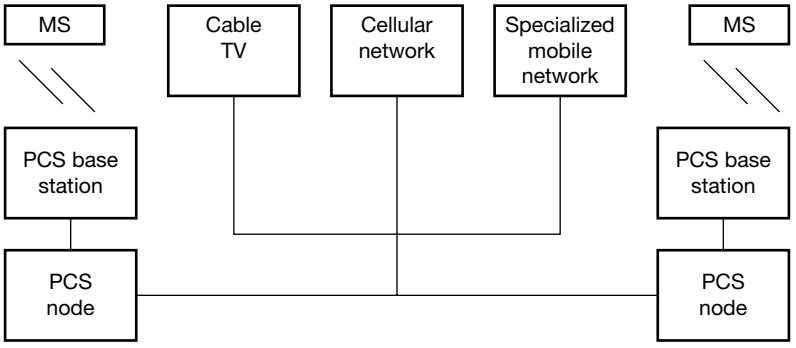


Figure 1.1 PCS Architecture

1.6 Context-aware computing

A context-aware computing system is one which has user, device and application interfaces which enable it to remain aware of various parameters like its surroundings, circumstances or actions. These parameters can be thought of as the present mobile network, surrounding devices or systems, changes in the state of the connecting network, etc. These could also mean physical parameters such as the present time of the day, presently remaining memory and battery power, presently available nearest connectivity, past sequence of actions, cached data records, etc.

The **context** of a mobile device represents the circumstances, situations, applications or physical environment under which it is being used. For example, the context is **student** when the device is used to download faculty lectures.

Context-aware computing leads to application-aware computing. This is because the application programming interfaces (APIs) are part of the context. For example, when using an e-mail ID, a mail-receiving or mail-sending application software is used for computing. An application can adapt itself to the context. For example, if context is a contact, the phone-talk application will adapt itself to use of the telephone number from the 'contact' and to the use of GSM or code division multiple access (CDMA) communication.

Context-aware computing also leads to pervasive or ubiquitous computing. In mobile device data-communication, context includes the existence of the service discovery protocol, radio interface and corresponding protocol. If the service discovery protocol senses the context and finds Bluetooth, then the device uses Bluetooth to communicate. Use of context in computing helps in reducing the possibility of errors and ambiguity in the actions. It also helps in deciding the expected system response on computation.

The five types of contexts that are important in context-aware computing are as follows:

Physical context: The context can be that of the physical environment. The parameters for defining a physical context are service disconnection, light level, noise level and signal strength. Assume a mobile phone is operating in a busy, congested area. If the device is aware of the surrounding noises, it can raise the speaker volume. If there is intermittent loss of connectivity during the conversation, the device can introduce background noises so that the user does not feel discomfort due to intermittent periods of silence.

Computing context: Computing context is defined by interrelationships and conditions of the network connectivity protocol in use. Examples of the latter could be Bluetooth, ZigBee, GSM, general packet radio service (GPRS) or CDMA. Computing context may also be bandwidth

and available resources. Examples of resources in a mobile device are keypad, display unit, printer and device cradle.

User context: The user context is defined as user location, user profiles, and persons near the user. It is based on the condition of the user, the primary intent of the systems and all other elements that allow users and computing systems to communicate.

Temporal context: Temporal context defines the interrelation between time and the occurrence of an event or action. A group of interface components has an intrinsic or extrinsic temporal context. For example, when a user presses a key to *add a contact* in his mobile device, the device should prompt him to *enter a number* as an input.

Structural context: It defines a sequence and structure formed by the elements or records. Graphical user interface (GUI) elements have structural context. Interrelation among the GUI elements depends on the structural positions on the display screen. For example, in a date, the hours are displayed on the left of the minutes.

1.7 Outline of the book

This book discusses both the theory and practice of mobile computing, so that the reader gets a complete idea of not only the techniques available to facilitate mobile computing, but also how to program and implement applications based on them.

Chapter 2 deals with the basics of wireless and cellular communication. The various wireless frequencies present in the electromagnetic spectrum, like radio, microwave, infrared and light, and their characteristic features and applications are presented. Satellite communication is discussed with reference to geostationary, medium-orbit and low-orbit satellites. The various generations of cellular phone communication are given in detail, as they form the basis of all communication for the handheld devices used in mobile computing.

Chapter 3 discusses wireless local area networks (WLAN). The most popular WLAN is the IEEE Standard 802.11. Its various extensions and modifications are dealt with in detail. The Bluetooth and infrared LANs are also presented. Both the versions of the European Standard HiperLAN are discussed. A comparison of all the above standards is presented to bring out their essential features and applications.

Chapter 4 deals with logical mobility. It discusses in detail the concept of process migration, which is the forerunner of mobile computing. The need for process migration and its various steps are presented.

Chapter 5 deals with physical mobility, its requirements and the challenges associated with it. It discusses the limitations of IP in providing for physical mobility. It shows how mobile IP and cellular IP overcome these problems in micro- and macro-mobility scenarios. The chapter also introduces mobile databases and their design issues. Finally, it looks in detail at the CODA file system developed to take into account disconnected operation in mobile computing.

Chapter 6 is on MANETs. The characteristics and classification of MANET are discussed in detail, along with their application. The proactive and reactive routing strategies for MANET are introduced. Three popular MANET routing protocols, namely, destination sequenced distance vector (DSDV), dynamic source routing (DSR) and adaptive on demand distance vector (AODV) are discussed in detail, with examples of their routing mechanisms. A performance comparison of DSR and AODV is also given.

Chapter 7 deals with WSNs. It shows how these are different from MANETs and gives their characteristics, architecture and some popular routing techniques developed for them. Case studies of the Mica mote sensor node and the TinyOS operating system used for it are also presented.

Chapter 8 discusses the handheld devices like PDAs and pocket computers used in mobile computing. It discusses the characteristics of various such devices, including Palm and HP devices. The operating systems used with such devices have certain special features. These are presented with respect to the Palm OS, the Windows CE and Windows Mobile operating systems.

Wide area mobile computing is the subject of Chapter 9, which presents what is now called the mobile Internet and the WAP, used to access the Internet on the move. The traditional Web programming model is compared with the wireless Web programming model. The WAP protocol stack is introduced and the WAP Gateway is discussed in detail, together with its design.

Chapter 10 revisits logical mobility in the form of mobile agents, their characteristics and architecture and highlights their differences with process migration, mobile codes and mobile objects. The two earliest and basic mobile agent platforms, namely, Aglets and Agent Tcl, are presented in detail. PMADE, a mobile agent platform developed at IIT Roorkee, is also presented. A discussion on the advantages of Java as a programming language for mobile agents is also given.

Chapter 11 discusses the most important and crucial issue of security in mobile computing systems. It highlights the security threats present in wireless systems. The security mechanisms present in IEEE 802.11, Bluetooth and WAP2.0 to take cognizance of and counter these threats are also discussed in this chapter.

Since this book is about mobile computing practice, the last chapter, Chapter 12, presents in detail as many as seven programming projects that can be designed and implemented by readers in different aspects of mobile computing. It thus provides an opportunity to have hands-on experience in designing and coding such systems.

The appendix gives some details of Java as a network programming language, and covers topics like socket programming, remote procedure call (RPC) and the Java RMI. Some examples are given to provide a clear understanding of these concepts.

1.8 Summary

Mobility is the hallmark of all animate beings and represents the movement from scarcity to resource-rich locations. In computing, mobility is characterized by logical or physical mobility and is represented by process migration, mobile agents or handheld-device communication. Mobile computing includes all these concepts, and it gives rises to a number of benefits, together with many technical issues and challenges. These have been dealt with in various ways, as discussed in this chapter, and the details are the subjects of the ensuing chapters.

In the next chapter, we shall concentrate on the various communication technologies that facilitate mobile computing.

Problems

1. What are the most important challenges facing mobile computing today? Discuss each of them in detail.
2. Distinguish mobile computing from distributed computing.
3. Go on to the Web and find about the state-of-the-art in mobile computing.

4. Do you have a pocket computer or PDA? If so, list the facilities it provides that can be listed under mobile computing applications.
5. The computer-networking architecture consists of seven layers, as given in the ISO OSI reference model. In your opinion, in which layer(s) should mobility be incorporated and why?
6. Recent conferences on mobile computing, such as ACM Mobicom and MobiSys, have published many articles on the subject. Read them and identify some of the current research challenges being addressed by researchers.
7. Do you think Java is suited for programming mobile computing systems? Explain your answer. (Do not look ahead into the later chapters of the book!)
8. What is your idea of a ubiquitous computing scenario for the home? Elaborate on this.
9. Discuss why security concerns in traditional systems are simpler than those in mobile systems. Give one example of a security threat that is present in the latter but not in the former.
10. Give one example where 'disconnected operation' may become imperative in a mobile computing scenario.

Multiple-choice questions

1. Which one of the following is 'computing that allows continuous access to remote resources even with the physical mobility of small computing devices such as laptops'?
   (a) Soft computing
   (b) Mobile computing
   (c) Remote computing
   (d) Ubiquitous computing
2. Pervasive computing is also called by which one of the following names?
   (a) Soft computing
   (b) Mobile computing
   (c) Remote computing
   (d) Ubiquitous computing
3. Wireless sensor networks are examples of which one of the following?
   (a) Soft computing
   (b) Mobile computing
   (c) Remote computing
   (d) Ubiquitous computing
4. Which one of the following can be characterized as 'mobility through a global network'?
   (a) Macro-mobility
   (b) Micro-mobility
   (c) Ad hoc mobility
   (d) None of the above
5. Mobility of a device in one single administrative domain of the global network is known as which one of the following?
   (a) Macro-mobility
   (b) Micro-mobility

(c) Ad hoc mobility
(d) None of the above

6. Which of the following is true for statements X and Y?
X: A mobile agent is a program that can move through a network and autonomously execute tasks on behalf of the users.
Y: Process migration is the act of transferring a process between two computers connected through a wired or wireless medium.
(a) X is true but Y is false
(b) X is false but Y is true
(c) Both X and Y are true
(d) Both X and Y are false

7. What is an Aglet?
(a) A wireless protocol
(b) A mobile agent
(c) A pervasive computing technique
(d) None of the above

8. Which one of the following is false for mobile agents?
(a) They are well suited for slow and unreliable links
(b) They cannot provide fault tolerance
(c) Unlike mobile code (applets), mobile agents carry data and thread of control
(d) They require agent environments

9. Which one of the following is not a wireless protocol?
(a) Bluetooth
(b) IrDA
(c) WAP
(d) CSMA/CD

10. Which one of the following is true for statements X and Y?
X: It is easier to provide security for a mobile system as compared to a stationary system.
Y: Security includes user authentication, data integrity and privacy, prevention of denial of service and non-repudiation.
(a) X is true but Y is false
(b) X is false but Y is true
(c) Both X and Y are true
(d) Both X and Y are false

Further reading

A.S. Tanenbaum (2005), *Computer Networks*, 4th ed. (Prentice Hall India).
A.T. Campbell (2000), 'Design, Implementation and Evaluation of Cellular IP', *IEEE Personal Communications*, 7 (August): 42–49.
C. Perkins (1998), *Mobile IP: Design Principles and Practice* (Addison-Wesley Longman).
D. Kotz et al. (1997), 'Agent Tcl: Targeting the Needs of Mobile Computing', *IEEE Internet Computing*, 1(4): 58–67.

D. Lange and M. Oshima (1998), 'Mobile Agents with Java: The Aglet API', *World Wide Web*, 1(3).

D. Milojicic et al. (1998), 'MASIF: The OMG Mobile Agent System Interoperability Facility', in Proceedings of the International Workshop on Mobile Agents (MA '98), Stuttgart.

D. Milojicic, F. Douglis and R. Wheeler (eds) (2000), *Mobility: Processes, Computers and Agents* (Addison-Wesley).

D.B. Lange and M. Oshima (1998), *Programming and Deploying Java Mobile Agents with Aglets* (Addison-Wesley).

D.P. Agrawal and Q.A. Zeng (2003), *Introduction to Wireless and Mobile Systems* (Thomson).

D.R. Cheriton (1984), 'The V-kernel: A Software Base for Distributed Systems', *IEEE Software*, 1(2): 19–42.

E. Pitoura and G. Samaras (1998), *Data Management for Mobile Computing* (Norwell, MA: Kluwer Academic Publishers).

E.R. Zayas (1987), 'Attacking the Process Migration Bottleneck', in Proceedings of the 11th ACM on Operating Systems Principles, pp. 13–24.

F. Adelstein et al. (eds) (2005), *Fundamentals of Mobile and Pervasive Computing* (Tata McGraw-Hill).

J. Kistler and M. Satyanarayan (1992), 'Disconnected Operation in the CODA Distributed System', *ACM Transactions on Computer Systems*, 10(1): 3–25.

M. Rozier and J.M. Legatheaux (1986), 'The Chorus Distributed Operating System: Some Design Issues', Y. Parker et al (eds.), in Proceedings of the NATO Advanced Study Institute on Distributed Operating Systems: Theory and Practice, Springer-Verlag, New York, August 1986, pp. 261–289.

M. Weiser (1991), 'The Computer of 21st Century', *Scientific American*, 265(3): 94–104.

M.C. Powell and B.P. Miller (1983), 'Process Migration in DEMOS/MP', *ACM SIGOPS OS Review*, 17(5): 110–119.

M.J. Acetta et al. (1986), 'Mach, a New Kernel Foundation for UNIX Development', in Proceedings of the Summer USENIX Conference, June 1986, pp. 93–113.

M.J. Litzkow, M. Livny, and M.W. Mutka (1988), 'Condor—A Hunter of Idle Workstations', in Proceedings of the 8th International Conference on Distributed Systems, June 1988, pp. 104–111.

R. Kamal (2007), *Mobile Computing* (Oxford University Press).

R.B. Patel (2002), 'Manual of PMADE' (Internal Report, Department of E&CE, IIT Roorkee, Uttarakhand, India).

Reza B'Far (2005), *Mobile Computing Principles: Designing and Developing Mobile Applications with UML and XML* (Cambridge University Press).

T. Imielinski and H.F. Korth (eds) (1996), *Mobile Computing* (Norwell, MA: Kluwer Academic Publishers).

U. Hausmann et al. (2003), *Principles of Mobile Computing*, 2nd ed. (Springer).

V. Kumar (2006), *Mobile Database Systems* (John Wiley).

W.R. Cockayne and M. Zyda (1998), *Mobile Agents* (Manning Publications).

Wireless and Cellular Communication

2

In this chapter, we will discuss the transmission technologies that form the basis of all mobile computing. In particular, we study in detail mobile or wireless communication and the different protocols that have been developed to physically or logically connect two mobile devices. Thus, this chapter looks at the physical layer technologies used in mobile computing, using what is called **unguided media**, as opposed to **guided media**, which consist of copper fibres, twisted pairs and optical fibres, and which are used for wired communication. The basis for all wireless transmission is the electromagnetic spectrum, in which lie the different frequency bands that are used for wireless communication. We will discuss in detail the characteristics of each of these frequency bands and the wireless and cellular communication systems enabled by them.

It is assumed here that the reader is familiar with the theoretical basis for data communication, that is, the terms *frequency*, *wavelength*, *channel*, *speed of light*, *bandwidth*, *the maximum data rate of a channel*, etc., and the relation between them.

For the sake of completeness, and because it is an important and relevant relation for this book, we must state here that the amount of information that a noisy channel can carry is governed by its bandwidth. According to Shannon, the maximum data rate of a noisy channel whose bandwidth is H Hz and whose signal-to-noise ratio is S/N is given by

$$\text{Maximum data rate (bits/sec)} = H \log_2 (1 + S/N)$$

For example, a channel of 3,000 Hz bandwidth and signal-to-thermal noise ratio of 30 dB can never transmit more than 30,000 bps.

For other related information, the uninitiated reader is referred to Tanenbaum (2003).

2.1 The electromagnetic spectr um

The electromagnetic spectrum is shown in Figure 2.1. The radio, microwave, infrared and visible light portions of the spectrum can all be used for transmitting information by modulating the wave's amplitude, frequency or phase. The higher frequencies, that is, ultraviolet light, X-rays and gamma rays, would give better results, but are normally not used because they are difficult to produce and modulate, do not propagate well through buildings and are harmful to humans. The various frequency bands have official International Telecommunication Union (ITU) names, as given in Figure 2.1, and are based on their wavelengths.

In this section, we give, very briefly, the characteristics, advantages and disadvantages of each of the above wave bands and see how they are used for wireless transmission.

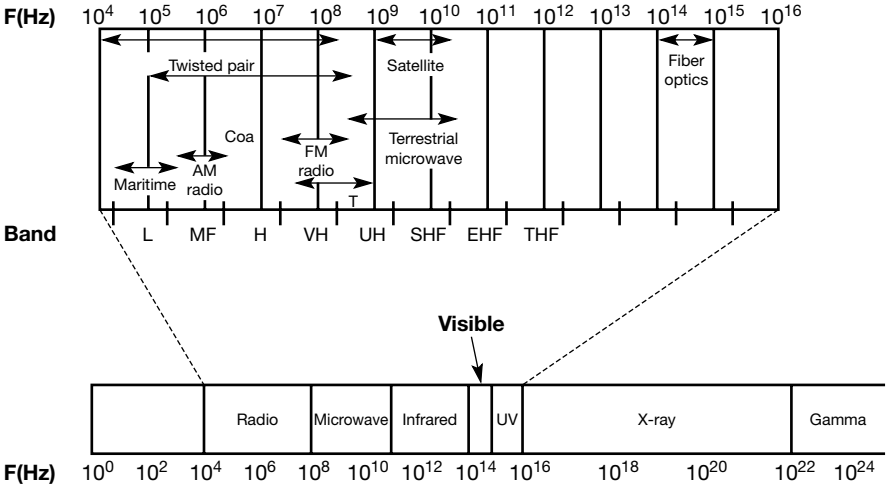


Figure 2.1 The Electromagnetic Spectr um

2.1.1 Radio waves

Radio waves are present at the lower end of the spectrum and are widely used for both indoor and outdoor communication. They have the advantage that they are omnidirectional and are able to travel long distances, penetrating easily through buildings. Their disadvantages are that they suffer from interference between users and from electrical equipment. They also exhibit frequency-dependent properties; that is, at low frequencies, they pass through objects, but attenuation in power occurs as distance from the source increases. On the other hand, high-frequency radio waves travel in straight lines and cannot penetrate through obstacles. Furthermore, rain and sleet absorb such waves.

2.1.2 Microwaves

Frequencies above 100 MHz are called microwaves. These have the advantage that they can be narrowly focused because they travel in straight lines. Thus, by properly aligning the sending and receiving antennae, they are able to give much higher signal-to-noise ratio. For the same reason, they are affected by the curvature of the earth if long-distance communication is to be used, making it necessary to build repeater towers for the transmitting antennae. Microwaves are less expensive to use than optical fibres and are therefore popular in mountainous and urban areas.

Microwaves have the disadvantage that they suffer from **multipath fading.** This is because they do not pass easily through buildings and obstacles and are refracted by the atmospheric layer; some waves therefore arrive out-of-phase with the direct ones, resulting in cancellation of the signal. The effect of this type of fading changes with weather and frequency.

2.1.3 Infrared waves

Unguided infrared and millimeter waves offer an alternative to the standard radio frequency communication for short ranges. However, they are subject to the following restrictions:

- Transmission distance of less then 2 miles
- Line-of-sight limitations
- Restricted to 16 Mbps throughput
- Presence of environmental disturbances, such as fog, dust and heavy rain

However, the advantages of this technology are as follows:

- Reasonable high bandwidth
- No government license required for operation
- Cost-effective
- Capable of traversing multiple paths without interferences
- More secure than radio
- Immune to radio frequency interference and electromagnetic interference

Infrared communication has very little use on the desktop. For example, it can be used for connecting notebook computers and printers, but is not used in computer-to-computer communication. The Infrared Data Association (IrDA) has defined a number of standards governing infrared wireless communication. These include the IrDA-data and IrDA-control standards. These will be discussed in detail in the next chapter.

2.1.4 Lightwaves

Unguided optical signalling has been around for many years. In recent years, coherent optical signalling using lasers mounted on rooftops has been used to connect the local area networks (LANs) in two buildings. The signals are inherently unidirectional, so each building requires a laser and photodetector. This scheme is very inexpensive and offers very high bandwidth. It is easy to install and does not require a license to operate. A major disadvantage is that laser beams cannot penetrate rain or thick fog. However, they work well on sunny days and can be effectively used for 'wireless outdoors'.

2.2 Communication satellites

Communication satellites have provided a very powerful wireless communication system since the first artificial satellite was put into orbit in 1962. A communication satellite is like a big microwave repeater in the sky. It consists of many transponders, each of which listens to some frequency spectrum, amplifies the incoming signal and rebroadcasts it at another frequency (to avoid interference with the incoming signal).

Mobile satellite services allow global coverage, because in these systems satellites play the role of mobile base stations (BSs). Satellite-based systems are categorized according to the orbital altitude of the satellite. This is shown in Figure 2.2.

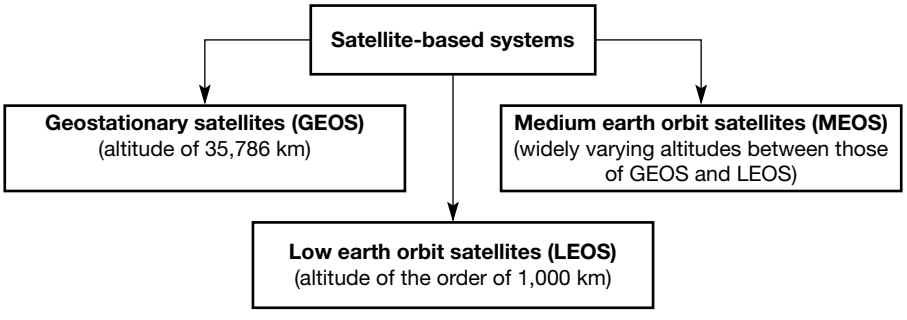


Figure 2.2 Satellite Systems

Table 2.1 The Main Satellite Bands

Band	Downlink	Uplink	Bandwidth	Problems
L	1.5 GHz	1.6 GHz	15 MHz	Low Bandwidth; crowded
S	2.2 GHz	2.2 GHz	70 MHz	Low Bandwidth; crowded
C	4.0 GHz	6.0 GHz	500 MHz	Terrestrial interference
Ku	11 GHz	14 GHz	500 MHz	Rain
Ka	20 GHz	30 GHz	3,500 MHz	Rain, equipment cost

The major advantage of GEOS systems is that contiguous global coverage up to 75 degrees latitude can be provided with just three satellites. Their main drawback is that they have a large 240–270 ms round-trip propagation delay and need higher radio frequency (RF) power. On the other hand, LEOS require less power but frequent handoffs. We shall discuss the characteristics of each of these in some detail below.

2.2.1 Geostationary satellites

Satellites at the altitude of 35,800 km in a circular equatorial orbit appear motionless in the sky. Such satellites are called geostationary satellites. With current technology, it is unwise to have geostationary satellites spaced much closer than 2 degrees in the 360-degree equatorial plane, to avoid interference.

ITU has allocated certain frequencies to satellite users. The main ones are listed in Table 2.1. The C band was the first to be designed for commercial satellite traffic. This band is already overcrowded because it is also used by the common carriers for terrestrial microwave link. The L and S bands were added by an international agreement in 2000. However, they are narrow and crowded. The next higher band available to commercial communication carriers is Ku (K under) band. This band is not (yet) congested, and at these frequencies, satellites can be placed as close as 1 degree. However, another problem exists: rain. Water is an excellent absorber of these short microwaves. Bandwidth has also been allocated in the Ka (K above) band for commercial satellite traffic, but the equipment needed to use it is still expensive.

A new development in the communication satellite world is the development of low-cost microstations, also called VSATs **(very small aperture terminals)**. These tiny terminals have 1 m or smaller antennas (versus 10 m for a standard GEO antenna) and can put out about 1 watt of power. In many VSAT systems, the microstations do not have enough power to communicate directly with one another (via the satellite of course). Instead, a special ground station called the **hub**, with a large, high-gain antenna, is needed to relay traffic between VSATs. See Figure 2.3.

2.2.2 Medium earth orbit satellites

MEOs are deployed much lower than the GEOs, and must be tracked as they move through the sky, as they drift slowly in longitude, taking about 6 hours to circle the earth. They have a smaller footprint on the ground and require less powerful transmitters to reach them. The 24 GPS (global positioning system) satellites orbiting at about 18,000 km above the earth are an example of MEO satellites.

2.2.3 Low earth orbit satellites

Moving down in altitude, we come to the LEO satellites. Due to their rapid motion, large numbers of them are needed for a complete system. On the other hand, because the satellites are so close to earth,

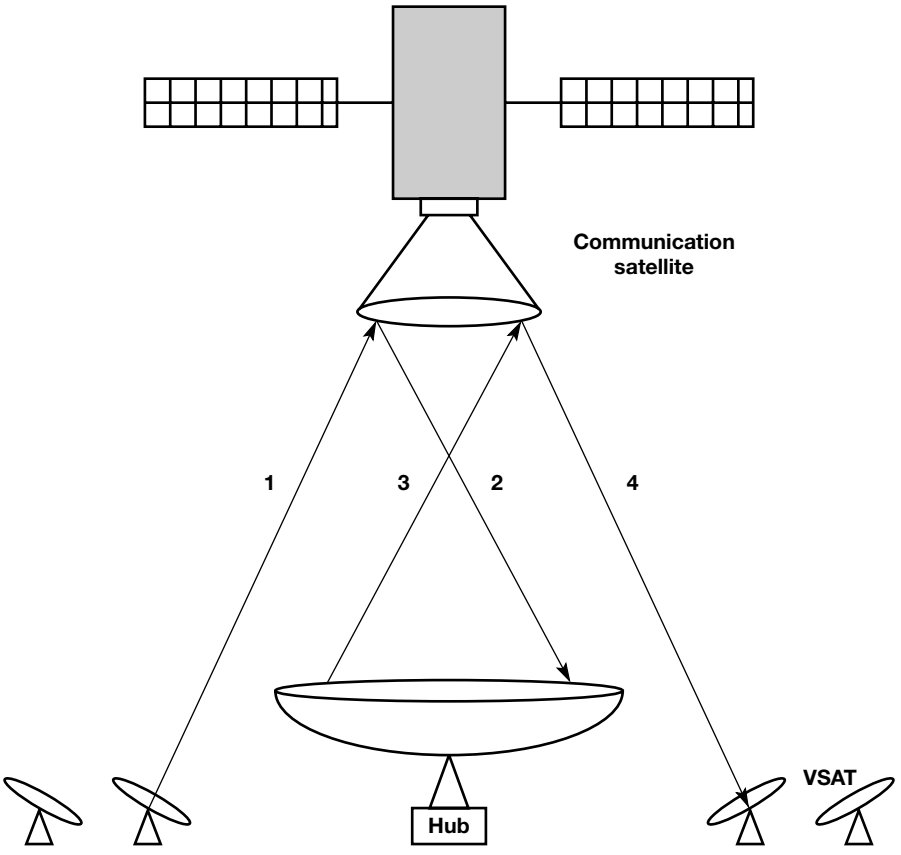


Figure 2.3 Hub and VSATs

the ground does not need much power, and the round-trip delay is only a few milliseconds. The examples are Iridium, Globalstar and Teledisc, of which only the last one is briefly discussed here.

Teledesic is targeted at bandwidth-hungry Internet users all over the world. The goal of the Teledesic system is to provide millions of concurrent Internet users with an uplink of as much as 100 Mbps and a downlink of up to 720 Mbps using a small, fixed, VSAT-type antenna, completely bypassing the telephone systems. It uses 30 satellites with large footprints, using the high-bandwidth Ka band, and packet-switching in space, with each satellite capable of routing packets to its neighbours. Users who want to send packets request and get assigned bandwidth dynamically, in about 50 ms.

2.3 Multiple-access schemes

In a wireless environment, there is a need to address the issue of simultaneous multiple access by many users or mobile stations (MSs) in the transmission range between the BS and themselves. Users are able to receive signals transmitted by others in the system. To accommodate a number of users, many traffic channels need to be made available. To provide simultaneous two-way communications (duplex communication), a forward (downlink) channel from BS to MS and a reverse (uplink) channel from MS to BS are necessary. Two types of duplex systems are used: frequency

division duplexing (FDD) divides the frequency used, and time division duplexing (TDD) divides the same frequency by time.

There are three basic ways in which many channels can be allocated within a given bandwidth. These are with respect to frequency, time and code division multiplexing, using three multiple-access techniques. These are frequency division multiple access, (FDMA), time division multiple access (TDMA) and code division multiple access (CDMA). FDMA mainly uses FDD, while TDMA and CDMA systems use either FDD or TDD. We will discuss these three techniques in this section along with their advantages and disadvantages.

A multiple-access technique is important in mobile cellular systems, so that an MS can distinguish a signal from the serving BS, and also discriminate the signals from an adjacent BS. Multiple-access techniques are based on the orthogonalization of signals.

An FDMA system is one which uses different carrier frequencies to transmit the signal for each user. If a system uses distinct time to transmit the signal for different users, it is a TDMA system. If a system uses different codes to transmit the signal for each user, it is a CDMA system.

2.3.1 FDMA—F requency division multiple access

In FDMA, the allocation of frequencies to channels can either be fixed (as in radio stations) or dynamic (demand-driven). Furthermore, channels can be assigned to the same frequency at all times, that is, pure FDMA, or change frequencies according to a certain pattern, that is, FDMA combined with TDMA. The latter is done in many wireless systems to circumvent narrowband interference at some frequencies, known as frequency hopping. The sender and the receiver agree on a hopping pattern, so that the receiver can tune to the right frequency. Hopping patterns are normally fixed for a long period.

As an example of FDMA, let us consider a mobile phone network based on the global system for mobile communication (GSM) standard for 900 MHz. There are 124 multiple-access channels per direction available at 900 MHz. The basic frequency allocation scheme is fixed and regulated by a national authority. All uplinks use the band between 890.2 and 915 MHz; all downlinks use 935.2 to 960 MHz. The BS allocates a certain frequency for uplink and downlink to establish a duplex channel with a mobile phone. Each channel (uplink and downlink) has a bandwidth of 200 KHz. Uplinks and downlinks have a fixed relation.

For a certain channel n,

$$\text{if the uplink frequency is } f_u = 890 \text{ MHz} + n \times 0.2 \text{ MHz},$$

$$\text{the downlink frequency is } f_d = f_u + 45 \text{ MHz, i.e., } f_d = 935 \text{ MHz} + n \times 0.2 \text{ MHz}.$$

2.3.2 TDMA—T ime division multiple access

TDMA offers a much more flexible scheme as compared with FDMA. Tuning to a certain frequency is not required, and the receiver can stay at the same frequency all the time. Very simple receivers and transmitters can thus be designed, since listening to many channels separated in time is easier than listening to different frequencies at the same time. Many different algorithms exist to control medium access using only one frequency. Almost all MAC schemes for wired networks like Ethernet, token ring, Asynchronous transfer mode (ATM), etc., work according to this principle.

In TDMA, synchronization between receiver and sender has to be achieved in the time domain. This can be done either by using a fixed pattern or by using a dynamic allocation. Fixed allocation is not efficient in cases where bandwidth requirement is variable.

Many systems like IS-54, IS-136, GSM and digital European cordless telecommunications (DECT) use TDMA with **fixed allocation**. For example, for the DECT cordless phone system, the BS uses 1 out of 12 slots for the downlink, whereas the MS uses 1 out of 12 different slots for the uplink. Uplink and downlink are separated in time. Up to 12 different MS can use the same frequency without interference. Each connection is allotted its own uplink and downlink pair. The pattern is repeated every 10 ms; that is, each slot has a duration of 417 μsec. This repetition guarantees access to the medium every 10 ms, independent of any other connection. A guard band is also used at the beginning and end of each slot to avoid collisions due to drifts in receiver and transmitter clock frequency or computational delays in placing the data in a slot.

Fixed access patterns are efficient for connections with a constant data rate, as in classical voice transmission with 32 or 64 Kbps duplex. But they are inefficient for bursty data or asymmetric connections, as in Web browsing, where no data transmission occurs while the page is being read, whereas clicking on a hyperlink triggers data transfer from the MS to the BS, followed by a large volume of data returned from the Web server. In such cases, demand-oriented TDMA schemes are used. In **demand-oriented TDMA**, the allocation is traffic dependent, and the BS can reserve time slots for an MS on demand.

2.3.3 CDMA—Code division multiple access

CDMA is the best technical solution available today and is the basis for the 3G mobile systems. It is also widely used in the United States in 2G mobile systems, competing with Digital advanced mobile phone system (D-AMPS). For example, Sprint personal communication services (PCS) uses CDMA, whereas AT&T Wireless uses D-AMPS. CDMA is also known as International Standard IS-95 or cdmaOne.

CDMA is completely different from FDMA and TDMA. Instead of dividing the allowed frequency range into a few hundred narrow channels, CDMA allows each station to transmit over the entire frequency spectrum all the time. Multiple simultaneous transmissions are separated using codes. Codes used by users should have a good **autocorrelation** and should be **orthogonal** to other codes. For details of these two terms, the reader is referred to Tanenbaum (2003). Autocorrelation helps a receiver to reconstruct the original data precisely even in the presence of distortion by noise, and orthogonality is necessary for two stations to share the medium without interference.

Tanenbaum (2003) has given a very good analogy to explain the concept of CDMA:

> An airport lounge has many pairs of people conversing. TDMA is comparable to all the people being in the middle of the room but taking turns speaking. FDMA is comparable to the people being in widely separated clumps, each clump holding its own conversation at the same time as, but still independent of, the others. CDMA is comparable to everybody being in the middle of the room talking at once, but with each pair in a different language. The French-speaking couple just hones in on the French, rejecting everything that is not French as noise.

In CDMA we extract only the desired signal and reject everything else as random noise. Here, each bit time is subdivided into m short intervals called **chips**. There are normally 64 or 128 chips per bit or longer. Each station is assigned a unique m-bit **code** called a chip sequence. Chip sequences in IS-95, for example, are $2^{42} - 1$ chips long, and the chipping sequence is 1228800 chips/s; that is, the code repeats after 41.425 days.

To transmit a 1 bit, a station sends its chip sequence. To transmit a 0 bit, it sends the 1-bit's complement of its chip sequence. No other patterns are permitted. Thus, for $m = 6$, if station A is assigned the chip sequence 010011, it sends a 1 bit by sending 010011 and a 0 bit by sending

101100. For pedagogical purposes, it is more convenient to use a bipolar notation, with binary 0 being −1 and binary 1 being +1.

To synchronize the sender and the receiver, the sender transmits a long predefined chip sequence so that the receiver can lock onto it. Transmissions that are not synchronized are treated as noise. The longer the chip sequence, the higher the probability of detecting it correctly in the presence of noise.

Implementation of the chip sequences and codes is complicated, and this is a major drawback with the CDMA scheme. But it is used for wireless mobile communication, as it operates in a much higher (1.25 MHz) band than D-AMPS and GSM, where it can support many more users than either of these systems.

For a good comparison of the above techniques, the reader is referred to Schiller (2006).

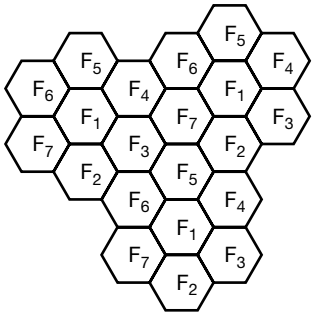
2.4 Cellular communication

Wireless communication using unguided media, that is, radio and microwave frequencies or satellites, has found widespread use in mobile phones. These are currently being used for voice communication, but soon they will find use in data communication. Cellular communication has undergone many generations, in which the communication bandwidths and data speeds have continuously increased. This has given rise to many applications that have benefited mobility. In this section, we discuss briefly these generations and how they have revolutionized not only mobile phone communication, but also mobile computing.

2.4.1 The first generation (1G): 1980

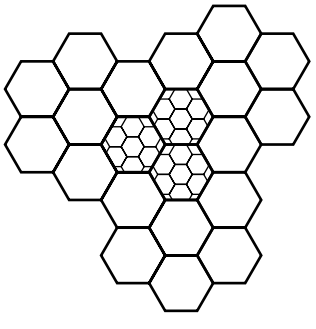
Analog cellular systems were the first generation of mobile telephone communication systems. They used analog frequency modulation for only voice (speech) transmission. The various systems that fall in this category are AMPS (Advanced Mobile Phone Service) (USA), Nordic Mobile Telephone (NMT)-900 (Sweden) and Cellular Digital Packet Data (CDPD), which is designed to provide packet data services on the top of existing AMPS.

The system architecture is such that a geographic region is divided into **cells**. The size of the cells in AMPS is about 10–20 km across, but is lesser in digital systems. Each cell uses some set of frequencies not used by its neighbours. Transmission frequencies are reused in nearby but not adjacent cells. Figure 2.4a illustrates the concept of frequency reuse. The cells are normally circular but are shown as hexagonal for ease of drawing.



(a)

Figure 2.4(a) Adjacent cells use different frequencies



(b)

Figure 2.4(b) Microcells add more users

If the area is overloaded, the power is reduced and the overloaded cells are split into smaller **microcells**. This allows for more frequency reuse and is shown in Figure 2.4b. At the centre of each cell is a BS to which all the telephones in the cell transmit. In a small system, all the BSs are connected to a single device which is called an MTSO (mobile telephone switching office) or MSC (mobile switching centre). In a larger system, several MTSOs may be used in a hierarchical manner.

Handoff: At any time instant, a mobile phone logically belongs to one cell and is under control of its BS. When it moves physically from the cell, the BS notices the phone's fading signal and finds out from other neighbouring BSs as to which one is getting the strongest signal. It then transfers ownership of the mobile to the BS of that cell. If a call is in progress, the mobile is asked to switch to the new channel used in that adjacent cell. This process is called *handoff* or *handover*. A BS is only a radio relay; the channel assignment is done by the MTSO.

Handoffs can be either soft or hard. In a **soft handoff**, the mobile is acquired by the new BS before the old one signs off. Thus, there is no loss of continuity. But it requires the mobile to be able to tune to two frequencies at the same time. Neither first- nor second-generation devices can do this. 3G CDMA systems provide soft handover, resulting in seamless connectivity to the mobile. In a **hard handoff**, the old BS drops the mobile before the new one acquires it. The call is disconnected abruptly if there is no available frequency with the new BS, or there is a call drop till the new frequency is received. This is noticeable by the user but is typically of very short duration of about 60 ms in GSM systems.

Different kinds of handover are possible when a mobile moves from one cell to another or when traffic through a specific stage becomes very high. Readers are referred to (Kamal, 2007) for details.

The AMPS system uses 832 full-duplex channels, each consisting of a pair of simplex channels (824–849 MHz for transmission and 869–894 MHz for reception). The individual cells use different frequencies, using a system referred to as FDMA. Here the maximum supported bit rate is 19.2 Kb/s.

Although 1G communication provided a good start, its main disadvantage was low speed due to low available frequencies, interference due to frequency reuse and poor security.

2.4.2 The second generation (2G): 1992

The first generation of mobile phones was analog; the second generation was digital. The term *PCS* is sometimes used in the marketing literature to indicate the second-generation systems. Sometimes PCS is classified as a 2.5-generation (2.5G) system separately. The various advantages of digital cellular are as follows:

1. It is more robust as it displays resistance to noise and crosstalk and has efficient error correction.
2. It exhibits the intelligence of the digital network.
3. It is more flexible and can be integrated with the wired digital network.
4. Reduced RF transmission power is needed.
5. Encryption can be provided for communication privacy.
6. System complexity is reduced.
7. User capacity is increased.

There are two basic technologies for managing shared access in digital cellular systems, which are further classified as shown in Figure 2.5.

The IS-54 standard is a North American standard based on TDMA. It contains the 30 KHz spacing of AMPS to make the evolution from analog to digital easier. Each channel provides a raw bit rate of 48.6 Kb/s.

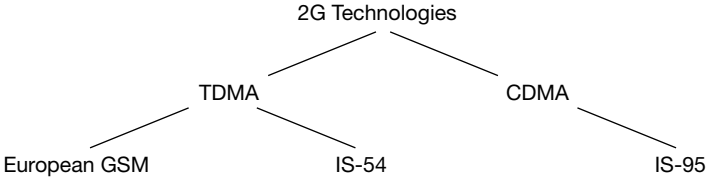


Figure 2.5 The 2G Technologies

The Pan-European GSM is based on TDMA with eight slots per radio channel. Each user transmits periodically in each of the slots with duration of 0.57 seconds. In the present version, GSM supports full-rate 22.8 Kb/s transmission.

In spite of many improvements over 1G, 2G still has many shortcomings. First, it still focuses only on low data rate speech service. Second, the capacity still does not satisfy the ever-growing demand, and finally multimedia service is still not provided.

2.4.3 The 2.5 generation (2.5G): 1996

As an interim step towards the higher data rates of 3G, whose technology differs considerably from the current cellular technology, some new techniques are being deployed as stopgap measures. They use incremental advances in cellular technology to increase the capacity of the currently deployed infrastructure.

Enhanced data rates for GSM evolution (EDGE) is a 2.5G system, with a data rate of 384 kbps, which is higher than GSM. The errors introduced by the high speeds necessitate the use of nine different schemes for modulation and error correction.

GPRS (general packet radio service) is another 2.5G scheme which is actually an overlay packet network over D-AMPS or GSM. In this scheme, voice Internet protocol (IP) packets can be exchanged between mobile senders and receivers, with speeds of 115 kbps. This is done by reserving some time slots on some frequencies for packet traffic. The BS can dynamically vary the number and location of the time slots, based on how much voice traffic is to be sent in the cell. The BS sends the packet received from the mobile unit, to the Internet, through a wired connection.

2.4.4 The third generation (3G): 2000 +

In 1992, 3G was envisaged by ITU as International Mobile Telecommunication (IMT2000), but it still has not seen the light of day. Its aim is to implement true 'anybody at any place' communication with 'anyone at any time'. IMT2000 is defined as a system aimed at 'the provision of worldwide mobile service through a limited number of wireless access points by combining various services and different systems'. It promises to connect up to 2 billion people worldwide by 2010 and offer data rates of up to 2 Mbps.

The frequency bands identified for IMT2000 are 1885–2025 MHz and 2110–2200 MHz. Its goals are to

- Support high mobile velocity (300–500 km/hour), compared with less than 100 km/hour in GSM.
- Support global wandering, as opposed to district and country in GSM.
- Support multimedia service, especially Internet service, 144 Kb/s (outdoor and higher velocity), 384 Kb/s (from outdoor to indoor, lower velocity), 2 Mb/s (indoor); speech with quality of service (QoS) and other services 4–100–200 Kbs/s (GSM, lower velocity).

- Convenience for transition and evolvement or innovation, compatibility of services with various fixed/mobile networks. High quality and security comparable to the fixed network.
- Highest spectrum availability, higher QoS, speech recognition technology, lower cost, higher security.
- Use the advantages of technologies such as adversity transmitting and receiving, multi-path combining, turbo code, channel estimation, signal-to-interference power ratio (SIR) measurement and Transmit power control (TPC), space-time technology, multi-user detection and interference cancellation, beam forming and smart antennas, and soft hand-off, etc.

Service targets for IMT-2000 are worldwide roaming, software radio and user identity module (smart card). Various services for users include multirate multimedia, that is, voice, image and high-speed data up to 2 Mbps.

2.4.5 The 3.5 generation (3.5G): 2000 +

The technical breakthrough towards 3.5G provides for an open architecture for service based on multimedia, and application of technologies such as smart antenna, software defined radio and TD-CDMA. The standardization focus has been moved from radio to network side, giving rise to the following advantages:

- Increased possibility to accommodate different types of radio in one system.
- A shift to the networking paradigm causes the problem of migrating legacy systems, so the effort for maintaining interoperability has been increased.
- Rapid upgrade of the standard for 4G, giving rise to more attention on the system evolution scenario.

2.4.6 The four th generation (4G): 2002 +

Between 1992 and 1995, there was a project in the European Community that was called Mobile Broadband System (MBS), which targeted future outdoor, cellular scenarios with high mobility and high data rates, to provide mobile multimedia communications. These systems will be the fourth mobile generation.

The European Radio communications Office (ERO) has proposed some features for 4G systems which include high bandwidth, ubiquity (connectivity everywhere), seamless integration with wired networks (especially IP), adaptive resource and spectrum management, software radio, besides high quality of multimedia service.

To implement the above features, innovative concepts are needed. The approach taken at the Mobile Multimedia Communication (MMC) project of the Delft University of Technology is to form a multi-disciplinary team in which user aspects get as much attention as the technological challenges. This MMC project has the following research goals:

- User interface and transparency
- **Compression:** Research has been carried out in two areas for source coding and two techniques have been proposed:
  1. H.263 for mobile video communication
  2. Compression of the shapes of video objects
- **Transmission protocols:** The MMC project uses a hierarchical protocol structure that provides different QoS to the various traffic streams in mobile multimedia communication. The hierarchy is realized with a hybrid TDM/FDM (time division multiplexing/frequency division

multiplexing) technique in which frames (the largest unit of data) are composed of packets, fragments and radio data units (RDUs).

- **Broadband radio transmission:** MMC transmission is located at the V band (from 40–75 GHz), centred at 60 GHz. New techniques for measurements have been proposed and carried out. The chosen modulation scheme is orthogonal frequency division multiplex (OFDM), which is specifically able to cope with the problems of the multipath reception.

However, 4G is still a distant dream, since as of today even 3.5G systems are yet to take off.

2.5 Summary

The electromagnetic spectrum contains all the frequencies that can be used in wireless communication and is the basis of all mobile computing. The different portions of the spectrum comprise radio waves, microwaves, infrared, and lightwaves, and their characteristics determine the data rates and applications in which each of these 'unguided' media can be used.

Communication satellites are an upcoming and useful long-range transmission system. Depending on their height of deployment, these can classified as geostationary orbit, medium earth orbit and low earth orbit and can be used in different applications.

Cellular communication has revolutionized the way mobile handhelds and phones are used. These handhelds are currently being used more for voice communication, but soon they will find widespread use for data. The first-generation systems were analog, and second-generation ones were digital with many options, like GSM, FDMA, TDMA and CDMA. There is a lot of talk about 3G, 3.5G and 4G systems, all of which are yet to take shape in reality. Each generation has improved on the capabilities of the older generation, with many new features added for broadband applications. Handover is an important aspect of all mobile systems and must be handled with proper care to provide seamless connectivity to mobile devices.

In the next chapter, we discuss wireless LAN (WLAN) standards, which are based on the short-range wireless communication technologies discussed in this chapter.

Problems

1. If a binary signal is sent over a 4 KHz channel whose signal-to-noise ratio is 20 dB, what is the maximum data rate achievable?
2. In a tabular form, compare radiowaves, microwaves and infrared waves, with respect to their data rates, transmission distance, interference and cost.
3. Repeat Question 2 by comparing the three satellite communication types, namely, GEOS, MEOS and LEOS.
4. Give typical applications for each of the three satellite systems.
5. Discuss how digital communication is better than analog communication.
6. Compare and contrast FDMA, TDMA and CDMA techniques.
7. Elaborate on the goals of IMT2000.
8. Identify the generation of your own mobile phone. Do you think it has the functionality discussed in this chapter for the relevant generation?
9. Differentiate between the two types of handoffs.

10. What are the features proposed for 4G systems?
11. Assume that 4G mobile technology handhelds are already available. Give some ideas for new features that can be added in 4G+ systems of tomorrow.

Multiple-choice questions

1. The higher frequencies, that is, ultraviolet light, X-rays and gamma rays, are normally not used for wireless transmission, because of which one of the following reasons?
   (a) They are difficult to produce and modulate
   (b) Do not propagate well through buildings
   (c) They are harmful to humans
   (d) All of the above
2. Which of the following is false for microwaves?
   (a) They travel in straight lines and are thus affected by the earth's curvature
   (b) They are relatively inexpensive to use
   (c) They can propagate well through buildings
   (d) They are preferred over optic fiber, especially in harsh terrain or urban areas
3. According to Shannon's theorem, the maximum data rate D of a noisy channel whose bandwidth is H Hz, and whose signal-to-noise ratio is S/N, is given by which one of the following formulae?
   (a) $D = H \log_2 (1 + S/N)$
   (b) $D = H (1 + \log_2 S/N)$
   (c) $D = 2H \log_2 (1 + S/N)$
   (d) None of the above
4. Which of the following is the correct sequence of waves in increasing order of frequencies?
   (a) Radio, microwaves, infrared, ultraviolet light, X-rays, gamma rays
   (b) Microwaves, radio, visible light, X-ray, ultraviolet light, gamma rays
   (c) Radio, microwaves, infrared, ultraviolet light, gamma rays, X-rays
   (d) Microwaves, radio, infrared, visible light, X-rays, gamma rays
5. Which one of the following is not true for infrared waves?
   (a) They are capable of traversing multiple paths without interferences
   (b) They are less secure than radio
   (c) They have reasonably high bandwidth
   (d) No government license is required for their operation
6. Globalstar satellites, which are close to the earth, do not need much power, and their round-trip delay is only a few milliseconds, are examples of which one of the following?
   (a) Geostationary satellites (GEOS)
   (b) Medium earth orbit satellites (MEOS)
   (c) Low earth orbit satellites (LEOS)
   (d) None of the above
7. To which one of the following generations does CDMA belong?
   (a) First generation
   (b) Second generation
   (c) Third generation
   (d) Fourth generation

8. Which one of the following is the multiple-access scheme used in GSM?
   (a) Time division multiple access (TDMA)
   (b) Frequency division multiple access (FDMA)
   (c) Code division multiple access (CDMA)
   (d) A combination of TDMA and FDMA
9. Which one of the following best characterizes IS 95?
   (a) a standard for cellular CDMA
   (b) a standard for cellular TDMA
   (c) a standard procedure for measuring indoor multipath propagation characteristics
   (d) a standard interconnecting base stations
10. The efficiency of a wireless system is given in which of the following units?
   (a) bits per second
   (b) bits per second per Hertz
   (c) bits per second per Hertz per km^2
   (d) None of the above

Further reading

A.S. Tanenbaum (2003), *Computer Networks*, 4th ed. (New Delhi, India: Pearson Education).

C. Shannon (1948), 'A Mathematical Theory of Communication', *Bell System Technical Journal*, 27 (July, October): 379–423, 623–656.

C.R. Casal, F. Schoute and R. Prasad, 'A Novel Concept for Fourth Generation Mobile Multimedia Communication', www.ubicom.tudelft.nl/MMC/Docs/VTC99.pdf (accessed November 2005)

———, 'Evolution towards Fourth Generation Mobile Multimedia Communication', www.ubicom.tudelft.nl/MMC/Docs/paper38.pdf (accessed March 2005)

D.P. Agrawal and Q.A. Zeng (2003), *Introduction to Wireless and Mobile Systems* (Thomson, Singapore).

J.F. Huber, D. Weiler and H. Brand (2000), 'UMTS, the Mobile Multimedia Vision for IMT-2000: A Focus on Standardization', *IEEE Communications Magazine*, 38 (September): 129–136.

J.H. Schiller (2006), *Mobile Communications*, 2nd ed. (Pearson Education, USA).

J.S. Lee and L.E. Miller (1998), *CDMA Systems Engineering Handbook* (London: Artech House).

R. Kamal (2007), *Mobile Computing* (Oxford University Press).

X. Zhou, 'Overview of the Third Generation Mobile Communications', www.meru.cecs.missouri.edu/workshop/zxb_pres1.ppt. (accessed January 2005)

3 Wireless Networ ks

With the advent and recent proliferation of handheld devices, wireless local area networks (WLANs) have become very popular. One can see them in offices, campus buildings, airports, hotels, restaurants, etc., facilitating continuous access to the Internet, through what has come to be known as the **wireless indoors**. Recently, the concept of the **wireless outdoors** has also emerged, which is concerned with the so-called last mile technology or wireless local loop (WLL) or fixed wireless access. To provide connectivity to millions of homes and businesses one has to lay fibre, coax, or category 5 twisted pair, which is a very daunting and costly affair. The provider uses a directed antenna and a transmitter of predefined power to ensure stable reception of high-frequency signals within a limited coverage area, such as an individual building.

WLL can be narrowband or wideband. Broadband wireless or wireless metropolitan area networks (WMANs) simply require erecting a big antenna on a hill just outside the town and installing antennas directed at it on customers' rooftops. We shall study both WLAN and WMAN standards in this chapter.

WLANs can operate in two configurations—with base stations or access points that are connected to the wired network, or without base stations, that is, mobile ad hoc networks (MANETs). MANETs are the subject of discussion in Chapter 6. Both configurations, however, use the short-range radio-wave transmission discussed in Section 2.2. See Figure 3.1.

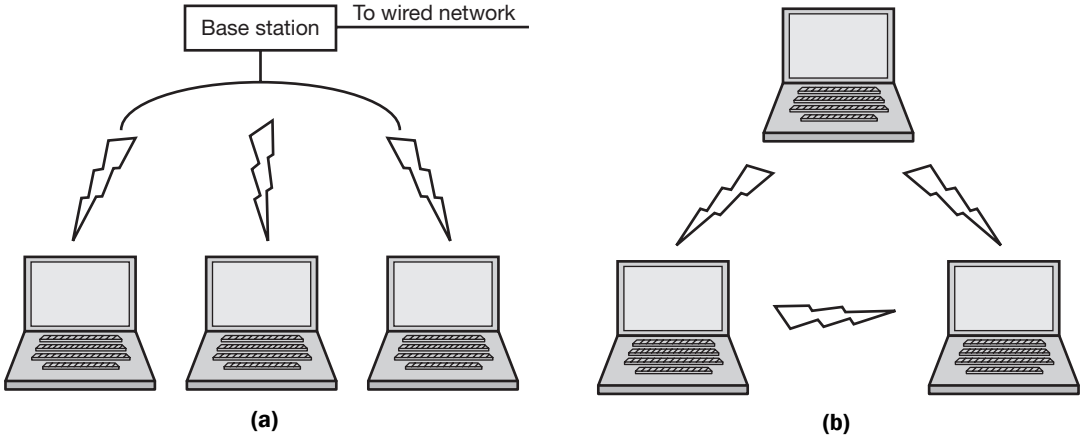


Figure 3.1 Wireless Networ ks **(a)** With Base Station **(b)** Without Base Station

When wireless networks were first developed, there were many challenges that had to be met. Some of them have been mentioned in Chapter 2. These challenges included finding an available worldwide frequency band, dealing with the finite range of signals, maintaining user privacy, taking limited battery life into account, understanding the implications of mobility, making the system economically viable, etc. We shall be dealing with all these issues in the book.

3.1 The need for ne w wireless standards

The main standard developed for WLANs is called the IEEE 802.11. So the question that arises here is: what is the need for a new standard, that is, why can't the universal Ethernet be used for WLANs? The answer lies in the many ways in which wireless operation differs from the traditional wired one. Some of these are discussed below.

1. Ethernet uses carrier sense multiple access with collision detection (CSMA/CD). An Ethernet station just waits until the ether is idle and starts transmitting. If it does not receive a noise burst back within the first 64 bytes, it assumes that the frame has been delivered correctly. But carrier sensing is not possible in the wireless environment. Also, not all stations are within the radio range of each other. Transmissions going on in one part of a cell may not be received elsewhere in the same cell. There are two problems encountered in this scenario—the problem of the hidden station and the problem of the exposed station.

a. The hidden station problem: Shown in Figure 3.2 is a WLAN containing stations A, B and C. C, which is not in the radio range of A, is transmitting to station B. If station A senses the channel, it will not hear anything because it is hidden from C. It falsely concludes that it may now start transmitting to B, resulting in a collision.

b. The exposed station problem: Consider the same WLAN, but now the scenario is as shown in Figure 3.3. A is transmitting to some station D not shown in the diagram. B is near A and can hear A sending. It falsely concludes that it cannot transmit to C, even though it can do so simultaneously. Thus, because of B's exposed location to A, it defers its transmission even when it need not.

2. Multipath fading (interference). This is due to reflection of radio signals by solid objects, which results in signals being received along multiple paths. This may cause interference, leading to data becoming error-prone in the wireless environment.

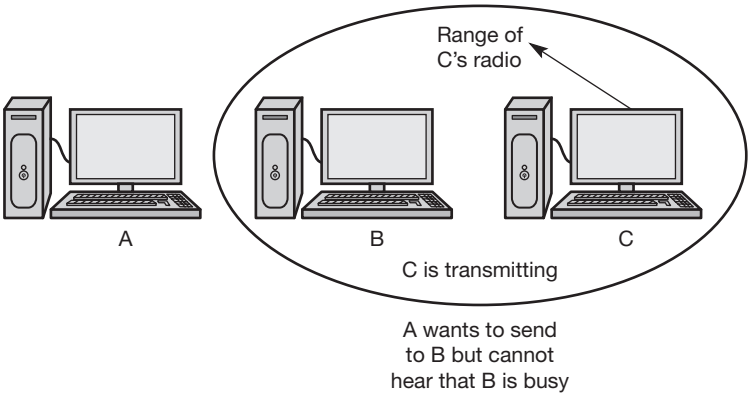


Figure 3.2 The Hidden Station Problem

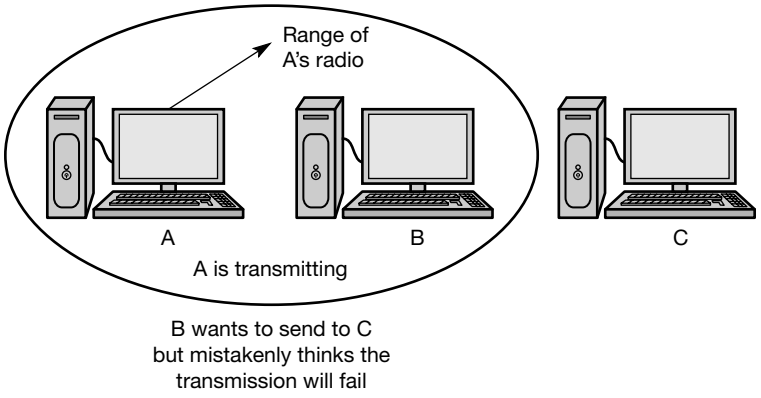


Figure 3.3 The Exposed Station Problem

3. No handoff in Ethernet. Handoff is the mechanism of allowing a mobile device continued access even when it moves from one cell (network) to another. This is a major requirement in wireless networks, but is not needed in the wired Ethernet.

4. Half-duplex transmission. Most radios are half duplex. They cannot transmit and listen for noise bursts at the same time in a single frequency. Thus, carrier sensing is not possible.

5. Absence of mobility-aware software. Software that is mobility-based or mobility-aware is yet to be made universally available. Until that happens, WLANs of mobile, handheld computers cannot be deployed as universally and simply as the standard Ethernet.

The above limitations of the standard Ethernet necessitated the development of a new standard for WLANs.

3.2 IEEE 802.11 WLAN standard

The IEEE 802.11 WLAN Standard is popularly known as the Wi-Fi standard. We shall now study in detail its protocol stack, frame structure and services. The physical layer radio-transmission techniques are beyond the scope of this book, but we shall mention them briefly here. Figure 3.6 shows the lower two layers of the IEEE 802.11 protocol stack. Here the data link layer consists of two sublayers, called the logical link control (LLC) layer and the medium access control (MAC) layer. The IEEE 802.11 protocol stack is discussed in detail below.

3.2.1 Physical layer

The 802.11 standard was developed in 1997 with data rates of 1 to 2 Mbps for WLANs. Initially, it had three possible modulation techniques for sending MAC frames from a sender station to a receiver station. Only some highlights of these techniques are given below and are as follows. For details, please refer to Tanenbaum (2003).

- **802.11: Infrared**, which uses diffused transmission at 0.85 or 0.95 microns. Two speeds are permitted, those of 1 Mbps and 2 Mbps. The advantage of infrared transmission, as seen in Chapter 2, is that infrared signals do not penetrate walls, so cells in adjacent rooms are well insulated from each other. But it is not good in sunlight, as sunlight swamps infrared signals. Further, bandwidth is limited.

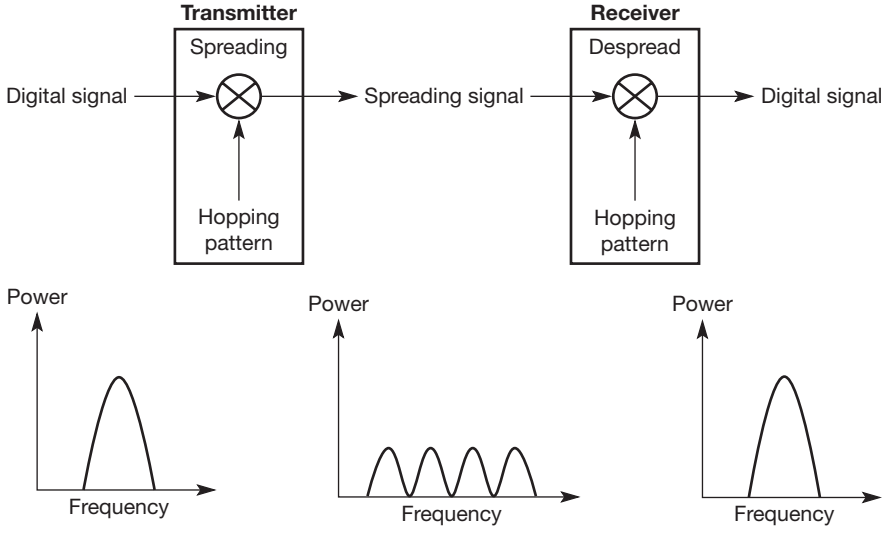


Figure 3.4 Concept of F requency Hopping Spread Spectr um (FHSS)

- **802.11: FHSS (frequency hopping spread spectrum)**, in which the transmitter hops from frequency to frequency hundreds of times per second. It uses 79 channels, each 1 MHz wide, starting at the low end of the 2.4 GHz ISM (industrial, scientific, medical applications) band. A pseudorandom generator is used to produce the sequence of hopped frequencies. Figure 3.4 shows the concept of FHSS. Stations need to use the same seed for the pseudorandom generator and stay synchronized in time to hop to the same frequencies. The dwell time, which is the amount of time spent at each frequency, is adjustable, but must be less than 400 msec. Since the hopping sequence and dwell time are not known, FHSS provides security against eavesdropping. It is resistant to multipath fading and is relatively insensitive to radio interference, which makes it popular for building-to-building links, that is, for wireless outdoors. Its main disadvantage is its low bandwidth and low power.
- **802.11: DSSS (direct sequence spread spectrum)** is like CDMA, but has some differences. It is also restricted to 1 or 2 Mbps. Each bit is transmitted as 11 chips in what is called a Barker sequence. Phase shift modulation is used at 1 or 2 Mbaud to transmit 1 or 2 bits per baud, when operating at 1 or 2 Mbps, respectively. The concept of DSSS is shown in Figure 3.5.

Subsequently, these speeds were considered too slow, and in 1999, two new standards were proposed. These are as follows:

- **802.11a: OFDM (orthogonal frequency division multiplexing)**, which uses the wider 5 GHz ISM frequency band to deliver up to 54 Mbps. In OFDM, which is a form of spread spectrum, but different from CDMA and FHSS, 52 different frequencies are used: four for synchronization and 48 for data. Splitting the signal into many narrow bands offers key advantages like better immunity to narrowband interference and the possibility of using non-contiguous bands. It also has good spectrum efficiency in terms of bits/Hz and good immunity to multipath fading.
- **802.11b: HR-DSSS (high-rate DSSS)** is another spread-spectrum technique, which uses 11 million chips/second to deliver data rates up to 11 Mbps in the 2.4 GHz band. Data rates

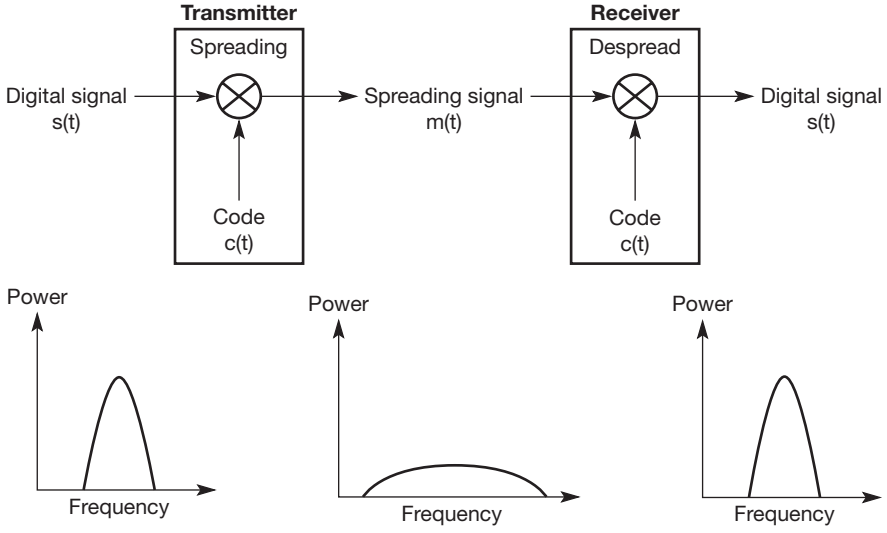


Figure 3.5 Direct Sequence Spread Spectr um (DSSS)

supported are 1, 2, 5.5 and 11 Mbps. These rates may be dynamically adapted during operation to achieve the optimum speed possible under current conditions of load and noise. Although it is incompatible with 802.11a and is much slower, its range is 7 times greater.

In 2001, another standard was proposed, which is

- **802.11g:** This uses the modulation technique of 802.11a, that is, OFDM, and the frequency band of 802.11b, so it theoretically delivers up to 54 Mbps data rates.

3.2.2 MAC layer

To overcome the hidden and exposed terminal problems of the CSMA/CD-based Ethernet, the MAC sublayer of 802.11 supports two modes of operation. These are the DCF and the PCF (which is optional) and are discussed below.

1. Distributed Coordination Function (DCF): As the name suggests, this mode does not use any central control like the Ethernet. But it uses CSMA/CA, that is, CSMA with collision avoidance, which itself supports two methods of operation.

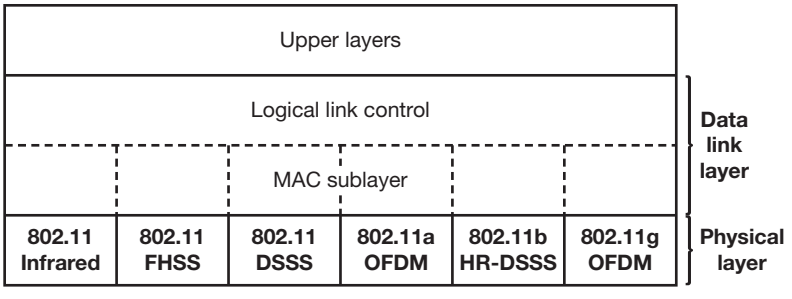


Figure 3.6 The Lower Layers of the IEEE 802.11 Protocol Stack

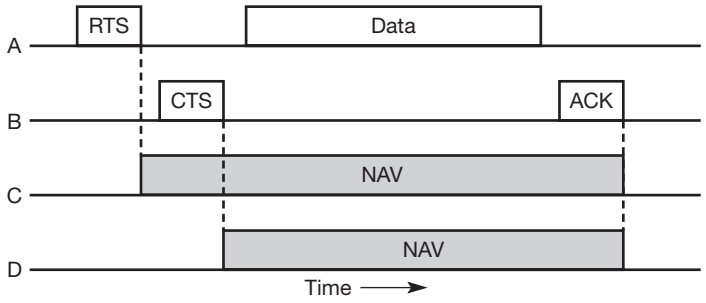


Figure 3.7 CSMA/CA V irtual Channel Sensing

The first method uses **physical channel sensing**. When a station wants to transmit, it senses the channel. If it is idle, it starts transmitting. It does not continue to sense the channel while transmitting, but sends the complete frame, which may be destroyed at the receiver due to interference there. If a collision occurs, the colliding stations wait for a random time, using the Ethernet binary exponential backoff (BEB) algorithm, and then try again later. If the receiver does not send an acknowledgement, the transmitter knows that a collision has occurred. There is no collision detection at the transmitter.

The second method is based on **multiple access with collision avoidance for wireless** (MACAW) and uses virtual channel sensing. It works as shown in Figure 3.7. Suppose there are four stations A, B, C and D in a network, such that B and C are within the range of A. D is not within A's range but is within the range of B.

Suppose A decides to send data to B. The protocol works as follows:

1. A sends a small 30 byte RTS (request to send) frame to B.
2. If B is ready to receive data, it responds with a CTS (clear to send) frame.
3. When A receives the CTS, it sends its data frame and starts an acknowledgement (ACK) timer.
4. If B correctly receives the data frame, it responds with an ACK frame and terminates the exchange.
5. In case A's ACK timer expires before it receives the ACK, the whole protocol is repeated.
6. C also receives the RTS frame, as it is in the range of A. It realizes that someone else wants to send data, so it stops transmitting till the data exchange is done.
7. D receives the CTS frame as it is in the range of B. Thus, it also maintains the same state as C.

Note that the signals shown in Figure 3.7 for C and D, called network allocation vector (NAV), are not transmitted. They are internal reminders to indicate that no data can be transmitted during that time. This is a kind of virtual channel busy signal, asserted by the stations themselves, using the NAV. The time for which they must wait can be calculated using the information present in the RTS and CTS frames.

Because of the noisy, wireless channel, the probability of the frame reaching the destination successfully decreases with frame length. For noisy channels, 802.11 allows frames to be fragmented into smaller pieces, each with its own checksum. Once the channel has been acquired using RTS and CTS, multiple fragments can be sent in a row (see Figure 3.8). The sequence of fragments is called a **fragment burst**. Fragmentation increases the throughput by allowing only bad fragments to be retransmitted, not the whole frame.

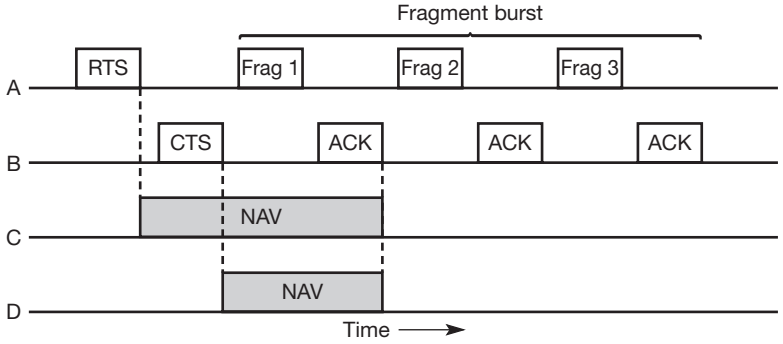


Figure 3.8 IEEE 802.11 F ragment Burst

2. Point Coordination Function (PCF): This mode of operation uses a central base station which polls other stations, asking them if they have any frames to send. No collisions can occur here. The base station broadcasts a **beacon frame** periodically, with the necessary system parameters, viz., hopping sequence, dwell times, clock synchronization, etc. It also invites new stations to sign up for the polling service. During signing up, the station is guaranteed a certain fraction of bandwidth to maintain quality of service (QoS). To save battery life, the base station can direct a mobile station to go into sleep state until explicitly awakened by the base station or the user. While the mobile station is asleep, the base station buffers any frames directed to it.

802.11 allows both PCF and DCF to coexist within one cell by carefully defining the interframe time interval. After a frame has been sent, a certain amount of time is required before any station may send another frame. Four different intervals are defined, each for a specific purpose. These are shown in Figure 3.9.

1. SIFS (short interframe spacing) is used to allow the parties in a single dialog to go first. This includes sending a CTS frame, ACK frame and fragment bursts.
2. PIFS (PCF interframe spacing) is used by exactly one station to respond after an SIFS interval. If the station fails to make use of its chance and the time PIFS elapses, the base station sends a beacon frame or poll frame.
3. DIFS (DCF interframe spacing) is the time after which the base station does not respond. Then any station may attempt to acquire the channel to send a new frame. The usual contention rules apply here.
4. EIFS (extended interframe spacing) is the time used by a station to report error if it has a received a bad or unknown frame.

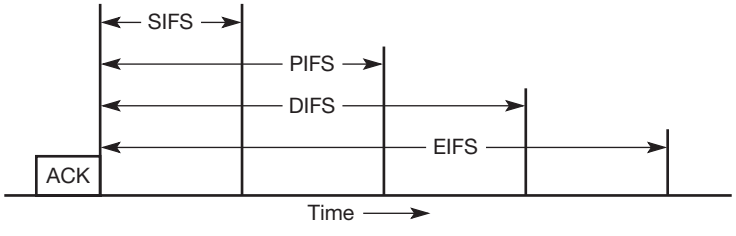


Figure 3.9 IEEE 802.11 Interframe Spacing

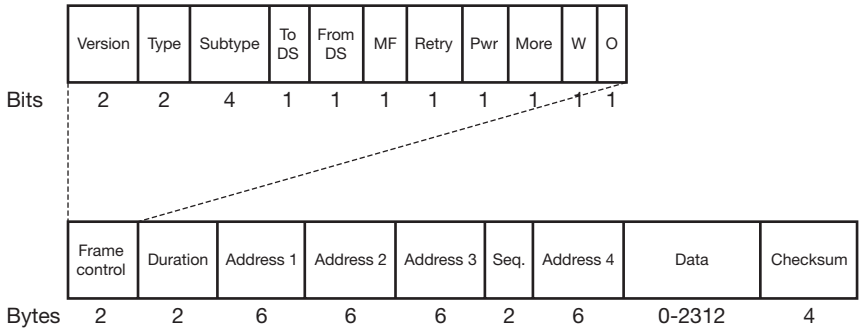


Figure 3.10 Data Frame Format for IEEE 802.11

3.2.3 Frame structure

The frame structure for the 802.11 data frame is shown in Figure 3.10.

- The *frame control* field is 2 bytes long and has various subfields.
- *Version* field gives the version of the protocol as usual.
- *Type* indicates whether it is a data, control or management frame.
- *Subtype* is concerned with the type of control frame.
- *To DS* and *From DS* indicate that the frame is going to or coming from the intercell distribution system, that is, the Ethernet.
- *MF* indicates more frames will follow.
- *Retry* indicates retransmission of a frame sent earlier.
- *More* indicates that the sender has additional frames to send.
- *Pwr* is the power management bit used by the base station to put the receiver into or take it out of sleep mode.
- *W* specifies that the wireless equivalent privacy (WEP) algorithm is used for encryption. This algorithm will be discussed in detail in Chapter 9.
- specifies to the receiver that a sequence of frames with this bit on must be processed strictly in order.
- *Duration* field tells for how long the frame and its ACK will occupy the channel.
- *Address 1* is the sender's address.
- *Address 2* is the receiver's address.
- *Address 3* is the sender's base station address.
- *Address 4* is the receiver's base station address.
- *Sequence* field allows fragment numbering—12 bits for frame and 4 bits to identify fragment.
- Management frames have a similar format, except that they have only one base address, because they are restricted to a single cell.
- Control frames are shorter. They have only one or two addresses and no *data* and *sequence* fields. *Subtype* is important here.

3.2.4 Services

A WLAN must provide **nine** types of services, five for distribution and four for the station.

The **distribution** services are concerned with managing membership within a cell and for interacting with stations outside the cell. They are as follows:

1. **Association:** This service is used when a mobile station wants to connect to the base station. The mobile station must identify itself and indicate the data rates supported by it, whether

it needs PCF services (polling) and what its power requirements are. The base station may accept or reject the mobile station. However, if it is accepted, the mobile station has to authenticate itself.

2. **Disassociation:** This service is used when either the mobile station or the base station wants to break the connection. The base station may do so for maintenance purposes or if it wants to go down. The mobile station may disassociate when it is leaving or shutting down.
3. **Re-association:** This service is used when a mobile station wants to change the base station, as when it moves from one cell to another.
4. **Distribution:** This service deals with routing of frames that are sent to the base station. If the destination of the frame is local, it is sent directly over the air; else, it is forwarded over the wired network.
5. **Integration:** This service is used if the frame has to be sent to a non-802.11 network. The frame must be translated from the 802.11 format to the format of the destination network.

The **station** services are concerned with working within the same cell. They are used after a mobile station has associated with a base station and are as follows:

1. **Authentication:** This service is used by the base station to check the identity of the mobile station. Initially, the standard did not require the base station to prove its own identity to the mobile station, but this defect in the standard is being corrected.
2. **De-authentication:** When a mobile station that has been authenticated wants to leave the network, it is de-authenticated by the base station. After de-authentication, it cannot use the network anymore.
3. **Privacy:** This service is used to ensure that the data in the wireless network is confidential. Encryption is used for this purpose.
4. **Data delivery:** This service is the one used by mobile stations to send and receive data. It is a reliable service requiring the higher layers to provide for error detection and correction.

3.3 Bluetooth

Bluetooth was developed in 1994 by a study interest group (SIG) consisting of IBM, Intel, Nokia and Toshiba for connecting mobile phones or computing and communication devices without the use of cables. In 2002 it was taken up by the IEEE wireless personal area network (WPAN) Committee as the IEEE 802.15 standard, for the physical and data link layers.

It is a short-range, low-cost and power-efficient radio-frequency-based wireless technology that supports both point-to-point and point-to-multipoint connections. It connects one handheld device to another Bluetooth-enabled device(s) within a 30-foot or 10-meter radius, such as mobile phones, laptops, printers and other accessories. It is like having a universal remote for the kinds of devices one uses every day and is oriented towards the mobile consumer wanting to do digital imaging and multimedia applications.

Bluetooth operates in the unlicensed ISM band, with slight locational variations. Its essential characteristics are summarized in Table 3.1. Bluetooth-enabled devices can automatically locate each other, but user action is necessary to make connections with other devices and to form networks.

Eight devices can be connected in a Bluetooth network, known as a **piconet.** One of them acts as the master and the others are called slaves. A **scatternet** is formed when two or more piconets connect via a bridge node. This is shown in Figure 3.11.

Table 3.1 Bluetooth Features

Characteristics	Description
Physical Layer	FHSS
Frequency Band	2.4–2.4835 GHz
Hop Frequency	1,600 hops/s
Data Rate	1 Mbps
Data and Network Security	Provides three levels of security, two levels of device trust and three levels of service security.
Operating Range	10 m
Throughput	Around 720 Kbps
Advantages	No wires and cables for many interfaces, can penetrate walls and other obstacles, uses low power and minimal hardware.
Disadvantages	May interfere with other ISM band technologies, has low data rates.

In addition to the seven active slaves, there can be up to 255 parked nodes (in low power state) in the net that can only respond to a beacon signal from the master. Slaves are dumb devices, doing what the master tells them to do. The piconet is a TDM system, with the master controlling the clock and determining which slave gets to communicate in which time slot. All communication is between master and slave, not between slave and slave.

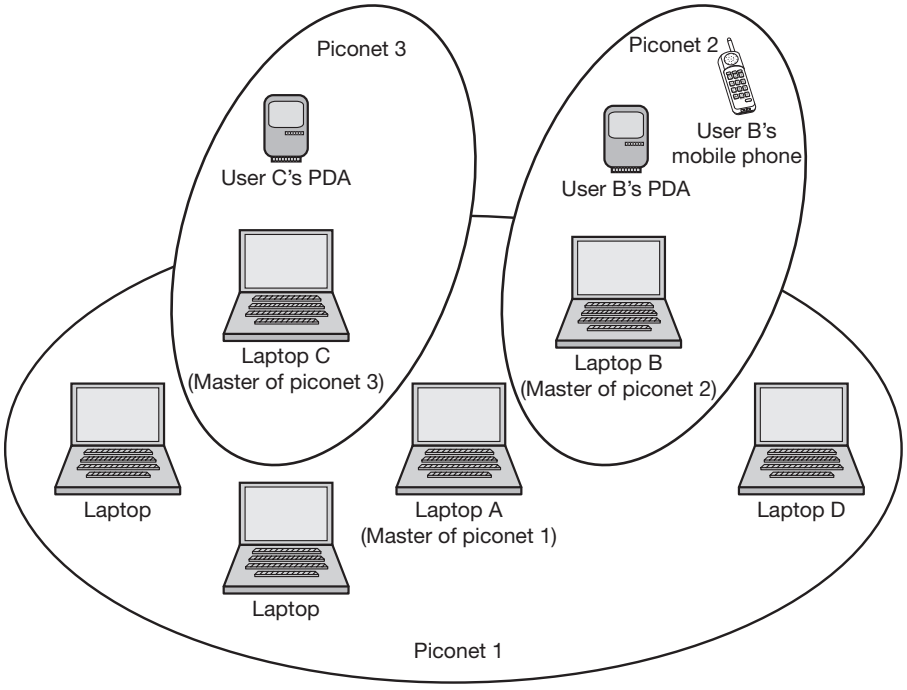


Figure 3.11 Bluetooth Scatter net

3.3.1 Advantages of Bluetooth

Bluetooth offers many advantages to users, both the home user and the small business user. This is because of its simple connectivity, which provides increased efficiency and reduced costs. These advantages are as follows:

1. Non-cable connections: Bluetooth technology has replaced cables by wireless for many short-range interconnections. These include mouse and keyboard computer connections; 12 Mbps and 480 Mbps USB (USB 1.1 and 2.0); printers and modems, at 4 Mbps; and wireless headsets and microphones for interfacing with PCs and mobile phones.

2. File sharing: File sharing between Bluetooth-enabled devices has become very simple, enabling Bluetooth-compatible laptops to share files with each other or Bluetooth-compatible mobile phones to act as wireless modems for laptops. Thus Bluetooth provides the laptop with complete networking facilities without using an electrical interface for the same.

3. Wireless synchronization: Bluetooth-enabled devices can automatically synchronize with each other wirelessly. This facilitates personal information present in address and appointment sheets to be transferred between PDAs, laptops, palmtops and mobile phones.

4. Wireless Internet connectivity: Since Bluetooth is supported by a variety of devices, Internet connectivity is possible when these devices connect together and use each other's capabilities. Thus a laptop with a Bluetooth connection can request a mobile phone to establish a dial-up connection and then access the Internet through that connection.

Bluetooth will soon be available in office appliances like PCs, faxes, printers and laptops; communication devices like cell phones, handsets, pagers, and headsets; and home systems like DVD players, cameras, refrigerators and microwave ovens. Many other exciting applications for Bluetooth include vending machines, banking and other electronic payment systems, wireless office and conference rooms, smart homes and in-vehicle communications and parking.

3.3.2 Bluetooth applications

The Bluetooth standard provides for 13 specific applications to be supported by Bluetooth V1.1. These are given in Table 3.2.

3.3.3 Bluetooth protocol stack

The Bluetooth protocol stack defines the software layers used for communication on top of the radio link. The lower layer defines the Bluetooth-specific components. The middle layer consists of the industry standard protocols that were adapted for Bluetooth use so that applications can be ported to Bluetooth easily. The top layer is the application layer. See Figure 3.12.

The Bluetooth radio layer defines the transmission characteristics. It sends bits from the master to the slave and vice versa. It is a low-power system with a range of 10 m. Bluetooth transceivers use Gaussian frequency shift keying (GFSK) modulation and employ FHSS with a hopping pattern of 1,600 hops/sec over 79 frequencies in a quasi-random fashion. Although the theoretical maximum data rate of a Bluetooth network is 1 Mbps, in reality it cannot support such data rates because of communication overhead. Second-generation Bluetooth technology is likely to go up to a data rate of 2 Mbps.

Bluetooth networks can support both data and speech channels. One asynchronous data channel can be combined with up to three simultaneous synchronous speech channels. A combination of packet-switching technology and circuit-switching technology is used in

Table 3.2 Bluetooth Applications

Name	Description
Generic access	procedures of secure link establishment and management
Service discovery	discovers what services other devices offer
Serial port	transport protocol for emulating a serial line
Generic object exchange	defines a client-server relationship for data movement
LAN access	protocol between a mobile and a fixed LAN
Dial-up networking	allows a laptop to call via mobile phone
Fax	as above, but to send faxes
Cordless telephony	connects a handset and its local base station
Intercom	provides digital walkie-talkie between two telephones
Headset	allows hands-free voice communication (between headset and base station)
Object push	a way to exchange simple objects, like business cards
File transfer	a more general file transfer facility
Synchronization	allows a PDA to synchronize with another computer

Bluetooth. Packet switching allows Bluetooth devices to send multiple data packets on the same path.

It is to be noted that because both 802.11 and Bluetooth operate in the 2.4 GHz ISM band, they interfere with each other, and since Bluetooth hops faster than 802.11, a Bluetooth is more likely to ruin 802.11 transmissions, rather than the other way around.

The Bluetooth baseband layer is the link-level protocol that allows links to be established between Bluetooth devices. It encapsulates the raw bit stream into frames. Many formats are possible. In one format, the master provides TDM time slots of 625 μs. The master's transmissions begin in the even slot and the slaves' transmissions begin in the odd ones, giving the master half the slots with all the slaves sharing the other half. Frames can be sized 1, 3, or 5 slots in length.

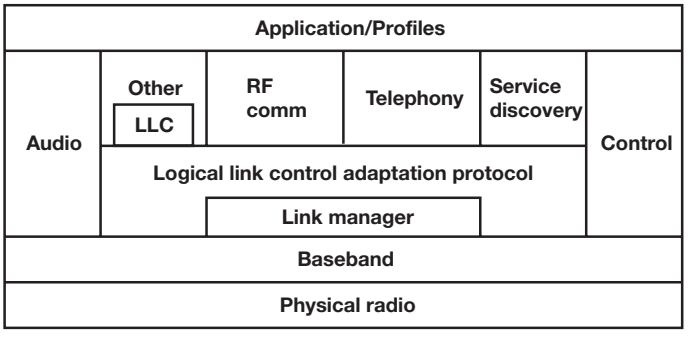


Figure 3.12 The Bluetooth Protocol Stack

A logical channel, called a link, is used to transfer data between the master and a slave. Two kinds of links are possible:

1. The **asynchronous connectionless** (ACL) link is used for packet-switched data coming from the L2CAP layer on the sending side and delivered to the L2CAP layer on the receiving side. ACL traffic is a best-effort one, with no guarantees given; frames can be lost and may have to be retransmitted. A slave can have only one ACL link with the master.

2. The **synchronous connection-oriented** (SCO) link is used for real-time data such as telephony. In this, the channel is allocated a fixed slot in each direction. Due to the time-critical nature of SCO links, frames are never retransmitted, and forward correction is used to provide high reliability. A slave can have up to 3 SCO channels with its master, and each link can transmit one 64 Kbps PCM audio channel.

The **link manager protocol (LMP)** is a data link layer protocol that communicates with its peer in the target device. It defines the messages used by software in the master for polling the client even when the master has no data to transport. Messages are exchanged between participating devices to set up and maintain links. LMP messages are transported in the payload data field of a Bluetooth data packet.

These messages also control authentication and encryption. Authentication is carried out using a challenge/response protocol between the verifier and claimant. After authentication, the participating devices must agree on common encryption parameters including encryption type and key.

The **logical link control and adaptation protocol (L2CAP)** is used by upper-layer protocols for data transport. It can be thought of as an adapter between upper and lower layers. It provides both connectionless and connection-oriented service. L2CAP messages are transported in the payload data field of Bluetooth data packets just as LMP messages are. L2CAP is responsible for upper-level protocol data segmentation and reassembly as well as transport of QoS information. It can accept blocks of data up to 64 kB in length and will reliably transport them to its peer in the target device using services provided by the baseband layer by creating a logical channel from one device to the other.

The **Bluetooth audio protocol** belongs to the data link level. It is the only protocol to use the SCO link for synchronous communication. An application can obtain synchronous services simply by opening a voice data link.

The **service discovery protocol (SDP)** is used to find new services as they become available and to deregister services that become unavailable. It is optimized for a fast-changing environment.

3.3.4 Bluetooth tracking ser vices

A piconet is formed in an ad hoc manner as devices come into proximity of one another. When a new device enters the piconet, it brings services with it that can be used by other devices. When a device leaves a piconet, its services become unavailable. A device already registered in a piconet may make new services available. An application running on a Bluetooth device can request services according to the class of service or through specification of service characteristics. It is also possible for an application to request a list of all available services.

An SDP client application can issue requests to the local SDP server or any SDP server in the piconet, which in turn will return the relevant information. All Bluetooth devices include an SDP server application that keeps track of the services available on that device. A service record,

describing attributes like service class ID, provider name, icon URL, service ID, etc., is kept for each service available on that device.

The **Radio frequency communication interface (RFCOMM)** provides an RS-232 serial port emulation to the application or to higher-level protocols. It provides a common programming interface for actual serially connected devices such as printers and modems as well as communication through a Bluetooth link. RFCOMM can support up to 60 concurrent connections.

The **Bluetooth telephony control protocol specification—binary (TCS-BIN)** defines the call-control signals for establishing a voice connection between Bluetooth devices. It has three major components. The call control component handles the establishment of a voice connection between two devices. Group management provides support for signaling among groups of devices in a piconet. The connectionless TCS component provides signaling for applications that do not require a synchronous voice connection. The **advanced telephony (AT)** commands enable telephony control. These commands are generally used for modem control, but they also support FAX transfer.

The remaining protocols in the stack use the RFCOMM interface. An Internet protocol (IP) stack including (user datagram protocol (UDP) and transmission control protocol (TCP) can be implemented using point-to-point protocol (PPP) as an interface to RFCOMM. The IP stack can be used to provide wireless access protocol (WAP) capability on Bluetooth devices. WAP provides value-added services such as email delivery across wireless links. This can be useful when updating email on a laptop. The laptop would be connected to the mobile phone using Bluetooth. The email would be transmitted to the mobile phone and from the phone to the laptop using WAP as a transport protocol.

The **object exchange protocol (OBEX)** is used to exchange data objects. It provides a session layer service for synchronization and file transfer applications. It can be implemented on Bluetooth either through the TCP/IP stack or through the RFCOMM interface. It can be used for calendar or email synchronization, generic file transfer or for pushing business card data from one Bluetooth device to another.

3.3.5 Bluetooth frame str ucture

The Bluetooth frame format is shown in Figure 3.13.

- The *access code* identifies the master so that slaves within the radio range of two masters can tell which traffic is for them.
- The 54-bit *header* contains an 18-bit header which is repeated 3 times for a total of 54 bits. This allows a receiver to examine all three copies of each bit and reject the one which does

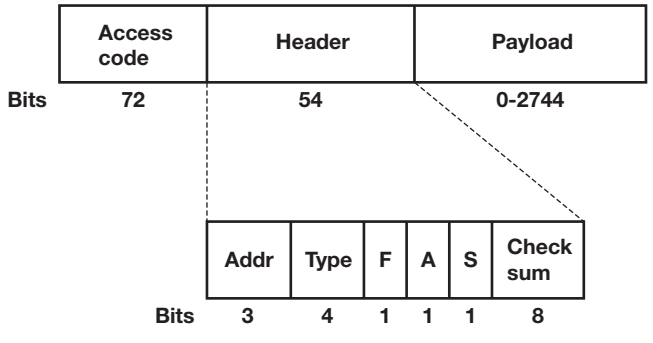


Figure 3.13 Bluetooth Frame Format

not have a majority. Note that such high redundancy is required for reliable transmission when using lower-power devices of 2.5 mW on a noisy channel.

- The *Address* field in the header identifies which of the eight devices the frame is intended for.
- The *Type* field identifies whether it is an ACL, SCO, poll or null frame type, the type of error correction used in the data field and how many slots long the frame is.
- The *Flow* bit is used for a primitive type of flow control and is asserted by a slave when its buffer is full and it cannot receive any more data.
- The *ACK* bit is used to piggyback an acknowledgement onto a frame.
- The *sequence* bit is used to number the frames to detect retransmissions in a stop-and-wait protocol.
- This is followed by the 8-bit *checksum*.
- The *data* field is up to 2744 bits in the case of a five-slot transmission and is 240 bits for a single time slot.

3.4 Infrared systems

The Infrared Data Association (IrDA) has defined a number of standards governing infrared wireless communication. These include the IrDA-data and IrDA-control standards. IrDA offers high data rates of 4 Mbps and 16 Mbps.

Infrared technology, introduced in Chapter 2, is similar to Bluetooth in that it is also useful for short-range communication. However, there are significant differences due to the characteristics of the transport medium. Infrared light emitters such as light-emitting diodes and laser diodes are directional devices. The emitted light propagates in a cone from the light source. This makes it easy to implement 'point and shoot' applications using IrDA. Bluetooth radio transmitters, however, are omnidirectional devices.

The IrDA protocol stack is shown in Figure 3.14.

In the **physical layer**, the infrared port allows half-duplex operation. IrDA achieves duplex operation by alternating the direction of the data link. The range of operation is up to 1 m. Low-power implementations allow ranges up to 30 cm. The hardware can operate in three modes: asynchronous serial infrared (from 9600 bps to 115.2 Kbps), synchronous serial infrared (1.152 Mbps) and synchronous infrared using pulse position modulation (4 Mbps).

The **infrared link access protocol (IrLAP)** provides connectionless and connection-oriented transport services to the upper layer. IrLAP provides for reliable communication between IrDA devices.

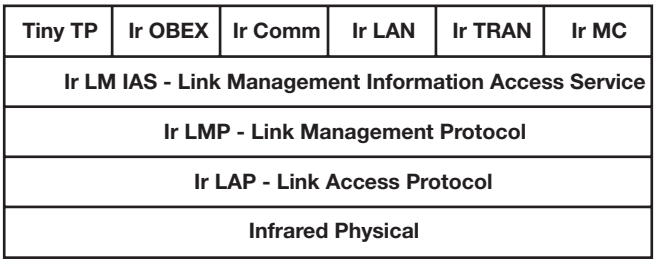


Figure 3.14 The IrDA Protocol Stack

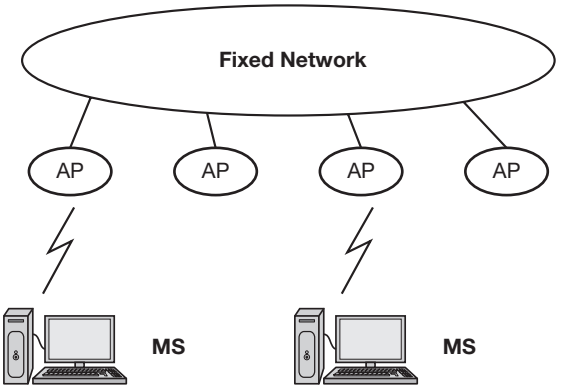


Figure 3.15 A Typical HiperLAN System

The **infrared link management protocol (IrLMP)** supports ad hoc connections with peer devices. IrLMP multiplexes data from multiple applications over the single data link. It also supports applications that request exclusive link access.

The **link management-information access service (IrLM-IAS)** layer is responsible for discovery. It determines which services another device has to offer by retrieving data from its discovery database.

A number of high-level protocols like Tiny-TP, IrOBEX, IrComm, IrLAN, IrTRAN, and IrMC provide services to the applications. These are beyond the scope of this book.

3.5 HiperLAN

The high-performance LAN (HiperLAN) is another wireless standard derived from traditional LAN environments and can support multimedia and asynchronous data effectively at high data rates of 23.5 Mbps. It is primarily a European standard and was published in 1995–96. It does not necessarily require any type of access point infrastructure for its operation, although a LAN extension via access points can be implemented. See Figure 3.15.

HiperLAN comes in two versions: the original HiperLAN/1, and HiperLAN/2, which was published in 2000. We shall describe below the salient points of each. For details, the reader is referred to the literature.

HiperLAN/1 employs 5.15 GHz and 17.1 GHz frequency bands and has a data rate of 23.5 Mbps, with coverage of 50 m and low mobility of <10 m/s.

- It supports a packet-oriented structure, which can be used for networks with or without a central control (base station-mobile station and ad hoc) and can support 25 audio connections at 32 Kbps with a maximum latency of 10 ms, 1 video connection of 2 Mbps with 100 ms latency and a data rate of 13.4 Mbps.
- It thus effectively supports Motion Pictures Expert Group (MPEG) or other state-of-the-art real-time digital audio and video standards.
- Because of its 5 GHz band, there is no conflict with home microwave ovens, etc. It uses CSMA/CA and per-session encryption.

- HiperLAN/1 describes the standards for service and protocols of the two lowest layers of the OSI model. Its MAC layer is compatible with the standard MAC service interface, enabling support for existing applications to remain unchanged.

HiperLAN/2 has been specifically developed to have a wired infrastructure, providing short-range wireless access to wired networks like IP and ATM. The two main differences between HiperLAN types 1 and 2 are as follows:

1. Type 1 has a distributed MAC with QoS provisions, whereas type 2 has a centralized scheduled MAC.
2. Type 1 is based on GMSK (gaussian minimum shift keying) whereas type 2 is based on OFDM.

The highlights of HiperLAN/2 are as follows:

1. QoS, to build multiservice networks, using connection-oriented service.
2. Strong security: it uses key negotiation, authentication (X.509) and encryption using (digital encryption standard (DES) or triple DES.
3. Automatic handoff when moving between LAN and WAN.
4. Increased data rates of up to 54 Mbps (using OFDM) and throughput.
5. Ease of use, deployment and maintenance.
6. Affordability and scalability.
7. Built-in interoperation with Ethernet, ATM, PPP and Internet.
   - The mobile station communicates with one access point (AP) at a time over an air interface.
   - The AP is basically a radio base station that covers an area of about 30 to 150 m.
   - Control is centralized to the AP, which informs the mobile station to transmit data using TDD (time division duplexing) and dynamic TDMA.
   - Selective repeat automatic request repeat (ARQ) is used to increase reliability over the radio link.
   - A mobile station may at any time request the AP and enter into a negotiated low-power state for a sleep period.

3.6 The IEEE 802.16 WiMAX standard

WiMAX, which stands for worldwide interoperability for microwave access, has been designed by the WiMAX Forum and standardized as IEEE 802.16. As mentioned earlier, it was originally conceived as a last-mile technology and also called WLL. However, in the case of WiMAX, the 'mile' is typically around 1 to 6 miles, but going up to a maximum of 30 miles. This classifies WiMAX as a MAN standard. Since it was designed to fulfill the last-mile role, WiMAX today does not incorporate mobility, and is characterized as 'fixed' wireless. However, work to add mobility into it, in the form of the standard IEEE 802.16e, is already nearing completion.

It is to be noted that WiMAX's client systems, called **subscriber stations**, are not end-user computing devices, but systems that multiplex the communication of all computing devices in a particular building. The data rates provided by WiMAX can go up to 150 Mbps to a single client.

WiMAX is heavily influenced by the ISO OSI Reference Model, including the terminology, sublayers, service primitives, etc., and is thus quite complicated. It defines several physical layer protocols. The original WiMAX physical layer protocol was designed for frequencies in the 10- to 66 GHz

range, making the communication line-of-sight (LOS). It was later extended to near LOS and non-LOS situations, by adding several other physical layer protocols using frequencies below 11 GHz.

A WiMAX base station uses multiple antennas pointed in different directions. The area covered by one antenna's signal is called a *sector*. The upstream and downstream channels are shared among the many subscriber stations in a given sector and also among the many WiMAX *connections* that each subscriber can have with the base station. Unlike 802.11 and the Ethernet, WiMAX is connection-oriented. This enables WiMAX to offer a variety of QoS guarantees with respect to latency and jitter. Thus it supports bursty data and high-quality telephone and high-volume multimedia.

Sharing of upstream and downstream channels is done by dividing them into equal-sized slots. A WiMAX frame generally takes up multiple slots, different frames taking a different number of slots. The downstream channel (from base station to subscribers) is easy to subdivide into connections since only the base station sends on that channel. The station simply sends frames, one after the other. Each subscriber station in the sector receives all the frames, but ignores those not addressed to one of its connections.

In the upstream direction, the connection is handled depending on its QoS parameters. Some connections get slots at a fixed rate, and some are polled to determine how many slots they need at any time, while others request slots whenever needed. Contention in the last category is dealt with by placing requests in a limited number of upstream slots set aside for this purpose. The exponential backoff algorithm is used to minimize the chance of a collision.

It is to be emphasized here that although both IEEE 802.11 and 802.16 operate in similar environments having been designed to provide high-bandwidth wireless communications, they differ in many ways. The main difference is that while 802.16 provides service to buildings which are 'fixed', 802.11 deals with 'mobile' notebooks.

3.7 Comparison of wireless technologies

Table 3.3 gives a comparison of the four wireless technologies studied above.

Table 3.3 Comparison of Wireless T echnologies

Parameter	IrDA	Bluetooth	IEEE 802.11	WiMAX
Technology	Infrared	radio radio		microwave
	—	FHSS	FHSS, OFDM, DSSS	FDD, TDD
Frequency range	—	2.4 GHz	2.4 GHz, 5 GHz	2–66 GHz
Speed	115.2 Kbps	1–4 Mbps	up to 54 Mbps	up to 70 Mbps
Range	1 m	10 m	300 m in air	10 km in air
Media access	—	master-slave	distributed/ central CSMA/ CA, MACAW	OFDMA
Frame size	—	350 bytes	2344 bytes	5–20 ms
Typical use	TV remote	peripheral-to-laptop link	PDA-to-BS link	building-to-tower link

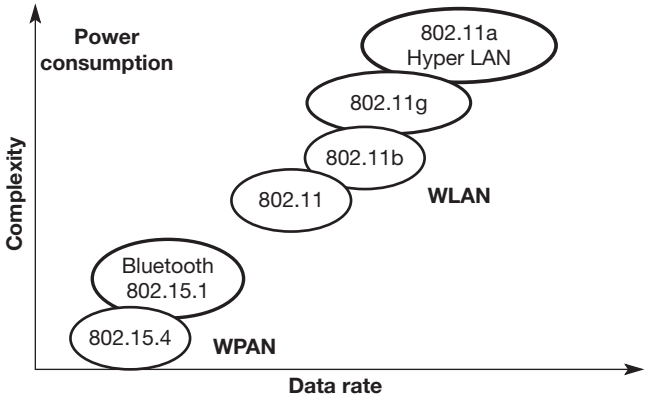


Figure 3.16 The Scope of V arious WLAN and WP AN Standards

The wireless technologies discussed above can be interworked to create heterogeneous wireless networks. For example, WPANs and WLANs will enable an extension of 3G cellular systems into devices without direct cellular access. Devices connected in a WPAN can utilize a combination of 3G access and WLAN by selecting whichever access is best at a given time. Thus, 3G, WLAN and WPAN technologies are not competitive, but give a choice to the user to select the best connectivity for his purpose.

Figure 3.16 shows the scope of the various standards discussed in this chapter.

3.8 Summary

WLAN standards are now becoming common but have their own problems, like the hidden terminal and exposed terminal problems. Since CSMA does not work, solutions like MACAW are used with it to give CSMA/CA.

IEEE 802.11 has emerged as the dominant WLAN technology. Its physical layer allows five different transmission modes, including infrared and various spread-spectrum schemes. It provides nine types of services, for intracell and intercell management.

To provide for last-mile connectivity, a new standard called IEEE 802.16 or WiMAX, also known as broadband wireless, has been introduced. This is used to connect subscribers in a building wirelessly, through a base station, in a kind of MAN system.

Bluetooth is also a wireless system, but with limited range and is currently used for connecting headsets and other peripherals to computers, wirelessly. It has many similarities with 802.11 but its applications are different. It has an elaborate protocol stack, which allows for both text and real-time data transmission.

The Infrared Data Association (IrDA) has given a standard for infrared systems, which can be used in point-and-shoot applications, unlike Bluetooth, which is omnidirectional. However, infrared systems have many limitations which restrict their use in mobile computing.

The HiperLAN is a European standard WLAN, and is available in two versions, with version 2 providing higher data rates and better QoS.

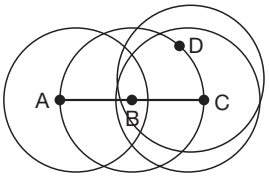
In the next chapter we shall look at logical mobility and migrating processes.

Problems

1. Discuss the IEEE 802.11 frame structure, clearly indicating its various fields and their uses.
2. Compare and contrast IEEE 802.11 and Bluetooth technologies.
3. Explain clearly why the wired Ethernet cannot be used for WLANs.
4. Discuss the hidden terminal and exposed terminal problems and their effect on WLANs.
5. Discuss the advantages and limitations of Bluetooth as a wireless standard.
6. Suppose that an 11 Mbps 802.11b LAN is transmitting 64-byte frames back-to-back over a radio channel with a bit error rate of 10^{-7}. How many frames per second will be damaged on average?
7. Consider Figure 3.5. Which of the last two stations is closest to A and why?
8. A Bluetooth device can be in two piconets at the same time. Is there any reason why one device cannot be master in both of them at the same time?
9. Several physical layer protocols are shown in Figure 3.4. Which of these is closest to the Bluetooth physical layer protocol? What is the biggest difference between the two?
10. Bluetooth supports two types of links between a master and a slave. What are they and what is each one used for?
11. What are the main differences between HiperLAN v1 and v2?
12. Discuss how the HiperLANs are different from the IEEE 802.11 standard.
13. Would you say the HiperLAN is closer to Bluetooth or to 802.11? Give reasons for your answer.

Multiple-choice questions

1. Wireless LANs will operate in which one of the following configurations?
   (a) With base stations only
   (b) Without base stations only
   (c) Either a or b
   (d) None of the above
2. Which one of the following is the main standard for WLANs?
   (a) IEEE 802.15
   (b) IEEE 802.3
   (c) IEEE 802.11
   (d) IEEE 802.16



3. In the above figure, which one of the following gives the hidden station problem?
   (a) When both B and C wants to transmit to station D
   (b) When only B wants to transmit to A

(c) When C wants to transmit to both B and D
(d) When A and C wants to transmit to B

4. In the same figure, which one of the following gives the exposed station problem?
(a) A and C want to transmit to B
(b) A is sending to some other station E and B wants to send to C
(c) Only B wants to transmit to A
(d) Both B and C wants to transmit to station D

5. Which one of the following is to be considered in WLANs?
(a) Multipath fading
(b) Handoff
(c) Mobility awareness
(d) All the above

6. NAV packets are transmitted by which one of the following stations in DCF?
(a) The station that wants to transmit data
(b) The station willing to receive data
(c) All the stations other than sender and receiver (virtually)
(d) There is nothing called NAV in DCF

7. In the 802.11 frame structure, subtype field is used for which one of the following?
(a) To differentiate control and data frames
(b) Used by the base station to send receiver into sleep
(c) To tell receiver this is the last frame
(d) None of the above

8. Which one of the following is the main standard for Bluetooth?
(a) IEEE 802.15
(b) IEEE 802.3
(c) IEEE 802.11
(d) IEEE 802.16

9. Which one of the following gives the number of nodes in a Scatternet?
(a) One
(b) None
(c) At least two
(d) None of the above

10. The Bluetooth baseband layer is responsible for which one of the following?
(a) Defining transmission characteristics
(b) Communicating with its peer in the target device
(c) Sending authentication information
(d) Establishing links with other Bluetooth devices

Further reading

A.S. Tanenbaum (2003), *Computer Networks*, 4th ed. (New Delhi: Pearson Education)

B.P. Crow, I. Widjaja, J.G. Kim and P.T. Sakai (1997), 'IEEE 802.11 Wireless Local Area Networks', *IEEE Communications Magazine*, 35 (September): 116–126.

C. Severance (1999), 'IEEE 802.11: Wireless Is Coming Home', *Computer*, 32 (November): 126–127.

D.P. Agarwal and Q. Zeng (2003), *Introduction to Wireless and Mobile Systems* (Singapore: Thompson Asia).

ETSI, 'High-performance radio local area network (HIPER-LAN) type 1; functional specification', http://weapp.etsi.org/pda/home.asp?wkiid=6956.

J. Haartsen (2000), 'The Bluetooth Radio System', *IEEE Personal Communications,* 7 (February): 82–85.

L.L. Peterson and B.S. Davie (2007), *Computer Networks—A Systems Approach*, 4th ed. (New Delhi: Elsevier).

M. Johnson, 'HiperLAN/2—the broadband radio transmission technology operating in the 5 GHz frequency band', www.hiperlan2.com/site/specific/whitepaper.exe. (accessed June 2006)

S. Kapp (2002), '802.11: Leaving the Wire Behind', *IEEE Internet Computing*, 6 (January–February): 82–85.

T. Karygiannis and L. Owens (2002), 'Wireless Network Security, 802.11, Bluetooth & Handheld Devices', NIST Special Publication No. 800–48.

The Bluetooth Special Interest Group, 'Baseband Specifications', www.bluetooth.com. (accessed May 2006)

The Infrared Data Association Web site, www.irda.org.

U. Hausmann, L. Merck, M.S. Nicklous, and T. Stober (2003), *Principles of Mobile Computing*, 2nd ed. (New Delhi: Springer).

Logical Mobility I— Migrating Processes

4

The concept of logical mobility was introduced in Chapter 1, where it was mentioned that abstract entities like software processes and agents may be allowed to move or migrate from their parent locations to other, maybe remote, locations for various reasons. In this chapter, we shall discuss logical mobility, as represented by migrating processes. Mobile agents are the subject of Chapter 7.

A **process** is a key concept in operating systems (OS). It is an active entity and is defined as a program in execution. In a single-processor system, a process continues to execute on the only processor in the system, but in a multiprocessing environment, like a cluster, many processes execute simultaneously. Frequently, a process executing on one processor may be required to migrate to another local processor, normally for purposes of load balancing or reliability.

In a mobile system or network, a process is frequently required to migrate to a remote location for continuation of easy and fast access. This is termed 'logical mobility'. There are many design issues involved in process migration. Although most programming languages nowadays provide the programmer with constructs and primitives to ease process migration, it is necessary for a student of mobile computing to know the basics of process migration.

In this chapter, we study precisely these design issues and understand the nitty-gritty of processes, their constituents and how process migration takes place. We also discuss the advantages and applications of process migration. We conclude with a discussion on several available alternatives to process migration.

4.1 What is a process?

A process consists of data, a stack, register contents, and the state specific to the underlying OS, such as parameters related to process, memory, device and file management. A process can have one or more threads of control. Threads, also called light-weight processes, consist of their own stack and register contents, but share a process's address space and some of the OS-specific states, such as signals.

The **task** concept was introduced as a generalization of the process concept, whereby a process is decoupled into a task and a number of threads. A traditional process is represented by a task with one thread of control.

A process can be categorized as a system process or an application process. Figure 4.1 shows the system process layout of a typical distributed operating system (DOS).

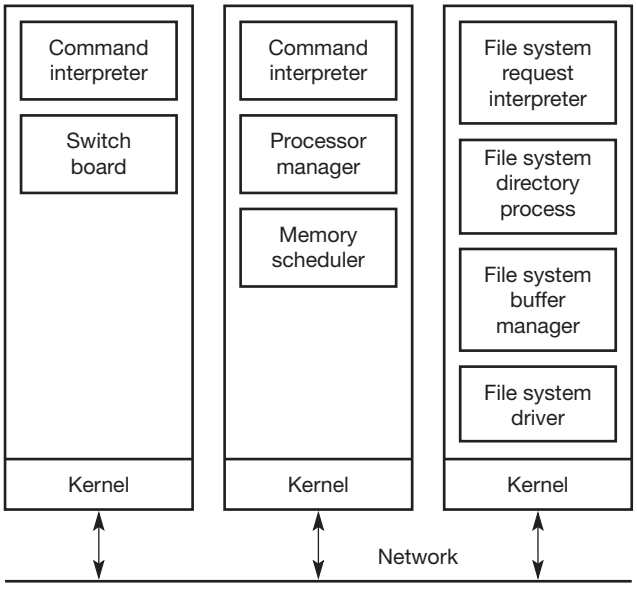


Figure 4.1 System Process La yout in a DOS

4.2 Process migration

Consider the case of a process which has to be moved from a source node to a destination node. Process migration consists of extracting the state of the process on the source node, transferring it to the destination node, where a new instance of the process is created, and updating the connections with other processes on communicating nodes. The transferred state includes the process's address space, execution point (register contents), communication state (e.g. open files and message channels) and other OS-dependent states.

Process migration represents transferring a process between two machines during execution of its threads. During migration, two instances of the migrating process exist—the **source instance** is the original process, and the **destination instance** is the new process created on the destination node.

After migration, the destination instance becomes a **migrated process**. In systems with a home node, a process that is running on other machines may be called a **remote** process (from the perspective of the home node) or a **foreign** process. Figure 4.2 gives a high-level view of process migration.

4.3 The steps in process migration

Once the decision has been made to migrate a process, the following steps are to be followed. These steps are shown pictorially in Figure 4.3 (1 to 8).

1. **A migration request is issued to a remote node.** After negotiation, migration has been accepted.
2. **A process is detached from its source node** by suspending its execution, declaring it to be in a migrating state, and temporarily redirecting communication, as described in the following step.

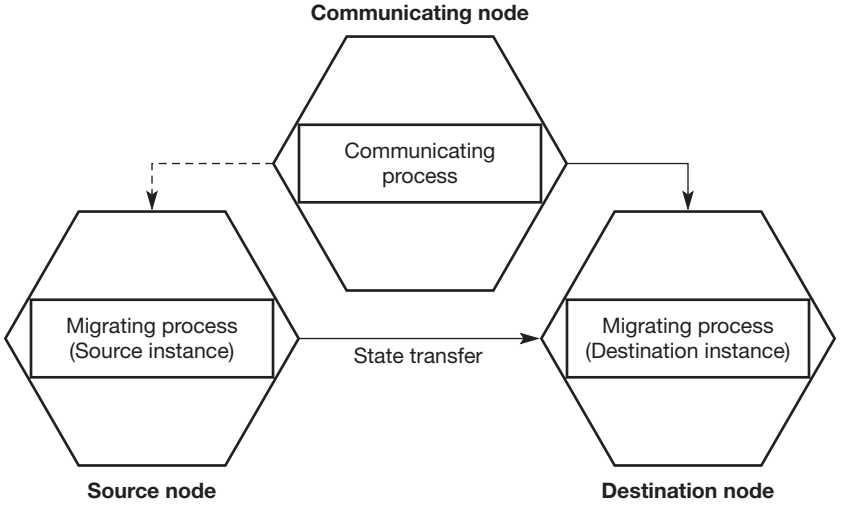
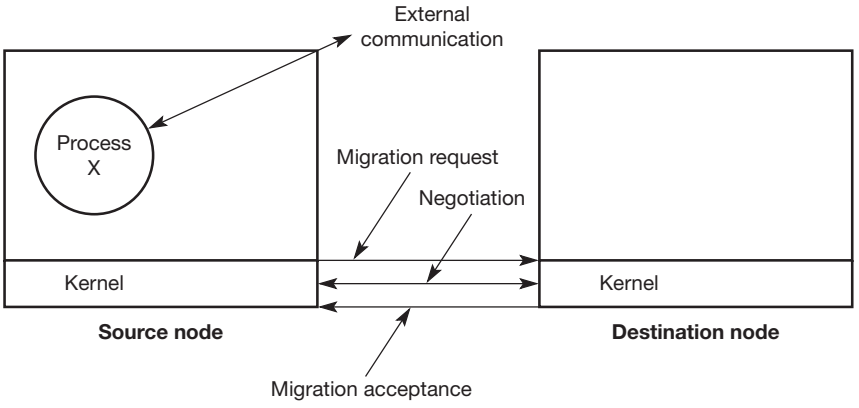
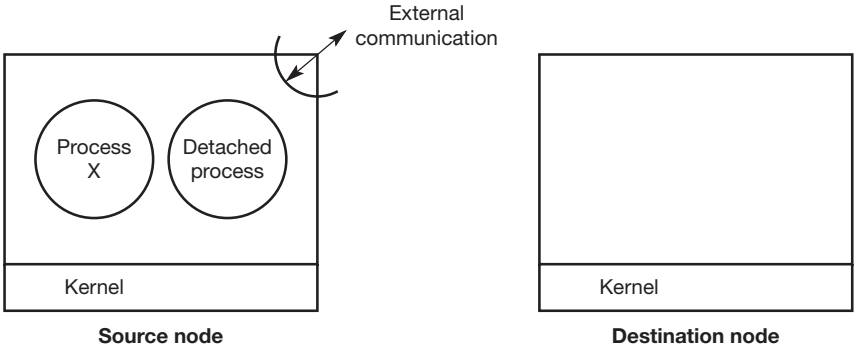


Figure 4.2 High-level View of Migration

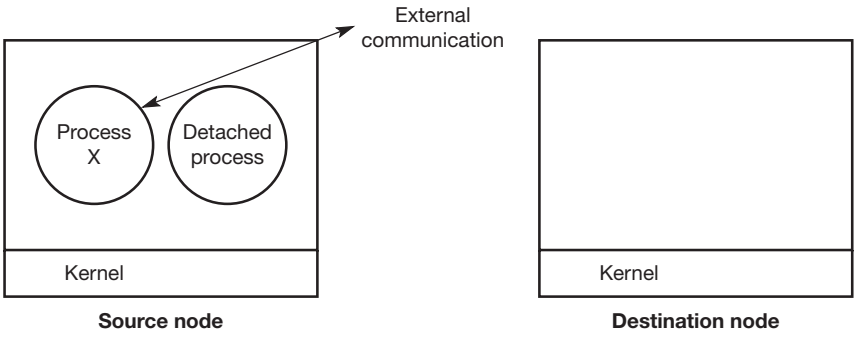
3. **Communication is temporarily redirected** by queuing up arriving messages directed to the migrated process. These are delivered to the process after migration. This step continues in parallel with steps 4, 5 and 6, as long as there are additional incoming messages. Once the communication channels are enabled after migration (as a result of step 7), the migrated process is known to the external world.
4. **The process state is extracted.** The state includes its memory contents, processor state (register contents), communication state (e.g. opened files and message channels) and relevant kernel context. The communication state and kernel context are OS-dependent. Some of the local OS internal state is not transferable. The process state is retained on the source node until the end of migration. In some systems, it remains there even after migration completes. Processor dependencies, such as register and stack contents, have to be eliminated in the case of heterogeneous migration.
5. **A destination process instance is created** into which the transferred state will be imported. It is not activated until a sufficient amount of state has been transferred from the source process instance. After that, the destination instance will be promoted into a regular process.
6. **The state is transferred and imported into a new instance** on the remote node. All of the state need not be transferred at once. Some of it could be lazily brought over after migration is completed.
7. **Some means of forwarding references** to the migrated process must be maintained. This is required in order to communicate with the process or to control it. It can be achieved by registering the current location at the **home** node (e.g. in Sprite), by searching for the migrated process (e.g. in the V Kernel, at the communication protocol level) or by forwarding messages across all visited nodes (e.g. in Charlotte). This step also enables migrated communication channels at the destination, and it ends step 3, as communication is permanently redirected.
8. **The new instance is resumed** when sufficient state has been transferred and imported. With this step, process migration completes. Once all of the state has been transferred from the original instance, it may be deleted on the source node.



1. A migration request is issued to a remote node

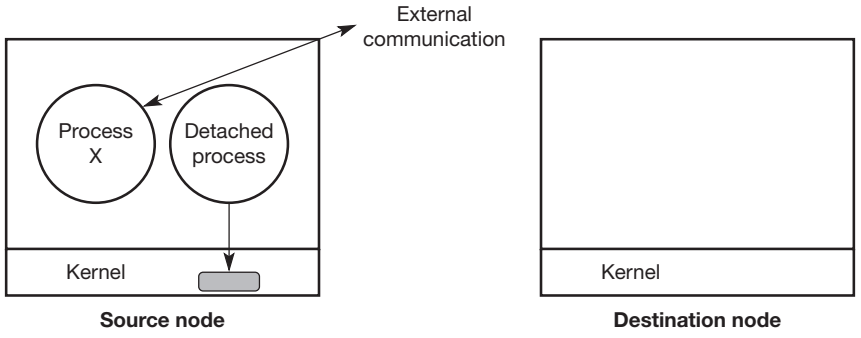


2. A process is detached from its source node

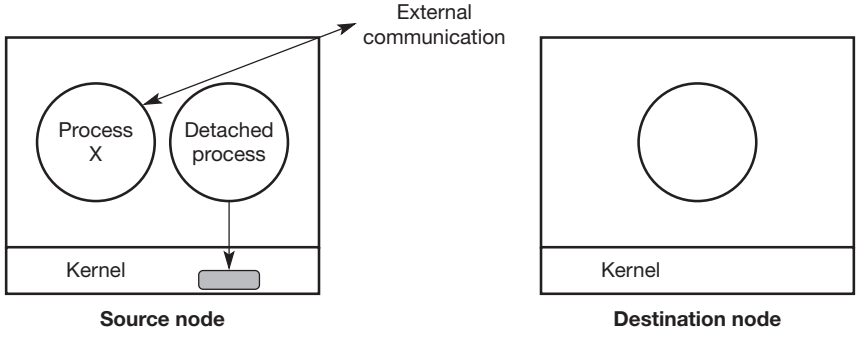


3. Communication is redirected

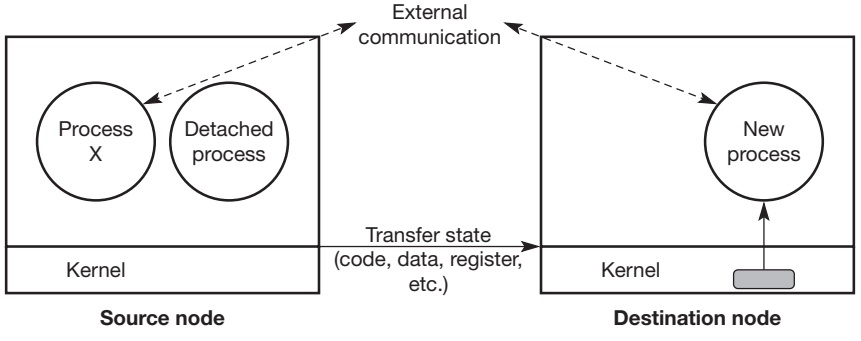
Figure 4.3(1–8) Steps in Mo ving a Process (*Continued*)



4. The process state is extracted

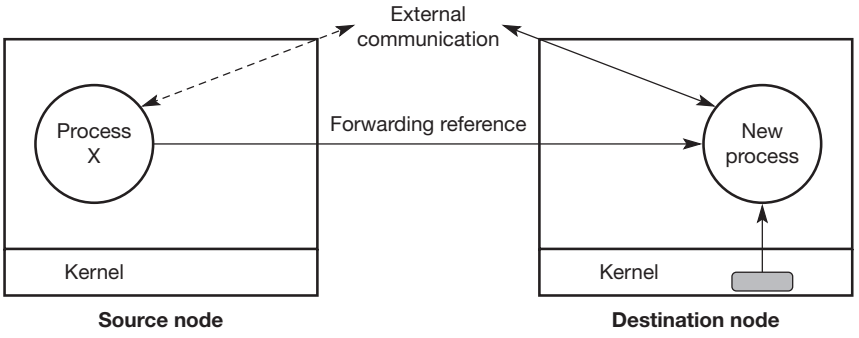


5. A destination process instance is created

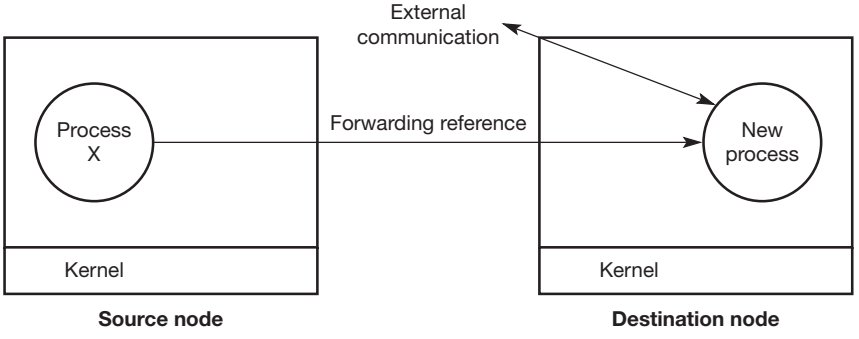


6. State is transferred and imported into a new instance

Figure 4.3(1–8) (*Continued*)



7. Some means of forwarding references



8. The new instance is resumed

Figure 4.3(1–8) (*Continued*)

4.4 The advantages of process migration

The advantages of process migration are many and varied, especially so nowadays, with the proliferation in distributed and networking systems. These are discussed below.

1. **Dynamic load distribution** is possible in multiprocessing systems to balance the load on the different processors/nodes, by migrating processes from overloaded nodes to less loaded ones.
2. **Fault resilience** can be achieved in such systems, by migrating processes from nodes that may have experienced a partial failure or are likely to fail completely in the immediate future.
3. **Improved system administration** can be achieved by migrating processes from the nodes that are about to be shut down or otherwise made unavailable.
4. **Data access locality** is possible to provide in wireless or fixed mobile systems, by migrating processes closer to the source of some data as the user moves from one network or cell to another.
5. **Resource sharing** is possible on a grid, by migration of a process to a specific node that is equipped with a special hardware device, large amount of free memory or some other unique resource.
6. **Mobile computing** also increases the demand for migration. Users may want to migrate running applications from a host to their mobile computer as they connect to a network at their current location or back again when they disconnect.

4.5 Applications of process migration

Based on the above advantages, process migration has a number of applications, as can be seen from the list given below.

1. **Parallelizable applications** can be started on certain nodes, and then migrated at the application level or by a systemwide migration facility in response to factors like load balancing. The parallel virtual machine (PVM) is an example of application-level support for parallel invocation and interprocess communication. Migratory PVM (MPVM) extends PVM to allow instances of a parallel application to migrate among nodes.
2. **Long-running applications**, which can run for days or even weeks, can suffer various interruptions. For example, there may be partial node failures or administrative shutdowns. Process migration can relocate these applications transparently to prevent interruption.
3. **Generic multiuser workloads** can benefit greatly from process migration. For example, in an undergraduate computer laboratory, as users come and go, the load on individual nodes varies widely. Dynamic process migration can automatically spread processes across all nodes.
4. A **pre-emptable application**, such as that used in resource sharing, fault resilience, exploitation of resource locality, accessing more processing power, system administration or mobile computing, can either migrate itself or be migrated by another authority. This type of application is most common in various systems such as MOSIX, Sprite, Mach and LSF (load sharing facility).
5. **Migration-aware applications** are those that have been coded to explicitly take advantage of process migration. These can be automatically redistributed if the load becomes uneven on different nodes. This can happen if processes are dynamically created or if there are many more processes than nodes.
6. **Network and mobile computing applications** are the most recent examples of the use of migration, in the form of mobile agents and mobile objects.

4.6 Alternatives to process migration

Process migration has many advantages and wide applications as seen above. But there are other options available to users. We discuss these alternatives and their advantages and disadvantages.

1. **Remote execution** is the most frequently used alternative to process migration. Remote execution can be as simple as the invocation of some code on a remote node, or it can involve transferring the code to the remote node and inheriting some of the process environment, such as variables and opened files. Remote execution is usually faster than migration because it does not incur the cost of transferring a large process state (such as the address space, which is created anew in the case of remote execution). For small address spaces, the costs for remote execution and migration can be similar.

   Allowing a process to run on the source node for some period of time has some advantages. Thus short-lived processes that are not worth migrating are naturally filtered out. Also, the longer a process runs, the more information about its behaviour is available, such as whether and with whom it communicates. Based on this additional information, scheduling policies can make more appropriate decisions. However, remote execution allows creation of the remote instances only at the time of process creation, and process migration allows moving the process at an arbitrary time.

2. **Cloning** of a process is useful in cases where the child process inherits state from the parent process. Cloning is typically achieved using a **remote fork** mechanism. A remote fork, followed by the termination of the parent, resembles process migration. The complexity of cloning is similar to migration, because the same amount of the process state is inherited (e.g. open files and address space). In migration, the parent is terminated. In cloning, both parent and child may continue to access the same state, introducing a distributed-shared state, which is complex and costly to maintain. Many systems use remote forking.
3. **Mobile agents** are becoming increasingly popular. The mobility of agents on the Web emphasizes safety and security issues more than complexity, performance, transparency and heterogeneity. Mobile agents are implemented on top of safe languages, such as Java, Telescript and Tcl/Tk. We discuss mobile agents later in the book.

   Compared to process migration, mobile agents are less complex to implement, because they do not have to support OS semantics. Performance requirements are also different, owing to the wide-area network communication cost, which is the dominant factor. Heterogeneity is taken care of at the language level. The early results and the wide interest in the area of mobile agents indicate that they have a bright future.
4. **Object migration at the middleware level** is also possible. Because of the increasing costs of OS development and the lack of standard solutions for distributed systems and heterogeneity, middleware-level solutions have generated more interest.

Distributed objects are supported in middleware systems such as distributed computing environment (DCE) and common object request broker architecture (CORBA), as discussed in the appendix of this book. One reason is that the early heterogeneity of these systems did not adequately support mobility. However, some systems support mobility at the middleware level, such as DC++ and the Object Management Group (OMG) mobile agent system interoperability facility (MASIF) specification for mobile agents, based on OMG CORBA, which we discuss in the appendix.

4.7 Summary

A process is a key concept in operating systems. Process migration consists of extracting the state of a process at the source node and transferring it to the destination node. The DEMOS multiprocessor used the first instance of process migration and showed the way forward to implement logical mobility for mobile systems. There are many advantages and applications of migrating processes, especially in mobile computing. Alternatives to process migration include mobile agents and process cloning.

In Chapter 5, we shall look at physical mobility as represented by mobile handheld systems.

Problems

1. What are the components of a process that need to be moved when a process is migrated? Discuss.
2. Implement process migration using a suitable C/C++ program, using the various steps discussed in this chapter.

3. Repeat Problem 2 above in Java.
4. Implement any sorting algorithm in Java and receive 100 random input data from an input file and sort them in ascending/descending order. During sorting, the following cases are required to be considered:
   (a) When sorting index is at 10, capture the contents of parameters that are required to start the execution from the stopped execution and save them in a file/vector and stop the execution.
   (b) Start execution once again and take the starting position as mentioned in (a).
   (c) Repeat steps (a) and (b) until data are sorted.
5. As mentioned in Problem 4, transfer the captured states and code of the program across the network and resume the execution. After completion, receive the result back at the client.
6. Write a Java program that counts the number of threads running in a program. Is it feasible to save the threads and shut down the program and restart the threads where they are before closing the execution? Why?
7. Write a Java program to generate 10 threads and divide them into three groups and then start them in the assigned group.

Multiple-choice questions

1. Which one of the following is called a 'process'?
   (a) Program in execution
   (b) Code
   (c) State specific to underlying operating system
   (d) Object
2. Which one of the following is not a component of the system process's kernel in a distributed operating system?
   (a) Command interpreter
   (b) Process manager
   (c) Memory scheduler
   (d) File manager
3. Which one of the following is an instance of a migrated process?
   (a) Destination instance
   (b) Remote instance
   (c) Correspondent instance
   (d) Source instance
4. Which one of the following is not a step in process migration?
   (a) Process is detached from its source node
   (b) State is transferred and imported into a new instance on the remote node
   (c) Migration request is issued to a source node
   (d) Destination process instance is created
5. Which one of the following is an advantage of process migration?
   (a) Fault resilience
   (b) Data sharing

(c) Memory sharing
(d) Improved resource administration

6. Which one of following is not an application of process migration?
   (a) Parallelizable application
   (b) Resource sharing application
   (c) Migration-aware application
   (d) Network and mobile computing application
7. Which one of the following is an alternative to process migration?
   (a) Generic multiuser workloads
   (b) Migration-aware applications
   (c) Remote execution
   (d) Object migration at the top level
8. Which one of the following does a thread consist of?
   (a) Process's address space
   (b) Own address space
   (c) Own data space
   (d) Own signals
9. Which one of the following is meant by a process?
   (a) Program in waiting state
   (b) Process is passive entity
   (c) Program in execution
   (d) Program in suspended state
10. Which one of the following is not included in the transferred state of a migrated process?
    (a) Process's address space
    (b) Signals
    (c) Execution point
    (d) Communication state

Further reading

F. Douglis and J. Ousterhout (1989), 'Process Migration in Sprite: A Status Report', *IEEE TCOS Newsletter*, 3(1): 8–10.

J.M. Smith (2001), 'A Survey of Progress Migration Mechanisms', Technical Report TRCUCS-324-88, CU, New York.

M.L. Powell and B.P. Miller (1983), 'Process Migration in DEMOS/MP', in Proceedings of the 9th ACM Symposium on Operating System Principles, pp. 110–119.

Physical Mobility 5

Recent advances in very-large-scale integration (VLSI) technology have enabled portable computers to be built and equipped with wireless interfaces. This facility allows networked communication to be possible even while being mobile and has given rise to the new computing paradigm called mobile computing. Technological advances in wireless networking have enhanced the utility of carrying a computing device and allowing versatile communication and fast notification of important events. Mobile computing is characterized by true physical mobility, which permits continuous access to the services and resources of land-based networks. It therefore gives much more flexibility than cellular phones or pagers.

The combination of networking and physical mobility gives rise to new applications and services, some of which are as follows:

1. Software for supporting meetings called on-the-fly
2. Personalized electronic bulletin boards for different people
3. Automatic and self-adjusting lighting and heating for homes and offices
4. Software to guide users in unknown surroundings

In this chapter, we shall dwell upon the problems associated with providing continuous Internet access to mobile devices. We shall study some existing protocols and see how they are inadequate in supporting mobility. A few modified and enhanced protocols that support mobility will be discussed, and finally we shall do a case study of a system that provides for such mobility.

5.1 The requirements for ph ysical mobility

There are three important issues related to physical mobility. These are discussed in detail below.

5.1.1 Wireless communication

Mobile computers require both wireless and wired network access. However, as discussed in Chapter 2, wireless communication is more difficult to achieve than wired communication, because the environment interacts with the signal, blocking its path and introducing noise and echoes. Hence quality suffers, resulting in lower bandwidth, higher error rates and more frequent disconnections.

Furthermore, communication delays are more frequent due to retransmissions, timeouts and error-control processing. The wireless connection can also be lost or degraded by mobility if the user enters high-interference areas. Since the number of devices in a cell varies dynamically due

to a large concentration of mobile users, network capacity can become overloaded at conventions and public events.

Disconnection is a major problem for mobile computing. If the network frequently disconnects, clients in a distributed file systems environment may block waiting for servers. Hence it becomes necessary to make computing more autonomous and asynchronous in operation. The asynchronous Remote Procedure Call paradigm can be used to hide round-trip latency and short disconnections. It is also possible to allow pre-fetching and lazy write-back techniques to decouple communication from computing, to mask network failures. This has been done in the CODA file system, which uses on-board caches to hold whole files rather than blocks, and which will be discussed in detail later in this chapter.

We have seen in Chapter 2 that wireless networks deliver much lower bandwidth than wired ones. For example, the figure is 1 Mbps for infrared communication, 2 Mbps for radio communication and only 9–14 Kbps for cellular telephony. On the other hand, Ethernet provides speeds of 10 Mbps, fibre distributed data interface (FDDI) 100 Mbps and asynchronous transfer mode (ATM) 155 Mbps.

Furthermore, network bandwidth is divided among users sharing a cell; hence, bandwidth usable per user is important, not raw transmission bandwidth.

To improve network capacity, one can reduce transmission ranges to fit more cells. Certain software techniques also help. For example, one can use compression and combining of short requests into large ones, to improve throughput. Also, pre-fetching and lazy write-back techniques can be used, together with scheduling communication intelligently.

Network quality also changes when moving from one wireless network to another, giving rise to heterogeneous networks. For example, a meeting room has better wireless facilities than a corridor. Secondly, multiple transceivers may be on different frequencies, and protocols on networks may be different. Finally, there may be a need to switch interfaces while moving outside from inside since infrared does not work outside.

We have also mentioned earlier that because of easy and open connectivity on wireless links, and because users are allowed to cross security domains, more complex security systems are needed.

There is a need for using encryption in software or hardware and managing secret encryption keys using automated software like Kerberos. But for authentication of mobile users when roaming, Kerberos needs to be modified to allow authorization control and accounting.

5.1.2 Mobility

The ability to change locations while connected to a network increases the volatility of information. Data considered static for a stationary computer becomes dynamic for mobile computing; for example, a stationary computer can be configured statically to use the nearest server, but a mobile computer needs some mechanism to determine which server to use.

The main problems introduced by mobility are as follows:

1. **Address migration:** As people move, their mobile computers will have to use different network service access points (NSAPs). Thus, dynamically changing addresses are needed. But in the Internet protocol (IP) protocol, host names are bound with network addresses, and moving to a new location means getting a new name. To communicate with a mobile computer, messages must be sent to its most recent address. This is possible with mobile IP, as we shall shortly see.

2. **Location-dependent information:** The local name server, the available printer, the time zone and real-world information, like 'where is the nearest library', are dynamically changing data that must be made available when the computer moves. The challenge is thus to provide all these without violating privacy.
3. **Migrating locality:** Mobile computing gives rise to a new kind of locality that migrates as users move. For example, a server that was found after great effort to be the nearest one at a particular location may not be so after further movement. Also, physical distance between two points does not necessarily reflect network distance. Hence, communication locality becomes more important than load balancing.

5.1.3 Portability

Desktop computers are not designed to be carried, so their design is liberal in the use of space, power, cabling and heat dissipation. On the other hand, handheld devices need to have wristwatch properties—they need to be small, light weight and water resistant and have long battery life. Concessions can be made to the user in the above characteristics, to increase functionality, but only in a limited manner, to keep it mobile. Any added, specialized hardware device, like a hardware encryption chip, should justify its consumption of power and space.

Design pressures caused by portability constraints are as follows:

1. **Low power:** Batteries are the largest single source of weight. But reducing battery size compromises its use, because it leads to frequent recharging, the need for spare batteries, or results in less use of the mobile computer.

   It has been found that $P \propto CV^2 F$,

   Where
   P is the power consumption of the device
   C is the interline capacitance of the VLSI processor chip in the device,
   V is the voltage swing and
   F is clock frequency of the processor chip.

   To reduce P, we need to reduce all three parameters. Thus,

   To reduce C, greater levels of VLSI are to be used.
   To reduce V, the chip has to be redesigned to operate at lower voltages and
   To reduce F, we must have smaller frequency chips.

   Personal digital assistant (PDA) products use all three methods to conserve power. Also, efficient operation can conserve power. For example, one must turn down screen lighting when not in use. Table 5.1 shows typical power consumption values of the components of a portable computer and its accessories, and Table 5.2 shows the power consumption breakdown, by subsystems, of the portable Compaq LTE 386 computer.
2. **Risks to data:** Portability increases the risk of physical damage, unauthorized access, loss and theft and can lead to breach in privacy or total data loss. So it is necessary to minimize the essential data on the device, encrypt the data on disks and removable memory cards or use replicated file systems and make backup copies.
3. **Small user interfaces:** Current windowing techniques are not feasible when user area is small. One cannot have too many windows open, or too many icons (buttons). Also, window

Table 5.1 Power Consumption of Components of a P ortable Computer and Accessories

Device	in watts
Base system (2 MB, 25 MHz cpu)	3.650
(2 MB, 10 MHz cpu)	3.150
(2 MB, 5 MHz cpu)	2.800
Screen backlight	1.425
Hard drive motor	1.100
Maths coprocessor	0.650
Floppy drive	0.500
External KB	0.490
LCD screen	0.315
Additional memor y (per MB)	0.050
Serial por t	0.035
Parallel por t	0.030
Accessories	
PCMCIA modem 14.4 Kbps	1.365
9.6 Kbps	0.625
2.4 Kbps	0.565
Infrared networ k	0.250
Cellular telephone (active)	5.400
(passive)	0.300
PCMCIA hard drive	0.7–3.0

title bars and borders are difficult to operate with pointing devices. So it is necessary to use handwriting or gestures or voice recognition analog devices.

In fact, speech production and recognition systems are ideal user interfaces for mobile computers, as they allow hands-free and eye-free operation, but they may compromise on disturbance, privacy and noisy environments.

Table 5.2 Power Consumption of Subsystems of a P ortable Computer (Compaq L TE 386)

System	% power
Displa y edge light	35%
CPU/memor y	31%
Hard disk	10%
Floppy disc	8%
Displa y	5%
Keyboard	1%

The mouse cannot be used with handheld devices, so pens are the standard input devices for PDAs. But problems like parallax and obscuring of the screen area by the hand may have to be taken care of.

4. **Small storage capacity:** Disks cannot be used in PDAs as they consume more power and are susceptible to volatility. Solutions include compressing file systems, accessing remote storage over the network, sharing code libraries and compressing virtual memory pages. A novel approach to reduce program size is to interpret script languages instead of executing compiled object code.

5.2 Overview of IPv4 and IPv6

Before we discuss new mobility-aware protocols, we need to review the existing IPs, IPv4 and IPv6, because they have certain limitations where mobile systems are concerned. These limitations have been addressed and removed in the new protocols which have been developed to support mobile systems.

5.2.1 IPv4

IP version 4, or IP as it is better known, is the glue that holds the Internet together. Its basic function is to deliver data from a source to a destination, independent of the physical location of the two. In this book, we shall not discuss the details of IP. Instead, we shall concentrate on its addressing mechanism only, since this is where the real problem lies for mobility.

IP identifies each host on a network uniquely, using an IP address that designates its physical attachment to the Internet. IPv4 addresses are 32-bit long integers and are represented in a dotted decimal format (e.g. 172.16.19.11) for ease of use. In CIDR (classless inter-domain routing) addressing, every IPv4 address contains the network number and the host number.

For example, if we consider a machine with IP address 172.16.19.11, then 172.16 gives the network number (decimal 11280) and 19.11 is the host number (decimal 4175).

All routers on the Internet have routing tables telling them which line to use to get to network 172.16. Whenever a packet comes in with destination IP address of the form 172.16.xxx.yyy, it goes out on that line.

Every IP packet consists of an IP header and an IP payload. The header contains the IP addresses of the sending node and the receiving node along with some other information. To correctly deliver these packets, IP executes two major steps: packet routing and packet forwarding. Packet routing involves use of protocols like border gateway protocol (BGP), routing information protocol (RIP) and open shortest path first (OSPF) to decide the route that each packet has to travel.

The route is decided using a routing table having the 2-tuple pairs

(destination address, next hop)

at each router. Destination addresses are compared with a pair contained in the routing table. Packet forwarding involves use of protocols like address resolution protocol (ARP), proxy ARP, etc. to deliver the packet to the end node once it has arrived at the destination network. This is typically done to discover the hardware address of the host corresponding to its IP address.

Suppose now this machine moves to some distant site. Then it will have to change its address. But the packets destined for it will continue to be routed to its original (home) local area network (LAN), and the user will no longer get e-mail, etc. Moreover, the higher-level protocols

(TCP/UDP) require the IP address of a host to be fixed for identifying and maintaining existing connections.

One solution to the above problem is to give a new IP address to the machine corresponding to its new location. But this cannot be done every time, because a large number of people, programs and databases would have to be informed of the change. In section 5.3, we shall discuss mobile IP, which has become the standard protocol to facilitate physical mobility of hosts.

5.2.2 IPv6

IP version 6 or IPv6 was developed in 1993, mainly to take care of the rapidly increasing number of machines connecting to the Internet. But it also provides other features, as given below:

1. It provides for 16-byte addresses, in place of the 4-byte addresses of IPv4.
2. It is a simpler protocol that allows routers to process packets faster by reducing the size of routing tables.
3. It provides better security through two 'extension headers'. The 'authentication' header provides a mechanism by which the receiver of a packet can be sure who sent the packet. The 'encrypted security payload' header makes it possible to encrypt the packet so that only the intended recipient can read it.

The result is a lean network layer protocol, making it fast, flexible, secure and with large addresses, all major requirements for mobile and pervasive computing. Most important, IPv6 has support for mobility. This will be discussed later.

The **mobile IP** is an extension to the IP, proposed by the Internet Engineering Task Force (IETF), and addresses the issue of mobility. We discuss it in the next section.

5.3 Mobile IP

The mobile IP protocol was proposed by Charles Perkins et al. in 2002, to provide mobility support for IP networks. It allows transparent routing of IP datagrams to mobile nodes in the Internet. In mobile IP, each mobile node is always identified by its home address, regardless of its current point of attachment to the Internet. Every mobile node is associated with a care-of address, which provides information about its current point of attachment to the Internet.

5.3.1 Goals of mobile IP

The major goals envisaged for mobile IP were as follows:

1. To continue to work with the existing TCP/IP protocol suite.
2. To provide Internet-wide mobility, allowing a host the same IP address, called 'home address'.
3. To optimize local area mobility without sacrificing performance or functionality of the general case.
4. To leave the transport layer and higher protocols untouched.
5. To ensure that no application needs to change in order to run on or to be used from mobile hosts (MHs).
6. To ensure that the infrastructure, that is, non-MHs, routers, routing protocols, etc. are not changed either.

7. To see that mobility is handled at the network layer.
8. To ensure that the solution scales well and minimizes potential points of failure, and
9. To ensure minimum power consumption, since mobile nodes are likely to be battery powered.

5.3.2 Applicability

Mobile IP enables nodes to move from one IP subnet to another. It is suitable for mobility across homogeneous as well as heterogeneous media, that is, from one Ethernet segment to another, as well as from an Ethernet segment to a wireless LAN. The only requirement is that the mobile node's IP address remains the same after such a movement.

To provide for mobility, additional protocol support is required at the application layer, for automatic configuration, service discovery, link awareness and environment awareness. This implies that computing activities should not be disrupted when the mobile device changes its point of attachment. In other words, all needed reconnections must occur automatically and without user interaction.

5.3.3 Mobility suppor t in IPv4

Mobile IP introduces the following new functional entities, which are required for both IPv4 and IPv6.

Mobile node: This is a host or router that changes its point of attachment from one network or subnetwork to another. A mobile node (or host) may change its location without changing its IP address.

Home agent: This is a router on a mobile node's home network, which tunnels or redirects datagrams for delivery to the mobile node, when it is away from home, and maintains the current location information for the mobile node.

The home agent maintains the mobility binding in a **mobility binding table** where each entry is identified by the 3-tuple

(permanent home address, temporary care-of address, association lifetime)

Figure 5.1 shows a mobility binding table. The purpose of this table is to map a mobile node's home address with its care-of address and forward packets accordingly.

Foreign agent: This is a router on a mobile node's visited network, which provides routing services to the mobile node while it is registered there. The foreign agent (FA) detunnels and delivers datagrams to the mobile node that were tunneled by the mobile node's home agent.

Home address	Care-of address	Lifetime (in sec)
131.193.171.4	128.172.23.78	200
131.193.171.2	119.123.56.78	150

Figure 5.1 Mobility Binding T able

Home address	Home agent address	Media address	Lifetime (in sec)
131.193.44.14	131.193.44.7	00-60-08-95-66-E1	150
131.193.33.19	131.193.33.1	00-60-08-68-A2-56	200

Figure 5.2 Visitor List

The FA maintains a **visitor list** which contains information about the mobile nodes currently visiting that network. Each entry in the visitor list is identified by the 4-tuple:

(permanent home address, home agent address, media address of the mobile node, association lifetime)

Figure 5.2 shows an instance of a visitor list.
In a typical scenario, the care-of address of a mobile node is the FA's IP address.
The basic mobile IP protocol has four distinct stages. These are as follows:

1. **Agent discovery:** Agent discovery consists of the following steps:
   - **a.** Mobility agents advertise their presence by periodically broadcasting agent advertisement messages. An agent advertisement message lists one or more care-of addresses and a flag indicating whether it is a home agent or a FA.
   - **b.** The mobile node (MH) receiving the agent advertisement message observes whether the message is from its own home agent and determines whether it is on the home network or a foreign network.
   - **c.** If a mobile node does not wish to wait for the periodic advertisement, it can send out agent solicitation messages that will be responded by a mobility agent.
2. **Registration:** Registration consists of the following steps:
   - **a.** If a mobile node discovers that it is on the home network, it operates without any mobility services.
   - **b.** If the mobile node is on a new network, it registers with the FA by sending a registration request message which includes the permanent IP address of the MH and the IP address of its home agent.
   - **c.** The FA in turn performs the registration process on behalf of the MH by sending a registration request containing the permanent IP address of the mobile node and the IP address of the FA to the home agent.
   - **d.** When the home agent receives the registration request, it updates the mobility binding by associating the care-of address of the mobile node with its home address.
   - **e.** The home agent then sends an acknowledgement to the FA.
   - **f.** The FA in turn updates its visitor list by inserting the entry for the mobile node and relays the reply to the mobile node.

   Figure 5.3 illustrates the registration process.
3. **In service:** This stage can be subdivided into the following steps:
   - **a.** When a correspondent node (CN) wants to communicate with the mobile node, it sends an IP packet addressed to the permanent IP address of the mobile node.

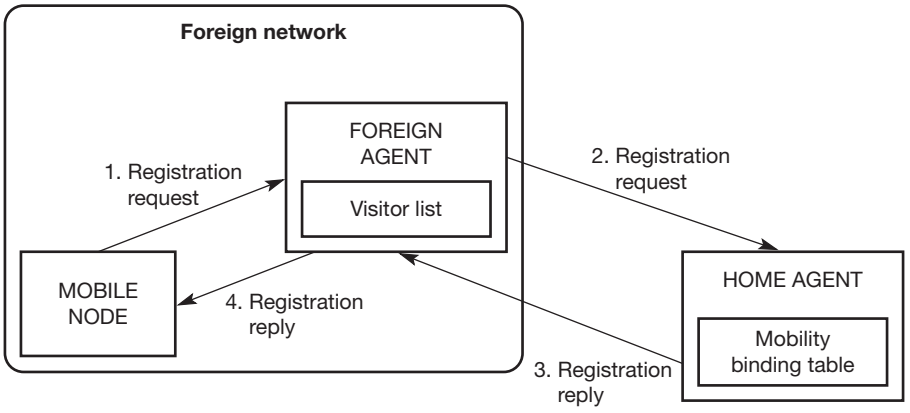


Figure 5.3 Registration Process in Mobile IP

b. The home agent intercepts this packet and consults the mobility binding table to find out if the mobile node is currently visiting any other network.

c. The home agent finds out the mobile node's care-of address and constructs a new IP header that contains the mobile node's care-of address as the destination IP address. The original IP packet is put into the payload of this IP packet. It then sends the packet. This process of encapsulating one IP packet into the payload of another is known as **IP-within-IP** encapsulation or **tunnelling**.

d. When the encapsulated packet reaches the mobile node's current network, the FA decapsulates the packet and finds out the mobile node's home address. It then consults the visitor list to see if it has an entry for that mobile node.

e. If there is an entry for the mobile node on the visitor list, the FA retrieves the corresponding media address and relays it to the mobile node.

f. When the mobile node wants to send a message to a CN, it forwards the packet to the FA, which in turn relays the packet to the CN using normal IP routing. This is called triangular routing. Triangular routing can be avoided and the routing optimized (optionally) in Mobile IPv4 by having the CN send subsequent data packets directly to MH after getting its COA from the home agent in the first data packet.

g. The FA continues serving the mobile node until the granted lifetime expires. If the mobile node wants to continue the service, it has to reissue the registration request.

Figure 5.4 illustrates the tunneling operation.

4. **Deregistration:** If a mobile node wants to drop its care-of address, it has to deregister with its home agent. It achieves this by sending a registration request with the lifetime set to zero. There is no need for deregistering with the FA as registration automatically expires when lifetime becomes zero. However, if the mobile node visits a new network, the old foreign network does not know the new care-of address of the mobile node. Thus, packets already forwarded by the home agent to the old FA of the mobile node are lost.

Security is provided in the system, as follows:

1. **Authentication:** The home agent has to be certain that the registration was originated by the mobile node and not by some malicious node. To ensure this, each mobile node and home

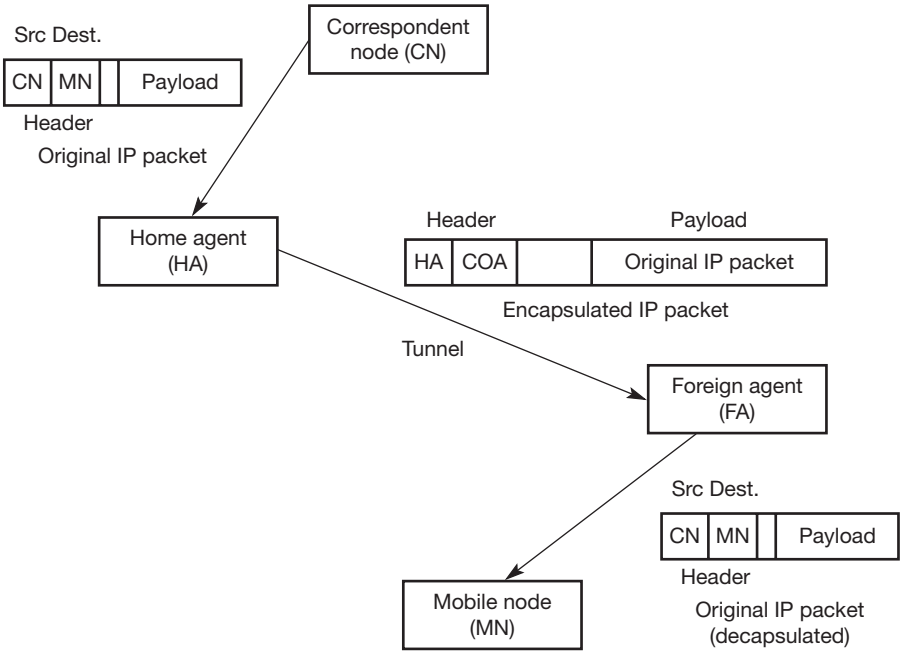


Figure 5.4 Tunneling in Mobile IP

agent have to share a security association. This is done by using the Message Digest 5 (MD5) with 128 keys, which creates a secure digital signature for the registration requests.

2. To overcome replay attacks, each separate registration request is required to have unique data. This is accomplished by having an 'identification' field which changes with every new registration. This could be a timestamp or a pseudorandom number (called a *nonce*).

5.3.4 Mobility suppor t in IPv6

IPv6 mobility support is like that of mobile IPv4, using the concepts of home networks, home agents and encapsulation. But IPv6 has several features that are used to streamline mobility support. These are as follows:

- Route optimization is built as a fundamental part of mobile IPv6 unlike in mobile IPv4, where it is an optional extension that may not be supported by all nodes. In IPv6, a CN is allowed to have the latest binding, so that it can directly tunnel packets to the mobile node without going via home agents. Binding updates to a CN are delivered directly from the mobile node instead of from home agents. It is also possible for the CN not to use tunnelling at all. Instead, it could use IPv6 routing headers, which are similar to the source routing option in IPv4.
- Foreign agents are not needed in mobile IPv6. This is because IPv6 uses the concept of neighbour discovery by the mobile node. Address auto-configuration is another feature used by the mobile nodes to function in any location without the services of any special router in that location.
- In mobile IPv4, when a mobile node communicates with a CN, it puts its home address as the source address of the packet. 'Ingress-filtering' routers present in the network will filter out such a packet, since it finds that the source address of the packet is different from the

network from which the packet actually originated. This problem is tackled in mobile IPv6 by putting the care-of address as the source address, not the home address. There is also a home address destination option, which allows the use of the care-of address to be transparent over the IP layer.

- Security is automatically provided in IPv6 nodes since they are expected to implement cryptographic and strong authentication and encryption features.

Some additional features supported by IPv6 are as follows:

- smooth handoff, which in IPv4, is specified only for the FA, as part of route optimization
- renumbering of home networks if the network addresses in the home network have changed
- automatic home agent discovery for the mobile node. Here, a home agent, who may want to stop operations for reasons like overloading, etc., sends switch messages to the mobile node to get a new home agent. This may also be done if the old home agent has changed or its address has changed.

5.4 Cellular IP

In mobile IP, packets addressed to an MH are delivered using the regular IP routing mechanism, to a temporary address assigned to the MH at its actual point of attachment. This approach results in a simple and scalable scheme that offers global mobility. Mobile IP is not appropriate, however, for **seamless mobility**. This is because after each migration, a local address must be obtained and communicated to a possibly distant location directory or home agent. Support for seamless mobility is needed in order to provide good service quality to mobile users, particularly in pico-cellular environments, where the rate of handoff and associated signalling load grows rapidly.

Thus, mobile IP is not suited to environments where mobility is frequent and restricted to small cells (**micro-mobility**). With frequent handoff, **micro-mobility protocols** have been proposed to handle local movement of MHs without interaction with the mobile-IP-enabled Internet. This has the benefit of reducing delay and packet loss during handoff, and eliminating registration between MHs and distant home agents when MHs remain inside their local coverage areas.

Eliminating registration reduces the signalling load on the core network in support of mobility. Similarly, reducing signalling is necessary for the wireless Internet to scale to very large volumes of wireless subscribers. With cellular IP (as in HAWAII and EMA), a visiting mobile device can use the same care-of address in the whole domain.

As in the case of the cellular phones, wireless IP nodes do not actively communicate most of the time. They are only switched on, ready for service and constantly reachable by the wireless Internet. Hence, MHs are normally in an idle state but passively connected to the network infrastructure. It is sufficient for the wireless Internet to know only the approximate location of its idle users. Their exact location becomes important only when data need to be forwarded to them. The network then should be able to efficiently search and find these users in a scalable and timely manner. In cellular telephony systems, this process is called **paging**.

As the number of mobile subscribers grows, the need to provide efficient location tracking of idle users and paging of active communications also grows. In order to achieve scalable location management, the wireless Internet must handle the location tracking of active and idle MHs independently. Support for **passive connectivity** balances a number of important design considerations.

For example, only keeping the approximate location information of idle users requires significantly less signalling. This not only reduces the load over the core network, but also preserves the battery power of MHs.

Thus, cellular IP incorporates a number of important design features present in cellular networks, like mobility management, passive connectivity, paging and fast handoff control. But it implements them around the IP paradigm. Its salient characteristic is simplicity and minimal use of explicit signalling, enabling low-cost implementation of the protocol.

5.4.1 The cellular IP access networ k

It consists of access routers or base stations (BSs) interconnected by wired links and connected to the Internet by a gateway router (R). It can also contain nodes that have no radio interface but which merely serve as traffic concentrators or support mobility management functions. Mobility between gateways (i.e. cellular IP access networks) is managed by mobile IP, while mobility within access networks is handled by cellular IP. MHs attached to the network use the IP address of the gateway as their mobile IP care-of address. See Figure 5.5. All packets coming from mobile terminals are routed from the BS they are connected to, towards R, no matter what the destination of the packet is.

The Gateway R periodically broadcasts a beacon packet that is flooded through the cellular IP network to make sure that all nodes have a correct routing entry towards it. Nodes in the network record the neighbour they last received this beacon from and use it to route packets towards the Gateway R.

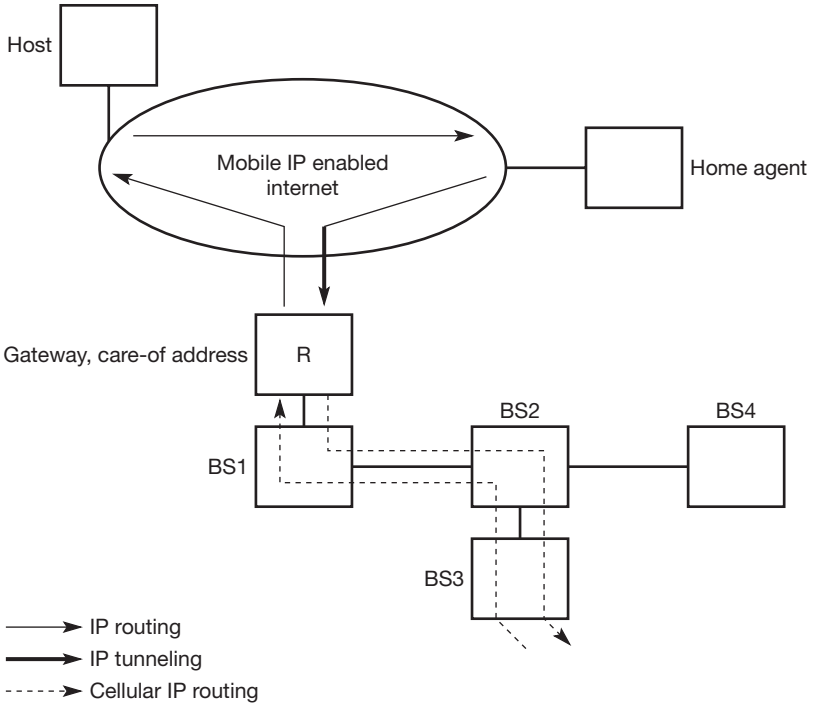


Figure 5.5 Cellular IP Access Networ k

In cellular IP, location management and handoff support are integrated with routing. To minimize control messaging, regular data packets transmitted by MHs are used to establish host location information. **Uplink** packets are routed from mobile to the gateway on a hop-by-hop basis. The path taken by these packets is cached in BSs. To route **downlink** packets addressed to an MH, the path used by recent packets transmitted by the host is reversed. When the MH has no data to transmit, it periodically sends empty IP packets to the gateway to maintain its downlink routing state. Following the principle of passive connectivity, MHs that have not received packets for a certain period of time allow their downlink soft-state routes to timeout.

5.4.2 Routing and paging cache

To send packets towards a mobile device, the nodes use two different caches, a **routing cache** and a **paging cache**, which is optional.

Routing caches are used to route packets towards a mobile terminal. Their entries are timed out after a system-specific time value. When a data packet from a mobile node enters a node, it puts a mapping into its routing cache for the mobile node's IP address and the neighbour node from which the packet came. The packet itself is forwarded to the gateway. All intervening nodes between the BS and the gateway have an entry in their routing caches for the mobile. To keep the caches valid, the mobile node must keep sending route update packets even if it has no data to send.

Paging caches are similar to routing caches but are used to find an idle mobile if a packet has to be sent to it but there is no entry for it in the routing cache. The timing interval of the paging cache is larger than that of the routing cache. Idle mobiles must keep sending paging updates at regular time intervals to the gateway to refresh the paging cache entries. If there is no entry for a mobile in either the routing or the paging caches, at any node in the network, the packet is forwarded to all downlinks by that node.

5.5 TCP for mobility

Transport protocols like TCP have been typically designed for fixed end systems and fixed, wired networks. Such networks use guided transmission media like coaxial cables, optical fibres and special hardware for routers. No transmission errors are introduced by this hardware.

The TCP **congestion control** mechanism in such networks is therefore based on the premise that packet loss is typically due to (temporary) overload situations. In such cases, the routers may discard packets as soon as their buffers are full. TCP recognizes congestion only indirectly via missing acknowledgements and slows down the transmission rate. This is called **slow start**. Retransmissions are not considered good, as they add to the congestion and make it even worse.

In **slow start**, the sender calculates a congestion window for a receiver and starts with a congestion window size equal to one segment. It exponentially increases the congestion window up to the congestion threshold, and then does a linear increase. A missing acknowledgement causes the reduction of the congestion threshold to one-half of the current congestion window size. The congestion window starts again with one segment, and the sender starts sending a single segment. The exponential growth starts once more up to the congestion threshold, then grows in linear fashion.

In TCP, a receiver sends an acknowledgement only after receiving a packet from the sender. If a sender receives several acknowledgements for the same packet, this is due to a gap

in packets received by the receiver. However, the receiver got all packets up to the gap and is actually receiving packets. Therefore, packet loss is not due to congestion, but is a simple packet loss. So the sender must transmit the missing packet(s) before the timer expires. This behavior is called **fast transmit**. The receipt of acknowledgements shows that there is no congestion to justify a slow start. It must continue with the current congestion window. This is called **fast recovery**.

Implications of mobility on TCP TCP assumes congestion if packets are dropped. This assumption does not apply in wireless networks, where packet loss is due to **transmission errors**. Furthermore, **mobility** itself can cause packet loss if, for example, a mobile node roams from one access point (AP) (e.g. FA in mobile IP) to another while there are still packets in transit, or to the wrong AP. If forwarding is not possible, the performance of an unchanged TCP degrades severely.

However, TCP cannot be changed fundamentally owing to the large base of installation in the fixed network. Enhancements to TCP for incorporating mobility have to remain compatible with the basic TCP mechanisms in order to keep the whole Internet together. We now discuss some classical enhancements that have been proposed to make TCP work in mobile networks and environments.

5.5.1 Indirect TCP

We have mentioned earlier that TCP performs poorly with wireless links, and that it cannot be changed for the fixed network. These facts led to the development of indirect TCP (I-TCP).

I-TCP segments a TCP connection into a fixed part and a wireless part. There are no changes to the TCP protocol for hosts connected to the wired Internet; millions of computers will continue to use this protocol. Figure 5.6 shows an example of an MH connected via a wireless link and an AP to the 'wired' Internet, where the CN resides.

Standard TCP is used between the CN and the AP. Its connection now ends at the AP, which acts as a proxy. Thus, the AP is seen as the MH for the fixed host and as the fixed host for the MH. Between the AP and the MH, a special TCP, adapted and optimized to wireless links, is used. The AP could be the FA of mobile IP (see section 5.3). FA is the proxy that relays all data in both directions.

If the CN sends a packet, the FA acknowledges it and sends it to the MH. If received, the MH in turn acknowledges it to the FA. If the packet is lost due to transmission errors on the wireless

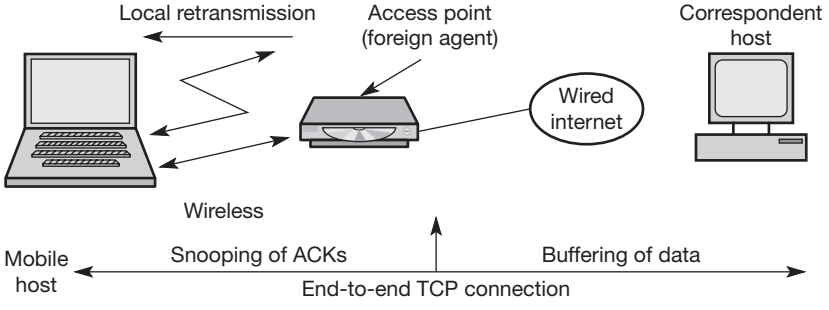


Figure 5.6 Two Segments of a TCP Connection in I-TCP

link, the CN would not notice, as it is the FA that would retransmit it locally to maintain reliable data transport.

If, on the other hand, the MH sends the packet, the FA acknowledges it and tries to forward it to the CN. If the packet is lost on the wireless link, the MH notices this much faster due to the lower round-trip-time and can directly retransmit the packet. Packet loss in the wired network is again handled by the FA, using standard TCP.

In both cases, hosts in the fixed part of the network do not notice the characteristics of the wireless part.

Using I-TCP has several advantages:

- No changes in the hosts on the fixed network are necessary. Similarly, no changes are required for the hosts in a wireless network. All current optimizations to TCP still work between the FA and the CN.
- Transmission errors, like lost packets on the wireless link, do not propagate into the fixed network because of the strict partitioning into two connections.
- It is simple to provide any new optimizations to improve TCP performance on the fixed network, as mobile TCP (M-TCP) is used only for one hop between, for example, the FA and the MH.
- Therefore, a very fast retransmission of packets is possible, since there is a very short known delay on the mobile hop. Any optimized TCP can use precise timeouts to guarantee retransmissions as fast as possible.

But I-TCP also has some disadvantages:

- The loss of end-to-end semantics of TCP occurs if the FA crashes. An acknowledgement to a sender does not now mean that a receiver really got a packet, but that only the FA received it. The CN has no knowledge about the FA at the mobile network.
- Higher handover latency is now possible due to buffering of data by one FA and forwarding it to a new FA, if the MH has moved on.

5.5.2 Snooping TCP

One of the disadvantages of I-TCP is that a single TCP connection is split into two, thus losing the original end-to-end semantic. Snooping TCP is a transparent extension of TCP, leaving its end-to-end connection intact.

The main concept in snooping TCP is that packets are buffered close to the MH for fast local retransmission in case of packet loss. This buffering can be done at the FA. Figure 5.7 shows how this is done. All packets destined for the MH are buffered at the FA, which also 'snoops' the packet flow in both directions. Lost packets on the wireless link (both directions) will be retransmitted

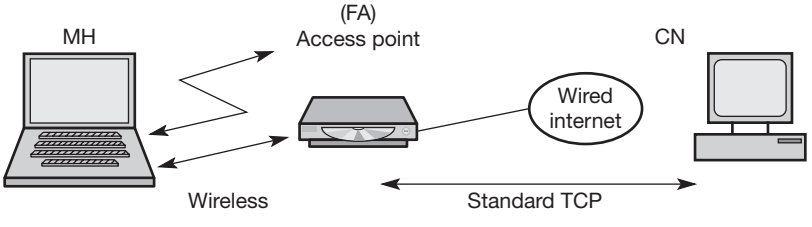


Figure 5.7 Snooping TCP

(locally) immediately by the MH or by the FA, respectively, from the buffer, performing a much faster retransmission compared to the CN. The timeouts for acknowledgments can be much shorter, because they consist of a one-hop delay and processing time.

Transparency is achieved in the following way. For data transfer to the MH, the FA buffers data until it receives an acknowledgement (ACK) from the MH. Packet loss is detected by duplicated ACKs or timeout. In this way, fast retransmission is possible, which is transparent to the fixed network. For data transfer from the MH, the FA detects packet loss on the wireless link via sequence numbers and answers directly with a negative acknowledgement (NACK) to the MH. The MH can now retransmit data with only a very short delay.

Snooping TCP has the following advantages:

- It maintains end-to-end semantics
- No change is required to the CN

But several problems also exist, giving rise to the following disadvantages:

- Snooping TCP does not isolate the wireless link as well as I-TCP.
- It may need change to MH to handle NACKs.
- Snooping may become useless if end-to-end encryption schemes are applied between CN and MH.

5.5.3 Mobile TCP

M-TCP addresses the other limitation of wireless links, besides high error rates. This is related to lengthy and/or frequent disconnections or when the MH cannot connect at all. However, the M-TCP approach has the same goals as I-TCP and snooping TCP: to prevent the sender window from shrinking, if bit errors or disconnection, but not congestion, cause problems. The main requirement is still to improve overall throughput.

M-TCP splits the TCP connection into two parts, just as I-TCP does. An unmodified TCP is used on the standard host-**supervisory host (SH)** connection, while an optimized TCP is used on the SH to MH connection. The SH is similar to the proxy in I-TCP. But a lower bit error rate is assumed on the wireless link. Hence there is no caching and no retransmission of data via the SH. A packet lost on the wireless link has to be retransmitted by the original sender to maintain the TCP end-to-end semantics.

The SH monitors all packets sent to MH and all ACKs returned from it. If no ACK is received for some time, a disconnection is detected and the sender window size is set to 0, so that the sender automatically goes into **persistent mode**, where its state cannot change. As soon as connectivity is detected again, the SH reopens the window at its old value, so that the sender can continue to send at full speed.

The wireless side uses an adapted TCP that recovers from packet loss much faster. This modified TCP does not use slow start, thus M-TCP needs a **bandwidth manager** to ensure fairness over the wireless link.

The advantages of M-TCP are as follows:

- It maintains TCP end-to-end semantics, since the SH does not send any ACK, but just forwards the same from the MH.
- It supports disconnection, by avoiding useless retransmissions, slow starts or breaking connections, by simply shrinking the sender's window size.
- No buffering is done in the SH, only forwarding.

The disadvantages are as follows:

- Since the SH does not act as a proxy as in I-TCP, packet loss on the wireless link is propagated into the fixed network.
- Modifications are required in the MH protocol software and also on new network elements like the bandwidth manager.

Many more modifications and enhancements have been suggested by researchers to optimize TCP for wireless and mobile systems. A good discussion can be found in Schiller (2006).

5.6 Mobile databases

Traditionally, we had large-scale commercial databases that were developed as centralized database systems. However, this trend changed as more and more distributed applications started to emerge. Distributed database applications usually involved a strong central database and powerful network administration. However, rapid advancements in wireless communication and VLSI technologies have changed this paradigm to a great extent.

Today, users desire that a mobile unit (cellphone, PDA, etc.) should have transaction management capability, which allows a user to perform day-to-day activities such as funds transfer, seat reservation and stock trading. In addition, they should be able to access information 24×7 from anywhere in any state, mobile or stationary. For example, a user should be able to access his or her account information, pay bills, buy and sell shares, etc., both while traveling as well as at home.

The above requirements led to mobile database systems. Mobile databases are those that allow the development and deployment of database applications for handheld devices, thus enabling relational database-based applications in the hands of mobile users. Mobile database technology allows employees using handheld devices to link to their corporate networks, download data, work offline and then connect back to the network to synchronize with the corporate database. For example, with a mobile database embedded in a handheld device, a package delivery worker can collect signatures after each delivery and send the information to a corporate database at the end of the day.

5.6.1 Design issues

The mobile computing paradigm directly affects the design, implementation and use of mobile databases. In this section, we list some of the issues that need to be addressed during their design.

- Communication costs play an important role in **database query processing**. The query optimization process requires selecting the best method of query evaluation. Mobility results in dynamically changing the communication costs, which complicate the optimization process. This in turn complicates query processing, especially for cases where **location** plays a key role, since it becomes difficult to determine the optimal location at which to send the result of the query.
- **User time** is a highly valuable commodity in most business applications today and should be taken into account while designing the mobile database. Database retrieval techniques should be fast and efficient.
- Similarly, **connection time** is the unit of monetary charge in most cellular systems. Alternatively, the number of bytes or packets transferred may be the unit of charge. Furthermore, charges based on **time of day** may also apply, and these may vary, based on whether communication occurs during peak or off-peak periods.

- **Energy consumption** has to be limited, because battery power is a scarce resource in a handheld device and should be optimized. This also requires that the query be processed speedily.

5.6.2 Problems in mobile databases

Mobile computing poses typical problems from the point of view of routing and query processing. As one of the major costs involved in wireless communication is based on connection time, there is an incentive for certain MHs to be disconnected for substantial periods. However, during the time of disconnection, the user may still be working on the host machine and may issue queries and updates on locally cached data. This situation creates several problems of the following types:

Recoverability: Updates entered at an MH machine, which is not connected, may be lost if the machine fails. This problem results from the fact that only one copy of information is kept at the local host, and simulation of storage that takes care of failure will be difficult to do.

Consistency: The locally cached data may become inconsistent, but the MH can discover this fact only when it is reconnected. Similarly, the updates occurring in the MH cannot be propagated until reconnection occurs.

Some of these and related issues are taken care of in the CODA system and have been discussed in detail in the next section.

5.6.3 Commercially available systems

Due to the proliferation in mobile handheld devices, many vendors have already come up with commercially available database management systems that have a small footprint. We list some of these here, but details are left for the interested reader. Note that SQL Anywhere from Sybase is the leader with 68 per cent market share.

- IBM's DB2 Everyplace 1.0 (DB2e)
- Oracle 9iLite
- Sybase's SQL Anywhere

These databases work on palmtops and handheld devices (Windows CE devices) providing a local data store for the relational data acquired from enterprise SQL databases. The main constraints for such databases relate to the size of the program as the handheld devices have RAM-oriented constraints. These commercially available mobile database systems allow a wide variety of platforms and data sources. They also allow users with handhelds to synchronize with open database connectivity (ODBC) database content, personal information management data and e-mail from Lotus Development's Notes or Microsoft's Exchange. They support either query-by-example (QBE) or SQL statements.

Till now, we have discussed methods for taking care of **user mobility**. Next, we shall discuss a system designed to take care of **disconnected operation** in wireless networks.

5.7 The CODA file system—A case study

CODA was developed for facilitating disconnected operation in mobile computing and was first demonstrated at Carnegie Mellon University (CMU), USA, in 1992. Disconnected operation is a mode of operation that enables a client to continue accessing critical data during temporary failures of a shared data, in a distributed file system like SUN, NFS, AFS, etc.

Mobile users need continuous connection, since they depend on remote resources like a shared repository to continue working even when the resource is inaccessible. With CODA, disconnected operation is possible and also efficient.

CODA uses data caching, a technique that is normally used for performance improvement. Here it is used for enhancing availability and is successful for 4 to 5 hours of disconnection, with a 100 MB local disk. Reconnection/updation takes just 1 minute.

The CODA environment consists of a large number of untrusted UNIX clients and a smaller number of trusted UNIX file servers, typical in an academic or research environment. CODA runs on IBM RTs, Decstation 3100s and 5000s, and 386-based laptops, and has been in use since April 1990. It had nearly 2 GB of triply replicated data at the end of 1992. Each CODA client has a local disk and cache size of 100 MB and can communicate with servers over a high-bandwidth network. Some clients may be portable, and may be temporarily unable to communicate with servers, due to network or server failure.

5.7.1 Cache manager Venus

CODA provides a single, location transparent shared UNIX file system where namespace is mapped to individual file servers in subtrees called **volumes.** At each client, a cache manager, **Venus,** dynamically obtains and caches volume mappings. CODA uses two distinct, but complementary, mechanisms for high availability:

1. **Server replication** allows volumes to have read-write replicas at more than one server, called volume storage group (VSG). At each client, what is currently accessible is its accessible volume storage group (AVSG). Venus uses a cache coherence protocol to guarantee that the latest copy of a file in the AVSG is available to each client, when opened. This guarantee is provided by servers notifying clients when their cached copies are no longer valid. Modifications are propagated in parallel to all AVSG sites and missing VSG sites.
2. **Disconnected operation** takes effect when the AVSG becomes empty. While disconnected, Venus services file system requests by relying only on the contents of the cache. But cache misses cannot be serviced. When disconnection ends, Venus propagates modifications and reverts to server replication. See Figure 5.8 for a typical scenario in CODA.

There are three servers, *A, B and C* as shown in Figure 5.8, which are connected through a network. They have replicas of a volume containing a file *x*. This file is potentially of interest to three clients, *a, b and c*. *c* is capable of wireless communication (indicated by the dotted line), as well as regular network communication. Proceeding clockwise, the steps in Figure 5.8 (a to f) show the value of *x* that is seen by each node as the connectivity of the system changes. Note that in step d, *c* operates in a disconnected manner.

5.7.2 Venus states

Venus exists in three different states, as shown in the state diagram of Figure 5.9.

In the **hoarding** state, Venus stores useful data in anticipation of disconnection. It also manages its cache so that the needs of connected and disconnected operation are balanced. For example, a user may be using a certain set of files but indicate that it will require another set of files later. Venus must cache both sets of files.

In the **emulation** state, Venus performs many actions normally handled by servers. It thus acts as a pseudoserver, and provides both access and semantic checks. It also generates temporary file identifiers (FIDs) for new objects, which can be assigned permanent ids on reintegration.

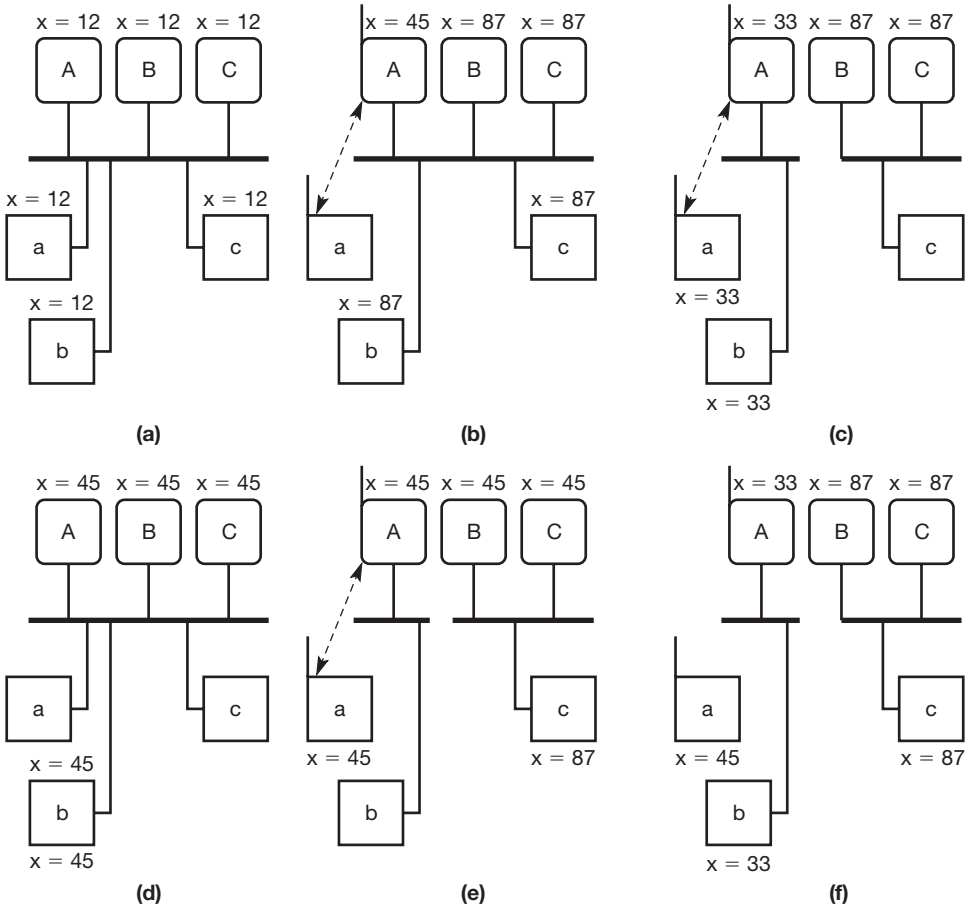


Figure 5.8(a–f) Stages of the COD A system for Disconnected Operation and Ser ver Replication

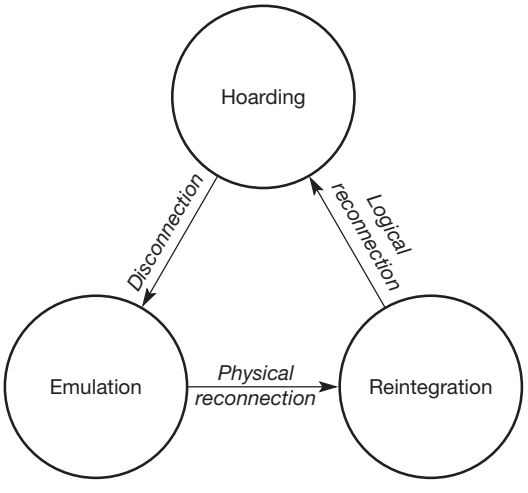


Figure 5.9 CODA State Diagram

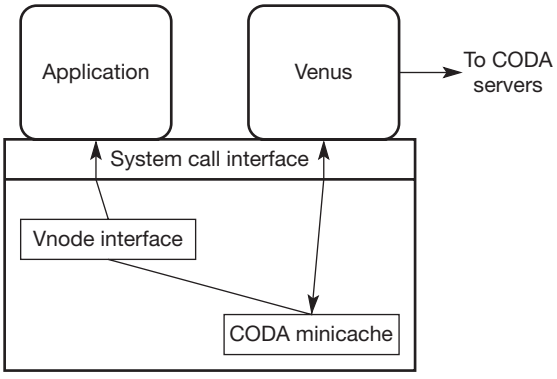


Figure 5.10 Client Str ucture in COD A

Updates accepted by Venus must be revalidated for integrity and protection by the real servers, because CODA trusts only servers, not clients.

Reintegration is a transitory state through which Venus passes from its pseudoserver role to that of cache manager. In this state, Venus propagates changes made during emulation and updates its cache to reflect current server state. Updation is performed a volume at a time, with all update activity in the volume suspended until completion.

The client structure of CODA is given in Figure 5.10. Venus is designed as a user level, rather than kernel-level process, because of its complexity and also to make it more portable and its code easy to debug.

UNIX file system calls are intercepted by Venus through the Sun Vnode interface. CODA uses a small in-kernel mini-cache, which filters out many Venus-kernel interactions and thus reduces overhead. However, the mini-cache does not support remote access, disconnected operation or server replications. These are therefore performed by Venus.

System calls on a CODA object are first forwarded for service to the mini-cache, if possible, and control is returned to the application. If not, the mini-cache calls Venus to service the call, by contacting the remote servers, if required. Control then returns to the application program from Venus, via the mini-cache. The mini-cache state is also updated by Venus, if there are events like callback breaks from CODA servers.

5.7.3 Design criteria

There were basically two main design criteria for CODA. One was the use of **conventional off-the-shelf hardware** and the other was **preserving transparency** by seamlessly integrating CODA into the UNIX environment, as discussed above.

Other less important design criteria were as follows:

1. scalability was included and prepared for a priori, not as an afterthought,
2. workstations should be portable, which is the main application for disconnected operation,
3. availability and consistency are obtained by using the concept of whole-file caching, not partial-file. This helps to have a simple failure model, since a cache miss can only occur on an **open**, never on a **read**, **write**, **seek** or **close**,
4. functionality to be placed on clients, not servers,
5. emphasis on avoidance of systemwide rapid change, without a requirement of consensus,

6. user assistance in augmenting the cache management policy for disconnected operation, since the user is good at predicting future file needs, and
7. no distinction between involuntary or voluntary disconnections, as it has the same mechanism for dealing with both.

5.8 Summary

Physical mobility is characterized by user movement. Two aspects are important here—change of network address and disconnected operation. To take care of the former, mobile IP has been developed as an extension of IPv4, the Internet network layer protocol, to facilitate mobile computing. In this, a care-of address is provided to the mobile user in the foreign network, so that a correspondent node can continue to communicate with it. Three new entities, namely, foreign agent, home agent and mobility binding are relevant here. IPv6 has certain built-in extensions which facilitate mobile computing and provide route optimizations.

Many changes and enhancements have also been made to TCP to accommodate mobility. These include indirect TCP, snooping TCP and mobile TCP. Each of these has its advantages and disadvantages, with respect to latency, end-end semantics, etc.

Mobile databases are the need of the hour for ease of database querying by mobile users, who may be disconnected for some time. The CODA file system was developed to facilitate disconnected operation. It uses off-the-shelf hardware and provides transparency; seamless integration into the UNIX provides these data for use when it is disconnected.

In the next chapter, we shall study mobile ad hoc networks and look at some of the new routing algorithms that have been designed for them.

Problems

1. What are the design issues to be considered in providing user mobility? Discuss.
2. Portability is a major issue with mobile users. What are the challenges introduced by portability in the hardware and software design of handheld computers?
3. All wireless systems are not necessarily mobile and vice versa. Name one example of a wireless system that is not mobile and one mobile system that is not wireless.
4. Show clearly how IP-in-IP tunnelling is used in mobile IP.
5. Is it possible to optimize triangle routing? Discuss clearly how it can be done.
6. How does a correspondent node communicate with a mobile node when mobile IP is in place in a network? Explain using a neat diagram.
7. Foreign agents are not required in mobile IPv6. Also automatic home agent discovery is possible. Explain why home agents are still required in mobile IPv6.
8. Explain clearly why cellular IP cannot be used in place of mobile IP.
9. A person who lives in New Delhi travels to Agra taking her portable computer with her. She finds that the LAN at her office in Agra is a wireless IP LAN, so she does not have to plug in. Is it still necessary to go through the whole business with home agents and foreign agents to make e-mail and other traffic arrive correctly?
10. Discuss, using a neat diagram, the different states of the CODA cache manager Venus.
11. What was the motivation for designing the CODA system? Discuss how it has been achieved.

Multiple-choice questions

1. Which one of the following is the mobility requirement for physical mobility?
   (a) Low power **(b)** Low bandwidth
   (c) Small user interface **(d)** Address migration

2. Which of following is the problem associated with the portability requirement for physical mobility?
   (a) Low power **(b)** Risk to data
   (c) Small user interface **(d)** All of the above

3. Which of the following subsystems consume the highest amount of power in a portable computer?
   (a) Display edge light **(b)** Keyboard
   (c) Hard disk **(d)** Display

4. Which one of the following is the problem in IPv4 addressing for physical mobility?
   (a) It has 32-bit addresses
   (b) It does not provide quality-of-service support
   (c) Its routing and forwarding requires a fixed IP determined by network
   (d) It has security issues

5. The mobility binding table in mobile IP is maintained by
   (a) Mobile node **(b)** Home agent
   (c) Foreign agent **(d)** All of the above

6. Which one of the following is one of the columns of the visitor list maintained by the foreign agent?
   (a) Home address **(b)** Media address
   (c) Life time **(d)** All of the above

7. Why is mobile IP not suitable for seamless mobility?
   (a) Route optimization is optional in mobile IP
   (b) After each migration, local address must to obtained and communicated to home agent
   (c) Mobile IP requires home and foreign agents
   (d) All of the above

8. What is the major difference between the routing and paging caches in cellular IP?
   (a) Paging cache is used to find an idle mobile if a packet is to be sent there, but there is no entry in routing cache.
   (b) Paging cache is related to mobile IP while routing cache is to cellular IP.
   (c) The timeout interval of routing cache is larger than that of the paging cache.
   (d) None of the above

9. Venus acts as psuedoserver in which of the following states in CODA?
   (a) Hoarding **(b)** Reintegration
   (c) Emulation **(d)** None of the above

10. Which of the following is NOT one of the main design rationales for CODA?
    (a) Preserving transparency **(b)** Scalability
    (c) Consistency **(d)** Low power

Further reading

A. Bakre and B. Badrinath (1995), 'I-TCP: Indirect TCP for Mobile Hosts', in Proceedings of the 15th International Conference on Distributed Computing Systems, IDDCS, Vancouver, Canada.

A.G. Valko (1999), 'Cellular IP: A New Approach to Internet Host Mobility', *ACM Computer Communication Review*, January 1999, 29(1): 50–65.

A.S. Tanenbaum (2005), *Computer Networks*, 4th ed. (Delhi, India: Prentice Hall India).

A.T. Campbell, J. Gomez, S. Kim, A.G. Valko, Chieh-Yih Wan, Z.R. Turanyi, et al (2000), 'Design, Implementation and Evaluation of Cellular IP', *IEEE Personal Communication,* August 2000, 7(4): 42–49.

C. Perkins (1998), *Mobile IP: Design Principles and Practice* (Upper Saddle River, NJ: Prentice Hall).

D. Milojicic, F. Douglis and R. Wheeler (eds) (2000), *Mobility: Processes, Computers and Agents,* 2nd print (Addison-Wesley USA).

H. Balakrishnan, S. Seshan and R.H. Katz (1995), 'Improving Reliable Transport and Handoff Performance in Cellular Wireless Networks', *Wireless Networks*, 1(4): 469–481.

J. Kistler and M. Satyanarayan (1992), 'Disconnected Operation in the CODA Distributed System', *ACM Transactions on Computer Systems*, 10(1): 3–25.

J.D. Solomon (1998), *Mobile IP: The Internet Unplugged* (Upper Saddle River, NJ: PTR Prentice Hall).

J.H. Schiller (2006), *Mobile Communications*, 2nd ed. (New Delhi, India: Pearson Education).

K. Brown and S. Singh (1997), 'M-TCP: TCP for Mobile Cellular Networks', *ACM Computer Communications Review*, 5(5).

'RFC 2002—IP Mobility Support' www.ietf.org/rfc/rfc2002.txt. (accessed October 1996).

'RFC 2003—IP Encapsulation within IP', www.ietf.org/rfc/rfc2003.txt. (accessed October 1996).

'RFC 2004—Minimal Encapsulation within IP', www.ietf.org/rfc/rfc2004.txt. (accessed October 1996).

www.oracle.com/technology/software/products/**lite**/index.html accessed July 07.

www.Publib.boulder.ibm.com/inforcenter/db2e accessed July 07.

www.sybase.com accessed July 07.

Mobile Ad Hoc Networ ks 6

The pervasive computing paradigm has introduced a pivotal shift in the design of communication architectures, client devices and network applications. A recent but rapidly maturing component of this paradigm is the design and use of wireless mobile ad hoc networks (MANETs). Another common term for a MANET is WANET (wireless ad hoc network). But we shall use the term MANET in this book.

MANETs, usually composed of small portable clients (or nodes), range in size from simple piconets used for device synchronization and distributed portable gaming to large-scale networks used for supporting military and emergency-response scenarios. Recently, perhaps the most popular WANET applications have been wireless sensor networks (WSNs), which are most often used for remotely collecting environmental and contextual data. WSNs are the subject of the next chapter.

We have seen until now that in a mobile system, hosts are mobile, but the routers are fixed. In the extreme case, as in MANETs, the routers themselves may be mobile. This is possible in a system consisting of military vehicles on a battlefield with no existing infrastructure, a fleet of ships at sea, emergency workers at a disaster site where the infrastructure has been destroyed, or a gathering of people with notebook computers in an area lacking the 802.11 Wi-Fi facility.

A MANET is a collection of wireless mobile hosts forming a temporary network, without the aid of any centralized administration or standard support services that are regularly available on the wide-area-network, to which the hosts may normally be connected. Figure 3.1(b) depicts an ad hoc network.

In this chapter, we will take an in-depth look at MANET characteristics, their classification and their advantages and also, most importantly, study the different routing techniques being developed for them.

6.1 MANET characteristics

Since a MANET has no underlying information infrastructure and is multihop in nature, every node in the network functions as a router. The nodes are free to move arbitrarily, and the topology of the network can be considered to be dynamic. Wireless connectivity between the nodes constrains the bandwidth of the network and lowers its capacity. Fading, noise, interference and congestion are the other results. Mobility of the nodes also results in limited power supply and the dependence on batteries or exhaustible means of power. Similarly, physical security

is limited as the nodes are more vulnerable to eavesdropping, spoofing and denial of service (DOS) attacks.

However, the advantages are numerous. Firstly, deployment is easy and speedy. Secondly, because there is no dependence on infrastructure, the network is robust and low-cost. Finally, MANETs form the basis of all pervasive and ubiquitous computing.

Typical applications of MANETs are in personal area networking (PAN) using cellphones or laptops or wearable computers, as these can be easily deployed in meeting rooms, sports stadiums, airports, aeroplanes and even boats. MANETs have useful application in military systems, where they can be deployed on soldiers or in tanks. In intelligent transportation systems, they provide easy vehicle-to-vehicle communication. Similarly, they can be used with ease in search-and-rescue systems, policing and fire fighting.

6.2 Classification of MANETs

Ad hoc networks can be classified into three categories, depending on the size of the coverage area. See Figure 6.1.

A BAN **(body area network)** can be correlated with a wearable computer. The components of the wearable computer are distributed on the body, and the BAN provides the connectivity among these devices. The communicating range of a BAN corresponds to the human body range, that is, around 1 to 2 meters. The main requirements of a BAN are the ability to interconnect heterogeneous devices like mobile phones, microphones, displays, etc.; the capability of auto-configuration and of service integration; and also the ability to interconnect with other BANs to exchange data with other people or PANs to access the Internet.

A PAN **(personal area network)** is a network in the environment around the person, while a BAN is devoted to the interconnection of one person through wearable devices. Its communication range is typically up to 10 meters. It connects mobile devices carried by users to other mobile and stationary devices. Wireless PAN technology makes use of the 2.4–2.484 GHz ISM band. Spread spectrum is typically employed at the physical layer to reduce interference and utilize bandwidth properly.

Wireless local area networks (WLANs) offer greater flexibility as compared with wired LANs and have a communication range typical of a single building or of a cluster of buildings in the range of 100–500 meters. Two different approaches can be followed in the implementation of LANs. These are the infrastructure-based approach and the ad hoc networking approach.

Infrastructure-based architecture imposes the existence of a centralized controller for each cell, often referred to as an access point, which is itself connected to the wired network. In the ad hoc approach, which is the topic of discussion here, a network is formed by the set of stations within the range of each other that dynamically configure themselves to set up a temporary network. No fixed controller is required, but one is dynamically elected from among the stations participating in the communication.

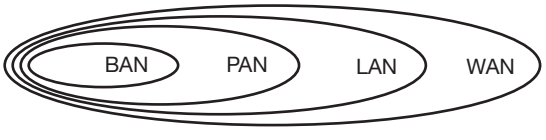


Figure 6.1 Categories of Ad Hoc Networ ks

6.3 Technologies for ad hoc networ ks

Currently, two main standards exist for ad hoc wireless networks for the lower data link and physical layers. These are the IEEE 802.11 standard for WLANs and Bluetooth, for short-range communication. Both these have been studied in Chapter 3, but we give their main features here again for completeness.

IEEE 802.11 specifies media access control (MAC) layer and physical layers for WLANs. The basic access methods in IEEE 802.11 MAC protocol are distribution coordination function (DCF), which is a carrier sense multiple access/collision avoidance (CSMA/CA) MAC protocol, and the point coordination function (PCF), which operates using a polling mechanism, where a point coordinator provides transmission rights to a single station at a time.

The DCF method is applicable to ad hoc networks. In the IEEE 802.11 standard, an ad hoc network is called an independent basic service set (IBSS). An IBSS enables two or more IEEE 802.11 stations to communicate directly without requiring the intervention of a centralized access point or an infrastructure network.

Bluetooth technology is a standard for low-cost, short-range radio links between mobile PCs, mobile phones and other portable devices. The Bluetooth system operates in the 2.4 GHz ISM band. A Bluetooth unit integrated into a device enables wireless ad hoc communication of voice and data in stationary and mobile environments. It is to be noted that MANETs are multihop networks and the source and destination nodes may not be in the radio range of each other. Thus, a route has to be established before data transmission.

6.4 Routing in MANETs

To date, a plethora of research efforts has addressed some significant challenges arising in the implementation of various aspects of MANET operation, most notably routing. In general, MANET routing protocols are expected to satisfy the following essential principles:

- Tolerance of unexpected network faults (e.g. device and link failures)
- Resilience to increasing traffic loads
- Minimal energy consumption (especially for smaller clients)

6.4.1 Traditional routing protocols

For wired networks, routing protocols can be classified into two types—distance vector routing protocols and link state routing protocols. We give the salient characteristics of each and discuss their shortcomings.

In the distance vector routing, routing tables are exchanged between neighbouring nodes. Routes are obtained by comparing distances to the destinations and by computing the shortest path to each host. RIP is an example of such a protocol and is extensively used on the Internet. However, it is only used inside an autonomous system (AS) and therefore scales poorly.

In the link state routing, neighbouring nodes only exchange link states. Routes are obtained by computing a graph of the most recent network topology, independently on every participating node, and computing the shortest distance to each host. Open Shortest Path First (OSPF2) is an example of such a protocol, which is extensively used on the Internet backbone as it has good scalability.

However, both the above protocols have several shortcomings, especially for ad hoc networks. For example, in ad hoc networks, there are only unidirectional links between nodes, and there may be more than one eligible path between the nodes. This gives rise to an increased

number of updates required. The protocols converge very slowly and hence show poor adaptation to rapid topology changes.

6.4.2 Requirements for routing protocols

The requirements for routing protocols for MANETs are very different from those for wired networks because of their unique nature. We have already seen in Chapter 2 that multiple-access schemes are not possible in wireless networks because of the hidden terminal and exposed terminal problems. There are many other requirements as indicated below.

Firstly, the routing protocol should operate in a distributed manner and be multihop and loop-free for best results. The primary cause for formation of routing loops in ad hoc networks is that nodes choose their next hops in a completely distributed fashion based on information that can be stale and therefore incorrect. This effect is more pronounced in ad hoc networks with high node mobility.

Secondly, the protocol should operate in a demand-based, that is, reactive or proactive mode because of the possibility of rapid changes in topology.

Thirdly, since the MANET size may increase to thousands of nodes, the routing protocol should be scalable.

Fourthly, since a node may not be in operating state most of the time, there should be provision for a 'sleep' period.

Fifthly, the protocol should support unidirectional links since transmission in both directions may not be same or possible.

Finally, security is a major concern here because of the ability for anyone to connect to the ad hoc network.

The above are some qualitative requirements for MANET routing protocols. The metrics for quantitative performance are concerned with, among other things, route acquisition time, end-to-end data throughput and delay, percentage of out-of-order delivery and efficiency and overheads of data transmission.

Thus, specific routing protocols for MANETs are desirable. Furthermore, there are two basic parts to the routing protocols. These have to do with **route discovery** and **route maintenance**. Route discovery is required to be done frequently because node mobility is high. To communicate with another node, the source node must first discover the route by initiating a route discovery procedure, using some kind of flooding. Route maintenance is to be done for the same reason, and also because route changes can occur due to factors like noise, interference, link breakage, etc. When the status of a link on a route changes, a mechanism is needed to modify the route information at the affected nodes. This is usually done by using hop-by-hop acknowledgements.

In the next section, we shall discuss routing protocols for MANETs in detail.

6.4.3 Classification of routing protocols

Routing protocols for MANETs may be unicast or multicast. Within unicast routing protocols, there are general protocols or those that use position information. Within general routing protocols, there are proactive, reactive and hybrid protocols. See Figure 6.2. Note that multicast and position-based protocols are also very much used in MANETs, but we will not discuss them in this book. The interested reader is referred to Mauve et al. (2003) and Lee et al. (2000) for more material on these topics.

Proactive or table-driven routing protocols are those that calculate all possible paths in the network independently of their use. The advantage of such a scheme is that when a packet needs to be

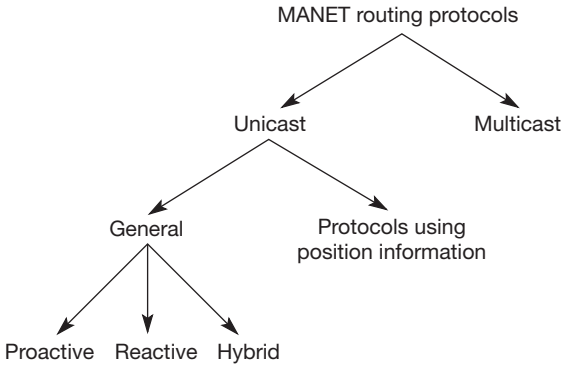


Figure 6.2 Classification of Routing Protocols for MANETs

forwarded, the route is already known and can be used immediately. Reactive or source-initiated on-demand routing protocols are those that invoke the route determination procedure only on demand. If data traffic is not generated by the nodes, then routing activity is totally absent. The hybrid class of routing protocols basically merges the features of proactive and reactive protocols.

In this chapter, we shall see one proactive and two reactive routing protocols for MANETs. For information on the other protocols the reader is referred to Perkins (2001).

6.5 Proactive routing protocols—The DSD V protocol

The most popular proactive routing protocol is the **destination-sequenced distance vector** routing protocol, also known as the DSDV protocol. This is a table-driven algorithm based on the classical Bellman-Ford Distance Vector routing algorithm used for wired networks. The improvements made to the Bellman-Ford algorithm include freedom from loops in routing tables. Loops are to be avoided in routing tables, as they lead to packets continually moving in the network, adding to the congestion of the network and delay of packets sent. There are four distinct phases of the DSDV protocol, as given below.

1. Route advertisements: Every mobile node in the network maintains a routing table in which all the possible destinations within the network and the number of hops to each destination are recorded. Each entry is marked with a sequence number assigned by the destination node. The sequence numbers enable the mobile nodes to distinguish stale routes from new ones, and also avoid the formation of routing loops. These routing tables are broadcast to its current neighbours periodically.

2. Routing table entry structure: The data broadcast by each mobile node contains the new sequence number and the following information for each new route:

- The destination address;
- The number of hops required to reach the destination; and
- The sequence number of the information received regarding that destination, and stamped by the destination

3. Responding to topology changes: To maintain the consistency of routing tables in a dynamically varying topology, routing table updates are periodically transmitted by a mobile

node to each of its current neighbours in the network. Updates are transmitted immediately when significant new information is available.

Routing information is advertised by broadcasting or multicasting the packets which are transmitted periodically and incrementally as topological changes are detected—for instance, when nodes move within the network. Data is also kept about the length of time between arrival of the **first** and the arrival of the **best** route for each destination. Based on this data, a decision may be made to delay advertising routes which are about to change soon. This helps to damp the fluctuations of the routing tables.

To help reduce the potentially large amount of network traffic that such updates can generate, route updates can employ two possible types of packets—**full dump** packets and **incremental dump** packets. Incremental dumps relay only that information which has changed since the last full dump, and fit in one network protocol data unit (NPDU) only. Full dumps carry all available routing information and can require multiple NPDUs. The mobile nodes maintain an additional table where they store the data sent in the incremental routing information packets.

4. Route selection criteria: When a mobile host receives new routing information, through an incremental packet, it compares it to that already available from previous routing packets. A route with a more recent sequence number is used. Routes with older sequence numbers are discarded. A route with a sequence number equal to an existing route is chosen if it has a 'better' metric, say cost, and the existing route discarded, or stored as less preferable.

6.5.1 Example of DSD V operation

Consider the ad hoc network given in Figure 6.3, which shows eight mobile nodes MH_1–MH_8, of which MH_1 has moved to a new location (shown by the dashed line). In this network, consider MH_4. Table 6.1 shows a possible structure of the forwarding table at MH_4. It contains the following four fields—the address of each destination node in the network, the next hop to it, the metric or cost to it and the sequence number of the packet with its creator.

Table 6.2 shows a typical advertised route table for MH_4. Suppose MH_1 now moves close to MH_8 and MH_7 and away from the others, especially MH_2. The new internal forwarding table at MH_4 now appears as shown in Table 6.3. Only the entry for MH_1 shows a new metric, but in the intervening time, many new sequence number entries have been received. The first entry must

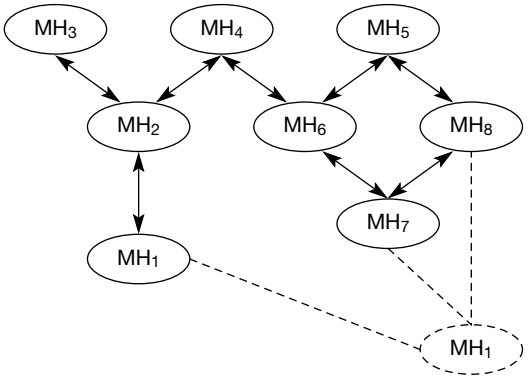


Figure 6.3 Movement in an Ad Hoc Networ k

Table 6.1 Structure of the MH_4 Forwarding Table

Destination	Next hop	Metric	Sequence number
MH_1	MH_2	2	$S400\_MH_1$
MH_2	MH_2	1	$S120\_MH_2$
MH_3	MH_2	2	$S500\_MH_3$
MH_4	MH_4	0	$S700\_MH_4$
MH_5	MH_6	2	$S390\_MH_5$
MH_6	MH_6	1	$S070\_MH_6$
MH_7	MH_6	2	$S120\_MH_7$
MH_8	MH_6	3	$S040\_MH_8$

Table 6.2 Advertised Route Table by MH_4

Destination	Metric	Sequence number
MH_1	2	$S400\_MH_1$
MH_2	1	$S120\_MH_2$
MH_3	2	$S500\_MH_3$
MH_4	0	$S700\_MH_4$
MH_5	2	$S390\_MH_5$
MH_6	1	$S070\_MH_6$
MH_7	2	$S120\_MH_7$
MH_8	3	$S040\_MH_8$

Table 6.3 Updated MH_4 Forwarding Table

Destination	Next hop	Metric	Sequence number
MH_1	MH_6	3	$S500\_MH_1$
MH_2	MH_2	1	$S230\_MH_2$
MH_3	MH_2	2	$S670\_MH_3$
MH_4	MH_4	0	$S820\_MH_4$
MH_5	MH_6	2	$S510\_MH_5$
MH_6	MH_6	1	$S180\_MH_6$
MH_7	MH_6	2	$S230\_MH_7$
MH_8	MH_6	3	$S170\_MH_8$

be advertised in subsequent incremental routing information updates until the next full dump occurs.

When MH_1 moved near MH_8 and MH_7, it started an incremental routing information update immediately and broadcast it to MH_6, which in turn sent an immediate update with new information for MH_1. MH_1, upon receiving this information, would broadcast it at intervals until the next full information dump. The updated routing table advertised by MH_4 is shown in Table 6.4.

Table 6.4 Updated MH_4 Advertised Table

Destination	Metric	Sequence number
MH_4	0	$S820\_MH_4$
MH_1	3	$S500\_MH_1$
MH_2	1	$S230\_MH_2$
MH_3	2	$S670\_MH_3$
MH_5	2	$S510\_MH_5$
MH_6	1	$S180\_MH_6$
MH_7	2	$S230\_MH_7$
MH_8	3	$S170\_MH_8$

In this advertisement, the information for MH_4 comes first, since it is doing the advertisement. Information for MH_1 is given next, because it is the only one which has significant route changes that affect it.

It has been found that the DSDV approach has many advantages. Its memory requirement is only **O(n)**, which is good for the small handheld systems usually used in ad hoc networks. It guarantees loop-free paths without requiring complex update protocols at nodes. Another useful feature of DSDV is that it can be used at either the network layer (layer 3) or below the network layer, but still above the MAC layer software in layer 2. In the latter case, certain additional information needs to be included along with the routing tables for convenient and efficient operation. Finally, though the worst-case convergence behaviour of DSDV is non-optimal, in the average case, convergence is expected to be quite rapid.

The main drawback of the DSDV protocol is that since it is a proactive protocol, routes tend to get stale and hence are rendered useless, in high-mobility scenarios. This disadvantage is removed in reactive routing protocols, which we will study next.

6.6 Reactive routing protocols

Reactive routing protocols are intended to maintain routing information about 'active' routes only. Routes are created when desired by the source node. Hence, the protocols are known as on-demand routing protocols. A route discovery procedure is needed before data transmission, giving rise to high latency. A separate route maintenance procedure is also necessary to adapt to link state changes. These concepts will be discussed in detail below.

However, no periodic routing advertisement messages are sent, thereby reducing network bandwidth overhead, particularly during periods when little or no significant host movement is taking place. Battery power is also conserved on the mobile hosts, both by not sending advertisements and by not needing to receive them (since a host could otherwise reduce its power usage by putting itself to 'sleep' or 'standby' mode when not busy with other tasks).

It must be noted that in a wireless environment, network transmission between two hosts does not necessarily work equally well in both directions, due to differing propagation or interference patterns around the two hosts.

Examples of reactive routing protocols are dynamic source routing (DSR), adaptive on-demand vector (AODV) and temporarily ordered routing algorithm (TORA). In this chapter, we

discuss only the DSR and AODV protocols in detail and give a comparison of the two. Both are distant relatives of the Bellman Ford distance vector routing algorithm. For a discussion on TORA, refer to Park and Corson (1997, 2001).

6.6.1 Dynamic source routing (DSR)

DSR is an example of a reactive routing protocol that does both route discovery and route maintenance. Source routing is a routing technique in which the sender of a packet determines the complete sequence of nodes through which to forward a packet; the sender explicitly lists this route in the packet's header, identifying each forwarding 'hop' by the address of the next node to which to transmit the packet on its way to the destination host.

Each router maintains a route cache in which it caches source routes that it has learnt. When one host sends a packet to another host, it first checks its route cache for a source route to the destination. If a route is found, the sender uses it to transmit the packet, else it may attempt to discover a route using the **route discovery** protocol. Each entry in the route cache has associated with it an expiration period, after which the entry is deleted from the cache.

While a router is using any source route, it monitors the continued correct operation of that route. If any intermediate host moves out of the transmission range or the next or previous hop along the route, the route can no longer be used to reach the destination. This monitoring of the correct operation of a route is called **route maintenance**. If a problem is detected with any route, route discovery is used again to discover a new route to the destination.

6.6.1.1 Route discovery in DSR

Route discovery allows any host in the ad hoc network to dynamically discover a route to any other host in the network, whether directly reachable within transmission range or reachable through one or more intermediate network hops through other hosts. The following steps are followed by DSR during route discovery.

1. Before the source (S) sends a data packet to the destination (D), it broadcasts a **route request (RREQ)** packet. RREQ contains the IDs of both S and D. In addition, it contains a **route record**, in which is accumulated a record of the sequence of hops taken by the RREQ as it is propagated through the network, and the sequence number of S, to identify a duplicate RREQ from S. All duplicates are discarded.
2. Every intermediate node appends its own ID in RREQ and then rebroadcasts it to all its neighbours.
3. If the intermediate node's route cache contains a route record for the destination, it sends the **route reply (RREP)** packet to S.
4. When D receives the first RREQ, it answers S with a RREP packet.
5. RREP goes back along the reverse of the route recorded in RREQ in case the links are bidirectional, otherwise the node piggybacks the RREP on a new RREQ.

To illustrate DSR route discovery, consider the network given in Figure 6.4.

Consider that the source node S has to send data to destination node D. S does not have path to D, so it initiates route discovery by sending RREQ to neighbours A and B. The procedure for route discovery as given above is illustrated in Figures 6.5(a–e), where the route records for RREQ are shown next to the nodes, as they build up at each hop (node). Figure 6.5a shows that A and B, which are neighbours of S (i.e. they are in the radio range of S), have received the RREQ packet and built their route records [S]. In Figure 6.5b, A and B have rebroadcast the RREQ packets to S, B,

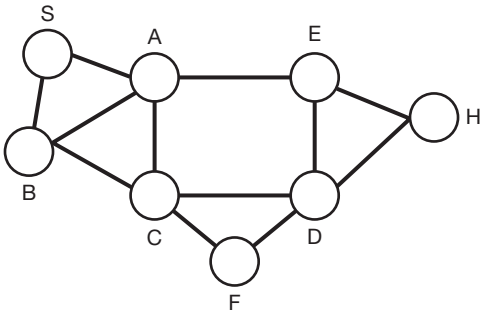


Figure 6.4 Ad Hoc Networ k for DSR Routing

C and E and S, A and C, respectively, after appending their own IDs in the RREQ packet. In turn, C and E have received these route records [S, A] and [S, B], respectively. This process goes on, as shown in Figure 6.5d, where the destination node D has received the route record [S, B, C] and [S, A, E, H]. D now chooses the shorter path given by [S, B, C] and answers with the RREP packet, which goes back to S by the reverse route [C, B, S].

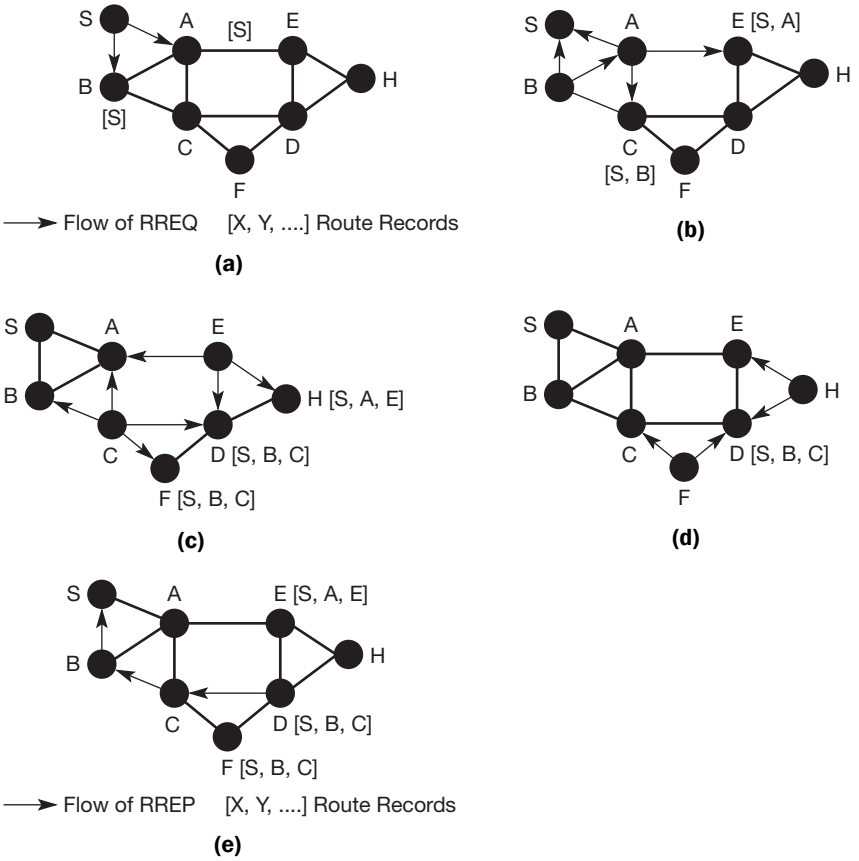


Figure 6.5(a–e) Route Disco very in DSR

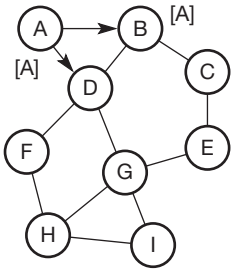


Figure 6.6(a) A does not ha ve a path to I, so it initiates route disco very by sending RREQ to neighbour s D and B.

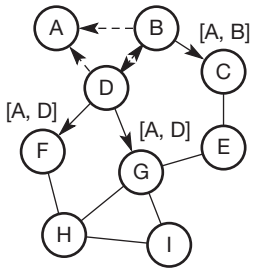


Figure 6.6(b) B and D broadcast RREQ to their neighbour s. A discards B' s and D's RREQ. D discards B' s RREQ and B discards D' s RREQ.

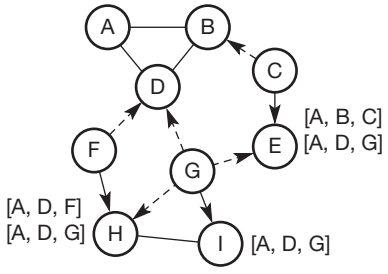


Figure 6.6(c) RREQ reaches the destination node I.

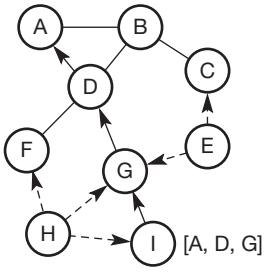
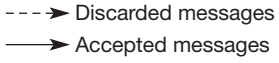


Figure 6.6(d) I unicasts RREP to G which, in turn, forwards it to D . A receives RREP from D and updates its routing table.



In Figures 6.6(a–d), another example of a network is taken to illustrate this procedure further.

6.6.1.2 Route maintenance in DSR

The route maintenance procedure monitors the operation of a route in use and informs the sender of any routing errors. Route maintenance is easy to provide in wireless networks, since at each hop, the sender can determine if that hop of the route is still working. The following steps are followed by DSR during route maintenance.

1. If any node detects failure of a link along the route, a **route error (RERR)** packet is sent to S. The RERR packet contains the addresses of the hosts at both ends of the link.
2. Intermediate nodes remove or truncate the invalid route to D on receipt of RERR.
3. Route discovery is reinitiated by S if it still wants to communicate with D.

6.6.1.3 Route cache in DSR

The presence of a route cache in the nodes has many advantages. These are given below:

1. Intermediate routes record the route backward to S contained in RREQ.
2. Intermediate nodes record the route backward to D contained in RREP.

3. Intermediate nodes can answer S with RREP if they have a valid route to D.
4. Every node records the source route contained in the data packet it overhears.
5. In response to a single-route discovery packet, as well as through routing information from other packets overheard, a node may learn and cache multiple routes to any destination.

This support for multiple routes allows the reaction to routing changes to be much more rapid, since a node with multiple routes to a destination can try another cached route if the one it has been using fails.

However, there are some disadvantages of the route cache. These are as follows:

1. Caching can result in faster route repair, but faster does not necessarily mean correct.
2. An intermediate node may send RREP using a stale cached route, thus polluting other caches.
3. If incorrect repairs occur often enough, caching performs poorly.

Hence, there is a need for a mechanism to determine when cached routes become stale. Many optimizations to the basic operation of DSR have been developed, including the mechanism mentioned above, but these are beyond the scope of this book. The interested reader is referred to Johnson (1994) and Johnson et al. (2001) for details. This paper also provides the results of a packet-level simulation of DSR, which shows that even for high rates of host movement, the overhead of the protocol is quite low, falling to just 1 percent of total data packets transmitted.

6.6.2 Adaptive on-demand distance vector protocol

AODV is a reactive, distance vector routing protocol, based on the distributed Bellman Ford routing algorithm. However, it provides a major improvement on DBF by achieving freedom from loops. We shall see how AODV achieves loop freedom.

AODV also attempts to improve on DSR by maintaining routing tables at the nodes, so that data packets do not have to contain routes. Later in the chapter, we shall compare and contrast DSR and AODV.

Since AODV is also a reactive routing protocol like DSR, a node wanting to send a packet must first discover a route to that destination. So a route discovery procedure is required. Similarly, in order to keep track of route changes due to high mobility or switching off of nodes, routes need to be maintained, as seen in DSR. Route discovery and maintenance techniques for AODV are discussed next.

6.6.2.1 Route discovery in AODV

The following steps are followed for route discovery in AODV.

1. Source S, who wants to send a data packet to destination D, first constructs and broadcasts an RREQ packet. The format of the RREQ packet is shown below in Figure 6.7.

Source address	Request ID	Destination address	Source sequence #	Destination sequence #	Hop count

Figure 6.7 Format of a Route Request P acket

Source address	Destination address	Destination sequence #	Hop count	Life time

Figure 6.8 Format of a Route Reply P acket

It contains the source and destination Internet protocol (IP) addresses, and a **Request ID**, which is a local counter maintained separately by each node and incremented each time a RREQ is broadcast. This is used to identify and discard duplicate RREQs. The **source sequence #** is used as a clock which increments whenever a RREQ is sent, and is used to tell new routes from old ones. The **destination sequence #** field shows the most recent value of the destination's sequence number that the source has seen, and helps to identify fresh routes from stale ones. The final field, **hop count**, will keep track of how many loops the packet has made. It is initialized to 0.

2. On receipt of RREQ, intermediate nodes inspect it to see if it is a duplicate, in which case it is rejected. If not, the (source address, request ID) pair is entered into the local history table. The destination is looked up in the routing table, and if a fresh route to it is known, an RREP packet is sent back to S. If not, it increments the hop count and rebroadcasts the RREQ. They also create a backward route towards S. This feature exists as an optimization technique in DSR, but is mandatory in AODV. A timer is also started for the newly made backward route entry. If it expires, the entry is deleted.
3. When D receives RREQ, it sends back (unicasts) an RREP to the node it got the RREQ from. The format of RREP is shown in Figure 6.8.
   Here, the **source address**, **destination address** and **hop count** are copied from the incoming RREQ packet, but the destination sequence # is taken from its counter in memory. The **lifetime** field controls how long the route is valid.
4. On receipt of RREP, intermediate nodes, on the way back, inspect the packet and enter into the local routing table, to create a backward route towards D, if the following conditions are met:
   a. No route to D is known.
   b. The sequence number for D in RREP is greater than the value in the routing table.
   c. The sequence numbers are equal, but the new route is shorter.
   In this way, all nodes on the reverse route learn the route to D for free, as a by-product of S's route discovery.
5. Intermediate nodes that got the original RREQ packet but were not on the reverse path discard the reverse route table entry when the associated timer expires.

6.6.2.2 Route maintenance in AODV

The following steps are followed for route maintenance in AODV.

1. For each entry in the routing table, every node maintains a list of its active neighbours.
2. A neighbour N of node X is considered active for a routing table entry, if N had replied to a 'hello' packet, within the interval called **active-route-timeout** that was forwarded using that entry.
3. When the next hop link in a routing table entry breaks, all active neighbours are informed.
4. Link failures are propagated by means of RERR packets, which also update destination sequence numbers.
5. RERR packets are also generated when a node X is unable to forward packet P from node S to node D on link (X, Y).

6. The incremented sequence number N is included in the RERR.
7. When node S receives the RERR, it initiates a new route discovery for D using a destination sequence number that is at least as large as N.

One example of AODV routing is shown in Figures 6.9(a–e). Another example is shown in Figures 6.10(a–d), to illustrate the procedure further.

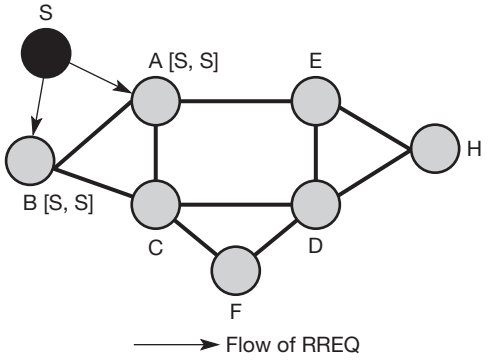


Figure 6.9(a)

Source—S
Destination—D

S does not ha ve a path to D , so it initiates route disco very by sending RREQ to neighbour s A and B.

A and B store the path to S in their routing table.

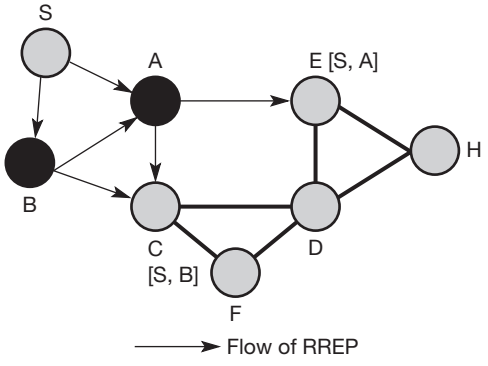


Figure 6.9(b)

A broadcasts RREQ to its neighbour s.
B broadcasts RREQ to its neighbour s.
All inter mediate nodes mak e entr y in their routing table for S and next hop to forw ard packet (for S).

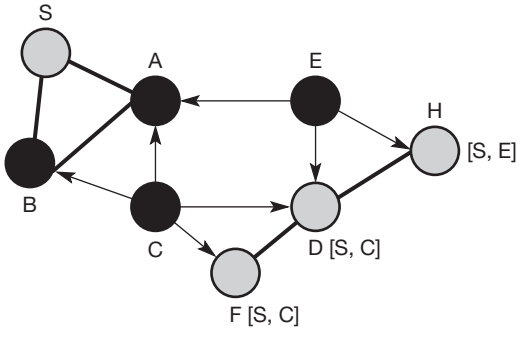


Figure 6.9(c)

D receives RREQ from S via C.

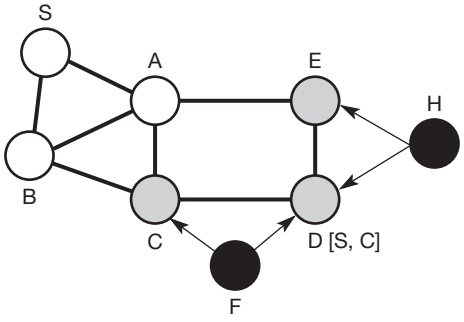


Figure 6.9(d)

D receives RREQ from S via H (decides to transmit RREP via C).

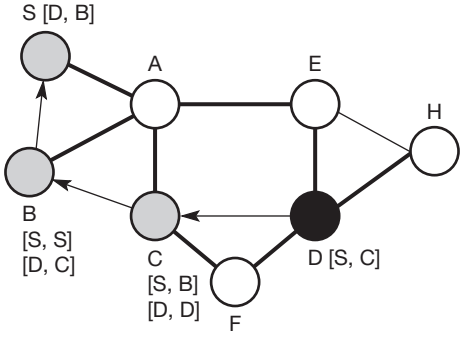


Figure 6.9(e)

Intermediate nodes record the entr y for D in their tables.

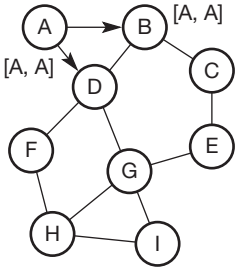


Figure 6.10(a) A does not ha ve a path to I, so it initiates route disco very by sending RREQ to neighbours D and B.

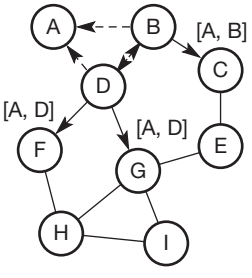


Figure 6.10(b) B and D broadcast RREQ to their neighbour s. A discards B' s and D's RREQ. D discards B' s RREQ and B discards D' s RREQ.

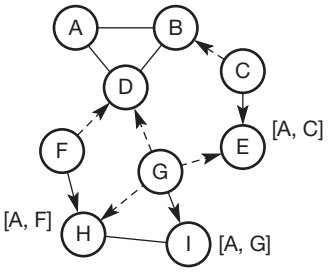


Figure 6.10(c) RREQ reaches the destination I. E receives C' s RREQ fir st, so it accepts C' s RREQ and discards G' s RREQ.

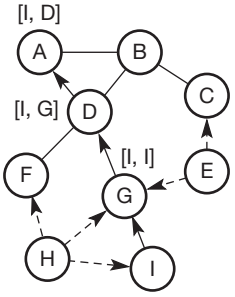


Figure 6.10(d) I unicasts RREP to G which, in turn, forwards it to D . A receives RREP from D and updates its routing table.

---➤ Discarded messages
——➤ Accepted messages

6.7 Comparison betw een DSR and A ODV

In this section, we will compare and contrast the routing techniques and mechanisms of DSR and AODV.

1. **General:** While DSR and AODV share the on-demand behaviour, many of their routing mechanisms are very different.
   - **a.** DSR uses source routing, whereas AODV uses a table-driven routing framework and destination sequence numbers.
   - **b.** By virtue of source routing, DSR has access to a significantly greater amount of routing information than AODV. For example, in DSR, by using a single request-reply cycle, the source can learn routes to each intermediate node on the route in addition to the intended destination.
   - **c.** DSR replies to all requests reaching a destination from a single request cycle. In AODV, on the other hand, the destination replies only once to the request arriving first and ignores the rest.
   - **d.** The route deletion activity using RERR is also conservative in AODV. By way of active neighbour list, RERR packets reach all nodes using a failed link on its route to any destination. In DSR, however, an RERR simply backtracks the data packet that meets a failed link. Nodes that are not on the upstream route of this data packet, but use the failed link, are not notified promptly.

   Table 6.5 below shows a comparison of the two protocols.
2. **Routing overhead:** DSR always demonstrates a lower routing overhead than AODV. The major contribution to routing overhead in AODV is from RERRs, while RREPs constitute a large fraction of routing overhead in DSR. AODV has more RERRs than DSR, and the converse is true for RREPs.
3. **Effect of mobility:** Link failures can happen very frequently in MANETs. They trigger new route discoveries in AODV, since it has at most one route per destination in its routing table. However, route discovery is delayed in DSR until all cached routes fail. With high mobility,

Table 6.5 Comparison betw een DSR and A ODV

Property	DSR	AODV
Type of routing	Source routing	Table-driven routing with destination sequence numbers
Amount of routing information	Greater	Lesser
Reply to requests	Replies to all requests reaching a destination from a single request cycle	Replies only once to the request ar riving first and ignores the rest
Route deletion activity using RERR	Fast, using backtracking	Conser vative, using active neighbour list

the chances of the caches becoming stale are quite high in DSR. Hence, cache staleness and high overhead together result in significant degradation in performance for DSR in high-mobility scenarios.

In Chapter 12, we shall see a case study of the simulation of DSDV and AODV protocols, which throws more light on their comparative performance.

6.8 Summary

Mobile ad hoc networks (MANETs) have gained popularity in recent years as they form the basis for pervasive or ubiquitous computing. They differ from the IEEE 802.11 wireless LANs in that they do not require any infrastructure (base station) for operation and can be easily deployed anywhere and at least cost.

MANET routing is becoming an important research area since traditional routing protocols cannot be used for MANETs because of their poor adaptation to rapidly changing topology and the need to support unidirectional links.

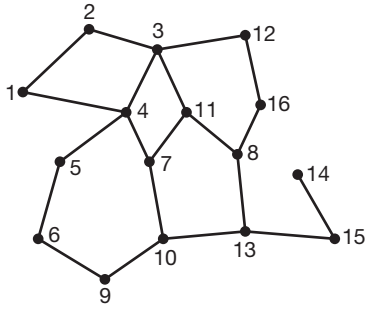
There are two types of routing protocols for MANETs. These are reactive and proactive, and both need the twin mechanisms of route discovery and route maintenance. However, in proactive protocols, route discovery is done continuously even if no data is to be sent, whereas in reactive protocols, it is done only when required, that is, just before data is to be sent. But route maintenance is required in both, since routes tend to become stale and useless because of the mobility of nodes.

Destination-sequenced distance vector (DSDV) is a proactive routing algorithm, whereas dynamic source routing (DSR) and adaptive on-demand vector (AODV) are reactive routing algorithms. Performance analysis has shown that DSR has less routing overhead as compared with AODV, but AODV works better in highly mobile environments.

In the next chapter, we shall study wireless sensor networks, which are like MANETs, because they are formed by mobile nodes, but are different from them because of their larger size and lower node mobility.

Problems

1. Is it possible to use circuit switching in ad hoc networks? Explain your answer.
2. Compare and contrast proactive and reactive routing mechanisms in ad hoc networks.
3. Differentiate between wireless cellular networks and ad hoc networks.
4. Consider the network shown below:



   Show how you can create a route from source node 6 to destination node 16, using the DSR algorithm.
5. Repeat Question 4 for the AODV algorithm.
6. Consider Figure 6.7(a). Give the routing table for source node D, showing for each destination, the next hop, distance in hops to it, and the active neighbours. If node G now moves away or is switched off, show how the routing table will change.
7. Indicate the purpose of 'Echo' and 'Hello' packets in routing.
8. Using a network simulator like NS2 or Qualnet, simulate the DSR and AODV routing algorithms and do a performance comparison of both. (Do not look ahead to Chapter 12, which gives details of this simulation.)
9. Mobile IP is also meant for mobile nodes. Can it be used to provide connectivity in ad hoc networks? Explain your answer.
10. Compare and contrast the routing mechanisms in mobile IP and AODV.

Multiple-choice questions

1. The destination-sequenced distance vector (DSDV) protocol can be viewed as which one of the following?
   (**a**) Reactive routing protocol
   (**b**) Proactive routing protocol
   (**c**) Hybrid routing protocol
   (**d**) Multicast routing protocol

2. Which one of the following is a type of MANET?
   (**a**) Personal area network
   (**b**) Body area network
   (**c**) Wireless LAN
   (**d**) All of the above

3. Which of the following is NOT true with respect to a MANET?
   (**a**) All the nodes in a MANET are free to move arbitrarily
   (**b**) Power consumption is a major issue in MANETs.
   (**c**) MANETs offer less mobility as compared to 802.11 Wi-Fi.
   (**d**) MANETs are more vulnerable to security threats than wired or 802.11 Wi-Fi networks.

4. Which one of the following is the memory requirement of destination-sequenced distance vector (DSDV) protocol?
   (**a**) O(n)
   (**b**) O(nlogn)
   (**c**) O(1)
   (**d**) $O(n^2)$

5. Which of the following is a reactive routing protocol for MANETs?
   (**a**) CSMA/CA
   (**b**) Dynamic source routing (DSR)
   (**c**) Link state routing protocol
   (**d**) DSDV

6. In the dynamic source routing (DSR) protocol, a route error (RERR) packet is sent during which of the following?
   (**a**) Route discovery
   (**b**) Route maintenance
   (**c**) Both of these
   (**d**) None of these

7. The major contribution to routing overhead in AODV comes from
   (**a**) RERR packets
   (**b**) RREP packets
   (**c**) RREQ packets
   (**d**) None of these

8. Which of the following fields is contained in the route request (RREQ) packet?
   (**a**) Destination IP address
   (**b**) Request ID
   (**c**) Source sequence number
   (**d**) All of the above

9. Which of the following is the correct order for the communicating range of the three types of networks?
   (**a**) PAN>WLAN>BAN
   (**b**) WLAN>BAN>PAN
   (**c**) WLAN>PAN>BAN
   (**d**) BAN>WLAN>PAN

10. Which one of the following statements is FALSE?
(**a**) AODV uses table-driven routing
(**b**) AODV outperforms DSR in highly mobile environments
(**c**) DSR has access to greater amount of routing information than AODV
(**d**) In DSR, the destination replies only once to the request arriving first and ignores multiple requests

Further reading

A.S. Tanenbaum (2005), *Computer Networks*, 4th ed. (New Delhi, India: Prentice Hall).

C. Cheng, R. Riley, S.P.R. Kumar and J.J. Garcia-Luna-Aceves (1989), 'A Loop-free Bellman Ford Routing Protocol without Bouncing Effect', in Proceedings of the ACM SIGCOMM'89, September, 224–237.

C.E. Perkins (ed) (2001), *Ad Hoc Networking* (Boston: Addison-Wesley).

C.E. Perkins, E.M. Royer and S.R. Das (2001), 'Ad Hoc On-demand Distance Vector (AODV) Routing', www.ietf.org/internet-drafts/draft-ietf-manet-aodv-09.txt (accessed June 2007).

C.E. Perkins and P. Bhagwat (1994), 'Highly Dynamic Destination Sequenced Distance Vector Routing Protocol (DSDV) for Mobile Computers', in Proceedings of the SIGCOMM'94 Conference on Communications Architectures, Protocols and Applications, August, pp. 234–244.

D.B. Johnson (1994), 'Routing in Ad Hoc Networks of Mobile Hosts', in Proceedings of the IEEE Workshop on Mobile Computing Systems and Applications.

D.B. Johnson, D.A. Maltz, Yih-Chun Hu and Jorjeta G. Jetcheva (2001), 'The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks', www.ietf.org/internet-drafts/draft-ietf-manet-dsr-05.txt (accessed June 2007).

D.P. Agarwal and Q. Zeng (2003), *Introduction to Wireless and Mobile Systems* (Singapore, Thompson Asia).

E. Royer and C.K. Toh (1999), 'A Review of Current Routing Protocols for Ad Hoc Mobile Wireless Networks', *IEEE Personal Communications*, 7(4): 46–55.

M. Mauve, H. Fubler, J Widmer and T. Lang (2003), 'Position-based Multicast Routing for Mobile Ad Hoc Networks', Technical Report TR03004, Department of Computer Science, University of Mannheim, Germany.

S.J. Lee, W. Su and M. Gerla (2000), 'On Demand Multicast Routing Protocol for Ad Hoc Networks', IETF Internet Draft, MANET Working Group.

V.D. Park and M.S. Corson (1997), 'A Highly Adaptive Distributed Routing Algorithm for Mobile and Wireless Networks', in Proceedings of the IEEE INFOCOM, 97, April, pp. 103–112.

V.D. Park and M.S. Corson (2001), 'Temporally-Ordered Routing Algorithm (TORA) Version 1 Functional Specification', www.ietf.org/internet-drafts/draft-ietf-manet-tora-spec-04.txt (accessed June 2007).

Wireless Sensor Networ ks 7

Advances in silicon technology have led to the development of next-generation, low-cost, low-power, multifunctional, sensor devices. These devices communicate wirelessly to transmit their readings. They are called wireless sensors and present a new facet in the field of communication and computer networks. Wireless sensors are compact devices that integrate communication, computation and microelectrical mechanical (MEMS) devices into a single chip.

A sensor network is a collection of communicating sensing devices or nodes. A large number of sensors can be spread across a geographical area and networked in many applications that require unattended operations, hence producing a wireless sensor network (WSN). The power of WSNs lies in the ability to deploy large numbers of such tiny sensor nodes. While the capability of any single device is minimal, the composition of hundreds of devices offers a significant opportunity for parallel, accurate and reliable data acquisition.

In 1999, Mark Weiser coined the term '**ubiquitous** or **pervasive computing**' to denote the kind of computing where computers become so small and so omnipresent that they fade into the background. WSNs are the basis for all pervasive computing. **Wearable computers** and **smart dust** are other synonyms for WSNs.

Unlike traditional wireless devices, wireless sensor nodes do not communicate directly with a base station, but rather operate in a peer-to-peer manner. The base station is usually a high-computing device, which aggregates data from multiple sensor nodes and processes them. All nodes in the network do not necessarily communicate at any particular time, and each node can only communicate with a few nearby nodes. Therefore, data collected by individual nodes is routed between the thousands of tiny sensor nodes in a multihop fashion until they reach the base station. See Figure 7.1 for a typical WSN connected to the Internet through a base station and transit network. The network has a routing protocol to control the routing of data messages between nodes. The routing protocol also attempts to get messages to the base station in an energy-efficient manner. Thus, sensor nodes act as routers as well as data originators.

7.1 Applications of wireless sensor networ ks

The applications of WSNs are innumerable, since each sensor node is capable of monitoring a wide variety of ambient conditions such as temperature, humidity, lightning condition, pressure and noise levels. Below, we have given some typical application areas for WSNs.

1. Military applications: Sensor nodes can be spread across a battlefield or enemy area and be programmed to track and monitor enemy troop movements or movement of terrorists and can

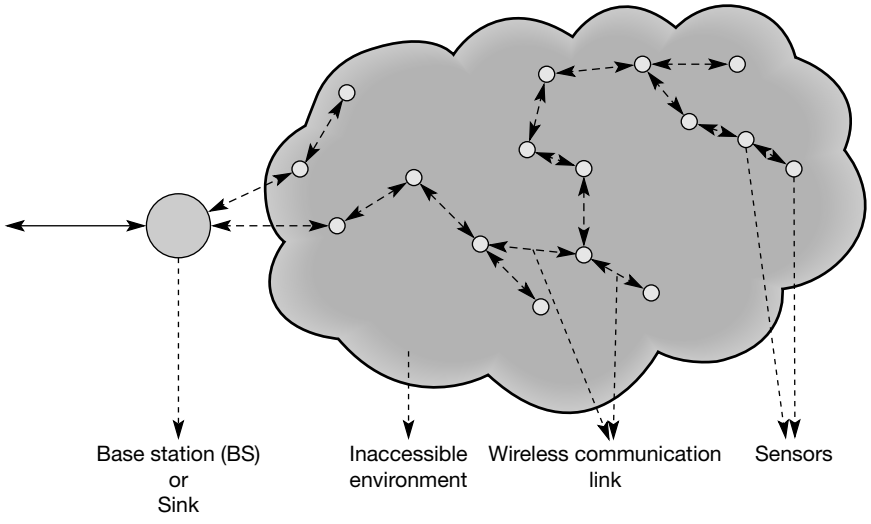


Figure 7.1 A Wireless Sensor Networ k

be used to locate their exact positions. They can also be used instead of mines to immediately detect any movement, thus safeguarding human lives.

Sensor networks can also be used to detect the use of biological or chemical weapons, and relay this information to commanders, so as to allow sufficient time for soldiers to take defensive measures in the field.

Improved battlefield communication is another benefit of WSN for military applications. Here, the soldier with a personal digital assistant (PDA) acquires an extended sense by interacting with the surrounding WSN. Target field imaging, security and tactical surveillance, and intrusion detection are other similar applications.

But military applications impose several special requirements on the WSN. Firstly, auto-deployment and self-organization should be supported. Secondly, detection of a sensor node should be difficult, otherwise an adversary may determine the location of sensor nodes easily, and compromise them. In such a situation, information with the compromised node can be stolen, or invalid information can be injected into the system.

Thus, secure communication is vital but may not be feasible in the form of large keys and robust protocols because of the limited computation power available. This is a major challenge in the deployment of WSN for military applications.

2. Environmental applications: Sensor networks are being increasingly used for environmental concerns. Examples include tracking the nesting habits of seabirds by monitoring a large geographic region with human presence, or attaching the sensors directly to large mammals to monitor their behaviour.

Monitoring of river currents is another application of WSNs, to measure their water inflow and mixture from various sources. Water-quality monitoring may also be useful to determine contamination with bacteria or other harmful pollutants.

A major application in this category is the spread of sensor nodes across a forest to monitor temperatures and give early warnings of fire outbreaks. Weather prediction, climate monitoring, distributed computing, pollution tracking, seismic detection, detecting ambient conditions such

as temperature, movement, sound, light, or the presence of certain objects, inventory control and disaster management are other similar applications.

The major requirement of environment sensors is the need for rugged operation in hostile surroundings and extended sleep periods to maximize the lifetime of the network.

3. Medical applications: WSNs can be used in medical applications by using the sensor node as a device that can reside on or within the human body and perform tasks that are currently done by costly machines. These include glucose monitors for continuous reading of insulin levels in diabetic patients; heart monitors for keeping track of the functioning of the heart, especially for patients with irregular heartbeats or coronary diseases; and artificial retinal and cortical implants to electronically transmit information to visually impaired persons.

Another example of the use of WSNs in medical applications is the vital statistics repository, which takes the form of a smart card holding medical information on persons with severe allergies to certain substances or medications.

Other medical applications include use in telemonitoring of human physiological data, tracking and monitoring doctors and patients inside a hospital and insurance cards.

Medical applications using sensors also have special requirements. They must be safe and biocompatible to merit continuous functioning inside the human body and not damage the tissues. The sensors must be designed for long-term operation and have enough power so that frequent surgery is avoided. This means that they must be highly fault-tolerant and provide redundancy and graceful degradation in failure scenarios.

4. Industrial applications: For use in industrial applications, low-cost sensor nodes could be attached to equipment to monitor performance. They could also be attached to parts as they move through an assembly pipeline on the shop floor. Thus, inefficiencies in plant process flow can be recognized quickly, rush orders could be expedited more easily and customer queries could be answered faster and more accurately.

Radio frequency identity (RFID) tags are already being placed on items like gasoline and other merchandise to allow fast and accurate scanning at checkout and for inventory tracking. These tags could be replaced by wireless sensors at fixed locations and used for tracking.

The use of sensor nodes for industrial and commercial applications requires their cost to be made very low so that they can be used in bulk. The protocols in use should, therefore, also be highly scalable.

5. Urban applications: WSNs can be used for various urban applications like transportation and traffic systems, auto-identification by driving license, parking availability, security monitors in shopping malls, parking garages, city streets and home security.

7.2 Differences from mobile ad hoc networks

In Chapter 6, we studied mobile ad hoc networks (MANETs), which are also made up of a number of wireless, mobile nodes. However, there are significant differences between MANETs and WSNs. These are as follows:

1. The number of sensor nodes in a sensor network is much more than that in an ad hoc network. Usually sensor networks consist of 1,000 to 10,000 sensor nodes covering the area.
2. Sensor nodes are generally static and cooperate together to transfer the sensed data.
3. In mobile ad hoc networks, the number of nodes is much less, but their mobility is very high.
4. Sensor nodes mainly use the broadcast communication paradigm, whereas most ad hoc networks are based on point-to-point communication.

5. Another difference between the two is that sensor nodes have a much lower power consumption requirement, of the order of 0.75 mW.

7.3 Design issues

There are many technical issues that need to be considered for the design of an effective WSN. These issues mainly include the following:

- **Low power consumption:** Sensor nodes are equipped with a battery to provide them with power. Therefore, protocols and algorithms designed for sensor networks should utilize minimal energy.
- **Low cost:** Most of the sensor networks applications require hundreds or thousands of nodes to be deployed. To correspond to that volume of production, the cost of the network components must be minimized.
- **Security:** Due to the wireless nature of transmission and the unattended nature of network, deployment, it may be easy for an adversary to attack the network. At the same time, the security features must be capable of implementation with inexpensive hardware and minimal resources.
- **Data throughput:** Sensor networks have limited data throughput that implies low communication efficiency, especially with any protocol overhead added to the packets. Therefore, communication protocols for sensor networks must be kept as simple as possible.

7.4 WSN architecture

Figure 7.2 shows the **general architecture** of a WSN. Sensor nodes, making up a sensor field, sense the data and route it to a sink. The sink is a special sensor node connected to a personal computer, which is capable of transferring the data over the Internet.

Alternatively, the sensor nodes may be arranged in a **hierarchical fashion**, with many nodes that are close to each other forming a cluster. Each cluster has a cluster head. See Figure 7.3. Clustering goes on at each level, till the base station (sink) is reached.

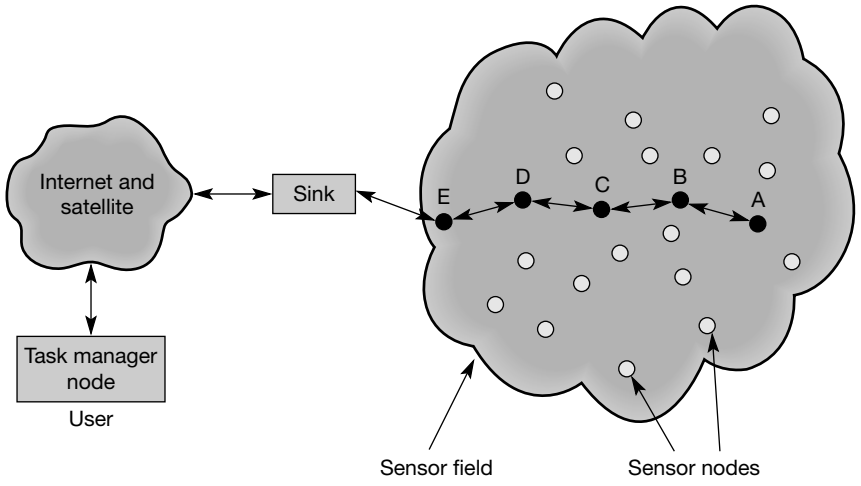


Figure 7.2 General Architecture of a Wireless Sensor Networ k

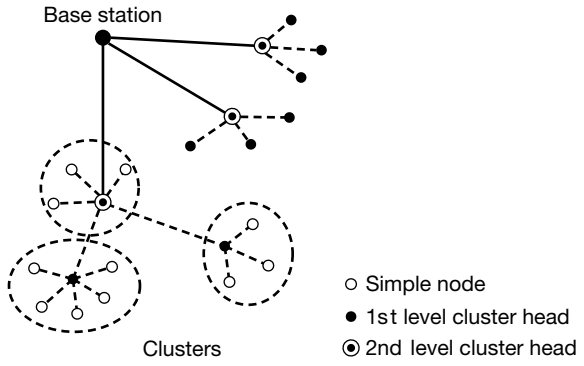


Figure 7.3 Cluster-based Hierarchical Architecture of WSNs

7.4.1 Sensor hardware components

The sensor node's hardware consists of five components—sensing hardware, processor, memory, power supply and transceiver, as shown in Figure 7.4. The sensing unit consists of a sensor which senses any physical attribute, like light, temperature, pressure, humidity, vibrations, etc., and converts it into an analog signal, which can be digitized using the A-D converter (ADC). This data is suitably processed by the processing unit, which consists of a processor and storage system. The transceiver consists of a transmitter and receiver which send and receive data.

A location-finding system is optionally a part of the sensor node and, together with the mobilizer, helps to give the location of the node, for example through a GPS system. This is useful in location-based protocols. The power unit (optional) powers the various units in the node.

7.4.2 WSN communications architecture

The WSN communication architecture is shown in Figure 7.5. It is a three-dimensional (3-D) architecture, in the sense that there are three **planes**, each of which helps to manage the usual five layers, namely, physical layer, data link layer, network layer, transport layer and application layer in its plane.

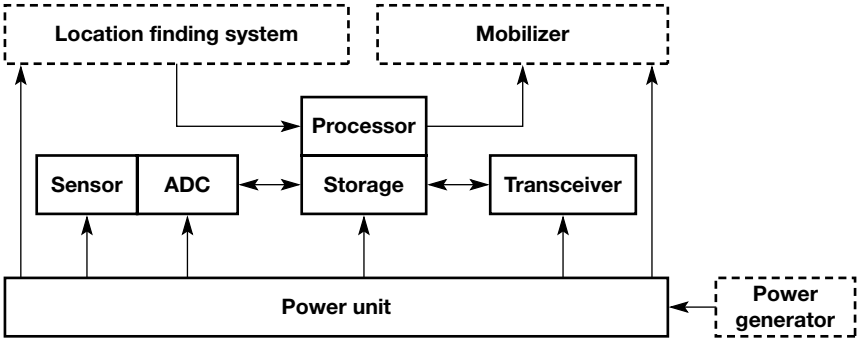


Figure 7.4 Hardware Components of a Sensor Node

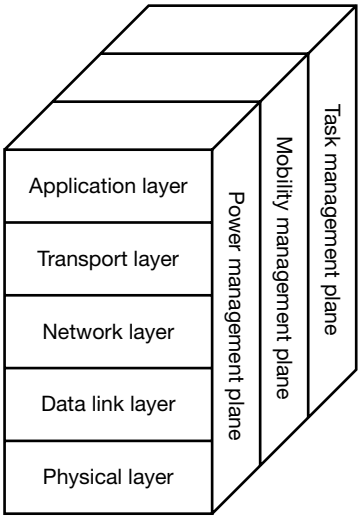


Figure 7.5 WSN Communications Architecture

The **power management plane** manages how a sensor node uses its power across all layers, since power efficiency is an essential constraint in WSNs. For example, a sensor node may turn off its receiver after receiving a message.

In some applications, sensor nodes may be mobile. In such applications, the **mobility management plane** manages mobility of nodes by detecting and registering their movement, so that a route back to the user is always maintained.

The **task management plane** balances and schedules the sensing tasks given to a specific region, since not all sensor nodes in that region are required to perform the sensing task at the same time.

7.5 Routing protocols for WSN

Many routing protocols have been proposed for WSNs. These have been classified into three categories, namely, data-centric protocols, hierarchical protocols and location-based protocols.

Data-centric protocols are query-based and use the concept of naming of desired data to eliminate many redundant transmissions. **Hierarchical protocols** cluster the nodes so that cluster heads can aggregate and reduce the data to save energy. **Location-based protocols** use position information to send the data to only the desired regions rather than to the whole network.

Some of these protocols are discussed briefly below. For details, the reader is referred to the concerned literature.

7.5.1 Data-centric protocols

Data-centric routing is different from traditional address-based routing, where routes are created between addressable nodes at the network layer. Here, the sink sends queries to certain regions and waits for data from the sensors located there. Since data are requested through queries, attribute-based naming is necessary to specify the properties of data. The following routing protocols belong to this category.

7.5.1.1 Flooding and gossiping

Flooding and gossiping are two classical mechanisms to transmit data in sensor networks without the need for any routing algorithms and topology maintenance. In **flooding**, each sensor that receives a data packet and broadcasts it to all its neighbours, and this process continues until the packet arrives at the destination, or the maximum number of hops for the packet is reached. **Gossiping** is an enhanced version of flooding, where the receiving node sends the packet only to a randomly selected neighbour, which picks another random neighbour to forward the packet to, and so on.

Although flooding is very easy to implement, it has several drawbacks. These include **implosion** caused by duplicate messages sent to same node, **overlap** caused when two nodes in the same region send similar packets to the same neighbour and **resource blindness** by consuming large amount of energy. Gossiping avoids the problem of implosion by selecting a random node to send the packet rather than broadcasting. However, this increases delays in the propagation of data through the nodes.

7.5.1.2 Sensor protocols for information via negotiation (SPIN)

The SPIN family of protocols is designed with the idea that sensor nodes operate more efficiently and conserve energy by sending data. SPIN can be used to efficiently disseminate information in a WSN. Conventional data dissemination approaches like flooding and gossiping waste valuable communication and energy resources by sending redundant information throughout the network. In addition, these protocols are not resource-aware or resource-adaptive. SPIN solves these shortcomings by using data negotiation and resource-adaptive algorithms.

Nodes in SPIN assign a high-level name to their data, called **metadata**, and perform metadata negotiations before any data is transmitted. This ensures that there is no redundant data sent throughout the network. SPIN also keeps track of the current energy level of the node and adapts the protocol it is running accordingly. Simulation results show that SPIN is more energy-efficient than flooding or gossiping. It also distributes data at the same or faster rate than either of these protocols.

SPIN has three types of messages—ADV (advertisement), REQ (request) and DATA (data). Before sending a DATA message, the sensor broadcasts an ADV message containing a descriptor. If a neighbour is interested in the data, it sends a REQ message for the DATA, which is then sent to it. See Figure 7.6. The neighbour sensor node then repeats this process. As a result, all the sensor nodes interested in the data get a copy.

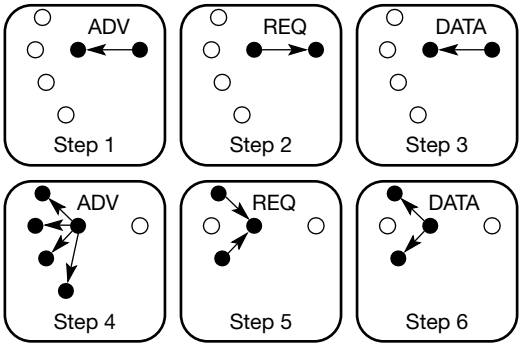


Figure 7.6 SPIN Protocol

SPIN has the following advantages:

1. Topological changes are localized, since each node needs to know only its single-hop neighbours.
2. As compared to flooding, it reduces energy dissipation by a factor of 3.5.
3. Metadata negotiation results in almost halving the redundant data.

However, the disadvantage of SPIN is that its data advertisement mechanism cannot guarantee delivery of data. If the nodes that are interested in the data are far away from the source node and the nodes between source and destination are not interested in that data, such data will not be delivered to the destination at all.

Therefore, SPIN cannot be used in applications such as intrusion detection, where reliable delivery of data packets is required at regular intervals.

7.5.2 Hierarchical protocols

As in traditional computer networks, scalability is a major design issue in sensor networks. A single-tier network can cause the gateway to be overloaded with an increase in number of sensors. This results in latency in communication and inadequate tracking of events. Thus, the single-gateway architecture is not scalable for a larger set of sensors, covering a wider physical area, since the sensors are typically not capable of long-haul communication. To allow the system to cope with additional load and to be able to cover a large wider physical area without degrading service, clustering has been used in some routing approaches.

The main aim of hierarchical routing is to efficiently reduce the energy consumption of sensor nodes by using multihop communication within a cluster. By performing data aggregation and fusion, the number of transmitted messages to the sink decreases. Cluster formation is based on the energy reserve of sensors and the sensor's proximity to the cluster head. Low-energy adaptive clustering hierarchy (LEACH) is one of the first hierarchical routing approaches developed for sensor networks.

7.5.2.1 Low-energy adaptive clustering hierarchy

LEACH is designed for sensor networks where an end user wants to remotely monitor the environment. The data from the individual nodes must be sent to a central base station, which may be located far from the sensor network and through which the end user can access the data. There are several desirable properties for protocols on such networks:

- They should use hundreds and thousands of nodes
- They should maximize system lifetime and network coverage
- They should use uniform, battery-operated nodes

Conventional network protocols, such as direct transmission, minimum transmission energy, multihop routing and clustering, all have drawbacks that do not allow them to achieve all these desirable properties. LEACH includes distributed cluster formation, local processing to reduce global communication and randomized rotation of the cluster heads. These features allow LEACH to achieve the desired properties. Simulations show that LEACH is an energy-efficient protocol that extends system lifetime.

LEACH is a clustering-based protocol that minimizes energy dissipation in sensor networks. Its operation is separated into two phases, the setup phase and the steady phase. In the setup phase, a group of sensor nodes selects a cluster head. During the steady phase, the sensor nodes

can begin sensing and transmitting data to their cluster head. The cluster heads aggregate data from the nodes in their cluster before sending them to the base station.

LEACH achieves over a factor of 7 reduction in energy dissipation compared to direct communication and a factor of 4–8 compared to the minimum transmission energy routing protocols. The nodes die randomly, and dynamic clustering increases the lifetime of the system. LEACH is completely distributed and requires no global knowledge of the network.

However, since it uses single-hop routing where each node can transmit directly to the cluster head and the sink, it is not applicable to large networks. Furthermore, dynamic clustering brings extra overhead, for example, head changes and advertisements, which reduce the gain in energy consumption. For more details of LEACH, the reader is referred to Heinzelman, Chandrakasan and Balakrishnan (2000).

7.5.2.2 PEGASIS

Power-efficient gathering in sensor information system (PEGASIS) is an improvement on LEACH. Instead of forming multiple clusters, PEGASIS forms **chains** from sensor nodes. Each node transmits and receives data from a neighbour, and only one node is selected from that chain to transmit to the base station (sink). Gathered data move from node to node, are aggregated and are eventually sent to the base station. The chain construction is performed in a greedy way.

Unlike LEACH, PEGASIS uses multihop routing by forming chains and by selecting only one node to transmit to the base station, instead of using multiple nodes. PEGASIS has been shown to outperform LEACH by about 100 per cent to 300 per cent for different network sizes and topologies. This is due to the elimination of the overhead caused by dynamic cluster formation in LEACH and by decreasing the number of transmissions and reception by using data aggregation.

However, PEGASIS introduces large delay for a far-off node in the chain. Also, the presence of a single leader can become a bottleneck.

7.5.2.3 TEEN and APTEEN

Threshold sensitive Energy Efficient sensor Network protocol (TEEN) is a hierarchical protocol designed to respond to sudden changes in the sensed attributes, such as temperature. Fast response is important for time-critical applications, in which the network operates in a reactive mode. TEEN uses a hierarchical approach along with the use of a datacentric mechanism. The sensor network architecture is based on a hierarchical grouping, where closer nodes form clusters, and this process goes on until the base station (sink) is reached.

After clustering is done, the cluster head broadcasts two thresholds to the nodes—the hard and soft thresholds for sensed attributes. The hard threshold is the minimum possible value of an attribute to trigger a sensor node to switch on its transmitter and transmit to the cluster head. It allows the nodes to transmit only when the sensed attribute is in the range of interest. This reduces the number of transmissions significantly. Once a node senses a value at or beyond the hard threshold, it transmits data only when the value changes by an amount equal to or greater than the soft threshold. As a result, the soft threshold further reduces the number of transmissions if there is little or no change in the value of sensed attribute. Both the hard and soft threshold values can be adjusted to control the number of packet transmissions.

However, TEEN is not good for applications where periodic reports are needed, since the user may not get any data at all if the thresholds are not reached.

The Adaptive Threshold sensitive Energy Efficient sensor Network protocol (APTEEN) is an extension to TEEN that captures periodic data collections and reacts to time-critical events. Details may be seen in Manjeshwar and Agrawal (2002).

7.5.3 Location-based protocols

Most routing protocols for sensor networks require location information for the nodes in order to calculate the distance between two particular nodes, so that energy consumption can be estimated. Since there is no addressing scheme for sensor networks like Internet protocol (IP) addresses and they are spatially deployed in a region, location information can be utilized in routing data in an energy-efficient way. For instance, if the region to be sensed is known, using the location of sensors, the query can be diffused only to that particular region. This will eliminate the number of transmissions significantly.

Some of these protocols were designed for mobile ad hoc networks, taking into consideration the mobility of nodes, but are also applicable to sensor networks with less or no mobility. These are the geographic adaptive fidelity (GAF), minimum energy communication network (MECN) and small minimum energy communication network (SMECN) protocols. We leave details of these to the interested reader.

7.6 Case study

In this section, we shall discuss a typical, commercially available, wireless sensor node kit, the **Mica mote**, its operating system TinyOS (tiny microthreading operating system) and its use in a typical application.

However, please note that the latest version of the WSN from Crossbow Systems is the Professional kit WSN-PRO 2400CA, based on a 2.4 Ghz processor. It has six packaged sensor nodes with MICA2 processor/radio board and light, temperature, humidity, barometric pressure and seismic sensor boards; one packaged Univeral System Bus (USB) base station/gateway; one unpackaged mote processor/radio board; one data acquisition board and housing; one USB programming board; and is preprogrammed with XMesh applications for out-of-the-box WSN experience.

7.6.1 The MICA mote

Current research in wireless sensor technology is geared towards the development of a hardware and software platform that combines sensing, communication and computing into a complete architecture. Crossbow Systems has spearheaded one such open source development in collaboration with researchers at the University of California, Berkeley (UC Berkeley). The first commercial generation of this platform was dubbed the Rene Mote, and the most recent sensor node is MICA2 mote.

The basic MICA2 hardware available commercially is a square inch in size and consumes a fraction of a watt of power. A Smart Dust mote is an electronic package composed of an integrated-circuit radio transmitter and receiver, microcontroller, random access memory (RAM), flash memory, standard sensors, analog-to-digital converter, power source and an antenna. The MOTE-KIT5040 is a combination eight-node kit for mote development (see Figure 7.7). It features Crossbow's latest generation MICA2 and MICA2DOT motes. Both motes are compatible with TinyOS. The kit includes the following:

1. Four MICA2 processor/radio boards (MPR300CB)
2. Four MICA2DOT quarter-sized processor radio boards

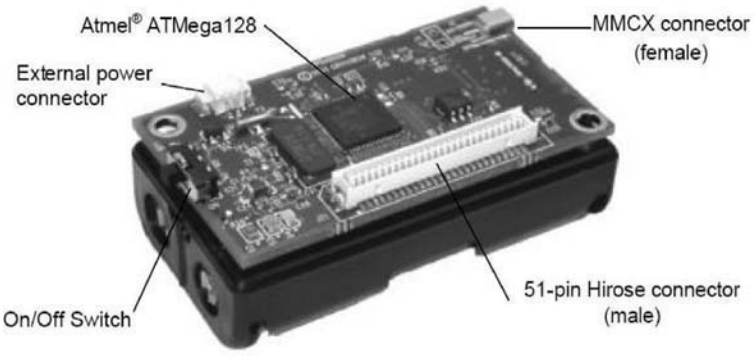


Figure 7.7 MICA2 Mote

3. Three MTS310 sensor boards (acceleration, magnetic, light, temperature, acoustic and sounder)
4. Two MDA500, MICA2DOT prototype and data acquisition boards
5. One MIB510 programming board
6. Serial interface board
7. Mote-view and mote-test software

A typical Mica mote uses a processor and radio board (MPR300CB), which is based on the Atmel Atmega 128L low-power microcontroller. Specifications of a typical Mica mote are given in Table 7.1.

7.6.2 TinyOS

The Mica mote uses a special operating system developed at UC Berkeley for sensor networks, called TinyOS. All software for the motes is developed on the TinyOS platform. TinyOS is an efficient and modular embedded software platform designed explicitly for networked sensor motes. It draws strongly from previous architectural work on lightweight thread support and

Table 7.1 Specifications of the Mica Mote

Processor/radio board	MPR300CB
Speed	44 MHz
Flash	128 KB
SRAM	4 KB
EEPROM	4 KB
Radio frequency	916 –433 MHz (ISM band)
Data rate	40 Kbits/sec max.
Power	0.75 mW
Radio range	100 ft, programmable
Power source	2 × AA batteries

efficient network interfaces. It has been developed on an open source software platform by UC Berkeley, with active support from a large community of users.

TinyOS is a component-based runtime environment designed to provide support for deeply embedded systems that require intensive concurrent operations while being constrained by minimal hardware resources, like size, power and efficiency. The TinyOS system, libraries and applications are written in NesC (network embedded systems C) language. NesC has a C-like syntax and is an extension to C designed to embody the structuring concepts and execution model of TinyOS.

Programs in NesC are built out of components, which are assembled (wired) to form whole programs. Components define two scopes, one for specification (containing the names of their interface instances) and one for their implementation. Components have internal concurrency in the form of tasks. Interfaces are bidirectional, that is, they specify a set of functions to be implemented by the interface's provider (commands) and a set to be implemented by the interface's user (events). This allows a single interface to represent a complex interaction between components (e.g. registration of interest in some event, followed by a callback when that event happens).

The machine to which the MIB is connected will contain the TonyOS code, that is, the code for the components. Application code is written using the component code that is already present in TinyOS. The NesC compiler is used to compile the program, whose output exe file is used to program the Mote's flash.

TinyOS maintains a two-level scheduling structure, so a small amount of processing associated with hardware events can be performed immediately while long-running tasks are interrupted. In TinyOS, each system module is designed to operate by continually responding to incoming events. When an event arrives, it brings the required execution context with it. When the event processing is completed, it is returned to the system.

7.7 Development wor k in WSN

Many of the author's undergraduate and postgraduate students at IIT Roorkee have worked on the hardware and software kits mentioned above. We give some details of each of these below.

In the work entitled 'Dynamic Keying in Tinysec: Development of Security Architecture for Wireless Sensor Networks', a new security architecture has been proposed for WSNs. The existing TinySec algorithm has been modified to incorporate the feature of a dynamically changing key. The proposed algorithm does intrusion detection and provides security against DOS (denial of service) attacks. TinyOS 1.0 has also been modified, and the required modules and interfaces have been added for implementing dynamic key change.

In another thesis, entitled 'Implementation and Performance Evaluation of Energy Efficient Routing for Wireless Sensor Networks', work has been carried out to develop an energy efficient multihop routing protocol for WSNs. This protocol is based on the Pulse protocol. It has been verified that the new protocol is more energy efficient than the pulse protocol and can be used for different types of applications.

The Mica mote kit has also been used successfully in a developmental project for designing a system to aid visually handicapped persons. This is discussed in detail in Chapter 12.

7.8 Summary

Wireless sensor networks (WSNs) consist of thousands of small computing nodes or **motes** equipped with radio transceivers and sensors, and are the basis of pervasive and ubiquitous computing. They can be quickly and easily deployed in many applications, ranging from military to

environmental. They differ from MANETs in that the number of nodes is much larger and the computing power is much lesser.

Many routing algorithms have been designed for WSNs. These can be divided into datacentric protocols, hierarchical protocols and protocols based on location. The more important ones among these are SPIN, LEACH, PEGASIS, TEEN and APTEEN.

The Mica mote kit was one of the earliest WSN nodes developed and has been usefully employed and tested for such diverse applications as aiding the movement of visually handicapped persons and for sensing and reporting fires. TinyOS is an example of a real-time operating system that has been specifically designed for such WSN nodes.

In the next chapter, we will discuss the handheld devices used in mobile computing, their characteristics and their underlying operating systems.

Problems

1. Explain clearly the differences between WSNs and other types of ad hoc networks.
2. Many applications have been given in the chapter for WSNs. Choose any given application and identify the design of the WSN for it.
3. For the above design, list the important security issues to be taken care of.
4. Compare and contrast the use and benefit of RFIDs vs. WSNs in a typical industrial environment.
5. Can the topology of a WSN change, even if there is no mobility? Discuss your answer.
6. Wireless sensor nodes are being increasingly deployed in the human body. What are the special security requirements of such biomedical sensors?
7. The energy consumed by different functions in a sensor is not the same. List the typical amounts of such consumption, for functions like aggregation, communication to cluster head, cluster head to BS, sleeping mode, sensing mode, etc.
8. Draw a table to compare and contrast the various routing algorithms used in WSN. List the items in terms of the parameters used for comparison.
9. Download the TinyOS or any other open source operating system available for WSN and implement and test a simple protocol with it using a simulator.
10. Suppose that a nuclear plant building of size 20×20 sq. metres and height 10 metres is to be fitted with wireless sensors. If the sensors have a transmit/receive range of 2 metres, what would be an efficient arrangement of the sensors? Discuss.

Multiple-choice questions

1. **Smart dust** is a synonym for which of the following?
   (a) Wireless sensor networks
   (b) Mobile ad hoc networks
   (c) Wearable computers
   (d) Both (a) and (c)
2. The kind of computing where computers become so small and so omnipresent that they fade into the background is known as
   (a) Soft computing
   (b) Pervasive computing

(c) Hard computing
(d) None of the above

3. Which one of the following is true for the statements X and Y?
X: Sensor nodes mainly use the broadcast communication paradigm
Y: Most ad hoc networks are based on point-to-point communication
(a) X is true but Y is false
(b) X is false but Y is true
(c) Both X and Y are true
(d) Both X and Y are false

4. Which of the following layers is not present in WSN (wireless sensor network) communications architecture?
(a) Physical layer
(b) Data link layer
(c) Presentation layer
(d) Application layer

5. Which one of the following balances and schedules the sensing tasks given to a specific region?
(a) Power management plane
(b) Task management plane
(c) Mobility management plane
(d) Balance management plan

6. Which of the following protocols are query-based and depend on the naming of desired data, which helps in eliminating many redundant transmissions?
(a) Data-centric protocols
(b) Hierarchical protocols
(c) Location-based protocols
(d) None of the above

7. A process where the receiving node sends the packet to a randomly selected neighbour, which picks another random neighbour to forward the packet to, and so on is known as
(a) Flooding
(b) SPIN
(c) LEACH
(d) Gossiping

8. Which one of the following algorithms avoids the problem of implosion by just selecting a random node to send the packet rather than broadcasting?
(a) Flooding
(b) LEACH
(c) Gossiping
(d) SPIN

9. Which one of the following is a hierarchical protocol designed to respond to sudden changes in the sensed attributes, such as temperature?
(a) LEACH
(b) PEGASIS
(c) TEEN
(d) Gossiping

10. The TinyOS system, libraries and applications are written in which one of the following languages?
(a) NesC
(b) C++
(c) Java
(d) C

Further reading

A. Buczak and V. Jamalabad (1998), 'Self-Organization of a Heterogeneous Sensor Network by Genetic Algorithms', in C.H. Dagli, S.R.T. Kumara and Y.C. Shin (eds), *Intelligent Engineering Systems through Artificial Neural Networks*, vol. 8 (New York: ASME Press), pp. 259–264.

A. Manjeshwar and D.P. Agrawal (2002), 'APTEEN: A Hybrid Protocol for Efficient Routing and Comprehensive Information Retrieval in Wireless Sensor Networks', in Proceedings of the 2nd International Workshop on Parallel and Distributed Computing Issues in Wireless Networks and Mobile Computing, Ft. Lauderdale, FL, April.

A. Perrig, R. Szewczyk, V. Wen, D. Culler and J.D. Tygar (2001), 'SPINS: Security Protocols for Sensor Networks', in Proceedings of Mobile Networking and Computing 2001.

A. Wolisz, H. Karl and A. Willig (eds) (2004), 'Wireless Sensor Networks', in Proceedings of 1st European Workshop, EWSN 2004, LNCS 2920 (Springer).

A.M. Mainwaring, D.E. Culler, J. polastre, R. Szewczyk and J. Anderson (2002), 'Wireless Sensor Networks for Habitat Monitoring', in ACM International Workshop on WSNs and Applications (WSNA'02), pp. 88–97.

Articles on sensors at www.sensormag.com/articles/ (accessed March 2007).

B.V. Prasad (2005), 'Dynamic Keying in Tinysec—Security Architecture for WSNs', M.Tech. thesis, IIT Roorkee.

C.R. Lin and M. Gerla (1997), 'Adaptive Clustering for Mobile Wireless Networks', *IEEE Journal on Selected Areas in Communications,* 15(7): 1265–1275.

F. Adelstein, S.K.S. Gupta, G.G. Richard III and L. Schwiebert (eds) (2005), *Fundamentals of Mobile and Pervasive Computing* (New Delhi, India: Tata McGraw-Hill).

I. Akyildiz, W. Su, Y. Sankarasubramaniam and E. Cayrici (2002), 'A Survey on Sensor Networks', *IEEE Communications Magazine*, 40(8): 102–105.

K. Akkaya and M. Younis (2005), 'A Survey on Routing Protocols for WSNs', *Elsevier Ad Hoc Network*, 3(3): 325–349.

L. Li and J.Y. Halpern (2001), 'Minimum Energy Mobile Wireless Networks Revisited', in Proceedings of IEEE International Conference on Communications (ICC01), Helsinki, Finland, June.

M. Bagli (2005), 'Performance Evaluation of Energy Efficient Routing for WSNs', M.Tech. thesis, IIT Roorkee.

M. Tubaishat and S. Madria (2003), 'Sensor Networks: An Overview', *IEEE Potentials*, 22(2): 20–23.

M. Weiser (1991), 'The Computer for the Twenty-First Century', *Scientific American*, 265(3): 94–110.

M. Weiser (1993), 'Hot Topics: Ubiquitous Computing', *IEEE Computer*, (October): 71–73.

P. Gupta and A. Chaturvedi (2006), 'Dhristi—Indoor Orientation for the Visually Impaired', B.Tech project report, IIT Roorkee.

Q. Jiang and D. Manivannan (2004), 'Routing Protocols for Sensor Networks', in IEEE Proceedings on Computers.

S. Hedetniemi and A. Liestman (1988), 'A Survey of Gossiping and Broadcasting in Communication Networks', *Networks,* 18(4): 319–349.

S. Lindsey and C.S. Raghavendra (1997), 'PEGASIS: Power Efficient Gathering in Sensor Information Systems', in Proceedings of the IEEE Aerospace Conference, Big Sky, Montana, March.

TinyOS documentation at http://today.cs.berkeley.edu/tos/ (accessed March 2006).

V. Rodoplu and T.H. Ming (1999), 'Minimum Energy Mobile Wireless Networks', *IEEE Journal of Selected Areas in Communications,* 17(8): 1333–1344.

W. Heinzelman, A. Chandrakasan and H. Balakrishnan (2000), 'Energy-efficient Communication Protocol for Wireless Sensor Networks', in Proceedings of the Hawaii International Conference System Sciences, Hawaii, January.

W. Heinzelman, J. Kulik and H. Balakrishnan (1999), 'Adaptive Protocols for Information Dissemination in Wireless Sensor Networks', in Proceedings of the 5th Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom99), Seattle, WA, August.

www.xbow.com/generalinfo (accessed April 2006).

Y. Xu, J. Heidemann and D. Estrin (2001), 'Geography-Informed Energy Conservation for Ad Hoc Routing', in Proceedings of the 7th Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom01), Rome, Italy, July.

Mobile Handheld Devices 8

Today, handheld devices or personal digital assistants (PDAs), also called palmtops or Pocket PCs, which are all small-sized extensions of desktops, comprise the largest group of Internet-connectable, pervasive devices available in the market. Their number is growing by the day, and many vendors now supply a variety of these devices. These vendors include, among others, Palm, Sony, Compaq, Dell, Toshiba, Motorola, Nokia, HP, Hyundai, etc. The market trend is today shifting rapidly from low-end devices to those with some form of wireless connectivity, whether for data or voice or both.

In this chapter, we shall dwell on the hardware aspects of mobile computing and discuss the types and characteristics of various handheld devices. We shall also see how the operating systems for these devices are different from those of desktop and laptop computers, and typically how the small available memory in them is managed.

8.1 Characteristics of PD As

All PDAs exhibit certain common characteristics, which make them small and lightweight, so that they easily fit into pockets. All have a stylus that is used to tap on the screens to activate applications, and most are simple, expandable, connected, fast and mobile. As a result of fast-changing technology and requirements, most PDAs in use tend to become obsolete very soon. Figure 8.1 shows a typical state-of-the-art PDA from Palm Inc.

Figure 8.1 The Palmone Tungsten E2 PD A

Table 8.1 Comparison of some handheld computer s

Device Class	EPOC	Palm	Pocket PC
A sample product	Psion Series 5mx	Palm i705	Casio E-200
Year	1999	2001	2002
Operating system	EPOC 32 bit	Palm OS 4.1	Pocket PC 2002
Size	17 × 9 × 2 cm	8 × 12 × 2 cm	13 × 7.8 × 1.6 cm
Weight	350 g	140–160 g	255 g
Data entry	Keyboard	Stylus	Stylus
Display size	12–14 cm	7.8 cm	21–25 cm
Display resolution	640 × 240 pixels Monochrome Touch screen	160 × 160 pixels Monochrome/256 colours, Touch screen	320 × 240 pixels 64 K colour s Touch screen
RAM	16 MB	8 MB	64 MB
Processor	32-bit RISC ARM710T 36 MHz	Motorola Dragonball VZ33 33 MHz	Intel Strong ARM^ 1110, 206 MHz
Peripherals	Serial por t, Infrared Compact flash Video recording	Serial Infrared Extension card slot	Serial, USB Infrared Compact flash, Voice recording , Audio speaker
Battery charge	> 10 hours	> 10 hours	> 12 hours
Applications	PIM*, e-mail Internet browser Database, Spreadsheet WAP+ (MC218 only)	PIM, e-mail HotSync, Graffiti	PIM, Active sync Handwriting recognition Internet Explorer Office Suite, Pocket Outlook Media player
Cost	$300	$80	$110

*Personal infor mation manager
+Wireless access protocol
^Advanced reduced instr uction set computer (RISC) machine, details in Section 8.1.1

See Table 8.1 for a list of the comparative features of PDAs that have been in operation in the past decade. We will discuss later in the chapter three of the latest and more popular PDAs, the Palm TX, the HP ipaq rx 4240 and the Nokia 9210.

It is obvious from Table 8.1 that, because of their small size, PDAs have several limitations. These are as follows:

1. Limited memory for running applications, typically 16 MB RAM
2. Less storage space, varying between 16 MB and 64 MB, with extension slots
3. Small battery power, lasting a few hours
4. Limited processing power, because slow processors are used, as mentioned in Chapter 1
5. Small screen, about 6 cm on a side, with a resolution of 320 × 320 pixels.

8.1.1 The ARM processor

Because the ARM processors are the most widely used processors in the handheld industry today, it is appropriate to give some details about their architecture and characteristics here.

The **ARM** architecture (previously, the **Advanced RISC Machine**, and prior to that **Acorn RISC Machine**) is a 32-bit RISC processor architecture developed by ARM Limited that is widely used in embedded designs. Because of their power-saving features, ARM CPUs are dominant in the mobile electronics market, where low power consumption is a critical design goal.

Today, the ARM family accounts for approximately 75 per cent of all embedded 32-bit RISC CPUs, making it one of the most widely used 32-bit architectures. ARM CPUs are found in most consumer electronics, including portable devices like PDAs, mobile phones, media players, handheld gaming units, and calculators, and computer peripherals like hard drives, desktop routers, etc.

The ARM design was started in 1983 as a development project at Acorn Computers Ltd. to build a compact RISC CPU. A key design goal was to achieve low-latency input/output (interrupt) handling used in Acorn's existing computer designs. The memory access architecture allowed developers to produce fast machines without the use of costly direct memory access hardware. The development samples called **ARM1** arrived in April 1985, and the first 'real' production systems **ARM2** arrived in 1986.

The ARM2 had a 32-bit data bus, a 32-bit (4 GB) address space and sixteen 32-bit registers. Program code had to lie within the first 64 MB of the memory, as the program counter was limited to 26 bits, using the top 6 bits of the 32-bit register as status flags. The ARM2 was considered the simplest useful 32-bit microprocessor, with only 30,000 transistors (compared with Motorola's older 68000 model, which had around 70,000 transistors). This simplicity was due to not having microcode and not including any cache. This resulted in its low power usage, while performing better than the Intel 80286. A successor, **ARM3**, was produced with a 4 KB cache, which further improved performance.

In the late 1980s, newer versions of the ARM core were developed, resulting in the **ARM6** released in 1991. DEC Systems licensed the ARM6 architecture and produced the **StrongARM**. At 233 MHz, this CPU drew only 1 watt of power and had 35,000 transistors.

The most successful implementation has been the ARM7TDMI, which is present in most microcontroller-equipped devices. The original design manufacturer combines the ARM core with a number of optional parts to produce a complete CPU, which gives good performance at low cost. As of January 2008, over 10 billion ARM cores have been built.

8.1.2 Network connectivity

PDAs can be connected to a wired network through either a direct wired connection, or a connection via a cradle, or else through a wireless connection, which is the most common.

Wireless connectivity to the PDA can be of three types. It could be through Wi-Fi–IEEE 802.11b connection, or through a cellular CDMA or GSM/GPRS connection, or through a Bluetooth connection. With the first two, it is mobile wireless networking, but with the third one, keyboards and cables are eliminated altogether. All three technologies have already been discussed in earlier chapters.

In this chapter, we shall discuss in detail two most recent PDAs available in the market today. These are from Palm Systems and Hewlett-Packard. We shall also look at the details of the operating systems that are used with these handhelds, namely, PalmOS, version 5.4, Symbian v. 9.25 and Windows CE (compact edition), which is the basis of the more recent Windows Mobile, versions 5 and 6.

8.2 Palm handhelds

Palm Computing Inc., a subsidiary of 3COM, manufactures PalmOS-based devices, together with companies like Symbol, IBM, Sony, Handspring, Acer, Alphasmart and Kyocera, which have licensed the Palm OS operating system.

In 1998, the Palm III was launched, which had 2 MB of memory, infrared-based communication and ran the Palm OS operating system. This was followed by the PalmIIIx, which had 4 MB of memory. Subsequently, the Palm m100, m125, m130, m500 and m515 were made with 8 MB of memory. Palm i705 has Internet and e-mail capabilities, and Palm m515 uses the Palm OS v4.1.

Next came the Palm Zire, Tungsten C, Tungsten T, Tungsten W and Tungsten T2, each with memory varying between 16 MB and 64 MB and running the PalmOS v5.0 and v5.2.1. All of them have IR communication capabilities, with the Tungsten C additionally supporting Wi-Fi and Palm Tungsten T and T2 supporting Bluetooth.

The Samsung colour communicator SPH-1300 and Kyocera QCP 7035 smart phone have a cellular phone and Palm in one device. The Handspring Visor family, the first ones to start in 1998, has five different variants and a USB interface. The latest are the Treo family devices, which combine a phone, organizer and have Internet facilities, together with a built-in keyboard. The latest Palm device in the market is the Palm TX, whose specifications are given in Table 8.2.

The Palm TX handheld has a large bright screen with built-in integrated Wi-Fi and Bluetooth technology. It has solid battery life. One can browse the Web and check e-mail from the office, campus or a home Wi-Fi network, as also from airports, cafes and hotels. One can also carry Word, Excel and PowerPoint files to get more work done anywhere. Web pages, presentations, spreadsheets, photos and videos can be seen on a large colour screen that rotates from landscape to portrait mode. The Palm TX handheld allows listening to MP3s and to read eBooks.

Note that the processor in the TX is only 316 MHz, which puts some constraints on high-performance activities such as playing video, opening large files or other processor-dependent tasks.

Table 8.2 Specifications and Salient F eatures of P alm TX

Processor	312 MHz Intel XScale PXA270 with WMMX
Operating system	Palm OS 5.4.9.3.40
Display	3.8″ 320 × 480 HVGA screen
Memory	128 MB flash memor y (101 MB a vailable)
Size and w eight	4.76″ × 3.08″ × 0.61″, 149 g (5.25 ounces)
Expansion	Single SDIO slot
Docking	PalmOne ‘Athena’ multiconnector ; USB cable; optional cradle
Communication	Integrated Bluetooth 1.1 and 802.11b Wi-F i
Audio	Internal monaural speak er; 3.5 mm headphone jack
Battery	Standard 3.7 volt, 1,250 milliamp-hour non-replacable Lithium-Ion batter y
Input	4 remappable application buttons; 5-w ay directional pad; touchscreen
Software	MobiTV, Aveenu, Documents To Go 7

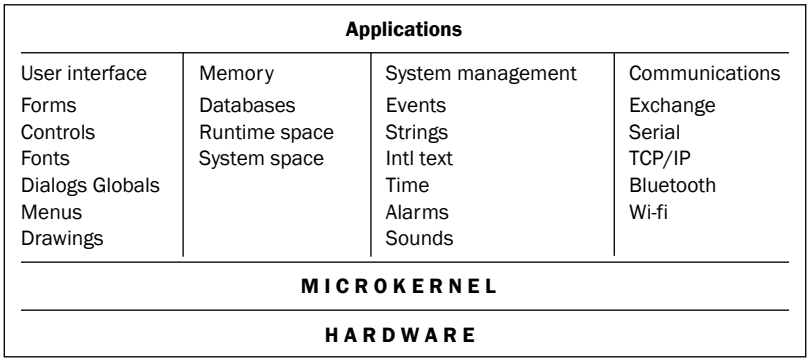


Figure 8.2 Palm OS Operating system

8.3 The Palm OS operating system

Palm OS is the product of Palm Inc. and has less than 10 per cent market share today, which has fallen from around 70 per cent earlier. The latest version is 5.4. It is based on a 32-bit architecture, which allows 4 GB of address space to be accessible and provides excellent power management, efficient memory utilization and ease in programming. It also has enhanced security features, like locking of the device, passwords, data encryption, wireless Internet and e-mail access. Figure 8.2 gives the hierarchical structure of the Palm OS operating system. We shall discuss some of the blocks shown in Figure 8.2 in some detail next.

8.3.1 Memory management

The total memory available to the Palm OS is 256 MB. All storage resides on memory cards, which are seen as logical units of RAM, ROM or both. See Figure 8.3 for the details.

1. **ROM:** The ROM is used for storage of the OS, built-in applications and default databases. On most devices, the ROM is actually a Flash RAM that can be programmatically changed. The size is mostly 2 MB, with 1 MB or more free memory, where custom applications can be inserted.
2. **RAM:** The RAM is divided into two separate fixed sized areas, a dynamic RAM that is used for temporary store of data without the need for persistence and a storage RAM that is used for permanent storage. Memory is allocated in chunks of maximum size of 64 KB. Relocatable memory blocks are used to combat fragmentation. Execute-in-phase (XIP) technology is used, so that the applications need not be transferred to a dynamic heap before execution.
3. **Database:** There is no traditional file system in PalmOS. Instead, a record database, which is an ordered list of records that can be sorted and searched efficiently, is used for application data. The resource database is an unordered list of records that is used to store application codes. A database manager is present for managing the database.
4. **Events and process management:** In PalmOS, only one application can run at a time, since there is no threading available. But it is possible to sublaunch other applications. For example, a 'search' application can invoke other applications like queues on a database. Palm OS is an event-driven operating system, where events can be user interface (UI) actions, like a tap on the screen, or system notifications, like a timer alarm or application-specific events like a database request.
5. **User interface:** The screen on the Palm devices is about 6 cm square and has a resolution of up to 320 $\times$ 320. There is no keyboard or mouse for input. A stylus is provided and either a

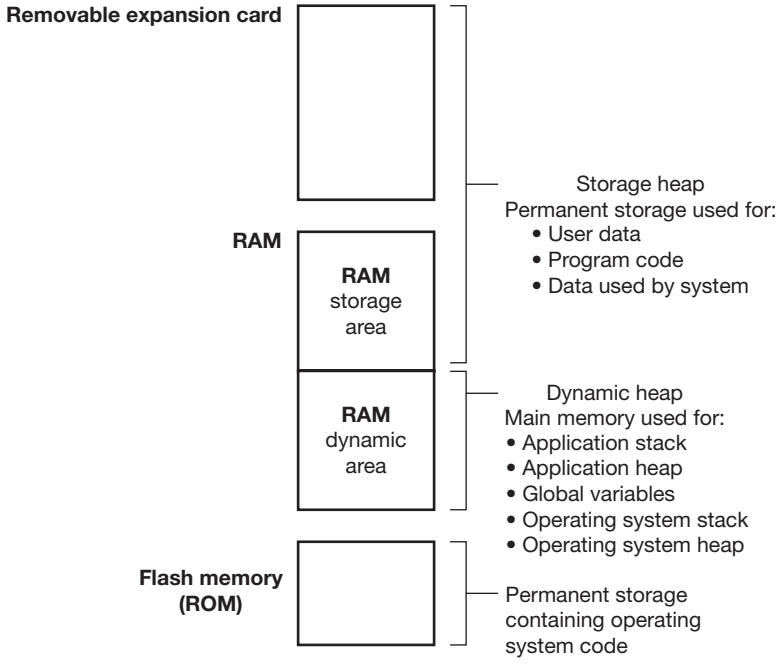


Figure 8.3 Memory Architecture in P alm OS

graffiti area or an on-screen keyboard for entering text. Displaying the right information is more important than fitting as much information on the screen as possible. However, a graphical UI is provided, with support for check boxes, radio buttons, lists and scrollbars. It is important to note that only one application uses the screen at a time.

8.3.2 Communication and networ king

The Palm TX can communicate with other devices in two ways. These are as follows:

1. **Wi-Fi:** The device quickly and automatically toggles Wi-Fi on and off as needed. This keeps the Wi-Fi 'on' all the time, while not excessively draining the battery. After the connection is idle for a certain period of time (or if the device is turned off), the Wi-Fi goes back to sleep. It also stores network parameters on a case-by-case basis. Suppose that at home one has a Wi-Fi network worked out with static Internet protocol (IP) addresses. At work, however, if the network uses automatically assigned dynamic IPs, the TX remembers the individual settings. It will use a static IP for the home network and a dynamic one for work.
2. **Bluetooth:** The interface provides the setup wizards for connecting to a mobile phone, PC, or network and has Bluetooth, global positioning system (GPS) and serial ports. Two libraries are provided to access the Internet through TCP/IP. These are the Net library and Internet library.

8.4 HP handhelds

Hewlett-Packard has taken mobile computing to new levels of compatibility and convenience. Starting with the HP Jornada 820 or 820e handheld PC (H/PC) in 2002, their latest is the **HP**

iPAQ rx 4240 Mobile device (2007), powered by the Microsoft Windows CE-based Windows Mobile operating system. Salient features of its hardware are as follows:

1. Samsung processor (400 MHz)
2. Windows Mobile 5.0 for Pocket PC, Premium Edition
3. 64-MB SDRAM, 128-MB Flash ROM
4. Up to 64-MB user-accessible memory
5. Integrated WLAN 802.11b/g and Bluetooth wireless technology
6. 2.8″ transmissive thin film transistor (TFT) screen with LED backlight
7. Secure digital (SD) slot
8. 1200 mAh battery
9. Integrated mini-USB charging/communications port

The following are some of its technical features, including software available and applications:

1. Up to 10 hours of battery life
2. Simple to synchronize with the desktop
3. Convenient to carry, with a weight of 1.115 kg, including batteries
4. Expandable with PC card accessories, USB devices, or an external monitor
5. Complete with built-in modem for simple and easy mobile computing
6. Has a dialup application for fast and easy access to e-mail and the Web
7. Has a viewer application that displays Pocket Outlook data at a glance, allowing one to navigate to or view data immediately
8. Internet Explorer v4.0 browser that includes support for hyper-text markup language (HTML 4.0), extensible markup language/extensible stylesheet language (XML/XSL), Dynamic HTML and animated graphic interchange formats (GIFs)
9. Windows media player for playing either Windows Media or MP3 music
10. New APIs with the crypto API (CAPI) for developing additional encryption software
11. Microsoft Message Queue (MSMQ) for exactly once-in-order message delivery providing reliable transactions while disconnected, and the SmartCard API enabling Smartcard applications such as additional security authentication, personal information (e.g. medical and financial) and loyalty points
12. SQL server for Windows CE

We shall now discuss the Window CE operating system and then move on to the Windows Mobile operating system, which is the latest offering from Microsoft.

8.5 Windows CE

Microsoft's Windows CE is an open, scalable, 32-bit operating system, designed to support a wide range of intelligent devices, from enterprise tools such as industrial controllers, communication hubs and point-of-sale terminals to consumer products such as cameras, telephones and home entertainment devices. There are four different versions of the Windows CE kernel for various CPUs like the NEC MIPS and its variants, the Intel/AMD X86, Hitachi SH-3 and SH-4 and the Intel StrongARM to handle different kinds of screens, keyboards, modems and peripheral devices, unlike the single version of PalmOS.

Windows CE was developed from the ground up, as a small footprint, highly customizable and modular operating system for embedded applications. Its kernel borrows from other

Microsoft 32-bit operating systems, but eliminates or replaces those features that are not needed for typical Windows CE applications. Hence, it is compact and customizable, and a version of the kernel can reside in less than 200K of ROM.

8.5.1 Memory architecture

In the Windows CE .NET-based system, the ROM stores the entire OS as well as the applications that come with the system, like Pocket Word, Pocket Excel. If a module is uncompressed, ROM-based modules are executed in place. If the ROM-based module is compressed, it is decompressed and then paged into RAM. All read/write data are loaded into RAM. The option to enable compression in ROM is controlled by the Original Equipment Manufacturer (OEM) Executing programs directly from ROM. This saves the program RAM and reduces the time needed to start an application, because the program does not have to be copied into RAM before it is launched.

Programs that are not in the ROM but are contained in the object store or on a flash memory storage card are not executed in place, but are paged into the RAM for execution. Depending on the OEM and the driver options on a specific Windows CE-based platform, the program module can be paged on-demand. One page can be brought in at a time, or the entire module can be loaded into ROM at once.

The RAM on a Windows CE device is divided into two areas—the object store and the program memory. The object store resembles a permanent, virtual RAM disk. Data in the object store is retained when the system is suspended or soft-reset, and devices typically have a backup power supply for the RAM to preserve data if the main supply is interrupted temporarily. When operation is resumed, the system looks for a previously created object store in RAM and uses it if one is found. Devices that do not have battery-backed RAM can use the hive-based registry to preserve data across boots.

The remaining RAM is devoted to program memory, which works like the RAM in PCs. It stores the heaps and stacks for the applications that are running. The maximum size of the RAM file system is 256 MB with a maximum size of 32 MB for a single file. However, there is a 16-MB limit on a database volume file. The maximum number of objects in the object store is approximately 4,000,000.

8.5.2 Memory management

Windows CE performs well on small devices, in a small pool of memory, on portable, battery-operated devices. It requires a little-endian, 32-bit processor and was built to run using paged memory. Each Windows CE process has a virtual address space of 32 MB, unlike the desktop versions of Windows, which provide a 2 GB process address space. However, applications that need to work with larger objects can have shared memory which is allocated from a gigabyte-sized part of the address space.

8.5.3 Processes and threads

As a preemptive, multitasking OS, Windows CE supports up to 32 processes running simultaneously within the system. The actual number of additional threads is limited only by the available system resources. Windows CE provides 256 priority levels that can be set on a thread. The top 248 levels can be protected from the application when an OEM enforces a trusted environment. All four synchronization objects from desktop versions of Windows are present in CE, namely, critical sections, mutexes, semaphores and events.

8.5.4 Scheduling

The Windows CE .NET scheduler maintains a priority list of each process and thread in the OS. Each process can contain multiple threads, and each of these threads composes a path of execution. The scheduler controls the order in which these different paths of execution are sequenced and allows them to interact in a predictable fashion. When interrupts occur, the scheduler takes them into account and reprioritizes threads accordingly. The scheduler can perform its work both from the kernel and in a predefined scheduling mechanism. An application developer can create a miniature custom scheduling system internally in a thread through the use of fibres in that thread.

8.5.5 Real-time performance

Windows CE is a real-time operating system (RTOS) having the following kernel capabilities:

- support for up to 3 different processes and 256 thread priority levels
- support for priority inversion
- support for nested interrupts to ensure that high-priority events are not delayed
- support for 1 ms system tick timing
- advanced thread timing and scheduling
- support for semaphores

Real-time performance is defined by the following:

- guaranteed upper bounds for the highest-priority thread
- guaranteed upper bound on delay in executing high-priority interrupt service routines (ISRs). The kernel has a few places where interrupts are turned off for a short, bounded time.
- fine control over the scheduler and how it schedules threads.

For further details on the security subsystem, system calls, kernel enhancements, etc., of Windows CE v3.0, the reader is referred to www.microsoft.com/windows/embedded for Windows CE OS.

8.6 The Windows Mobile operating system

Windows Mobile is a compact operating system combined with a suite of basic applications for mobile devices based on the Microsoft Win32 API. Devices which run Windows Mobile include Pocket PCs, smart phones and portable media centres. It is quite similar to desktop versions of Windows. Windows Mobile for Pocket PC provides the following standard features:

- The current date, owner information, upcoming appointments, e-mail messages and tasks are shown on the Today screen. Users can also select the other information they wish to display. It also includes the notification bar which has icons to notify the status of Bluetooth, etc.
- The Taskbar shows the current time, the volume and the connectivity status. The Start Button is designed like the Start Button on desktop versions of Windows. The Start Menu shows recently opened programs at the top, nine customizable menu entries and links to the program, settings, search and help.
- Mobile versions of Microsoft Office applications include Word Mobile and Excel Mobile. In Windows Mobile 5.0, PowerPoint Mobile has been included. These versions include many features which are used in desktop versions, but some others like inserting tables and images have not been included in pre-5.0 versions. Desktop versions of files can be converted to Office Mobile compatible versions.

- Outlook mobile includes Microsoft Outlook for desktop versions. It is included in a value-added CD-ROM with the Pocket PC hardware manufacturer.
- Windows Media Player is also included. All new Windows Mobile 6 devices have Version 10.2 of the player. In older devices, Version 10 is available for download for only specified devices. The player supports .WMA, .WMV, .MP3, and .AVI files. Currently, MPEG files are not supported, and WAV files are played in a separate player. Some versions are also capable of playing .M4A audio files.

Figure 8.4 shows a typical screen of the Windows Mobile 6, which was released in early 2007. It features three different versions—Windows Mobile 6 Standard for smart phones (phones without touch screens), Windows Mobile 6 Professional for PDAs with phone functionality (Pocket PC Phone Edition) and Windows Mobile 6 Classic for plain PDAs without cellular radios.

We give below a summary of its specifications:

- It is based on Windows CE 5.0 (version 5.2)
- Supports 800 × 480 and 320 × 320 resolution
- Operating system live update
- Improved remote desktop access
- Faster, easier application development and distribution
- VoIP (Internet calling) support
- Windows Live for Windows Mobile
- Microsoft Bluetooth stack
- Storage card encryption—of data stored in external removable storage cards
- Smartfilter to search faster through e-mails, contacts, songs, files, etc.
- Improved Internet sharing to easily set up the device as a modem.
- Outlook Mobile supports viewing HTML e-mail.
- Support for JavaScript
- Set 'Out of Office Replies' from the device.
- SQL Server Compact Edition in ROM.

Figure 8.4 Windows Mobile 6 Professional T oday Screen

8.7 Nokia handhelds

Numerous Nokia handhelds are available in the market today. Here, we shall discuss one of the latest products made available by Nokia, the Nokia 9210 Communicator.

The Nokia 9210 Communicator is a third-generation Communicator series smart phone produced by Nokia, introduced in 2000. It greatly improved on the second-generation Nokia 9110 Communicator, providing colour main screen, changing to Symbian OS platform and ARM processor. It is one of the few mobile phones able to send and receive fax. Specifications are given in Figure 8.5.

It is used as a normal, though bulky, mobile phone in closed mode; when it is flipped open, it can be used like a very small notebook computer with a half VGA screen. The earpiece and microphone are located on the back, so one must hold it with the front screen and keypad facing out to make a call. The phone also has speakerphone functionality.

8.7.1 Specifications of Nokia 9210

- **Main applications:** Mobile phone, desk application, messaging (SMS, fax, e-mail), Internet (Web, WAP), contacts (address book), calendar, office (word processor, spreadsheet, presentation viewer, file manager)
- **Extra applications:** Calculator, clock, games, recorder, and unit converter
- **Processor:** 32-bit 66 MHz ARM9-based RISC CPU running Symbian OS
- Includes PC Suite for the Nokia 9210 Communicator, running on Windows platform.

Nokia 9210 Communicator

Manufacturer	Nokia
Screen	4096-colour 640 × 200 LCD screen
Exterior screen	Mono
Memory	Application 14 MB, user 2 MB
Memory card	16 MB multimedia card
Networks	EGSM 900/1800 and HSCSD (43.2 Kbit/s)
Physical size	158 × 56 × 27 mm
Weight	244 g

Figure 8.5 Specifications of the Nokia 9210 Communicator

In 2005, the 9210 Communicator was replaced with

- Nokia 9300—which is smaller (132 × 51 × 21 mm) and lighter (167 g) than Nokia 9210, with similar features.
- Nokia 9500—which has additional features (Wi-Fi and camera) but the same size (148 × 57 × 24 mm) and weight (222 g).

Both new models include other improvements such as EDGE, colour 128 × 128 external display and Bluetooth. The Nokia 9290 is the American variant of the Europe-only Nokia 9210 Communicator phone.

8.7.2 Features

Some of the useful features present in the Nokia handheld are as follows:

1. It features an **integrated design** with the following characteristics:
   - Consistent menus and colour screen icons that create a PC-like environment.
   - High-resolution, colour TFT active matrix screen that displays up to 4,096 colours.
   - Full QWERTY keyboard that allows information to be input as in PCs.
   - Application keys that give easy access to the features most used.
2. It is equipped with Office facilities called **Wireless Office,** which allows it to
   - Create, edit and share documents and spreadsheets.
   - View presentations with the built-in presentation viewer.
   - Edit documents or check calendar while talking handsfree on the speakerphone.
   - Send documents and spreadsheets via e-mail, infrared (IR) or fax.
3. It is a **powerful PDA** which lets the user
   - Synchronize the calendar, to-do list and contacts with Microsoft Outlook and Lotus Notes using Nokia 9290 PC Suite 4.
   - Use the Calendar to check schedules, make appointments, keep track of anniversaries and birthdays, view and edit a to-do list, set alarms and send appointments to other people's calendars via SMS or e-mail.
   - Create, edit and manage all contact information such as phone and fax numbers, e-mail and street addresses. In addition, the user can attach a photograph to each contact card and synchronize with the desktop PC to keep current contact information at the fingertips.
4. It provides many **e-mail** features to
   - Send and receive e-mail with attachments.
   - Manage e-mail from anywhere.
   - Support all mail protocols like IMAP4, POP3, MIME1 & 2, MHTML and SMTP and is compatible with Microsoft Outlook, Lotus Notes, etc.
5. It provides **mobile multimedia** features to
   - Send, receive and view video clips, digital images and music, including AVI, WAV, MPEG and MP3 files, which can be converted and played on Nokia 9290 Communicator by using the multimedia converter application supplied on the Nokia 9290 Communicator CD or download.
   - Support JPEG, GIF, TIFF, BMP and other images.

- Use real video and Macromedia Flash on Nokia 9290 Communicator. Players are included on the Nokia 9290 Communicator CD or download.

6. **Internet access** is built-in using the 3.2 HTML browser to
   - Browse the Web wirelessly and see it in full colour.
   - Bookmark Web sites for easy access.
   - Support Frames and Secure Socket Layer (SSL) security.
7. It is also an **integrated cellular phone with speakerphone** and allows
   - Handsfree talking on the speakerphone.
   - Conference calls with up to five people.
   - Talking and using some other applications simultaneously.
   - Checking calendar and taking notes during a call.

   The **battery** used is the extended lithium-ion battery of 1,300 mAh.

8.8 Symbian operating sy stem

Symbian OS is an open operating system, designed for mobile devices, with associated libraries, UI frameworks and common tools, produced by Symbian Ltd. It is a descendant of Psion's EPOC and runs exclusively on ARM processors.

Symbian OS is the market leader in open operating systems for advanced data-enabled mobile phones, licensed by the world's leading mobile phone manufacturers. It is designed for the specific requirements of advanced 3G mobile phones and beyond. It combines the power of an integrated applications environment with advanced data services like mobile telephony. Symbian OS v9.5, the latest evolution of Symbian OS, delivers over 70 new features for high-performance smart phones.

8.8.1 Design

Symbian OS has its roots in the Psion Software's EPOC and is structured like many desktop operating systems with pre-emptive multitasking and memory protection. EPOC provides multitasking with server-based asynchronous serialised access based on events.

Symbian OS was built to follow three design rules in order to support extended, always-on operation:

- The integrity and security of user data is important.
- User time must not be wasted.
- All resources are scarce.

For hardware, the OS is optimized for low-power battery-based devices and for ROM-based systems. Applications and the OS follow an object-oriented design. Later, OS versions have a real-time kernel and a platform security model in versions 8 and 9.

There is a strong emphasis on **conserving resources**, using Symbian-specific programming features such as descriptors and a cleanup stack. All Symbian OS programming is event-based, and the CPU is switched off when applications are not directly dealing with an event. This is achieved through a programming feature called active objects. Similarly, the OS approach to threads vs. processes is driven by reducing overheads.

Symbian OS supports sufficiently **fast real-time response** such that it is possible to build a single-core phone around it, that is, a phone in which a single processor core executes both the

user applications and the signalling stack. This is a feature which is not available in Linux. This has allowed SymbianOS phones to become smaller, cheaper and more power efficient.

As of 2007, Symbian OS was the leading OS in the smart mobile device market with a 67 per cent share of the smart mobile device market, with Microsoft having 13 per cent through Windows CE and Windows Mobile and RIM having 10 per cent.

8.8.2 Symbian structure

The Symbian OS system model contains the following layers, from top to bottom:

1. UI framework layer
2. Application services layer containing Java ME
3. OS services layer containing the following services:
   a. Generic OS services
   b. Communications services
   c. Multimedia and graphics services
   d. Connectivity services
4. Base services layer
5. Kernel services and hardware interface layer

The **base services layer** is the lowest-level reachable by user-side operations; it includes the File Server and User Library, the Plug-In Framework which manages all plug-ins, Store, Central Repository, data base management system (DBMS), and cryptographic services. It also includes the Text Window Server and the Text Shell, the two basic services from which a completely functional port can be created without the need for any higher-layer services.

Symbian OS has a **microkernel architecture**, which provides for robustness, availability and responsiveness. It contains a scheduler, memory management and device drivers. Other services like networking, telephony and filesystem support are present in the OS services layer or base services layer. The inclusion of device drivers means the kernel is not a true microkernel, but a nanokernel, containing only the most basic primitives and supporting an extended kernel to implement any other abstractions.

Symbian OS is designed for **compatibility** with other devices, especially removable media file systems. Earlier the FAT was used as the internal file system, but an object-oriented persistence model has been placed over the underlying FAT, providing a POSIX-style interface and a streaming model.

There is a large **networking and communication subsystem**, which has servers for EPOC telephony, EPOC sockets and for serial communication. The subsystem also contains code for short-range communication links, such as Bluetooth, IrDA and USB.

There is also a large volume of **user interface (UI)** code. Only the base classes and substructure are contained in Symbian OS. The actual user interfaces are maintained by third parties. The OS also contains libraries for graphics, text layout and font rendering.

8.9 Summary

Personal digital assistants (PDAs) are small extensions of notebook computers, and represent the fastest-growing handheld systems that can be connected to the Internet today. They are characterized by small size, small memory, small screen size and low battery power. But the features provided and the functionality available in them are improving day by day.

The operating systems used in these PDAs are characterized as being real-time and having a small footprint. The most popular OS used with PDAs are the SymbianOS, PalmOS, the Windows CE and the Windows Mobile. These operating systems exhibit many unique features, including real-time and multitasking operation.

In the next chapter, we shall discuss the mobile Internet and the wireless Web to see how these handheld devices can be used to access the Internet in a wireless manner and the protocols available to facilitate the same.

Problems

1. Using the parameters given in Table 8.1, compare any commercially available PDA and notebook computer.
2. Explain what is meant by 'small footprint', with respect to operating systems.
3. Discuss the parameters by which one can compare operating systems for handhelds.
4. Compare the PalmOS, Windows CE and SymbianOS using the parameters identified in Problem 3.
5. Explain why it is better to have a real-time OS with PDAs.
6. Is Windows CE a real-time OS? Explain your answer.
7. Show how the Windows Mobile OS is superior to Windows CE.
8. Search the Internet and note down the specifications of the latest PDA available in the market. Compare its features with the Windows Mobile 6 Professional.
9. Compare and contrast the specifications of a typical mobile phone with a typical palmtop computer.
10. Windows CE is said to be a multitasking and multithreading OS. What is the meaning of this statement? Discuss the difference between threads and processes.

Multiple-choice questions

1. Which one of the following vendors is considered the most secure?
   (a) Palm **(b)** BlackBerry
   (c) Windows Pocket PC **(d)** Handspring Visor family
2. How much memory of Palm Zire 71 is user accessible?
   (a) 16 MB **(b)** 12 MB
   (c) 14 MB **(d)** 20 MB
3. Which of the following wireless communication techniques is used in personal area networks?
   (a) Wi-Fi 802.11b **(b)** CDMA
   (c) Bluetooth **(d)** AT&T Wireless
4. In which state is the Windows CE device not powered, so that all volatile RAM data and code are lost?
   (a) Suspended **(b)** Dead
   (c) Blocked **(d)** Terminated

5. Which one of the following companies manufactured Palm OS-based devices?
   (a) Kyocera **(b)** Sun Microsystems
   (c) Apple **(d)** HP
6. Which one of the following operating systems supports multithreading and multitasking?
   (a) Palm OS **(b)** Windows CE
   (c) Symbian **(d)** None of the above
7. A record database can be used to store which one of the following?
   (a) Unordered list of records **(b)** Ordered list of records
   (c) Application codes **(d)** None of the above
8. Which of the following is the memory capacity of Palm m515?
   (a) 16 MB **(b)** 24 MB
   (c) 8 MB **(d)** 20 MB
9. Which one of the following vendors is heavily depended on the third-party software?
   (a) Windows Pocket PC **(b)** BlackBerry
   (c) Palm **(d)** None of the above
10. What is the latest version of Palm handheld devices?
    (a) Palm m215 **(b)** Palm i705
    (c) Palm IIIx **(d)** Palm Zire

Further reading

http://en.wikipedia.org/wiki/ARM_architecture#cite_note1
U. Hansmann, L. Merck, M.S. Nicklous and T Stober (2003), *Principles of Mobile Computing*, 2nd ed. (New Delhi, India: Springer).
www.hp.com for information on Hewlett-Packard handhelds.
www.microsoft.com/mobile for an overview of Windows CE devices.
www.microsoft.com/pocketpc for Pocket PC information.
www.microsoft.com/windows/embedded for Windows CE OS.
www.motorola.com/consumer for information on Motorola's consumer products.
www.nokia.com for the Nokia homepage.
www.palm.com for information and downloads for Palm OS computing devices.
www.palmzone.com online magazine covering Palm OS-related topics.
www.psion.com for EPOC-powered devices, Psion devices, free downloads, etc.
www.symbian.com/news/pr/2008/pr200810018.html

The Mobile Internet and Wireless Web

9

Traditionally, the Internet has been accessed via a 2 Mbps cable modem from the office or home, and the content has been received on high-resolution screens via hypertext transfer protocol (HTTP) in hypertext markup language (HTML) format. But with the advent of and considerable interest in small portable devices and handhelds, capable of accessing the Web via wireless links like GPRS or GSM, focus has shifted to tiny displays of 96 × 64 pixels to receive content 24 × 7 × 365.

The wireless application protocol (WAP) is an open, global specification that empowers mobile users with wireless devices to easily access e-mail and the Web, through information and services instantly. The WAP Forum was initially set up as a consortium led by Nokia, Ericsson and Motorola in 1997, as an Industry Group for the purpose of extending the existing Internet standards for use with wireless communication. It was able to deliver services as early as 1999, through the WAP standard 1.1, providing speeds of 9.6 Kbps. The WAP standard 2.0 came in 2002 and could provide speeds of 384 Kbps. As of 2002, more than 500 companies from all parts of the industry, including network operators, device manufacturers, service providers and software vendors, are members of the WAP Forum.

In this chapter, we concentrate on the WAP and its gateway, but before we go into the details of WAP, let us take a look at the traditional Web programming model and see how it compares with the WAP programming model.

9.1 The Web programming model

The Internet World-Wide-Web (WWW) architecture provides a very flexible and powerful programming model, shown in Figure 9.1.

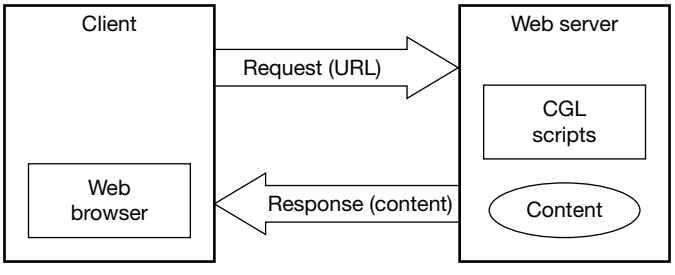


Figure 9.1 The Web Programming Model

In this model, applications and content are presented in standard data formats and are browsed by applications known as **Web browsers**. The Web browser is a networked application; that is, it sends requests for named data objects to a network server, and the network server responds with the data encoded using the standard formats. The WWW standards specify many of the mechanisms necessary to build a general-purpose application environment, including the following:

1. **Standard naming model:** All servers and content on the WWW are named with an Internet standard **uniform resource locator** (URL).
2. **Content typing:** All content on the WWW is given a specific type, thereby allowing Web browsers to correctly process the content, based on its type.
3. **Standard content formats:** All Web browsers support a set of standard content formats. These include the HTML, scripting languages like ECMAscript and Javascript, and a large number of other formats.
4. **Standard protocols:** Standard networking protocols allow any Web browser to communicate with any Web server. The most commonly used protocol on the WWW is the HTTP, operating on top of the TCP/IP protocol suite.

This infrastructure allows users to easily reach a large number of third-party applications and content services. It also allows application developers to easily create applications and content services for a large community of clients.

9.2 The WAP programming model

The WAP programming model, shown in Figure 9.2, is similar to the Web programming model, with certain enhancements and extensions to match the characteristics and constraints of the wireless environment. These are related to slow central processing units (CPUs), low-bandwidth connections, small memory and small screens. Wherever possible, existing standards have been adopted or have been used as the starting point for the WAP technology, as this provides flexibility to application developers. The use of proxies is one such feature in this direction.

The WWW programming model provides several benefits, including a familiar programming model, a proven architecture and use of existing tools like Web servers, extensible markup language (XML) tools, etc. Optimizations and extensions have been made in order to match the characteristics of the wireless environment. The most significant enhancements that WAP has added to the programming model are push mechanism and support for telephony (WTA).

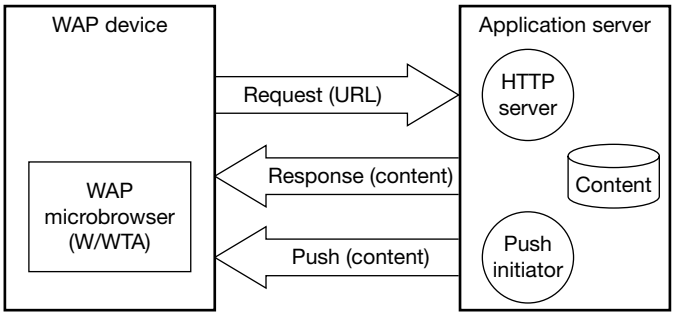


Figure 9.2 The WAP Programming Model

The classical request-response mechanism is commonly referred to as **pull**, where the client pulls data from the server, in contrast with the **push** mechanism, where the server pushes data to the client.

WAP content and applications are specified in a set of well-known content formats, based on the familiar WWW content formats. Content is transported using a set of standard communication protocols based on the WWW communication protocols. The WAP **microbrowser** in the wireless terminal coordinates the user interface and is analogous to a standard Web browser. WAP defines a set of standard components that enable communication between mobile terminals and network servers, including the following:

1. **Standard naming model:** WWW standard URLs are used to identify WAP content on origin servers. WWW standard **uniform resource indicators** (URIs) are used to identify local resources in a device, for example, call control functions.
2. **Content typing:** All WAP content is given a specific type consistent with WWW typing, thereby allowing WAP user agents to correctly process the content, based on its type.
3. **Standard content formats:** All WAP content formats are based on WWW technology and include display markup, calendar information, electronic business card objects, images and scripting languages, and a large number of formats.
4. **Standard protocols:** WAP communication protocols enable the communication of browser requests from the mobile terminal to the Web server. The WAP content types and protocols have been optimized for mass market, handheld wireless devices.

9.3 WAP protocol stack

The WAP 1.1 protocol stack is shown in Figure 9.3. It is derived from the ISO OSI reference model and has five different layers.

The application layer (WAE) provides an application environment for the development and execution of portable application and services. The microbrowser sits here but does not use HTML. Instead, it uses the wireless markup language (WML), which is an application of XML. Therefore, a WAP device can only access those pages that have been converted to WML. However, it is possible to design an on-the-fly HTML to WML filter to increase the set of pages available. This is called the WAP gateway and is the subject of discussion in Section 9.5.

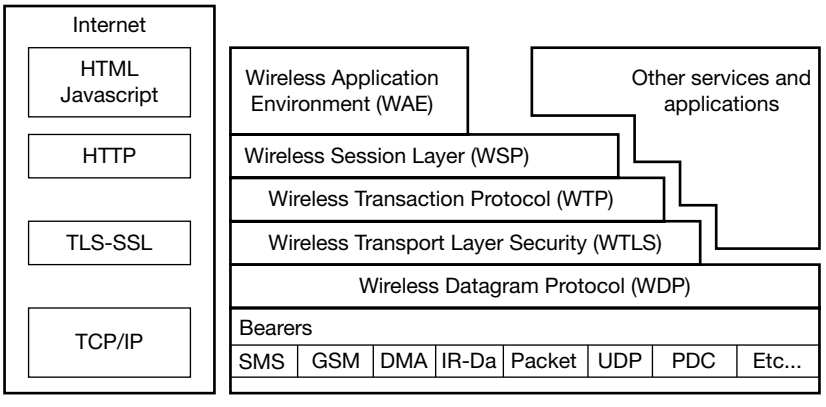


Figure 9.3 WAP 1.1 Protocol Stack

The session layer (WSP) supplies methods for the organized exchange of content in client/server applications. It is similar to HTTP/1.1, but with some restrictions and extensions for optimization purposes.

The transaction layer (WTP) provides different methods for performing transactions, to a varying degree of reliability. It manages requests and responses, either reliably or unreliably.

The security layer (WTLS) is an optional layer that provides, when present, authentication, privacy and secure connections between applications. WTLS is a subset of Netscape's Secure Socket Layer (SSL).

The transport layer (WDP) is the bottom layer of the WAP stack, which shelters the upper layers from bearer services offered by the operator. It is essentially user datagram protocol (UDP).

The bearer layer supports all the existing mobile phone systems, including GSM, DAMPS and CDMA. The WAP 1.1 data rate is 9,600 Bps.

WAP 1.1 is basically a circuit-switched system that can be used with a variety of different networks. But it comes with a high per-minute charge. Its other major drawback is that it does not support HTML, and hence content available to the WAP device is limited. Another major drawback with WAP 1.1 is that it does not support Internet standard protocols. Hence, a gateway is required to translate contents from the WAP environment to the Internet environment. This poses a major security risk, which we will discuss in Chapter 11.

9.4 Information-mode (I-mode)

In 1999, the Japanese NTT DoCoMo company launched the information-mode (i-mode) for wireless access to the Web, using special handsets. The i-mode system has a new transmission system, a new handset and a new language. The new transmission system is built on two separate networks—the existing circuit-switched mobile phone network for voice and a new packet-switched network built specifically for i-mode service. These are, however, not usable concurrently.

I-mode handsets are basically wireless terminals and are not user programmable. When the i-mode handset is switched on, the user is presented with a list of services categorized as e-mail, news, weather, sports, games, shopping, etc. Each of these services is to be subscribed and paid for, though nominally.

The language used in the i-mode is a compact version or subset of HTML, called cHTML. While most of the HTML tags are allowed here, the cHTML browser does not support Javascript, frames, style sheets, JPEG images, etc. However, the I-mode server is a full-blown system supporting CGI, Perl, PHP, JSP, ASP, etc.

For details of I-Mode and its comparison with WAP 1.1, the reader is referred to Tanenbaum (2005), Frengle (2002) and Vacca (2002).

9.5 WAP 2.0

The second-generation wireless Web, WAP 2.0, uses a packet-switching mechanism like GPRS. It has some new features, the most significant of which are as follows:

1. Push as well as pull model.
2. Support for integrating telephony into applications.
3. **Multimedia messaging:** WAP 2.0 supports the merger of voice and data in a variety of ways. Along with e-mail and telephony, multimedia messaging is supported.

4. **Inclusion of 264 pictograms:** Pictograms are included in a number of categories, namely, animals, appliances, dress, emotion, food, human body, gender, maps music, plants, etc.
5. **Interface to a storage device:** Flash ROM is supported as a storage device. A WAP-enabled wireless camera could use it for temporary image storage, before sending it on the Internet.
6. **Support for plug-ins in the browser:** Plug-ins extend the browser's capabilities, with provision for scripting languages, etc.

Various technical differences are also present between WAP 1.1 and 2.0. Firstly, WAP 2.0 continues to support the WAP 1.1 stack, but also supports the standard Internet stack with TCP and HTTP/1.1. However, four minor but compatible changes to TCP were made to simplify the code:

1. Use of a fixed 64-KB window
2. No slow start
3. Maximum MTU of 1500 bytes, and
4. A slightly different retransmission algorithm

Secondly, WAP 2.0 supports XHTML Basic, a markup language intended for small wireless devices like mobile phones, televisions, personal digital assistants (PDAs), vending machines, pagers, cars, game machines, watches, etc. It therefore does not support style sheets, scripts or frames, but most of the standard tags are there, which are defined in XML.

WAP 2.0 runs at 384 Kbps, which is still very slow, as compared to the 11 Mbps or 54 Mbps data rates offered by IEEE Standard 802.11, which is giving it huge competition.

9.6 WAP gate way

A gateway is an intermediary element, usually used to connect two different types of networks. It receives requests directly from the clients as if it were an origin server that the clients want to retrieve the information from. Clients are usually not aware that they are speaking to the gateway. This scenario is shown in Figure 9.4.

A WAP gateway acts as a bridge between the Internet world and the mobile world, by providing access to and translation of the Internet resources. The architecture of the protocol used at the WAP gateway is shown in Figure 9.5.

WAP uses coded binary data to improve transmission efficiency. The header and content are compactly compiled. For the traditional HTTP network, the packet header is in string format. In order to adapt to the WAP network, encoding and decoding techniques are required. The coder/decoder functionality within the gateway is used to convert the WML and WMLScript content going to and coming from the client into a form that is optimized for low-bandwidth networks. Figure 9.6 illustrates the functionality of the WAP gateway.

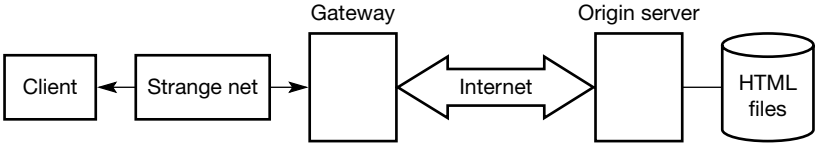


Figure 9.4 A Gate way

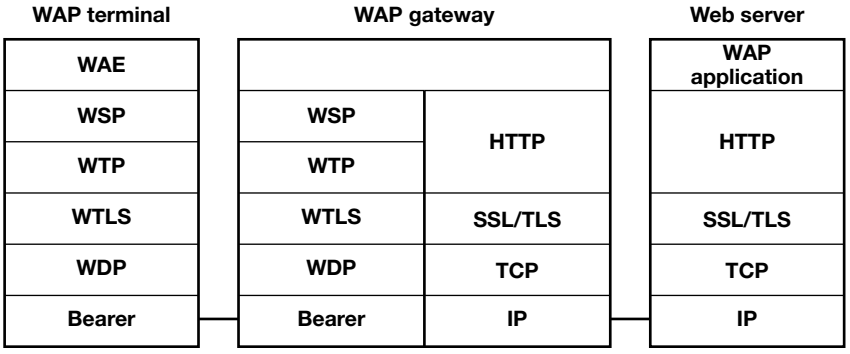


Figure 9.5 Architecture of the W AP Gate way Protocol

To translate the traditional HTTP requests and replies to the coded forms, a mapping table is required. The features provided by the WAP gateway are pull operation, push operation, on-the-fly image conversion (GIF, JPEG to wireless bit map protocol [WBMP]), WML to WAP binary XML (WBXML) (compressed format), access control, logging and cache service.

A WAP gateway does three tasks:

1. Header translation, which allows the client system to access the Internet via a different protocol.
2. Push operation, which allows the server to send the right information to the client, and
3. Content compilation, which allows compaction of the data for low-bandwidth support.

We discuss below these operations and their implementation in some detail.

9.6.1 Push operation

A push operation in WAP is done by a **push initiator** (PI) which transmits **push content** and **delivery instructions** to a **Push Proxy Gateway** (PPG), which then delivers the push content to the WAP client according to the delivery instructions. The PI is an application that runs on an ordinary Web server. It communicates with the PPG using the push access protocol (PAP); details

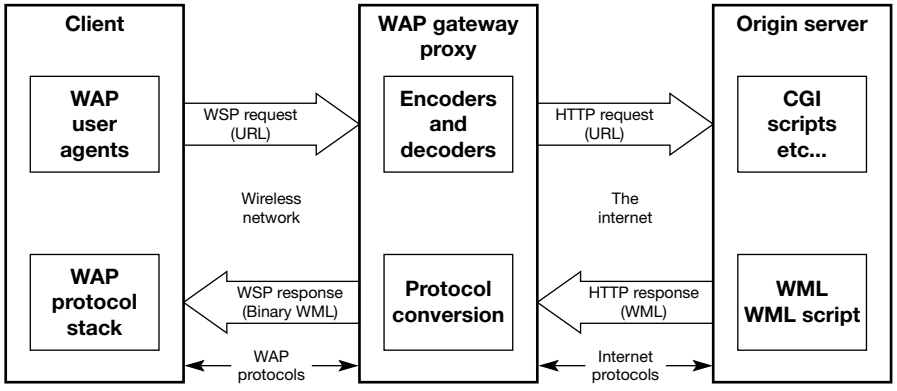


Figure 9.6 The WAP Gate way

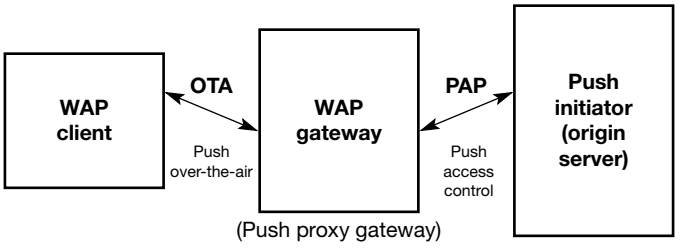


Figure 9.7 Push Architecture

of which are given in WAP Forum (2001). The PPG uses the **push over-the-air** (OTA) **protocol** to deliver the push content to the client. Push architecture is shown in Figure 9.7.

Push message types: Two push message content types are available. These are as follows:

a. **Service indication:** The service indication (SI) content type allows sending notifications to end users in an asynchronous manner. These may be about new e-mails, changes in stock price, news headlines, advertising, reminders, low prepaid balance, etc. This is shown in Figure 9.8.
b. **Service loading**: The service loading (SL) content type causes a user agent on a mobile client to load and execute a service that can be in the form of a WML deck. The SL contains a URI indicating the service to be loaded by the user agent without user intervention.

Basic steps involved in push operation:

- The PI (Web server or e-mail provider) instructs the WAP (push proxy) gateway to push an SI to the mobile client using the PAP. The PI sends the SI message with an appropriate header and an URI to the e-mail service.
- The push proxy/gateway sends the SI to the mobile client using the push OTA protocol.
- The mobile client receives the push containing the SI, and the message is presented to the end user.

HTTP transport: To send push request to WAP gateway: HTTP POST request method is used to transport the push request using PAP. The HTTP response always contains result code 202 (accepted for processing) when the HTTP transaction succeeds, although the response PAP document may contain a PAP error.

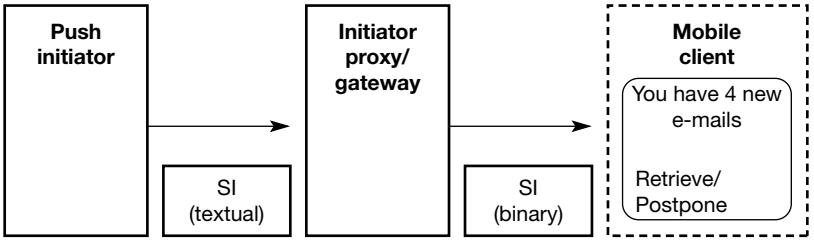


Figure 9.8 SI Message Over view

9.6.2 Push message for mat (using P AP)

The push message contains three entities that are bundled together in a message sent from the PI to the PPG. These are as follows:

1. **A control entity:** This is an XML document that contains delivery instructions for the PPG. It must be the first entity in the multipurpose Internet mail extensions (MIME) multipart/related message.
2. **A content entity:** This is a MIME body part containing the content to be sent to the wireless device. It is included only in the push submission and not in any other operation request or response. It must be the second entity in the MIME multipart/related message.
3. **An optional capability entity**.

We give a typical example code in XML for this below:

Example

```
Content-Type: multipart/related; type=application/xml;
boundary=WPL11woVbhESdfalYevGqpdzLCs

–WPL11woVbhESdfalYevGqpdzLCs
Content-Type: application/xml; charset=UTF-8

<?xml version="1.0"?>
<!DOCTYPE pap PUBLIC "-//WAPFORUM//DTD PAP 1.0//EN"
"http://www.wapforum.org/DTD/pap_1.0.dtd">
<pap>
<push-message push-id="100">
<address address-value="WAPPUSH=127.0.0.1/TYPE=IPv4@localhost"/>
</push-message>
</pap>
–WPL11woVbhESdfalYevGqpdzLCs
Content-Type: text/vnd.wap.si; charset=UTF-8

<?xml version="1.0"?>
<!DOCTYPE si PUBLIC "-//WAPFORUM//DTD SI
1.0//EN""http://www.wapforum.org/DTD/si.dtd"><si>
<indication href="yahoo.com" si-id="111" action="signal-medium">
this is message
</indication>
</si>
–WPL11woVbhESdfalYevGqpdzLCs–
```

The OTA SI format is shown in Figure 9.9.

TID	TYPE	Type_Specific_Content

Figure 9.9 OTA SI F ormat

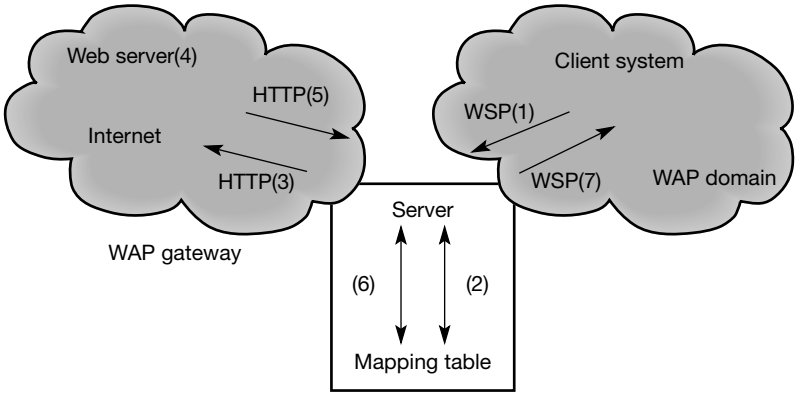


Figure 9.10 Pull Operation Architecture

Here

TID = push ID (unit 8)

TYPE = 0 × 06 (for push messages)

The field type_specific_content has four subfields, as given below:

- **Headers len field:** It specifies the length of the content-type and headers fields combined.
- **Content-type field:** It contains the content type of the data.
- **Headers field:** It contains the push headers.
- **Data field:** It contains the data pushed from the server.

Sending SI to the mobile: The specified message is then sent to the mobile in an UDP Packet, at port number 2948.

9.6.3 Pull operation

Details of the pull operation are shown in Figure 9.10. The steps involved in pull operation are given below:

1. The user agent (mobile) sends a URL request to a WAP gateway using the WSP protocol.
2. The WAP gateway decodes the request message and translates the request line and request header (in binary format) to HTTP format by a mapping table.
3. The WAP gateway creates a connection to the Web server and sends an HTTP request to it.
4. The HTTP request is processed by the Web server.
5. The Web server returns an HTTP reply message, which contains data.
6. The WAP gateway encodes the reply and translates the HTTP formatted reply line and reply header into WSP binary format using the mapping table.
7. The WAP gateway creates a WSP response containing WML and returns it to the client system.

9.7 Summary

The Internet today has become 'mobile', which implies that more and more mobile users are accessing it using their small handheld devices. But various extensions and enhancements are needed in the traditional Web programming model to match the characteristics and constraints

of these devices. These extensions have been provided by the wireless application protocol (WAP) programming model.

Two versions of the WAP are currently available, WAP 1.1 and WAP 2.0, with the latter providing many more facilities, including higher speeds and ease of use. Both require the WAP gateway, which translates Internet content for downloading on the small PDA.

In Japan, a radically different system called i-mode is used, which provides many services, standardized by the Japanese NTTDoCoMo.

The WAP gateway provides both push and pull operations, unlike the traditional Web, by the use of the push access protocol (PAP) and the over-the-air (OTA) protocols.

In the next chapter, we shall revisit logical mobility in the form of mobile agents and their characteristics.

Problems

1. Identify the requirements for a wireless application protocol, and compare and contrast WAP 1.1 with WAP 2.0, with respect to these requirements.
2. Indicate why the IEEE Standard 802.11 cannot be used in place of WAP 1.1. or WAP 2.0, although it is much faster than both.
3. Show clearly why an http server cannot be used in place of a wireless gateway.
4. WAP 2.0 supports XHTML Basic, which is intended for small wireless devices. It does not support scripts but most of the standard tags. Find out what these are and what their functions are.
5. Look up the relevant book, to find out the services provided by I-Mode, and show how these differ from WAP 1.1.
6. Compare and contrast XHTML and cHTML. Which is more user-friendly?
7. What are scripting languages, and how do they differ from normal programming languages?
8. Write a program in Javascript to accept a number and calculate its factorial. Indicate how this program is different from a normal Java or C++ program.
9. Repeat the program given in Question 8 using PHP script. Which is a better program and why?
10. Devise a scheme for transmitting English text efficiently over a wireless link to a WAP device.

Multiple-choice questions

1. Which one of the following does WAP stand for?
   (a) Wireless access protocol
   (b) Web access protocol
   (c) Web application protocol
   (d) Wireless application protocol
2. Which one of the following is not a WAP programming model component?
   (a) WAP microbrowser
   (b) HTTP server

(c) Push indicator
(d) Pop indicator

3. Which one of the following are the enhancements that WAP has added to the Web programming model?
(a) Push and telephony.
(b) Pop and telephony.
(c) Telephony support only
(d) Pop only

4. WTP stands for which one of the following?
(a) Wireless transport protocol
(b) Wireless transaction protocol
(c) Wired transaction protocol
(d) None of the above

5. Which one of the following features does not belong to WAP 2.0?
(a) Pop
(b) Push as well as pop
(c) Multimedia messaging
(d) Interfaces to storage devices

6. Which one of the following changes to TCP is incorporated in WAP 2.0?
(a) Window size variable
(b) Window size is fixed of size 64 KB
(c) No slow start
(d) Window size is fixed at 64 KB and no slow start

7. WAP 2.0 runs at which one of the following speeds?
(a) 364 Kbps
(b) 384 Kbps
(c) 54 Mbps
(d) 11 Mbps

8. Which one of the following features is provided by the WAP gateway?
(a) Simple gateway functionalities
(b) Simple gateway functionalities plus push operation
(c) Simple gateway functionalities plus on-the-fly image conversion
(d) All of the above

9. Which one of the following are WAP gateway components?
(a) Encoder only
(b) Decoder only
(c) Encoder, decoder and protocol conversion
(d) Encoder, decoder, protocol conversion and WAP protocol stack

10. Push architecture consists of which one of the following?
(a) Push indicator and WAP gateway
(b) Push locater and WAP gateway
(c) WAP gateway and push initiator
(d) None of the above

Further reading

A.S. Tanenbaum (2005), *Computer Networks*, 4th ed. (New Delhi, India: Prentice Hall India).

H. Mei, Y.M. Wen and W.J. Lin (2000), 'Turning an Http Server into a Wireless Internet Gateway', in Proceedings INET2000, Japan, July.

J.R. Vacca (2002), *I-Mode Crash Course* (New York: McGraw-Hill).

N. Frengle (2002), *I-Mode: A Primer* (New York: Hungry Minds).

WAP Forum (2001), 'Wireless Application Protocol Architecture Specification', WAP-210-WAPArch-20010712-a, July 2001, www.wapforum.org (accessed July 2006).

WAP Forum (2001), 'Wireless Application Protocol WAP 2.0 Technical White Paper', August 2001, www.wapforum.org (accessed July 2006).

www.ssimail.com/WAP_gateway.html (accessed November 2003).

www.wapforum.org for information on the WAP Forum (accessed July 2006).

Logical Mobility II—Mobile Agents 10

In Chapter 4, we looked at the migrating process, which can be considered to provide mobility in the logical domain. In this chapter, we revisit logical mobility by considering the mobile agent, which is an extension of the migrating process.

An **agent** is a computer program whose execution is contingent upon events and data conditions in its environment and which is not under continuous, direct control by a human user. Agents make an interesting topic of study because they draw on and integrate so many diverse disciplines of computer science, including objects and distributed object architectures, adaptive learning systems, artificial intelligence (AI), expert systems, genetic algorithms, distributed processing, distributed algorithms, collaborative online social environments and security.

Agent technology is also interesting for its potential to solve some nagging productivity problems that pester almost all modern computer users. Many agents are used as intelligent electronic gophers or automated errand boys. If we want them, for example, to search the Internet for information on a topic, or assemble and order a computer according to our desired specifications, they will do so and inform us when they have finished.

Agents are not a new paradigm, as they have been researched earlier in the area of distributed AI as intelligent agents. In this sense, they can be said to be the successors of the 'actors' of Hewitt. The Web and Java language have given agents a new impetus, in the form of agent-oriented programming.

Some of the many ways in which agents are useful are as follows:

- They simplify distributed computing
- They are intelligent **resource managers**
- They overcome user interface problems
- They act as **personal assistants** which adapt to the user
- They are particularly relevant to telecommunications. Telecommunications companies account for the largest portion of agent research.

In this chapter, we will study mobile agent technology and see the characteristics and uses of mobile agents, especially in relation to mobile computing. We shall see how agents are different from other mechanisms like process migration, client/server architectures, etc. We shall visit three mobile agent platforms and study their salient features. Finally, we shall discuss the merits and demerits of Java language vis-à-vis design support for mobile agents.

10.1 Mobile agents

Mobile agents and mobile code are a new paradigm for distributed computing that complements such technologies as distributed objects, remote compute servers and distributed programming systems such as programmable virtual machine (PVM) and multi-programming interface (MPI). Mobile agents are actually the end points of an incremental evolution of mobile abstractions, starting from simple mobile code. Mobile code is simply a code, as for example in applets. Next comes the mobile object, which is a code with data. Then comes the mobile process, which is a code with data and threads of control. Finally, we have the mobile agent, which has code, data and threads and carries the owner's authority with itself.

Agents that can move themselves to new locations are called mobile agents. More specifically, a mobile agent is an agent that can move at a time of its own choice, from machine to machine, in a heterogeneous network. However, moving to a new location will not be advantageous for all applications or in all network environments. Mobility is best viewed as simply another tool available to the agent programmer, to be used or ignored as the application and its target environment dictates.

Mobile agents are autonomous, intelligent programs that move through a network, searching for and interacting with services on the user's behalf. Mobile agent systems use specialized servers to interpret the agent's behaviour and communicate with other servers. A mobile agent has inherent navigational autonomy and can ask to be sent to some other nodes. It should be able to execute on every machine in a network, and the agent code should not have to be installed on every machine the agent could visit. Therefore, mobile agents use mobile code systems like Java and the Java virtual machine, where classes can be loaded at runtime over the network. Mobile agents may be one-hop, which migrate to one other place only, or multihop, which roam the network from place to place.

A mobile agent is composed of two different pieces. One piece is the code itself, which consists of the instructions that define the behaviour of the agent. The second piece is the current state of execution of the agent. Often, these two pieces are separate. For example, in a typical computer program, the code sits on disk while the executing state sits in RAM. A mobile agent, however, brings the two together. When an agent migrates to a new host, both its code and its state are transferred. Thus, the agent does not only remember what to do and how to do it, it also remembers what it was doing before as well. When an agent migrates to a new host, both of these pieces must be captured and packaged.

10.2 Characteristics of mobile agents

The following characteristics are typical of a mobile agent and its host:

- An agent must have its own unique identity.
- It must be able to determine what other agents are executing on the agent host (AH).
- It must be able to determine what messages other agents accept and send.
- An AH must allow agents to communicate with each other and itself.
- It must allow multiple agents to coexist and execute simultaneously.
- It must be able to negotiate the exchange of agents.
- It must be able to freeze an executing agent and transfer it to another host.
- It must be able to thaw an agent transferred from another and allow it to resume execution.
- It must prevent agents from directly interfering with each other.

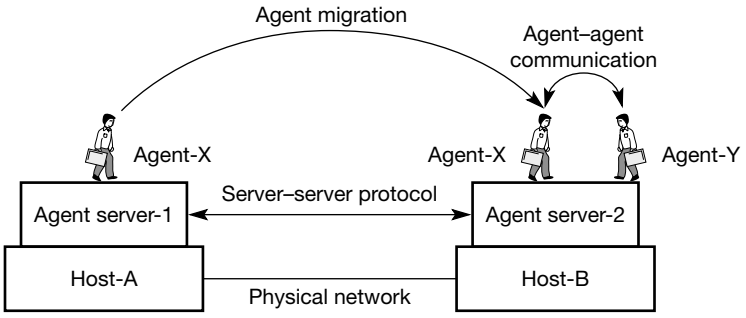


Figure 10.1 Architecture of a Mobile Agent System

10.2.1 Architecture

A simple mobile agent architecture is shown in Figure 10.1. We have two agents, X and Y, originally on host A and host B, supported by agent servers 1 and 2, respectively, and connected through a physical network. A server–server protocol exists between the two hosts to facilitate movement of their agents. When agent X moves to host B and wants to communicate with agent Y, the agent–agent communication protocol allows this to take place. Note that this is a basic mobile agent architecture, which does not support any kind of security for agents and their hosts.

10.2.2 Mobile code and agents

One of the main differences between mobile code, such as applets, and mobile agents is the itinerary. Whereas mobile code usually travels just from point A to point B, mobile agents have an itinerary and can travel sequentially to many sites. One natural application of mobile agents, therefore, is collecting information spread across many computers hooked to a network.

An example of this kind of application is a network backup tool that periodically must look at every disk attached to every computer hooked to a network. Here, a mobile agent could roam the network, collecting information about the backup status of each disk. It could then return to its point of origin and make a report.

10.2.3 Mobile agents and process migration

Mobile agents are programs that can migrate from host to host in a network, at times and to places of their own choosing. The state of the running program is saved, is transported to the new host and is restored, allowing the program to continue where it left off. Mobile agent systems differ from process migration systems discussed in Chapter 4 in that the agents move when they choose, typically through a 'jump' or 'go' statement; whereas in a process migration system, the system decides when and where to move the running process (typically to balance CPU load). Mobile agents differ from 'applets', which are programs downloaded as the result of a user action, then executed from beginning to end on one host.

10.2.4 Client/ser ver and mobile agent architectures

Mobile agents solve the client/server network bandwidth problem. By moving a query or transaction from the client to the server, the repetitive request/response handshake is eliminated.

Agents reduce design risk. Agents allow decisions about the location of the code (client vs. server) to be pushed towards the end of the development effort when more is known about how

the application will perform. In fact, the architecture even allows for changes after the system is built and in operation.

Agent architectures also solve the problems created by intermittent or unreliable network connections. Agents can be built quite easily that work 'offline' and communicate their results back when the application is 'online'.

10.3 Requirements for mobile agent sy stems

There are many technical challenges to implementing mobile agent systems. Most of these problems are in the structure of the computational medium and the environment the agents operate in. Servers must be designed, implemented, and deployed, that not only allow mobile agents to run, but also allow them to run safely.

10.3.1 Portability

Mobile agent code itself must be portable; when an agent arrives at a server, the server needs to be able to execute that agent. Commonly used computer languages such as C and C++ are not very portable. Compiled C code only works on the machine it was compiled for, and the source form is notoriously unportable. Portability can be achieved by running computer programs inside virtual machines interpreters, but overhead has limited the use of interpreted languages. Most mobile agent systems under development now rely at least in part on virtual machines to standardize the execution environment.

10.3.2 Ubiquity

In order for mobile agents to be successful, they need access to many different computer resources. Servers for agents must be commonplace; there needs to be a widely accepted framework for executing mobile agents deployed on many machines across the Internet. In practice, the requirement of ubiquity means that the execution environment needs to have market acceptability, be freely available and be unencumbered by restrictive intellectual property requirements.

10.3.3 Networ k communication

Mobile agents that live in the network need to be written in a language that makes network access simple. It must be easy to transfer objects across the network and to invoke methods of remote objects. Traditional computer languages treat networking structures as an afterthought, usually providing only a minimal socket library.

Languages that better support network access have typically not been widely used. This situation is improving with the current development of language-neutral distributed object frameworks such as Common Object Request Broker Architecture (CORBA) and OLE (object linking and embedding).

10.3.4 Ser ver security

A major concern specific to mobile agents is the protection of the servers running the agents. Running arbitrary programs on a machine is dangerous: A hostile program could destroy the hard drive, steal data or do all sorts of other undesirable things. This risk must be thoroughly addressed if mobile agent environments are to succeed. Two types of security are possible to protect servers from malfunctioning and hostile agents—physical and social.

Physical security refers to building servers for agents in such a way that the agents cannot harm the server. The 'laws of physics' of the server execution environment can be designed to

make dangerous operations difficult or impossible. Common approaches involve creating a 'sandbox' for visiting agents, restricting access to resources (preventing disk writes, for instance) and ensuring that the agent cannot escape those restrictions. This approach to security is attractive; when it works, it is entirely effective. But the viability of physical security in the face of design complexity and server implementation bugs is unclear. In addition, physical security is typically focused on protecting some underlying aspect of the server from the sandbox the agent is trapped in. But if multiple agents are put in the same sandbox, how can the server guarantee that one agent cannot harm another? As we put more trust in the computations that take place inside sandboxes, the security of those sandboxes themselves becomes important.

A second approach to server security is using social enforcement mechanisms to punish the creators of harmful agents. If a server administrator can find out who is responsible for a malicious agent, then that person can be held accountable via social mechanisms (such as lawsuits). Digital signature technology makes identifying the authors of agents possible. But there are limitations to a purely social approach to security. It may not be clear which agent is responsible for damage, nor will it be easy to determine ahead of time which agent authors are trustable. In practice, some combination of social and physical enforcement of server security will be useful.

10.3.5 Agent security

The complement of server security is agent security: Whether the agent can trust the server on which it is executing. A mobile agent might contain secret information such as proprietary data and algorithms. Worse, servers might have an incentive to subvert the computation of a visiting agent. In the Internet-based Digital Encryption Standard (DES) cracking effort currently under design, a major concern is protecting the computation from sites that pretend to do pieces of the problem but return false answers. Physical security answers to this problem are difficult. Secure, trusted hardware on the server could guarantee agent safety but is unlikely to be widely deployed. Agent programmers can protect their agents by obfuscating their code and by verifying the results of the remotely performed computation, but the general applicability of these techniques are unknown. Social solutions may be possible in the form of reputation systems for servers. This area of security has largely been unexamined.

10.3.6 Resource accounting

If economic control and incentive are going to be factors in Netwide resource use, some mechanism to account for the resources that an agent uses and a way for receiving payment for those resources is necessary. In theory, these requirements are not difficult to meet. Servers can keep track of the resource usage of agents, explicitly accounting CPU, memory, bandwidth and disk usage. Digital cash systems can be used to pay for services. In practice, these technologies are not widely deployed, and the overhead they impose presents an engineering challenge.

10.4 Mobile agent platfor ms

Many mobile agent systems have been developed over the last two decades. These include aglets, Agent Tcl, Voyager and Concordia. See Nelson (1999) for details of these systems, along with their comparison. In this chapter, we shall study only two of the more popular mobile agent platforms available to the user. These are the Aglets and Agent Tcl. We also present briefly the PMADE (platform for mobile agent development and execution) agent platform developed at IIT Roorkee.

10.4.1 Aglets

Aglets are a Java-based framework for mobile agents, designed by IBM. With aglets, one can construct objects that can move from one host on the network to another; that is, an aglet that executes on one host can suddenly halt execution, dispatch to a remote host and start executing again. When the aglet moves, it takes along its program code as well as the states of all the objects it is carrying. The aglet framework has been developed by the IBM Tokyo Research Laboratory, Japan.

10.4.1.1 The aglet object model

The aglet object model is different from the distributed object model, because here computation itself is transmitted, while in distributed object models, the requests for remote methods are transmitted. Figure 10.2 shows the aglet object model.

This model defines a set of abstractions and the behaviour needed to leverage mobile agent technology in wide area networks like the Internet.

An aglet is a mobile Java object that visits aglet-enabled hosts in a computer network. It is **autonomous**, since it runs in its own thread of execution after arriving at a host, and **reactive** because of its ability to respond to incoming messages. The aglet life cycle model is shown in Figure 10.3.

A **context** is an aglet's workplace. It is a stationary object that provides a means for maintaining and managing running aglets in a uniform execution environment, where the host system is secured against malicious aglets. One node in a computer network may host multiple contexts.

A **proxy** is a representative of an aglet. It serves as a shield for the aglet that protects it from direct access to its public methods. The proxy also provides location transparency for the aglet. That is, it can hide the real location of the aglet.

A **message** is an object exchanged between aglets through proxies only. It allows for synchronous as well as asynchronous message passing between aglets. Message passing can be used

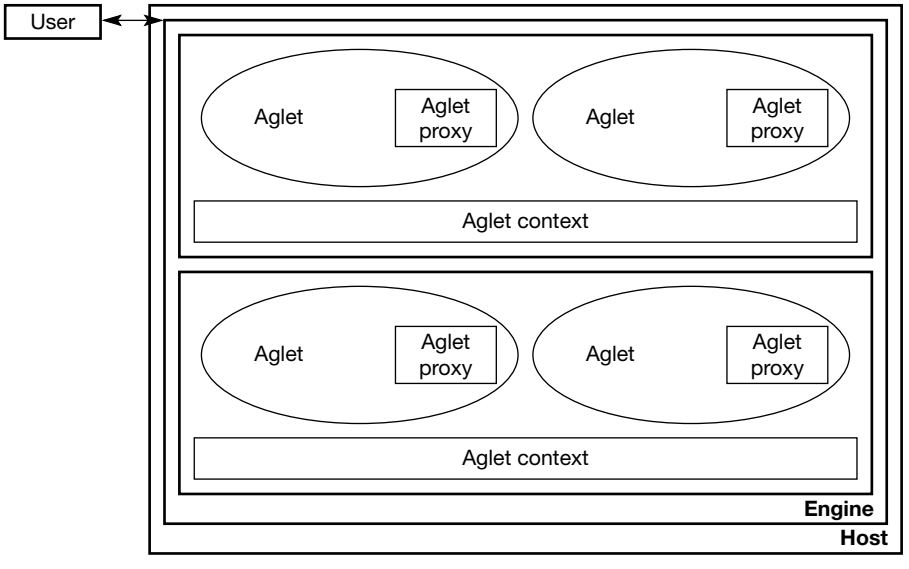


Figure 10.2 The Aglet Object Model

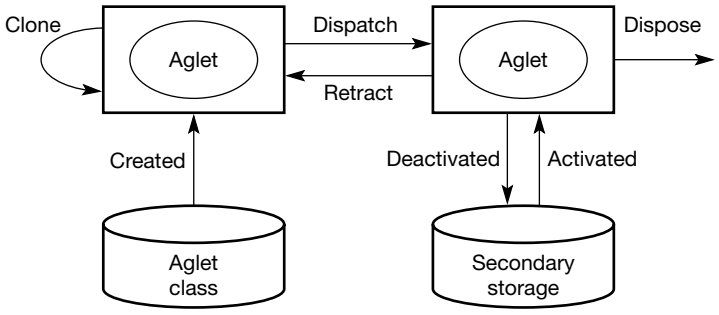


Figure 10.3 The Aglet Life Cycle Model

by aglets to collaborate and exchange information in a loosely coupled fashion. The **message manager** allows for concurrency control of incoming messages.

An **itinerary** is an aglet's travel plan. It provides a convenient abstraction for non-trivial patterns and routing.

An **identifier** is bound to each aglet. This is globally unique and immutable throughout the lifetime of the aglet.

The **creation** of an aglet takes place in a context. The new aglet is assigned an identifier, inserted in the context, and initialized. The aglet starts executing as soon as it has successfully been initialized.

The **cloning** of an aglet produces an almost identical copy of the original aglet in the same context. The only differences are the assigned identifier, and that the execution restarts in the new aglet. Notice that execution threads are not cloned.

Dispatching an aglet from one context to another will remove it from its current context and insert it into the destination context, where it will restart execution (execution threads will not migrate). We say that the aglet has been pushed to its new context.

The **retraction** of an aglet will pull (remove) it from its current context and insert it into the context from which the retraction was requested.

The **deactivation** of an aglet is the ability to temporarily remove it from its current context and store it in secondary storage. **Activation** of an aglet will restore it in a context.

The **disposal** of an aglet will halt its current execution and remove it from its current context.

Messaging between aglets involves sending, receiving and handling messages synchronously as well as asynchronously.

A **naming mechanism** automatically assigns immutable identities to new aglets. These identities should be guaranteed to be globally unique.

10.4.1.2 Aglet communication

- Aglets use messages and events for communication and interactions.
- Message communication can be either synchronous or asynchronous.
- Events can be used for either reactive or proactive agents; for example, an aglet agent can listen for some event such as a change in stock value.
- An aglet package supplies some default events, like cloning events, mobility events and persistence events. Aglet API (application-programming interface) is a Java package consisting of classes and interfaces, namely, aglet, aglet proxy and aglet context and message.

10.4.1.3 The aglet event model

The aglet programming model is event-based. The model allows the programmer to 'plug in' customized listeners into an aglet. Three kinds of listeners exist:

- **Clone listener:** Listens for cloning events. One can customize this listener to take specific actions when an aglet is about to be cloned, when the clone is actually created, and after the cloning has taken place.
- **Mobility listener:** Listens for mobility events. One can use this listener to take action when an aglet is about to be dispatched to another context, when it is about to be retracted from a context, and when it actually arrives in a new context.
- **Persistence listener:** Listens for persistence events. It allows the programmer to take action when an aglet is about to be deactivated and after it has been activated.

Application: Aglets have been applied in various scenarios, including e-commerce, the e-market place, air tickets and package tours.

10.4.2 Agent Tcl

Agent Tcl is a powerful Internet agent system that runs on UNIX workstations and allows the rapid development of complex agents. It has been specially developed to target the needs of mobile computers. The architecture builds on the server model of Telescript, supports multiple languages, and the transport mechanisms of two predecessor systems at Dartmouth.

10.4.2.1 Agent Tcl architecture

The architecture has four levels. See Figure 10.4(a). The lowest level is an API for the available transport mechanisms. The second level is a server that runs at each network site to which agents can be sent. Figure 10.4(b) shows the various support systems available with Agent Tcl. These are explained in detail later.

The server performs the following tasks:

1. **Status:** The server keeps track of the agents that are running on its machine and answers queries about their status.
2. **Migration:** The server accepts each incoming agent, authenticates the identity of the owner and passes the authenticated agent to the appropriate interpreter for execution.

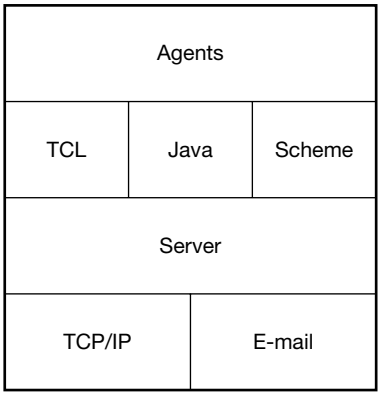


Figure 10.4(a) Architecture of Agent Tcl

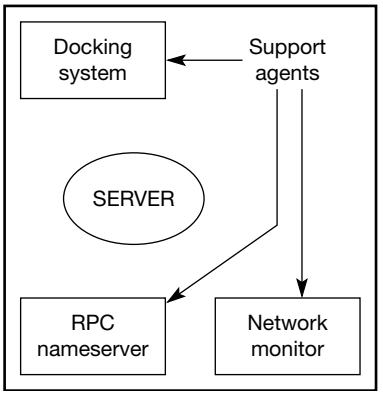


Figure 10.4(b) Tcl Suppor t Agents

3. **Low-level interagent communication:** The server provides a hierarchical namespace for agents and allows them to send messages to each other within this namespace. The topmost division of the namespace is the symbolic name of the agent's network location. A **message** is an arbitrary sequence of bytes with no predefined syntax or semantics except for two types of distinguished messages.

   An **event** message provides asynchronous notification of an important occurrence, while a **connection** message requests or rejects the establishment of a direct connection. A direct connection is a named message stream between agents and is more convenient and efficient than message passing for long interactions, since the programmer can wait for messages on a particular stream, and the server often can hand control the stream to the interpreter. The server buffers incoming messages, selects the best transport mechanism for outgoing messages and creates a named message stream once a connection request has been accepted.
4. **Non-volatile store:** The server provides access to a non-volatile store so that agents can back up their internal state as desired. The server restores the agents from the non-volatile store in the event of machine failure.

Agents provide all services that are available within the system, whether mobile or stationary. At each Agent Tcl site, a server resides and handles the management of local agents and incoming mobile agents. The server also provides mechanisms for enforcing security, providing a hierarchical namespace in which agents can be referenced and allowing agents to address each other locally.

Agents move between sites by issuing the mobility primitive *agent_jump*. This command packages the program context of the agent and transfers it to a destination site where the server restarts it at the instruction after the *agent_jump* command. The method in which the agent is transported is determined by the transport system advocated by the local site server, for example, TCP/IP, e-mail and so on.

An **interpreter** that is appropriate to the source language of the mobile agent handles agent execution and provides an execution environment for the agents. Agents can be written in a language that supports interpretation, but the authors indicate that compiled agents might be possible in a limited capacity. For example, an agent written and compiled in C might be able to execute but not migrate due to its platform dependence.

In Agent Tcl, the interpreter of Tcl was extended to support three extra modules:

- **A security module** to prevent agents from performing malicious actions against the system.
- **A server API module** that allows communication with the server to facilitate migration, communication and checkpointing. It contains mostly a C/C++ library, that is shared among all interpreters and also language-specific stubs.
- **A state capture module** to capture and restore the state component of an agent when it wishes to migrate to a new location.

Agent Tcl uses pretty good privacy (PGP) to authenticate servers and protect agents and data while in transport. However, since there is no automated procedure for distributing PGP public keys, each server must possess the keys of all servers from which it might receive agents in advance. Based upon this authentication, a **resource manager** assigns an agent an appropriate set of access permissions in order to protect resources. **Built-in resources** are directly accessible through the language primitives (such as the system clock, the CPU and such like), but **indirect resources** are managed and protected through a third-party agent. Safe Tcl is used to protect built-in resources and provide two interpreters for the language.

A **trusted interpreter** is one in which all of the Tcl commands are made available to the agent. **An untrusted interpreter** is one in which the more 'dangerous' commands are removed, such as opening files, creating network connections and invoking the computer's bell. However, to prevent agents from being too restricted, agent Tcl uses a modified version of SafeTcl in which dangerous commands are removed and replaced with a link to a secure version. The secure version of each command either uses an access control list to determine whether or not the agent can execute the command (matched against the agent's access permissions) or severely restricts the operation of the original command.

Agents use the *agent_meet* command to initiate a communication with another agent and *agent_accept* to complete and synchronize the two agents. In addition, agents can communicate in an asynchronous fashion using the *agent_send* and *agent_receive* primitives.

Agent-level support system This module provides high-level functionality to the agents. It is shown in Figure 10.4(b).

Firstly, it allows navigation of agents between remote systems, by providing docking stations. Docking is shown in Figure 10.5. Here, mobile computers (laptops) are shown as pentagons and permanently connected machines are shown as hexagons. The docking system is used to pair each mobile computer with a permanently connected 'dock' machine. An agent jumping to or from a laptop is shown in Figure 10.6. When a mobile agent is unable to migrate to a laptop machine (1), it waits at its dock machine (2). When the laptop reconnects (3), it forwards its address to the 'dock' (4), which sends all waiting agents to it (5).

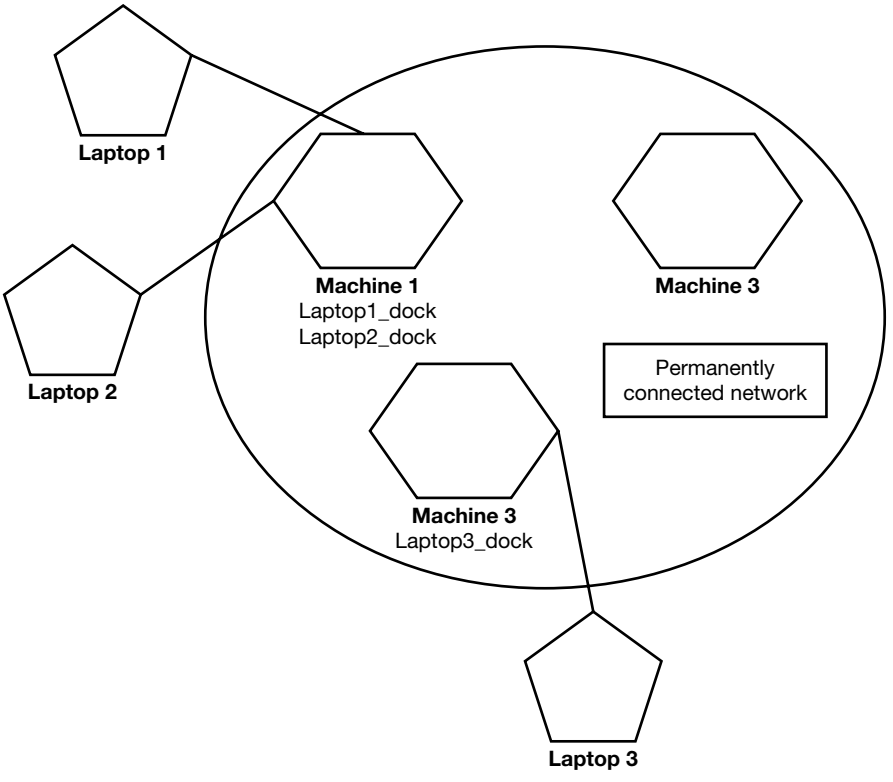


Figure 10.5 Docking System in Agent Tcl

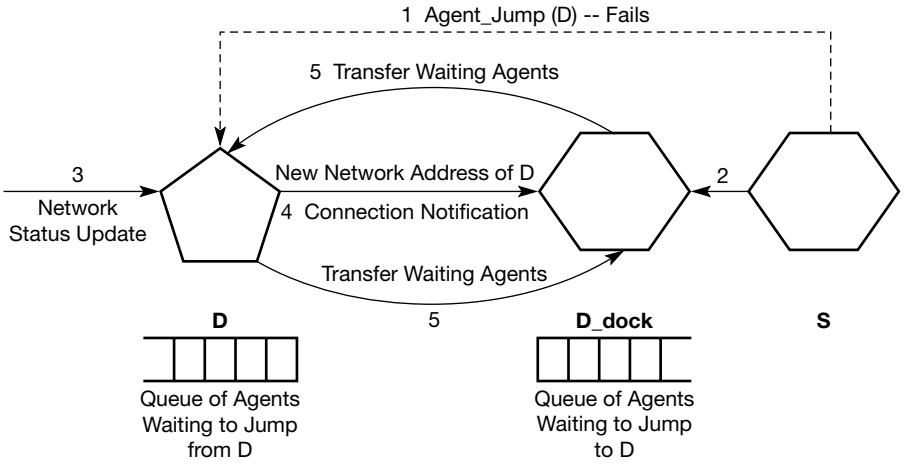


Figure 10.6 Agent Jumping from Laptop

Secondly, a high-level communication mechanism using remote procedure call (RPC) is provided, over and above the simple low-level messages and streams.

Thirdly, it also provides a resource management function which can be used, for example, for accounting in commerce applications and control to prevent denial-of-service attacks. However, no protection is provided for agents from malicious machines.

10.4.2.2 Agent Tcl applications

The Agent Tcl architecture has been used in many information-retrieval applications, like in technical reports, for text-based medical records or in three-dimensional drawings of mechanical parts.

We give a sample code in Figure 10.7(a) to illustrate the suitability of Agent Tcl in a network management application, where the agent makes a list of all users logged onto some machines on a network and shows the result to the owner. The UNIX *who* command is used for this purpose. Figure 10.7(b) shows three machines, Beta, Mu and Theta, connected in a network for this application.

10.4.3 PMADE

PMADE is a mobile agent platform designed and implemented as a research work by students. It is a Java-based platform and has provision for agent–agent communication, agent itineraries, and agent and host security, besides agent development and execution.

```
agent_begin;        registration on home machine
set output { };             set machine list {mu,  theta,. . .}
for each machine $machinelist {
      agent_jump $machine
      append output [exec who]
      }
agent_jump $agent (home)
#display results
agent_end;                  finished
```

Figure 10.7(a) Sample Code for Agent Tcl Application

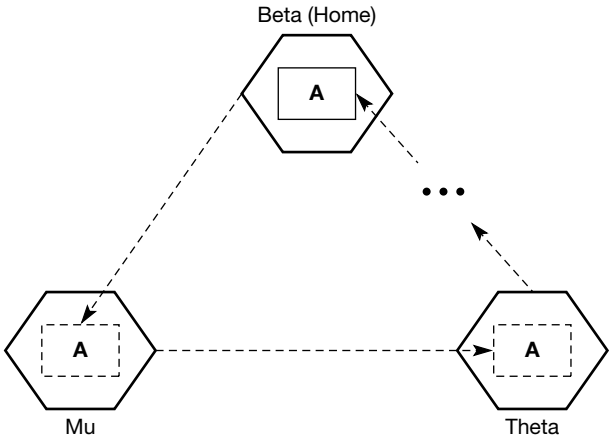


Figure 10.7(b) Machine Setup for Agent Tcl Application

The main blocks of PMADE are shown in Figure 10.8. Each node of the network has a server called agent host (AH), which accepts and executes incoming agents, and a client called agent submitter (AS), which submits the agent on behalf of the user to the AH. When a user wants to perform a task, he/she submits the agent, designed to perform that task, to the AS on the user system. The AS establishes a connection with the specified AH, where the user is registered, submits the agent and goes offline.

The AH examines the nature of the agent and, if required, clones it and forwards it to other active AHs in the network. It then goes on to execute one clone. The execution of the agent depends on its nature and state. It can be transferred from one AH to another whenever required. On completion of execution, it submits its results to the AH, which in turn stores the results, until the AS receives them for the user. AS and AH are discussed in detail in the following sections.

10.4.3.1 Agent submitter

The AS plays a crucial role in formulating and dispatching agents to the AH. It acts as an interface between the user and the AH. One of its primary jobs is to attach a signature to the invoked agent. It retrieves the static Internet protocol (IP) address of the host on which it is running and binds it to an agent signature.

The AS also receives replies from the AH for user requests. It keeps track of agents and maintains a profile that it submits to the AH from the user's system. This architecture allows a user to go offline after submitting its agents and receive results on reconnection to the network. The AS functions as follows:

- Receives an agent and verifies whether it is listed in its database.
- If found, checks the status of AH, connects to it, and sends the agent to the AH.
- If connection is unsuccessful, sends an appropriate reply to user.
- If agent reaches AH successfully, receives acknowledgement from the AH.
- Receives results from AH and stores on the local disk for future references by the client.

The various components of AS are shown in Figure 10.9 and discussed below.

The **result manager** provides a means to transport agent results from a remote AH. It receives results from an AH and stores them in the secondary storage device for future references by

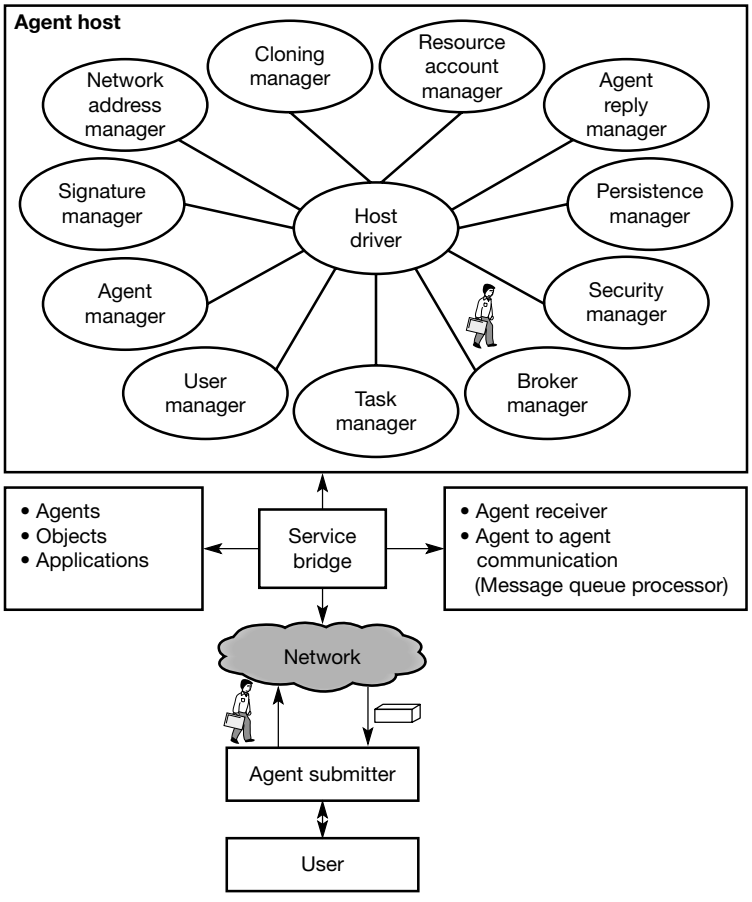


Figure 10.8 Architecture of PMADE

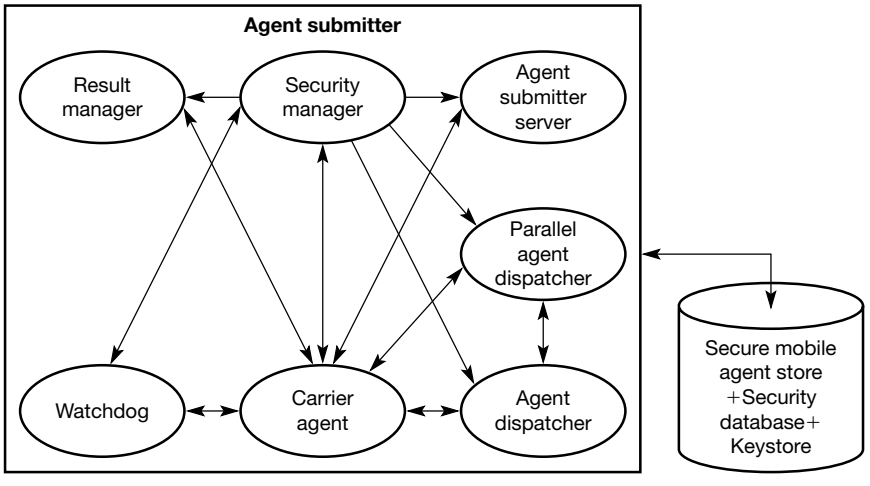


Figure 10.9 Architecture of Agent Submitter

the user. It uses the IP address of the last visited host, the agent name and version of the agent and creates a file to store the result. When a user enquires about the result of its dispatched agent, it calls the security manager to decrypt the result before displaying it. The security manager is discussed later.

The **carrier agent** provides the basic functionality of a mini Web server specialized to load class files from the file systems. It uses default host port 8081, which can be changed if required.

The **agent dispatcher** provides an interface for dispatching an agent on its mission. On instruction from an AS/AH/agent, it first checks the status of the host where the agent has to be transferred. If the host is active, the agent is transferred and an appropriate message is returned. If the host is passive or some problem occurs during transfer, it receives information about the next host from the itinerary. It maintains a copy of every agent it dispatches, that is, its code, method, data, certificate, etc.

The **watchdog** helps an agent launcher to trace the agent if it does not report back in a specified period of time, so that it can be restarted by the AS. It maintains information about the agents' currently visited host and where it is migrating next. This information helps the AS to identify the portion of the job the agent has completed at any time.

The **agent submitter server** is used if an agent has been tampered with. A fresh copy of the tampered agent's method is sent to the requesting host. It uses the agent's stored information to provide a runtime solution to the tampered agent; that is, it provides a sealed method to the host from where request is generated. Thus, it maintains the integrity and confidentiality of the agent.

The **parallel agent dispatcher** is used to identify groups of methods of similar tasks. Such a group is considered to be one subagent. It identifies how many threads need to be started to dispatch the subagents. Each thread calls the agent dispatcher to dispatch a subagent to the first active host in their group. The thread stops when it has finished transferring the subagent associated with it. These subagents follow their itineraries to perform their assigned tasks on the network, submit their results to the result manager and die.

10.4.3.2 Agent host

The AH is the key component of PMADE. It consists of the manager modules and the host driver. The host driver is the basic utility module, lying below the manager modules, and is responsible for driving the AH, by ensuring proper coordination between the various managers and by making them work in tandem.

The Service Bridge provides a hosting facility to the carrier agent, which regularly monitors requests issued by the AH, to transfer agents to other AHs. Various manager modules help to perform functions like agent transfer, execution, communication, etc. These managers are grouped into four categories and discussed next.

10.4.3.3 Communication managers

Agent manager This manager receives the agent's parameters like agent name, version, signature and sending entity address from the previous AS/AH and sends them to the signature manager to check whether the agent is a duplicate or a new one. The result of the check is sent to the requesting AS/AH.

Network address manager This manager is called when an agent has to travel to other AHs. It keeps track of the AHs to which it is supposed to communicate directly for the purpose of

forwarding agent clones. It does this by using a database of IP addresses/URLs, which is maintained by the administrator. A list of active AHs is also maintained to identify hosts on which AH services are currently running. For this, it accesses the address database. It tries to contact those addresses by calling the space status module of the resource account manager. If a connection is established, an acknowledgement is received from the remote AH, and an entry is made in the list. A new list of active AHs is prepared every time an agent wants to travel, and the old list is removed.

Agent reply manager This manager sends/receives results to/from an AH/AS. It keeps on polling to establish a connection with the AS to whom the results are to be sent, if it is unreachable. When an AS comes online, the agent reply manager ensures that results are delivered to it. For this, it maintains a table of all the agent signatures for which it has to send a reply and a buffer to store the results. It is also required to reinject those agents back into their itineraries that had transferred themselves to it when other hosts were unreachable. It provides base server functionality to PMADE and executes special agent code to reroute the agent when a failure occurs in the network. This manager works like a temporary shelter provider to agents.

10.4.3.4 State managers

Task manager When an AS/AH has to send an agent, it calls the task manager of the remote AH. The task manager uses the services of the user manager for authentication and the agent manager for transferring the agent. It creates and maintains a multilevel queue of agents, depending on their nature. It reserves the power to terminate and pre-empt the resources held by an agent when it runs in an infinite loop or tries to access protected area. Such an agent is terminated, and its owner is informed. Entries of the agent are removed from the signature table in FIFO (first-in, first-out) fashion, after a fixed interval of time. This is an optional property and can be set or reset by the system administrator. It also dispatches an agent, which has finished its assigned task on the current AH, to another AH for further processing.

Signature manager This manager waits in a loop for an agent to arrive. When the agent arrives, it generates a suitable acknowledgement and sends it to the agent manager. It also does bookkeeping for the AH and maintains records of signatures of all agents that have been accepted, so that duplicates can be avoided.

Cloning manager This manager makes 'clones' of agents received from remote hosts and dispatches them by starting multiple threads. Each thread dispatches one clone to the hosts. When the clones finish their assigned tasks, their results are stored in the result store, and the thread is stopped.

Resource account manager This manager is responsible for monitoring the resources used by agents on the AH. It calculates the cost of resources used by an agent, on the basis of parameters such as processor time consumed, bandwidth utilized and amount of primary and secondary storage used. These parameters can be modelled to suit the specific needs of the consumer and the service provider. It also performs distributed garbage collection for agents running on the AH. The main components of this manager are as follows:

(a) **Space status module:** This module is activated when a request is received from the remote/local AH that wants to forward an agent for execution. It indicates the status of availability of execution space and sends an appropriate acknowledgement to the AH, indicating whether the request is granted or rejected.

(b) **Agent execution resource metering:** This module actually receives an agent from the local or remote AH and provides an execution environment to it. During agent execution, it measures its execution time, as also the amount of primary and secondary storage used by it. If required by the application developer, it also authenticates clones against tampering.
(c) **Garbage collection:** This is an important function, since agents arrive and leave repeatedly. When an agent arrives at an AH, it is registered; and if it does not get immediate charge of the CPU and other required devices, it waits in a queue. The class file of the received agent is maintained in secondary storage and its other attributes in a queue. Its signature is maintained in the Accidental Crash Database, and it is registered in the Registration domain (directory). When the agent finishes its assigned task, its result is saved in the Result Store, and all these entries are removed from the AH. Entries made in the primary memory are also removed.

Broker manager Brokering is a major requirement in e-commerce applications and can benefit vastly by the use of agents. There may be various static agents and applications running on an AH, providing different services at a cost. The broker manager helps to guide an incoming agent to the appropriate static agent on the host. It maintains a record of all these service agents, the services offered and their charges, etc., and advertises these to broker managers on other hosts. It also facilitates code on demand service to agents and AH.

10.4.3.5 Persistence manager

As discussed in Section 2, some mechanisms are required in mobile agent systems (MAS) to take care of agents when the system crashes. The most common mechanism is checkpointing. The persistence manager is called whenever an AH goes down or comes up again. It allows the system to go offline for regular maintenance, yet not lose agents running on the AH. Hence, it saves the AH from restarting agent execution from scratch. When an AH has to be shut down for some reason, it saves the states of all agents residing on it. When the AH comes up again, it retrieves the agents, along with their states, from the secondary storage and restores their status, as it was before going offline.

10.4.3.6 Security manager

PMADE has been designed to provide an integrated approach to security. We have implemented mechanisms to protect local resources from malicious agents and agents from malicious hosts and for communication among agents across the network and in local host entities. A large variety of mechanisms, policies and tools is available in PMADE to achieve flexible levels of security. These properties make it suitable for the design and implementation of distributed services in several application areas, namely, mobile computing, distributed database retrieval, network management and information distribution.

Details about the security manager of PMADE are given in Patel and Garg (2005). Here, it suffices to say that a **user manager** is used which is responsible for authenticating a user's account. It deals with the identification and verification of users on the AH. It is invoked when an AH receives a connection request from an AS/AH and sends an acknowledgement to the requesting AS/AH. It keeps records of all user certificates in a database.

It has to be emphasized that there are many issues of importance to mobile agents and their systems. These have to do with agent communication, agent itineraries, agent security, agent naming and agent location, to name the more important ones. But these are beyond the scope of this book. The interested reader is advised to see the related references listed in the Further Reading for more material on these topics.

10.5 Java and mobile agents

Java language has changed the Web overnight and offers unique capabilities that are fuelling the development of mobile agent systems. Java is a powerful tool for mobile agent development, as seen in the following discussion.

10.5.1 Advantages of Ja va

1. Platform independence: Java is designed to operate in heterogeneous environments. To enable a Java application to execute anywhere on the network, the computer generates architecture neutral byte code as opposed to non-portable native code. For this code to be executed on a given computer, the Java runtime system must be present. This allows us to create a mobile agent without knowledge of the types of computers it will run on.

2. Dynamic class loading: This mechanism allows the virtual machine to load and define classes at runtime. It provides a protective namespace for each agent, thus allowing agents to execute safely and independently of each other. The class-loading mechanism is extensible and enables classes to be loaded via the network.

3. Multithreaded programming: An agent is by definition autonomous and executes independently of other agents residing within the same place. Allowing each agent to execute in its own lightweight process, also called 'thread execution', is a way to enable agents to behave autonomously. Fortunately, Java not only allows multithreaded programming, it also supports a set of synchronization primitives that is built into the language and enables agent interaction.

4. Object serialization: A key feature of mobile agents is that they can be serialized and deserialized. Java provides a built-in serialization mechanism that can represent the state of an object in a serialized form, sufficiently detailed for the object to be reconstructed later. The serialized form must be able to identify the Java class from which the object's state was saved and to restore the state in a new instance. Objects often refer to other objects. To maintain the object structure, these other objects must be stored and retrieved at the same time. When an object is stored, all the objects in the graph that are reachable from that object are also stored.

5. Reflection: Java code can discover information about the fields, methods and constructors of loaded classes and can use reflected fields, methods and constructors to operate on their underlying counterparts in objects, all within the security restrictions. Reflection accommodates the need for agents to be smart about themselves and other agents.

6. Secure execution: The execution environment in the server should be designed so as to make dangerous operations difficult or impossible. One of the approaches towards solving this problem involves creating a Java-provided 'sandbox' for visiting agents. Once the agent is inside the sandbox, it has restricted access to resources, as we saw in Chapter 2. Thus, disk writes are prevented, and access to blocks of memory is limited. This results in host protection from malicious agents. This concept does not, however, address the issue of protection of multiple agents from each other, once they are in the sandbox. To circumvent this problem, we can have a separate 'child sandbox' for each incoming agent and one 'parent sandbox' that houses all these child sandboxes.

10.5.2 Shortcomings of Ja va

All the above features make Java a very useful language for designing mobile agents. However, Java has a number of shortcomings also, as discussed below:

Measuring CPU bandwidth and memory usage in a Java interpreter is fairly simple, although the problem becomes more difficult in compiled code. The thread scheduler is probably the right place to add in the CPU accounting, the garbage collector can determine

memory usage and the socket classes can track bandwidth usage. Implementing these systems efficiently will be challenging.

Java also currently has **no explicit support for digital cash payments**. The cryptographic basics of these systems are possible in any language; preliminary Java implementations do exist. A secondary issue is the actual distribution of digital cash. Is there an easy and secure way to give an agent digital money to carry along with it as it executes? Efficiency of handling payments is also a concern.

Java does **not support preservation and resumption of full execution state**. Information such as status of PC and frame stack is forbidden territory for Java programs. Hence, for a mobile agent to properly resume a computation on a remote host, it must rely on internal attribute values and external events to direct it. An embedded automation can keep track of the agent's travels and ensure that computations are properly halted and resumed.

In spite of the above shortcomings, Java is still the best-known option for designing efficient and secure mobile agents. Some details of network programming in Java are given in the appendix at the end of this book.

10.6 Summary

Mobile agents are becoming very popular for distributed, network and Web-based programming, especially where heterogeneous systems are involved, since they are autonomous entities and work on behalf of the owner. There are many advantages in using mobile agents, as compared to traditional client-server programming. However, mobile agent deployment also requires that the systems they run on be secure and support a suitable environment for their creation, transport and storage.

A variety of today's online applications are amenable to mobile agent-based implementations, including network management, e-commerce, information retrieval, software distribution and system administration.

Many mobile agent platforms have been developed over the years, the more popular of them being the aglets, the Agent Tcl and the PMADE system developed at IIT Roorkee by the author and her research student. Many more agent systems are available today, but discussion about these is beyond the scope of this book.

Java is very useful as a programming language for implementing mobile agents because of its numerous characteristics like platform independence, object serialization, dynamic class loading, multithreading and secure execution.

In the next chapter, we will discuss security issues for wireless and mobile systems.

Problems

1. Discuss how mobile agents are different from mobile code and mobile objects.
2. How is the mobile agent paradigm superior to the client-server paradigm? Explain.
3. Give the requirements for the design and deployment of mobile agent systems.
4. Discuss why Java language is suitable for programming mobile agents.
5. Download the aglet environment and API from the IBM site, and design and implement a 5-node network monitoring system using it. Include features such as finding out what resources are available on a node and what its current CPU utilization is.

6. Using aglets, design and implement an information retrieval system where an agent is sent to various ecommerce/vendors sites to collect information about the price of a commodity.
7. Design and implement a software distribution system using mobile agents, which takes a piece of software code and distributes it to various nodes in a network.
8. Repeat Question 5 using Agent Tcl.
9. Repeat Question 6 using Agent Tcl.
10. Compare and contrast the features of PMADE, aglets and Agent Tcl.

Multiple-choice questions

1. An agent is said to be reactive by which one of the following properties?
   (a) Ability to respond to incoming messages
   (b) Runs in its own thread of execution
   (c) Autonomous
   (d) Has inherent navigational autonomy
2. Which one of the following is not considered as a requirement for mobile agent systems?
   (a) Portability
   (b) Security
   (c) Ubiquity
   (d) Resource availability
3. Which one of the following features of Java enables agent interaction?
   (a) Synchronization
   (b) Reflection
   (c) Multithreading
   (d) Object serialization
4. The *agent_accept* command in Agent_Tcl is used for which one of the following functions?
   (a) Initiate a communication
   (b) Synchronize the two agents
   (c) Start agent execution on remote machine
   (d) Negotiate agent exchange
5. Which one of the following is considered as an end point of an incremental evolution of mobile abstractions?
   (a) Mobile code
   (b) Mobile agent
   (c) Mobile process
   (d) Mobile object
6. Agent–agent communication allows an agent to do which one of the following actions?
   (a) Transfer it to another host
   (b) Communicate with other agent
   (c) Bring it from another host
   (d) Determine what messages it accepts
7. Which one of the following does an itinerary not provide to a mobile agent?
   (a) Convenient abstraction for non-trivial patterns and routing
   (b) A travel plan

(c) Parallel processing by cloning
(d) Message sending to other agent

8. Aglet programming model is considered to be which one of the following?
(a) Event-based
(b) Process-based
(c) Object-based
(d) Thread-based

9. Agent Tcl uses pretty good privacy (PGP) for which one of the following functions?
(a) To encrypt servers
(b) To authenticate servers
(c) To authorize servers
(d) None of the above

10. Which one of the following does the agent level support system in Agent Tcl provide?
(a) Navigation of agents between remote systems
(b) High-level communication mechanism
(c) Provides a resource management
(d) All of the above

Further reading

A. Bieszczad, B. Pagurek and T. White (1998), 'Mobile Agents for Network Management', *IEEE Communications Surveys*, 1(1): 2–9.

A. Carzaniga, G.P. Picco and G. Vigna (1997), 'Designing Distributed Applications with Mobile Code Paradigms', 19th International Conference on Software Engineering, Boston, Massachusetts, May 17–23.

C. Hewitt (1977), 'Viewing Control Structures as Patterns of Passing Messages', *Artificial Intelligence*, 8(3): 323–64.

D. Kotz, R. Gray, S. Nog, D. Rus, S. Chawla and G. Cybenko (1997), 'Agent TCL: Targeting the Needs of Mobile Computers', *IEEE Internet Computing*, 1(4): 58–67.

D.B. Lange and M. Oshima (1998), *Programming and Deploying Java Mobile Agents with Aglets* (Addison-Wesley, USA).

D. Lange and M. Oshuna (1998), 'Mobile Agents with Java: The Aglet API', *World Wide Web*, 1(3).

E.N. Elnozahy and Z.W. Manetho (1992), 'Transparent Rollback-Recovery with Low Overhead, Limited Rollback, and Fast Output Commit', *IEEE Transactions on Computers, Special Issue on Fault-Tolerant Computing*, 41(5): 526–531.

E.N. Elnozahy, A. Lorenzo and B.J. David (2002), 'A Survey of Rollback-Recovery Protocols in Message-Passing Systems', *ACM Computing Surveys*, 34(3): 375–408.

E.R. Harold (2000), *Java Network Programming* (Cambridge: O'Reilly).

J. Nelson (1999), *Programming Mobile Objects with Java* (New York: Wiley).

L. Moreau (1999), 'A Distributed Garbage Collector with Diffusion Tree Reorganization and Mobile Objects', *ACM Sigplan Notices*, 34(1): 204–215.

M. Feridun and J. Krause (2001), 'A Framework for Distributed Management with Mobile Components', *Computer Networks*, 35: 25–38.

M. Hughes, M. Shoffner and D. Hammer (2001), *Java Network Programming* (Pune, India: Computer Hut).

M. Kasbekar, C. Narayanan and C.R. Das (1999), 'Selective Checkpointing and Rollbacks in Multithreaded Object-Oriented Environment', *IEEE Transactions on Reliability,* 48(4): 325–337.

M. Shapiro, P. Dickman and D. PlainFossé (1992), 'SSP Chains: Robust, Distributed References Supporting Acyclic Garbage Collection', Technical Report 1799, INRIA, Rocquencourt, France, November.

N.M. Karnik and A.R. Tripathi (1998), 'Design Issues in Mobile-Agent Programming Systems', *IEEE Concurrency*, 6(3): 52–61.

R.B. Patel (2002), 'Manual of PMADE', Internal Report, Department of E&CE, Indian Institute of Technology, Roorkee, India, January.

R.B. Patel and K. Garg (2003), 'A Security Framework for Mobile Agent Systems', in Proceedings of the 6th International Conference on Business Information Systems (BIS 2003), Colorado, Spring, USA, June 4–6, pp. 49–56.

R.B. Patel (2004), 'Development of a Platform for Mobile Agents Development and Execution', Ph. D. thesis, IIT Roorkee, India.

R.B. Patel and K. Garg (2005), 'A Flexible Security Framework for Mobile Agent Systems', *Journal of Control and Intelligent Systems*, 33(2): 175–183.

T. Gschwind, M. Feridun and S. Pleisch (1999), 'ADK—Building Mobile Agents for Network and Systems Management from Reusable Components', in Proceedings of 1st International Symposium on Agent Systems and Applications, Palm Springs, California, October 3–6, pp. 13–21.

W.R. Cockayne and M. Zyda (1998), *Mobile Agents* (Manning Pubs, USA).

This page intentionally left blank

Security Issues in Mobile Computing 11

No book on mobile computing can be considered complete without a discussion on security. Since mobile computing systems are based on wireless networks and wireless communication technologies, the security of such systems is a major concern. We have already mentioned earlier that larger security challenges are present in wireless networks than in conventional wired networks. Security becomes mandatory in mobile systems due to the existence of hackers, viruses, intruders and Internet-based attackers, who have easy access to such systems through the broadcast nature of wireless channels. Much of the security problem in wireless networks can be traced to the base stations or access points (APs), which operate without any security at all and can be easily taken out and put into wired (Ethernet) nets, exposing the data on it to everyone within its radio range.

It is worth mentioning here that the dissimilarities between wired and wireless networks make it difficult to implement the existing firewalls and other intrusion detection systems (IDS) of wired networks in wireless networks. The most significant difference is that data in a wireless network are transmitted over the broadcast medium, rendering it available to all nodes in the vicinity. Another major difference lies in the communication link, which is slower and of lower bandwidth. Limited battery constraints and higher costs are also major drawbacks in mobile, portable systems, giving rise to frequent disconnections, as discussed in earlier chapters.

In spite of the above differences, network security in wireless systems can still be divided roughly into the following four traditional, intertwined areas:

- **Secrecy (or confidentiality or privacy):** Keeping information out of the hands of unauthorized users.
- **Authentication:** Making sure that one is communicating with the intended person.
- **Non-repudiation:** Using signatures and ensuring that a person cannot go back on his/her earlier communication.
- **Integrity control:** Ensuring that the received message was not tampered with.

We shall assume here that the reader is familiar with the basic concepts and techniques of cryptography, which is the basis of most network security protocols. See Stallings (2000) for an excellent discussion on public and private key cryptography. Furthermore, it is not our intention in this book to discuss in detail security techniques like digital signatures, message digests, firewalls, Internet Protocol Security (IPSec), virtual private networks (VPNs) or authentication systems like Kerberos. These are assumed as prerequisites for understanding the discussion on security in this book and can be found in detail in Tanenbaum (2003) and Stallings (2000).

In this chapter, we shall discuss the general security threats to wireless networks and see how these have been addressed in the various wireless LAN/PAN standards discussed in earlier chapters.

11.1 Security threats to wireless networ ks

As mentioned earlier, wireless networks are vulnerable to many more security threats than traditional wired networks. These are as follows:

1. **Accidental attack:** This gives rise to exposure due to frequent failure of devices and components, because of their small sizes and capabilities.
2. **Passive attack:** Here, the goal of the intruder is only to monitor or get information that is being transmitted. Attacks may include releasing message content or traffic characteristics. Since no data are altered, passive attacks are difficult to detect.
3. **Active attack:** In this type of attack, modification of data or false data transmission takes place, (man-in-the-middle attack) giving rise to masquerade or replay. Denial of service (DoS) is possible where there is either temporary prevention of communication facilities or disruption of the entire network. This is done by flooding it with a large number of messages to degrade the performance of the system.
4. **Unauthorized usage:** This attack takes place because of the growing use of the Internet, which leaves the network vulnerable to hackers, viruses and intruders. It can be prevented by using proper user authentication techniques.
5. **Broadcast based:** As mentioned earlier, an eavesdropper is able to tap the communication into the wireless communication channels, by positioning itself within transmission range.
6. **Device vulnerability:** Mobile devices can be hijacked easily, and if secret IDs or codes are embedded in the device, hackers may get access to private information stored on it and to other network resources.
7. **Heterogeneity:** Mobile nodes need to adjust to potentially different physical communication protocols as they move to different locations.
8. **Resource depletion/exhaustion:** In mobile systems, resources like processing power and battery life are very limited. Hence, techniques such as public key cryptography cannot be used during normal operations to conserve power.

   The device may also be left open to an attack that reduces the normal lifespan of the battery. A DoS attack may consume and waste all the power in the battery, leaving the unit unable to function. In ad hoc networks, these attacks can cause routing nodes in the network to fail, making the network partially unreachable.
9. **Detectability:** Mobile systems used in the military do not want to be detected. Even if strong encryption is being used and the data cannot be deciphered, just detecting the signal puts the mobile user at risk if its position can be located. The device can be jammed by local radio frequency (RF) interference or the user attacked.
10. **Theft of service:** It is very easy to install wireless LANs by just taking them 'out of the box' and by plugging them into the network so that they work. In such systems, security settings are either disabled by default or factory-set default passwords are commonly known. Unauthorized, nearby users, malicious or otherwise, can get a dynamically assigned Internet protocol (IP) address and connect to the Internet.
11. **War driving/walking:** This is like the popular war game called war dialing, which was an earlier technique for searching phone numbers with modems attached to them. As wireless LANs gain popularity, hackers can find them by just taking a notebook computer or pocket PC fitted with a wireless card and some detection software like **netstumbler**, **kismet**, **airsnort**, etc., an optional global positioning system (GPS) and driving/walking round the city. This information is then used to build a network from the identified APs.

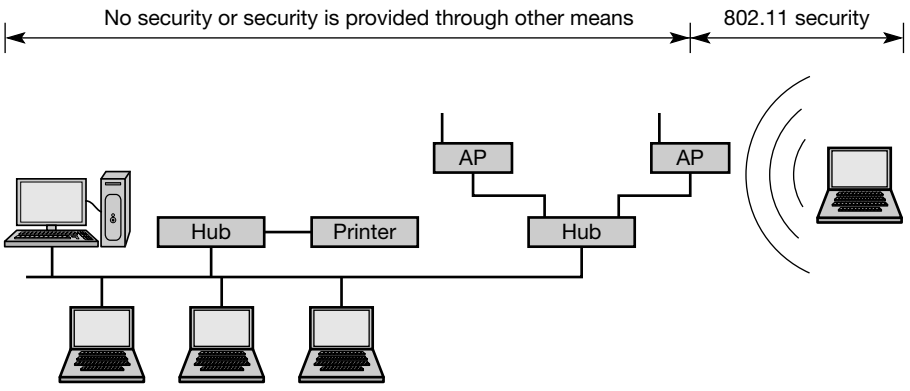


Figure 11.1 Wireless Security of 802.11 in a T ypical Networ k

11.2 IEEE 802.11 security through WEP

This section discusses the built-in security features of 802.11. The IEEE 802.11 specification identified several services to provide a secure operating environment. The security services are provided by the wired equivalent privacy (WEP) protocol to protect link-level data during wireless transmission between clients and APs. WEP does not provide end-to-end security. Only the wireless portion of the connection is made secure, as shown in Figure 11.1. WEP is discussed in detail below.

11.2.1 WEP security features of 802.11 wireless LANs

IEEE defines three basic security services of authentication, confidentiality and integrity for WLANs as given in Karygiannis and Owens (2002). These are given in detail below:

11.2.1.1 Authentication

When wireless users attempt to gain access to a wired network, they must first be validated to make sure they are who they claim to be. This is called authentication. The IEEE 802.11 specification provides for two types of authentication—open-system authentication and shared-key authentication. The highlights of these are shown in Figure 11.2 as a taxonomy.

In **Open System authentication**, a client station exchanges messages with an access point (AP). The AP sends a query as a 'challenge' to the station. If the station sends the correct

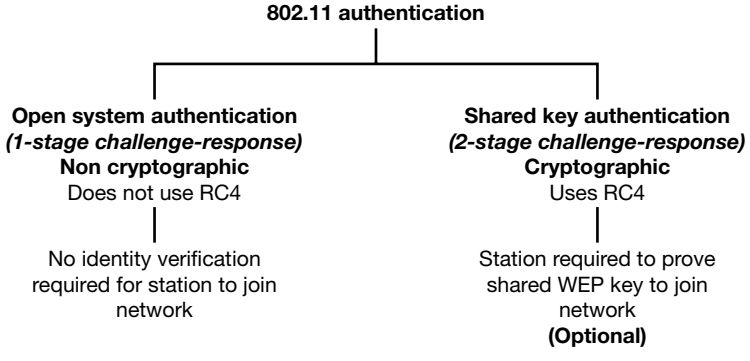


Figure 11.2 802.11 Authentication T echniques— a Taxonomy

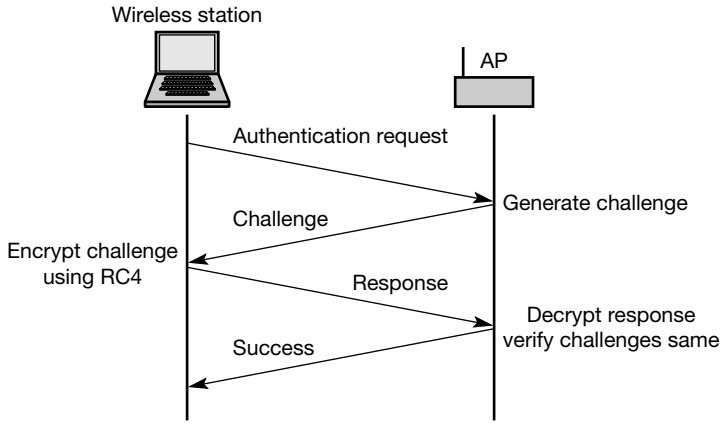


Figure 11.3 Shared-key Authentication

'response', i.e., the correct MAC address fields, it is considered authenticated. Note that there is no cryptographic validation here. Hence open-system authentication is highly vulnerable to unauthorized access and attack. The 802.11 specification only requires this type of 'authentication'. However, this technique cannot be really called authentication, as the AP accepts the mobile station without verifying its identity.

In **Shared key authentication**, another basic 'challenge-response' technique is used which is based on cryptography. In this scheme, shown in Figure 11.3, a random challenge (or nonce) is generated by the AP and sent to the wireless client. The client uses a cryptographic key that is shared with the AP to encrypt the challenge and returns the result to the AP. The AP decrypts this result and if the decrypted value is the same as the random challenge it had sent, it allows access. The 128-bit challenge text is generated using the RC4 symmetric key, stream cipher algorithm as given in Tanenbaum (2002). Unlike open-key authentication, shared key authentication is optional in the IEEE 802.11 specification.

Both the above authentication methods have limitations. Firstly, they do not provide mutual authentication, i.e., the AP authenticates the wireless client, but the client does not authenticate the AP. The mobile station must trust that it is communicating with a legitimate AP. Secondly, the simple challenge-response schemes used in these techniques are known to be weak and suffer from attacks like the 'man-in-the-middle' attack.

11.2.1.2 Confidentiality

The aim of providing confidentiality, or privacy, is to prevent information being eavesdropped during transfer, as is done in a wired network. Eavesdropping is a purely passive attack which must be avoided.

The 802.11 standard for WEP also uses the RC4 symmetric key, stream cipher algorithm and is shown in Figure 11.4. At the wireless station side, a pseudo-random data sequence, called a 'keystream', is obtained by concatenating a 24-bit Initialization Vector (IV) to a shared 40-bit key and passing the same through the RC4 algorithm. Then the payload, which consists of the plain-text, together with the CRC generated by the CRC generating algorithm, is X-ORed with the keystream to generate the ciphertext. At the AP side, the procedure is performed in reverse to get back the plaintext.

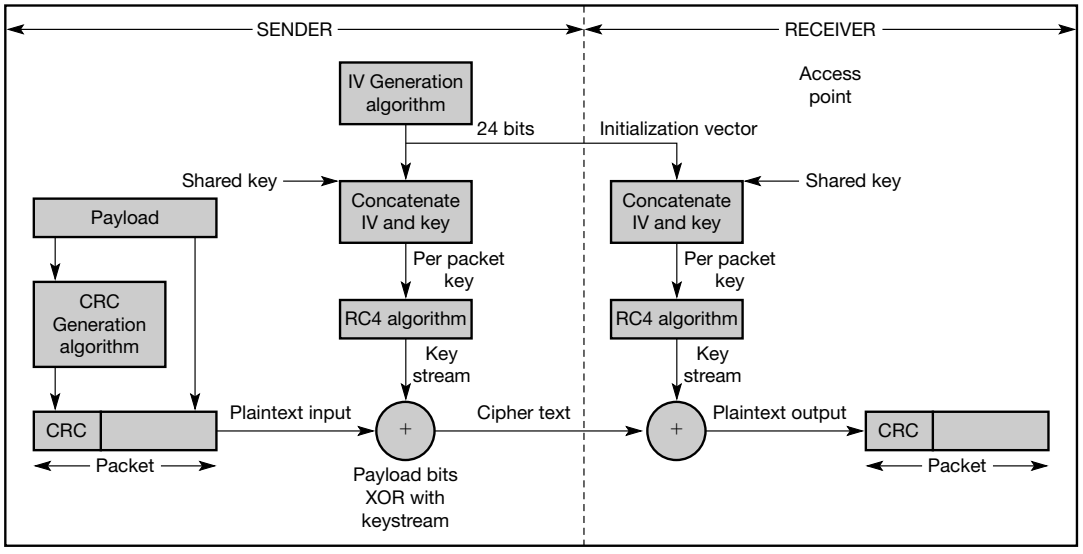


Figure 11.4 WEP Privacy and Integrity Using RC4 Algorithm

In this way, data can be protected from eavesdropping, during transmission over the wireless link using WEP. WEP is applied to all data above the 802.11 WLAN layers to protect Transmission Control Protocol/Internet Protocol (TCP/IP), Internet Packet Exchange (IPX) and Hyper Text Transfer Protocol (HTTP) traffic.

The 802.11 standard WEP supports only a 40-bit cryptographic keys size for the shared key, but nonstandard extensions of WEP that support key lengths upto 104 bits are also prevalent. It may be noted that increasing the key size increases the security of a cryptographic technique.

11.2.1.3 Integrity

Another goal of WEP is to ensure that data/messages between the wireless clients and the AP are not modified in transit in an active attack, i.e., their 'integrity' is not compromised. The IEEE 802.11 specification provides such a data integrity service so that an active adversary 'in the middle' can be thwarted. The same procedure that is used for providing confidentiality, as shown in Figure 11.4, is used at the wireless client side, to provide such data integrity. At the receiving AP, decryption is performed and the CRC is recomputed on the received message. This is compared with the one computed with the original message. If the CRCs do not match, this indicates an integrity violation and the packet is discarded.

Note that the simple CRC is not as cryptographically secure as a hash or message authentication code (Tanenbaum, 2003). The IEEE 802.11 specification also does not provide for key management mechanisms like generating, distributing, storing, loading, etc. of keys. Keys must either be preloaded by the manufacturer or exchanged in advance over a wired backbone network. The base station or the mobile station could also choose a random key and send it over the air, encrypted with the other's public key. Such keys generally remain stable for months or years.

The main drawback of the WEP algorithm is that the same key is shared by all wireless clients, so there is no way to distinguish one from another. Also all users can read each others' data.

These drawbacks have resulted in many instances of attacks on WEP since its implementation. These have exploited either the cryptographic weakness of RC4, or the fact that many of the keys have the property that it is possible to derive some key bits from the keystream.

11.3 Bluetooth security

Bluetooth, as discussed in Chapter 2, has a much shorter range than 802.11, but security is still an issue. If two people occupy adjacent offices in a building and have their mobiles equipped with Bluetooth-enabled wireless keyboards and/or printers, each could read and capture everything the other types or prints, including incoming and outgoing e-mails, confidential reports, etc., if no security is provided.

However, Bluetooth wireless technology puts great emphasis on wireless security so that users can feel secure while making their connections. The Bluetooth Special Interest Group (SIG), made up of more than 4,000 member manufacturers, has a Bluetooth security experts group of engineers from its member companies who provide critical security information and feedback that is taken into account as the Bluetooth wireless specification evolves.

Security for the Bluetooth radio path is depicted in Figure 11.5. As shown in the diagram, security is provided on the various wireless links—on the radio paths only. Link authentication and encryption is provided, but end-to-end security is not possible without providing higher-layer security solutions on top of Bluetooth. In the example provided, security services are provided between the personal digital assistant (PDA) and the printer, between the cell phone and the laptop, and between the laptop and the desktop.

The three basic security services defined by the Bluetooth specifications are briefly discussed below:

- **Authentication:** Identity verification of communicating devices is the first goal of Bluetooth. This security service addresses the question, 'Do I know with whom I am communicating'? An abort mechanism is provided if the device cannot authenticate itself properly.
- **Confidentiality:** Confidentiality, or privacy, is the second security goal of Bluetooth. The requirement is to prevent passive attacks on information, like eavesdropping. This security service addresses the question, 'Are only authorized devices allowed to view my data'?

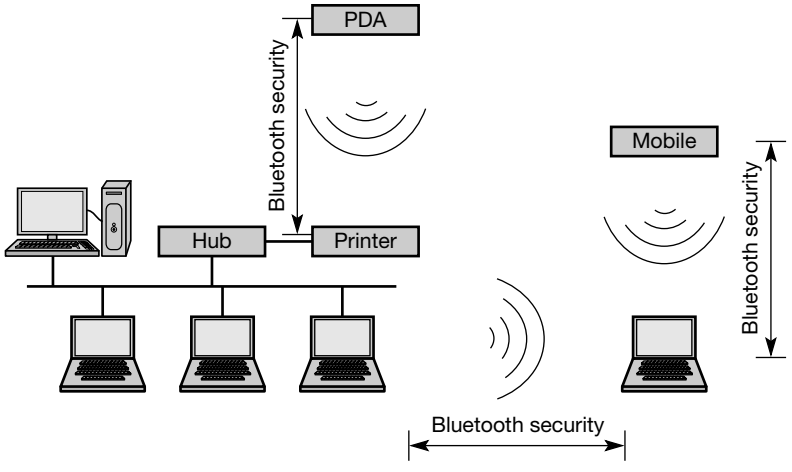


Figure 11.5 Bluetooth Radio Security

- **Authorization:** The third goal of Bluetooth is to allow the control in the use of system resources. This security service addresses the question, 'Is this device authorized to use the requested resource'?

Other security services such as audit and non-repudiation are provided in Bluetooth, and they must be provided through other means, if necessary.

Bluetooth uses a frequency-hopping scheme with 1,600 hops/second combined with power control at the radio link to limit the transmit range. These features provide Bluetooth with some protection from eavesdropping and malicious access. The frequency-hopping scheme, which is a technique to avoid interference, makes it difficult for an adversary to locate the Bluetooth transmission. The power control feature makes it necessary for a potential adversary to be close to the Bluetooth network to carry out an attack.

Three modes of security are provided in Bluetooth for implementing the above security services. These are determined by the product or device manufacturer. These modes are as follows:

SecurityMode1: This is a non-secure option

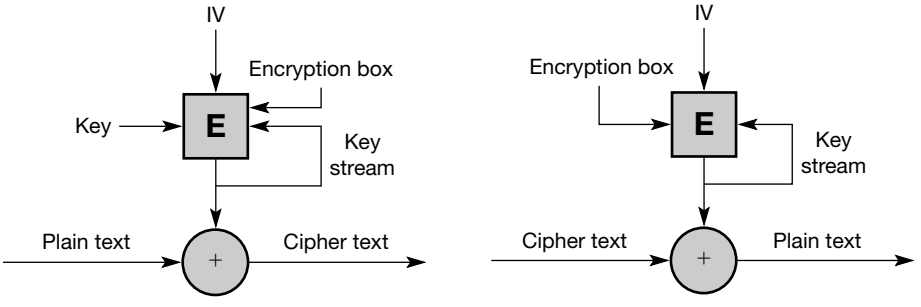
SecurityMode2: In this mode, the enforced security is at service level

SecurityMode3: In this mode, security is enforced at link level

Devices and services also have different security levels. For devices, there are two levels—trusted device and untrusted device. A trusted device, having been paired with one's other device, has unrestricted access to all services. Regarding services, three security levels are defined—services that require authorization and authentication, services that require authentication only and services that are open to all devices.

Bluetooth security starts when a newly arrived slave asks for a channel with the master. The two devices have a shared secret key in advance, which may be hardwired by the manufacturer (for a headset and mobile sold as a unit), or the headset may have a hardwired key and the mobile user may have to enter it in the device as a decimal number. These shared keys are called passkeys.

To establish the channel, the slave and the master each check to see if the other has the passkey and then negotiate whether the channel will be encrypted or integrity will be controlled or both. A random 128-bit session key is then selected. Encryption uses the E0 stream cipher shown in Figure 11.6. The plaintext is XORed with the keystream to generate the ciphertext as shown.



(a) Encryption **(b)** Decryption

Figure 11.6 A Stream Cipher

The above mechanism has several weaknesses and is prone to attacks like the 'man-in-the-middle' attack. A major security issue is that Bluetooth authenticates only devices and not users, so theft of data is a real danger. However, Bluetooth also implements security in the upper layers, so in the event of a breach of link level security, some security may still remain. This is especially so for applications in which a PIN code is required to be entered from a keyboard to complete a transaction.

11.4 WAP 2.0 security

One major drawback of wireless application protocol (WAP) 1.x was that it did not provide end-to-end security. This was because it used non-standard protocols, because of which the WAP gateway, which translates the WAP content to standard Internet content, presented a major 'gap', where momentarily data were present in its plaintext form while being translated. This posed a severe security threat to data.

However, the move to Internet standards is thorough in WAP 2.0, from protocol level to transport layer to session layer, and, last but not the least, in the security layer, promoting transport layer security (TLS), the successor to secure sockets layer (SSL) over the wireless transport layer security (WTLS).

In WAP 2.0, the presence of WTLS means that there is support and services for Internet protocols in the WAP environment. Hence, such translation is not required, and transport layer end-to-end security is assured. The network layer is IP based so that there is full support for IPSec.

There are three main components in IPSec as given below:

- The **authentication header (AH)** provides message integrity.
- The **encapsulating security payload (ESP)** provides confidentiality.
- The **Internet key exchange (IKE)** defines a complex, hybrid protocol to negotiate and provide authenticated keying material for the connections (or security associations as they are called in IPSec) in a protected manner.

Details of IPSec are beyond the scope of this book and can be seen in Tanenbaum (2003).

In the transport layer, TCP connections can be protected by transport layer security (TLS), which is an Internet Engineering Task Force (IETF) standard given in request for comment (RFC) 2246 and is an improvement over SSL. At the application layer, WAP 2.0 uses HTTP client authentication. Application layer crypto libraries provide for integrity control and non-repudiation.

Thus, WAP 2.0 security services can be considered to fare better than 802.11 and Bluetooth security.

11.5 Summary

Larger security challenges are present in wireless networks than conventional wired networks. These include active and passive attacks, device vulnerability, unauthorized usage, resource depletion/exhaustion, detectability, war driving/chalking, etc.

For IEEE 802.11 Wi-Fi networks, WEP provides security against authentication, integrity and privacy. There are three modes of security for Bluetooth access between two devices—non-secure mode, service-level enforced security mode and link-level enforced security mode. Furthermore, Bluetooth authenticates only devices and not users, so theft is a real danger. However, Bluetooth also implements security in the upper layers, so in the event of a breach of link-level security,

some security may still remain. WAP 2.0 provides security at all the layers, and thus WAP 2.0 security services can be considered to fare better than 802.11 and Bluetooth security.

In the next chapter, we will discuss some design and programming projects that can be carried out by readers in the area of mobile/wireless computing, to get a better insight into the topics discussed in the book.

Problems

1. Many security threats are listed in this chapter. Discuss these in detail and show why some of them are unique to wireless systems.
2. What is meant by 'authentication'? Discuss how authentication is carried out in
   (a) IEEE 802.11 WLANs
   (b) Bluetooth networks
   (c) WAP 2.0
3. What do you understand by the term 'Confidentiality'? Show how confidentiality is maintained in each of the wireless networks mentioned above.
4. Repeat Q2 for Integrity.
5. Differentiate between the three modes of security in Bluetooth and indicate in which application each may be used.
6. Compare and contrast the security features found in the three wireless standards discussed in this chapter.
7. Study the 'authentication header' (AH) of the IPSec protocol mentioned in this chapter from one of the references listed at the end of the chapter to find out how authentication is performed between sender and receiver in IPSec.
8. Similarly, study how the encapsulating security payload (ESP) header works in IPSec and then differentiate between AH and ESP.
9. The IPSec protocol uses private or symmetric key cryptography. In your opinion, why is this more suitable than public or asymmetric key cryptography?
10. Some people confuse between authorization and authentication. Find out what the difference is and then indicate which is the more difficult problem to solve and why.

Multiple-choice questions

1. Which one of the following deals with signatures, and ensures that a person cannot go back on his/her earlier communication?
   (a) Secrecy
   (b) Authentication
   (c) Non-repudiation
   (d) Integrity control
2. If the goal of the intruder is to monitor or obtain information that is being transmitted, then this type of attack is known as
   (a) Accidental attack
   (b) Active attack

(c) Heterogeneity attack
(d) Passive attack

3. Exposure due to frequent failure of devices and components is known as
(a) Accidental attack
(b) Active attack
(c) Heterogeneity attack
(d) Passive attack

4. In which of the following types of IEEE 802.11 authentication techniques, does the access point accept a mobile station without verifying its identity?
(a) Open-system authentication
(b) Shared-key authentication
(c) Both
(d) None of the above

5. Which of the following is True for the statements X and Y for the wired equivalent privacy (WEP) protocol?
X: WEP does not provide end-to-end security.
Y: WEP integrity uses a simple CRC which is not a cryptographically secure mechanism.
(a) X is true but Y is false
(b) X is false but Y is true
(c) Both X and Y are true
(d) Both X and Y are false

6. Which one of the following security services is not addressed by Bluetooth?
(a) Authentication
(b) Confidentiality
(c) Authorization
(d) Non-repudiation

7. Which of the following is True for the statements X and Y?
X: Bluetooth authenticates only users and not devices.
Y: Bluetooth provides a frequency-hopping scheme.
(a) X is true but Y is false
(b) X is false but Y is true
(c) Both X and Y are true
(d) Both X and Y are false

8. At the application layer, WAP 2.0 uses which one of the following authentication clients?
(a) HTTP client authentication
(b) FTP client authentication
(c) IPX client authentication
(d) None of the above

9. As defined in the IEEE 802.11 standard, WEP supports which one of the following cryptographic keys size for the shared key?
(a) 40-bits
(b) 42-bits
(c) 44-bits
(d) 48-bits

10. Which of the following is not a mode of security for Bluetooth access between two devices?
(a) Non-secure
(b) Service-level enforced security.
(c) Link-level enforced security.
(d) User-level enforced security.

Further reading

A. Aziz and W. Diffie (1993), *Privacy and Authentication for Wireless Local Area Networks* (Sun Microsystems Inc), http://piggy.cs.nthu.edu.tw/paper/Mobile/PS/priv+auth-wirel-LAN.ps.gz.

A.S. Tanenbaum (2003), *Computer Networks*, 4th ed. (New Delhi, India: Pearson Education).

D.P. Agarwal and Q. Zeng (2003), *Introduction to Wireless and Mobile Systems* (Singapore: Thompson Asia Pvt Ltd.).

F. Adelstein, S.K.S. Gupta, G.G. Richard III and L. Schwiebert (eds) (2005), *Fundamentals of Mobile and Pervasive Computing* (New Delhi, India: Tata McGraw-Hill).

IETF Working Group, 'IP Security Protocol (IPSec)', www.ietf.org/html.charters/ipsec-charter.html (accessed December 2006).

M.J. Ranum (1996), *Internet Attacks,* www.clark.pub.mjr (accessed April 2006).

N. Borisov, I. Goldberg and D. Wagner (2001), 'Intercepting Mobile Communications: The Insecurity of 802.11', 7th International Conference on Mobile Computing and Networking, ACM, pp. 180–188.

S. Fluhrer, I. Mantin and A. Shamir (2001), 'Weakness in the Key Scheduling Algorithm of RC4', in Proceedings of the 8th Annual Workshop on selected Areas in Cryptography.

S. Kent and R. Atkinson, 'IP Authentication Header (RFC 2402)', www.ietf.org/rfc/rfc2402.txt (accessed May 2007).

S. Kent and R. Atkinson, 'IP Encapsulating Security Payload (ESP) (RFC 2406)', www.ietf.org/rfc/rfc2406.txt (accessed Jan. 2006).

S. Uskela (1997), Security in Wireless Local Area Networks', Technical Report, Department of Electrical and Communications Engineering, Helsinki University of Technology, 1997, www.tml.tkk.fi/Opinnot/Tik-110.501/1997.

T. Dierks and C. Allen (1999), 'The TLS Protocol, Version 1.0', RFC 2246, www.ietf.org/rfc/rfc2246.txt (accessed January 1999).

T. Karygiannis and L. Owens (2002), *Wireless Network Security, 802.11, Bluetooth and Handheld Devices* (NIST Special Publication), pp. 800–848.

T. Karygiannis et al (2006), 'Detecting Critical Nodes for IDS', 2nd International Workshop on Security in Pervasive and Ubiquitous Computing, France.

V. Bharghavan (1994), 'Secure Wireless LANs', ACM Conference on Computers and Communications Security '94, University of California at Berkley, http://shiva.crhc.uiuc.edu/Papers/ccs94.ps.gz.

W. Stallings (2000), *Network Security and Cryptography* (New Delhi, India: Prentice Hall of India).

This page intentionally left blank

Design and Programming Projects 12

In this chapter, we present some design projects that are based on the topics discussed in the preceding chapters and which can be implemented by readers. These projects have been designed and implemented by the final-year undergraduate and postgraduate students at IIT Roorkee as part of the mobile computing course curriculum. These will help to provide the necessary hands-on experience to students of a mobile computing course.

The following projects will be discussed briefly, with only some useful suggestions being provided for their design and implementation.

1. Implementation of mobile Internet protocol (IP)
2. Performance comparison of ad hoc on-demand distance vector (AODV) and dynamic source routing (DSR) routing protocols through simulation
3. Design and implementation of a Bluetooth chatting application
4. Design of a wireless application protocol (WAP) gateway
5. Using mobile agents for network monitoring
6. An IEEE 802.11 local area network (LAN) for a typical student hostel
7. An application using wireless sensor networks

Note that only some aspects of the design and implementation of each project are included in this chapter. The details are left as hands-on exercises for the readers. Some aspects are present in the relevant Further Readings given at the end of the chapter.

12.1 Implementation of mobile IP

Problem statement: To implement mobile IP over the Linux platform

Design

The following entities have to be designed:

1. **Mobile node (MN):** A host that may change its point of attachment from one network/subnetwork to another through the Internet. It is preassigned a fixed home address on a home network, which other correspondent hosts will use to address their packets to this host, regardless of its current location.
2. **Home agent (HA):** A router that maintains a list of registered mobile nodes in a visitor list. It is used to forward packets addressed to the MNs through the network when the MNs are away from home. After checking with the current mobility bindings for a particular MN, it encapsulates datagrams and sends it to the MN's current temporary address.

3. **Foreign agent (FA):** A router that assists a locally reachable MN that is away from its home network. It delivers information between the MN and the HA.
4. **Care-of address (COA):** An address which identifies the MN's current location. It can be viewed as the end of a tunnel directed towards an MN. It can be either assigned dynamically or associated with its FA.
5. **Correspondent node (CN):** A node that sends packets addressed to the MN.
6. **Home address:** A permanent IP address assigned to the MN. It remains unchanged regardless of where the MN is attached to the Internet.
7. **Mobility agent:** An agent which supports mobility. It could be either an HA or an FA.
8. **Tunnel:** The path taken by encapsulated packets from the HA to the FA.

The following services will be supported:

1. **Agent discovery:** HAs and FAs broadcast their availability on each link to where they can provide service. A newly arrived MN can send a solicitation on the link to learn if any prospective agents are present.
2. **Registration:** When the MN is away from Home, it registers its COA with its HA so that the latter knows where to forward its packets. Depending on the network configuration, the MN could register either directly with its HA or indirectly with the help of the FA.
3. **Encapsulation:** The process of enclosing an IP datagram inside another, with the outer IP header containing the COA of the MN. The IP datagram itself remains intact and untouched throughout the enclosing process.
4. **Decapsulation:** The process of stripping the outer IP header of the incoming packets so that the enclosed datagram can be accessed and delivered to the proper destination.

Implementation

Test environment

A platform is to be created to support the functions required in order to test the functionality of mobile IP. Its characteristics are as follows:

Operating system: Redhat Linux v9.0 does not include all the features, called **modules,** required for several network operations in the implementation. These are provided by the distribution in source code format. Thus, it is necessary to compile a new kernel in order to include these features.

The following modules are needed:

- Packet socket (CONFIG_PACKET)
- Kernel user netlink socket (CONFIG_NETLINK)
- Routing messages (CONFIG_RTNETLINK)
- IP socket filtering (CONFIG_FILTER)
- IP tunnelling (CONFIG_NET_IPIP)
- IP policy routing (CONFIG_IP_MULTIPLE_TABLES)
- IP advanced router (CONFIG_IP_ADVANCED_ROUTER)
- GRE (generic routing encapsulation) tunnels (CONFIG_NET_IPGRE)
- Loadable module support (CONFIG_MODULES)
- Netfilter (CONFIG_NETFILTER)

- Connection tracking (CONFIG_IP_NF_CONNTRACK)
- IP tables support (CONFIG_IP_NF_IPTABLES)

Dynamic host configuration protocol (DHCP) server

To create subnetworks, each serving computer should have the ability to assign IP addresses and passing parameters to any client. It is also required to assign a co-located COA.

Network address translation/Internet protocol (NAT/IP) masquerading

After installing and adjusting the DHCP server, the next step is to give Internet access to any subnetwork created, that is, to any DHCP client. This is done by setting up the NAT functionality in the form of Linux IP Masquerade.

Apart from compiling a new kernel, a few more adjustments have to be done on the operating system so that the mobile IP mechanisms work properly. These are as follows:

1. Disabling the **spoof protection** feature so that MN's packets are not rejected.
2. Activating **IP forwarding** so that HA and FA can forward their packets.
3. Activating **proxy ARP** (address resolution protocol) to enable HA to reply to ARP requests sent by nodes seeking the MN at its Home network.

Hardware

Three computers can be used with the following typical configuration:

CPU: Intel premium 2.4 GHz

RAM: 128 MB

Hard disc: 40 GB

Ethernet cards: Realtech 8139 C+

Two Ethernet cards (eth0 and eth1) are used on one of these computers to provide access to the Internet (through eth0, external network interface) and to create subnetworks for the client computers connecting to eth1 (internal network interface) by using private/reusable IP addresses of the form 192.168.x.x.

Typical test-bed topology

Home network

Network address:	192.168.208.0
Mask:	255.255.255.0
Broadcast:	195.168.208.254

Home agent

Interface:	eth0
IP address:	192.168.208.50
Network address:	192.168.208.0
Mask:	255.255.255.0
Broadcast:	192.168.208.255
Default gateway:	192.168.208.254

Foreign network	
Network address:	192.168.1.0
Mask:	255.255.255.0
Broadcast:	195.168.1.255
Foreign agent	
Interface:	External-eth0
IP address:	192.168.208.52
Network address:	192.168.208.0
Mask:	255.255.255.0
Broadcast:	192.168.208.255
Default gateway:	192.168.208.254
Interface	
Interface:	Internal-eth1
IP address:	192.168.1.1
Network address:	192.168.1.0
Mask:	255.255.255.0
Broadcast:	192.168.1.255
Mobile node	
Interface:	eth0
IP address:	192.168.208.51
Network address:	192.168.208.0
Mask:	255.255.255.0
Broadcast:	192.168.208.255
Default gateway:	192.168.208.254

The network setup is shown in Figure 12.1. The student should implement this setup and get the results.

12.2 Comparison between AODV and DSR protocols

Problem statement: To simulate the AODV and DSR routing protocols for mobile ad hoc networks (MANETs) and compare their performance

The following metrics will be used to compare the two protocols:

1. **End-to-end packet delivery ratio:** This is the ratio of the number of packets received by the destination node to the number of packets sent by the source node.
2. **Routing overhead:** This is the total number of routing packets transmitted during the simulation.
3. **Path optimality:** This is the difference between the number of hops a packet takes to reach the destination node and the length of the shortest path.
4. **Scalability:** This is the ability of the protocol to adapt to the increase in the number of nodes in the network.

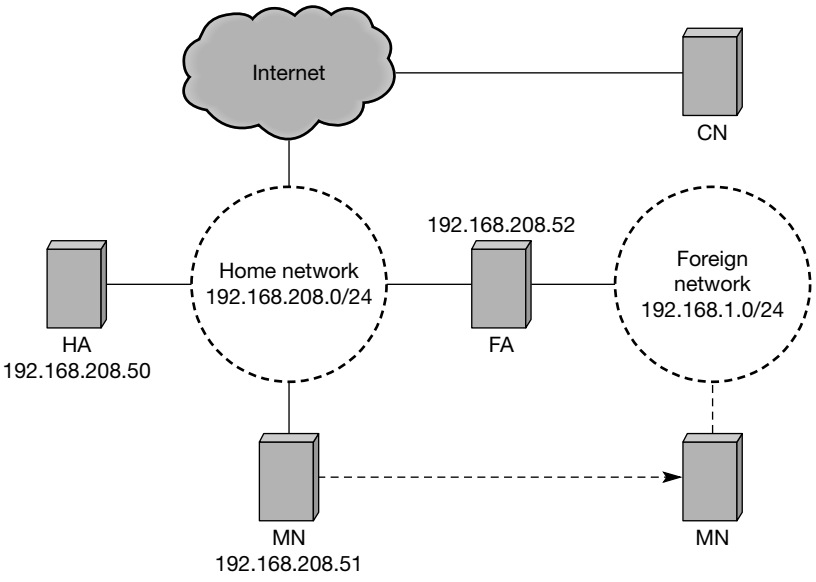


Figure 12.1 The Network Setup

Software used

1. Network simulator (NS2.29)
2. Java programming language under Unix operating system

Design

Simulation model: Discrete event simulation will be used in NS2.

Simulation parameters

Node transmission range:	250 m and 100 m
Simulation time:	900 s
Number of nodes:	6, 10, 20 and 30 nodes
Speed of node movement:	1, 10 and 20 m/s.
Area of node movement:	1500 m × 300 m
Traffic type:	CBR, varied as 4, 10, 20 and 1,000 Kbps/node
Data packet size:	1024 bytes
Pause times:	5, 30 and 90 s.

Constants used in AODV simulation

Time for which the broadcast ID of route request (RREQ) packet is kept:	1 s.
Time before a broken link is deleted from routing table:	1 s.
Time for which a route is considered active:	3 s.
Route maintenance beacons:	1 s.

Constants used in DSR simulation

Time between retransmitted route requests:	500 ms.
Time to hold packets awaiting routes:	3 s.

Simulation

1. The **simulation module** will contain the common functions for both AODV and DSR. Thus, there will be modules for initializing the simulation parameters to create the topology with the nodes, to configure each node with specified parameters, to provide mobility to the nodes, to create sending and receiving agents for each node, to create specified traffic between the nodes and to sequence the vents in the simulation.
2. Modules for simulating the **DSR and AODV** mechanisms.
3. The **classes** required for performance comparison will include the main performance class and classes to keep track of simulation time elapsed, to place the events in accordance with their time of occurrence, to provide the time for each event, to keep track of all statistical details of the simulation, for providing user interfaces for AODV and DSR, to keep track of all node movements, to deal with their RREQ, for timeout, to generate packets and to maintain a route cache for DSR.

Note: In the above project, we have used the popular NS2 simulator. Another useful simulation tool that is now available is the **Qualnet version 4** from Scalable Network Technologies, USA. This has been used by some graduating students of computer science. They have found it to be very user friendly and especially suited for simulating wireless networks, as it has many support libraries for the same. We give in Appendix B a comparison of the salient features of both NS2 and Qualnet4.

12.3 Bluetooth application

Problem statement: To develop a JABWT (Java API for Bluetooth wireless technology)-based chat room application for mobile devices

Hardware required: A JABWT-capable device (such as the Nokia 6600 phone) that supports the J2ME MIDP 1.0 profile

We shall call the application BChat. When BChat is launched, it will search and join any existing chat room within the Bluetooth effective range, or create a new chat room if it is the first active BChat in that range. We implement a virtual chat room formed by a network of BChat applications. Users can start messaging with each other within the same virtual chat room when there is more than one party connected to each other. If one user sends a message over the air, all parties of the chat room will receive the message. Users can join and leave the chat room at anytime.

Software required

J2ME wireless toolkit

J2 SDK

Nokia SDK

The Java Community Process introduced the first standardized API specification for Bluetooth in 2000. This specification (JSR-82), Java API for Bluetooth wireless technology (JABWT), establishes a common ground for rapid Bluetooth application development. Developers can write

Bluetooth applications independent of hardware vendors. Most important, JABWT-compatible applications are portable across various JABWT-equipped devices.

The most commonly used profiles and functions included in JABWT are generic access profile, service discovery profile, serial port profile, generic object exchange profile and their respective protocols. These profiles allow JABWT applications to perform the following functionalities:

- **Generic access profile:** It provides the basic building blocks of a Bluetooth application, such as local device, remote device, Bluetooth address and device discovery.
- **Service discovery profile:** It provides the ability to find available services to access remote functionalities.
- **Serial port profile:** It provides a stream-based connectivity between Bluetooth applications.
- **Generic object exchange (OBEX) profile:** It provides support for OBEX protocol, which allows applications to exchange simple objects such as business card data.

To simplify the programming model, two entities, the Bluetooth configuration centre (BCC) and the service discovery database (SDDB), are abstracted from the Java API. BCC includes the capabilities that globally configure the Bluetooth stack and prevent one application from adversely affecting another. SDDB is an abstract database of service records, a collection of attributes that describe Bluetooth services. Java applications interact with SDDB indirectly via the update and retrieval of service records.

We make the following assumptions to simplify the application:

- There is only one chat room that exists within the effective Bluetooth range.
- There is no security imposed when joining a chat room.
- Users run one instance of BlueChat on a device at any given time.

JABWT provides a familiar API to J2ME developers for accessing Bluetooth facilities and is integrated with the J2ME generic connection framework. Like many other network protocols, the Bluetooth connection model employs client/server architecture. Our BlueChat application, on the other hand, operates in a peer-to-peer manner. Each running instance of BlueChat (or a node) can serve as a client and a server at the same time. It behaves as a client when BlueChat starts up; it searches and connects to existing running BlueChat devices. Once connected, it makes itself available for future clients to connect to. In such cases, it serves as a server for future client connections.

Figure 12.2 shows the network relationship between three BlueChat applications. To logically represent an active BlueChat node, we use the concept of endpoint to encapsulate all the connectivity attributes of a node. An endpoint represents a unique message delivery destination and source regardless of whether it is a server or a client.

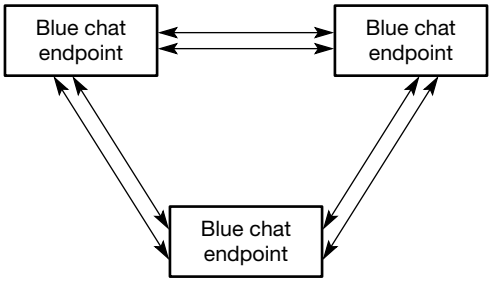


Figure 12.2 Bluetooth Chat Connectivity

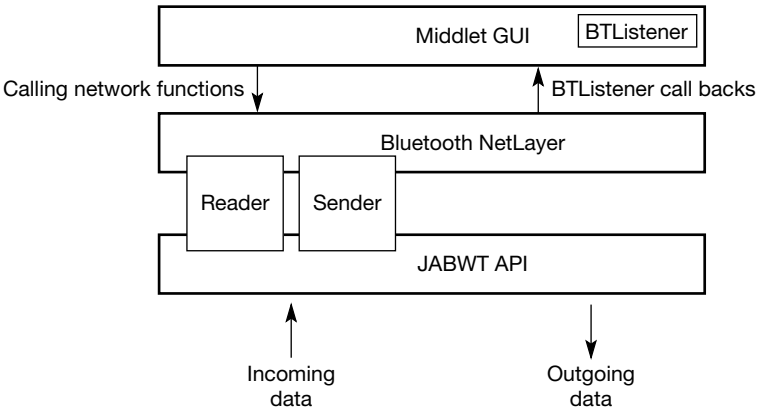


Figure 12.3 Network Layer

A Bluetooth connection differs from a regular socket connection by its unique device and service discovery processes. Bluetooth applications start the device discovery process to identify connectable devices, followed by a service discovery process to obtain a reference (URL) to suitable services. To hide these complexities from the graphical user interface (GUI) elements, a network layer is introduced to serve as a façade to the Bluetooth API.

The GUI can access Bluetooth connectivity via a simplified interface, which does all the discovery and connection establishment behind the scenes. This network layer also provides the functionality to send messages to and receive messages from other endpoints. A call back interface is in place to report any network activity back to the GUI. Figure 12.3 illustrates the relationship between various layers and components in BlueChat.

The communication channel between each connected BlueChat endpoint is a structured data stream connection. We put together a simple protocol to coordinate the activity between each endpoint. This protocol includes the following features:

- **Initial handshake:** Each point must handshake with each other when the connection is first established. This ensures that the connecting device is a BlueChat node rather than a mistakenly connected application. During the handshake, we also exchange the screen names of the users (see Figure 12.4).
- **Delivery of text message:** Each sent text message is delivered to all endpoints connected to the BlueChat network.
- **Termination handshake:** If the user quits the chat room gracefully, a termination token is sent to all the other endpoints to indicate its intention. We can clean up the necessary network and runtime resources associated with the leaving endpoint upon receiving this token. However, if the user walks away from effective range and becomes inaccessible, a termination token is not sent. Other active endpoints will discover that the leaving party is inaccessible when the connections are lost and will clean up the resources (see Figure 12.5).

Implementation

The **NetLayer class** is to be implemented for the BlueChat networking layer. It does most of the Bluetooth-related work and provides the following functionality:

- Initializes the Bluetooth stack
- Registers BlueChat services to the Bluetooth device

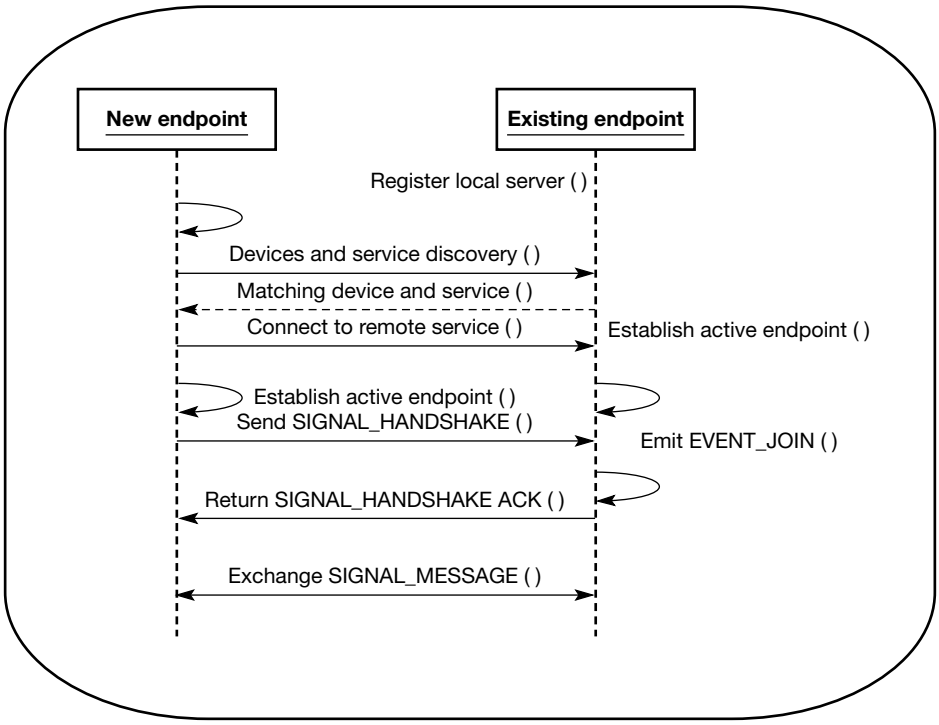


Figure 12.4 Establish Two Endpoints

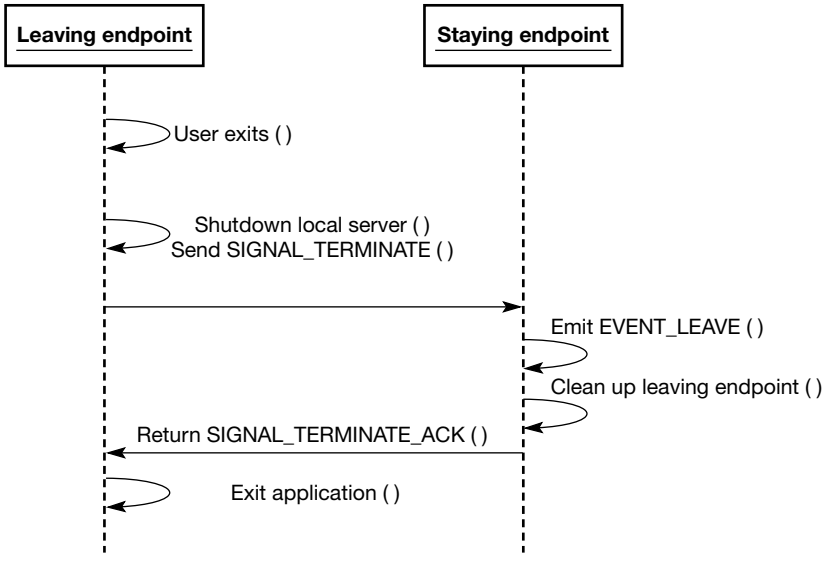


Figure 12.5 Terminate Two Endpoints

- Searches for nearby devices
- Searches for BlueChat services on nearby devices
- Establishes endpoint connectivity for found BlueChat services
- Manages the life cycle of all endpoints

The Bluetooth stack can be initialized by calling LocalDevice.getLocalDevice( ). LocalDevice is a singleton that uniquely represents the underlying Bluetooth device implementation and can be used to gain access to other Bluetooth features including the following:

- Discovery agent (via getDiscoveryAgent( ))
- Bluetooth physical network address (via getBluetoothAddress( ))
- SDDB (via getRecord( ) and updateRecord( ))

The BlueChat NetLayer's initial work is to create and register a BlueChat service to a local device. A Bluetooth service is an entry point for other Bluetooth clients to access available functionalities. Since each BlueChat endpoint can act as a server, it must register its service in order to make this server available to other BlueChat clients. JABWT utilizes the MIDP generic connection framework to instantiate a server connection. A BlueChat application needs to instantiate a Serial Port Profile connection, basically a stream-based connection that allows two BlueChat applications to exchange data using Java input and output streams.

After a server connection is created, a corresponding ServiceRecord is created for this service. A ServiceRecord is a collection of attributes that describes our service, and these attributes are searchable by clients. We can use localDevice.getRecord (server) to retrieve the newly created ServiceRecord.

The server.acceptAndOpen( ) method notifies the Bluetooth implementation that the application is ready to accept incoming connections and make the service available. This also instructs the underlying implementation to store the ServiceRecord object in the SDDB, which occurs when server.acceptAndOpen( ) is first invoked. Any subsequent change to the ServiceRecord must be reflected in the SDDB by using localDevice.updateRecord( ).

Now our BlueChat application is ready to accept a connection. If there is an existing chat room available, BlueChat should join the existing network by searching for other BlueChat services on each individual device and by connecting to their services. Three steps must be taken to perform this action.

1. Search for an available device by using DiscoveryAgent, another singleton in JABWT.
2. For each available device, search for available and matching services by using LIAC discovery mode.
3. For each available and matching service, connect to the service and perform the initial handshake.

The device discovery and service discovery processes are performed in an asynchronous manner. A Bluetooth application must provide a callback object for the JABWT implementation to notify when devices or services are found.

To logically represent all the parties in the chat room, we implement the **class EndPoint**. From the application-level perspective, an endpoint encapsulates information for each actively connected BlueChat user and device. BlueChat uses EndPoint to identify which user to send a message to, and from which user a message is received.

When a user exits BlueChat, the application sends a termination token (SIGNAL_TERMINATE) to all connected parties to signal that the endpoint is no longer active. All receiving parties

must return an acknowledgement (SIGNAL_TERMINATE_ACK) and remove the leaving endpoint from the active endpoint list.

A GUI, based on the MIDP LCDUI API, can also be designed to provide a simple interface to send and receive messages.

12.4 Design of a WAP gateway

Problem statement: To design and implement a WAP gateway for push and pull operations

Design gateway setup

For pull operations, it requires

1. Mobile: Nokia toolkit (mobile simulator) 4.0
2. Web server: MS IIS (Microsoft Internet Information Server)
3. WAP gateway

For push operations, it requires

1. Mobile: Nokia toolkit (mobile simulator) 4.0
2. Push initiator (PI): Openwave PI
3. WAP gateway

Gateway structure implementation

See Flowcharts in Figures 12.6 and 12.7.

Language used: JDK 1.5 Packages used: Nokia SDK APIs

Implementation for pull operation

- Create a server (bind at port 9200) listening for PULL request from client.
- After receiving a request, the WAP gateway decodes the request message and translates the request line and request header (in binary format) to HTTP format by a mapping table (using wireless markup language [WML] encoder, WML compiler, WML decoder).
- Establish the connection with the desired HTTP server and forward the request to it.
- The Web server returns an HTTP reply message, which contains data.
- The WAP gateway encodes the received reply message and translates the HTTP well-known formatted reply line and reply header to WSP binary format using the mapping table (using WML encoder, WMLS compiler, WML decoder).
- The WAP gateway returns a WSP response containing the WML to the client system.

Implementation for push operation

- A push initiator submits a push message to a WAP gateway. The push message is a multipart type, which contains an XML control entity and a content entity, and may contain RDF-format capabilities entity defined in the user agent profile.
- The gateway determines to accept or to reject the received push message. If the push access protocol (PAP) push message element is not valid with respect to its document type definition (DTD), the push proxy gateway (PPG) must reject it.
- The gateway reports the acceptance/rejection result in the response which is an XML document.
- The gateway parses control information of push message to determine where and how it should be delivered.
- If there is no error, the WAP gateway delivers push primitives.

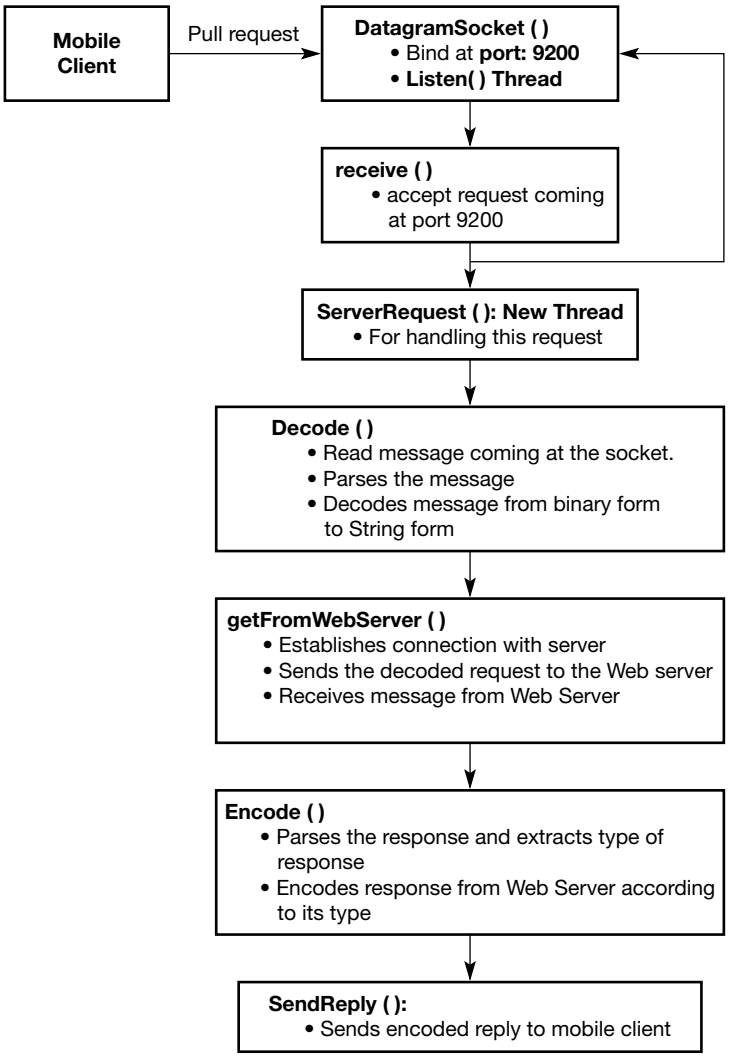


Figure 12.6 Flow Chart for Pull Operation

- The gateway sends a result notification to inform the PI of the outcome of the push submission. This notification reports whether the push message was sent, delivered, expired, cancelled or there occurred an error.

12.5 Mobile agents for networ k monitoring

Problem statement: To design and implement a network monitoring system through information retrieval, using Java aglets

Design

In this application, we design a flexible architecture which forms a layer over conventional SNMP-based network management. This ensures that the advantages of SNMP are not lost and

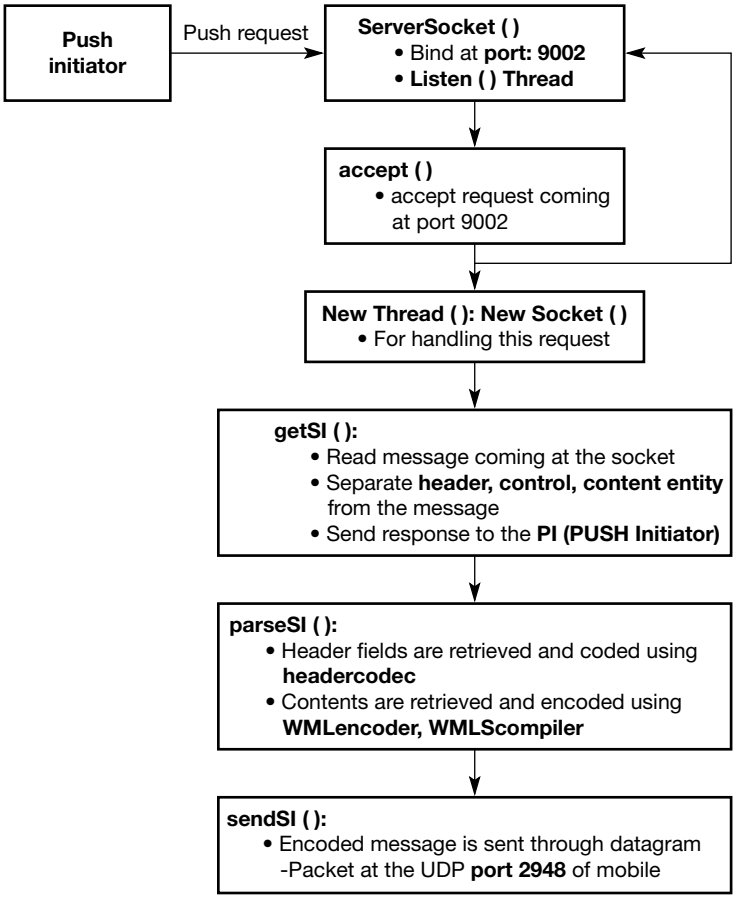


Figure 12.7 Flow Chart for Push Operation

also serves the purpose of managing legacy SNMP-based systems. The manager is given the flexibility of deciding whether to use a sequential or parallel approach to monitor the network. In the sequential approach, only one mobile agent is created with traverses through all the nodes; while in the parallel approach, as many mobile agents are created, as there are clients, with a different agent going to each client.

Environment used

We use the aglet API which is a Java package (`aglet`) consisting of classes and interfaces, most notably `Aglet`, `AgletProxy`, `AgletContext`, and `Message`. It is shown in Figure 12.8.

The aglet class is the key class in the aglet API. This is the abstract class that the aglet developer uses as a base class when he or she creates customized aglets. The `Aglet` class defines methods for controlling its own life cycle, namely, methods for cloning, dispatching, deactivating, and disposing of itself.

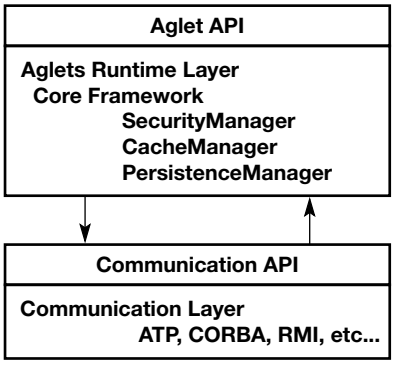


Figure 12.8 The Aglet API

The `AgletProxy` interface acts as a handle of an aglet and provides a common way of accessing the aglet behind it. Since an aglet class has several public methods that should not be accessed directly from other aglets for security reasons, any aglet that wants to communicate with other aglets has to first obtain the proxy, and then interact through this interface. In other words, the aglet proxy acts as a shield object that protects an aglet from malicious aglets. Another important role of the `AgletProxy` interface is to provide the aglet with location transparency.

The `AgletContext` interface is used by an aglet to get information about its environment and to send messages to the environment, including other aglets currently active in that environment. It provides a means for maintaining and managing running aglets in an environment where the host system is secured against malicious aglets.

Aglets communicate by exchanging objects of the `Message` class. A string field named 'kind' distinguishes messages. This field is set when the message is created. The second parameter of the message constructor is an optional message argument.

IBM's Aglets Workbench comes with a graphical user interface for the context. It is called **Tahiti server (aglets-2.0.2)** and will be used for creating aglets. It performs the following tasks and needs to be installed and configured.

- Creates and loads aglets
- Dispatches and retracts
- Process of creating an aglet
- Loading the class file
- Instantiating the aglet
- Establishing the aglet in the context
- Invoking onCreation()
- Invoking run()

Implementation

(a) Sequential itinerary (hop by hop): In the sequential itinerary, the agent moves to the clients given in the list given by the AdminGUIAglet, one by one. It instantiates a stationary agent (StationaryAglet) over the client environment and communicates with it. It sends the

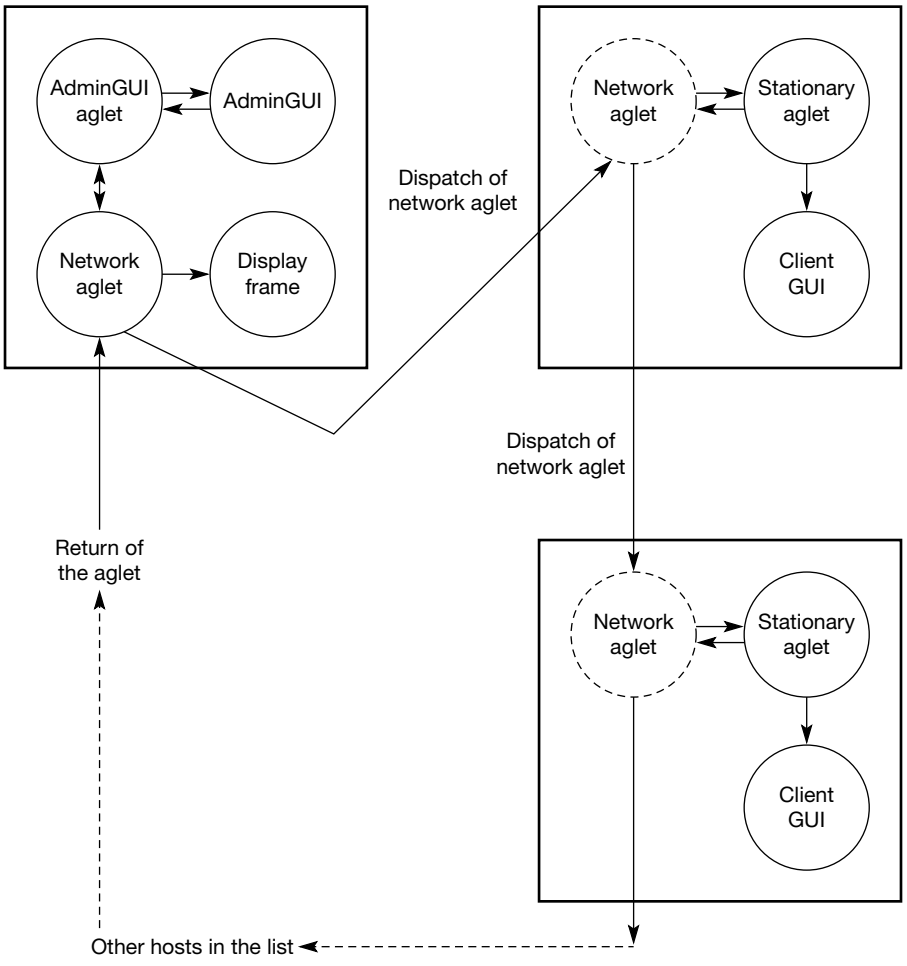


Figure 12.9 Class Diagram of Sequential Itinerar y

GetLocalInfo message to get the information regarding the client and synchronizes the date and time of client with the server. If any client is unreachable, then the network aglet stores this information and moves on to the next host in the list. The Class Diagram for the Sequential Itinerary is shown in Figure 12.9.

(b) Parallel itinerary (single hop): For the Parallel Itinerary, the AdminGUIAglet creates as many SimpleAglet agents as the number of clients in the list. Every simple agent moves to its related client and instantiates a stationary agent (StationaryAglet) over there. It communicates with the stationary agent via messages to get local info and to set date and time. After doing so, it returns to the admin host where the gathered information is displayed. The class diagram for the parallel itinerary is shown in Figure 12.10.

The various classes implemented in the application are shown in these class diagrams and are self-explanatory.

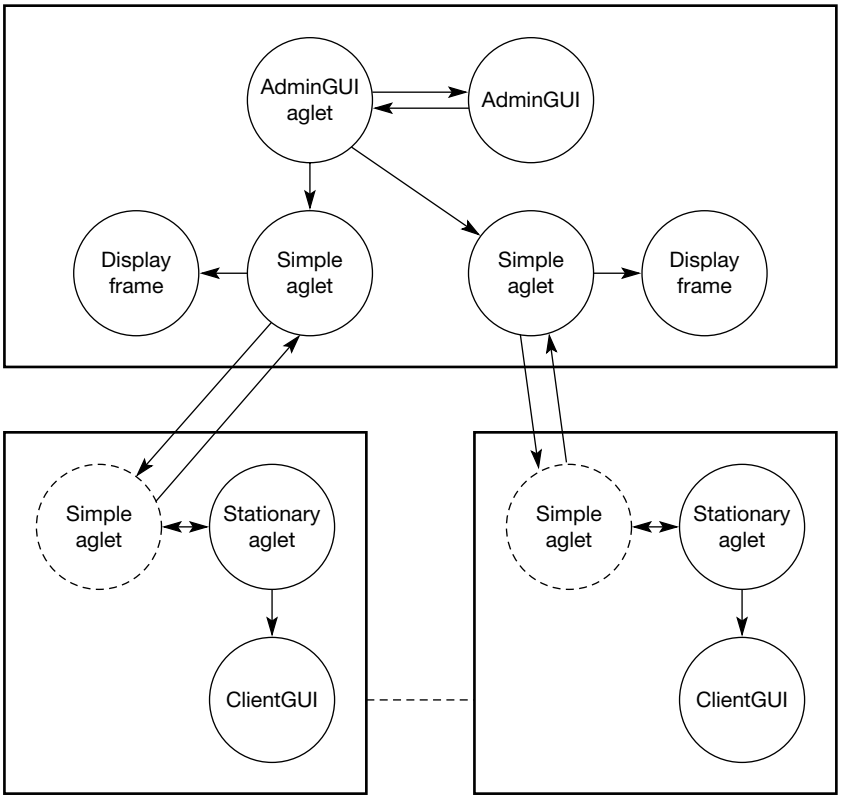


Figure 12.10 Class Diagram of P arallel Itinerar y

12.6 An IEEE 802.11 LAN for a typical student hostel

Problem statement: To design a wireless access application for a typical student hostel of capacity 500 students, for Web surfing and media-streaming facilities

Design choices/decisions

For this project, the following predesign decisions have to be made:

(a) A **site survey** is first required in order to assess the physical size of the hostel (in this case the hostel called Govind Bhawan at IIT Rorkee) so that the number and locations of the access points (APs) can be finalized.

(b) The choice of wireless technologies, namely, IEEE 802.11a or b or g, is to be made.
We choose **802.11g** for our LAN for the following reasons:
- It has a higher data rate than that of 802.11b
- The range supported is greater than that of 802.11a
- It is backward compliant with 802.11b devices/cards
- We assume that other devices operating in the ISM band, like microwave ovens and Bluetooth-enabled devices, are not present. Hence, interference is less.

(c) It is assumed that the hostel already has a **wired Ethernet LAN** in the cyber cafe room in the hostel, with the requisite distribution switch.

(d) The type of **power over Ethernet (POE)** to be used is to be decided.

POE technology enables an AP to receive electrical power and data over the same cable. A power-sourcing device, which connects directly to the power source and is situated between the LAN switch and the AP, will detect the AP and inject the right amount of electrical current over unused pairs in the Cat5 Ethernet cable. Besides, cost-savings POE allows flexible AP mounting locations, easier deployment and better reliability because of a fewer number of cables.

We choose **hub-based POE** to support multiple (11) APs from a source signal using only one power outlet.

Design

In this section, we propose an example design. It would be appreciated that this is just one of many possible solutions. Note that the design is based on the devices/equipment from a particular vendor.

The layout of the APs is shown in Figure 12.11. The 11 APs are connected to the distribution switch through Cat5 cable. About 500 users can be supported with a maximum speed of 500 Kbps

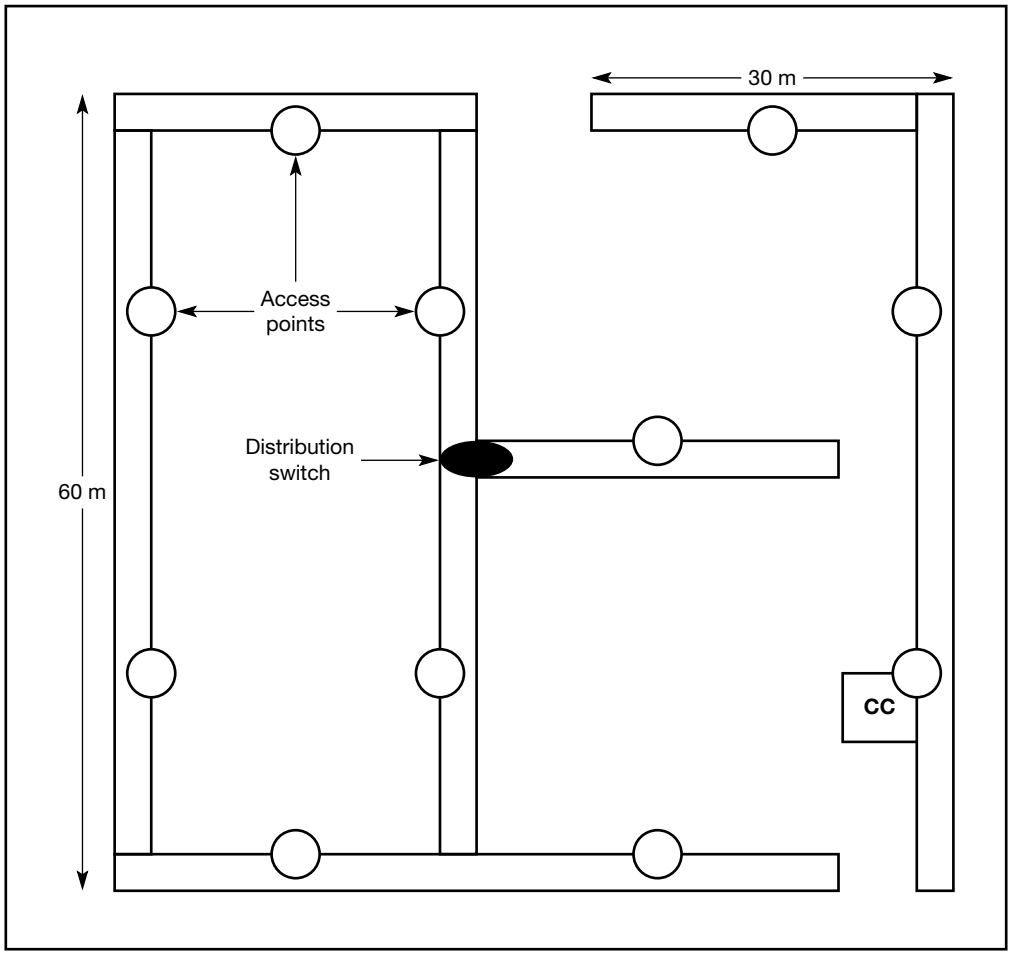


Figure 12.11 The Hostel Design La yout for APs

of wireless access, which is sufficient for gaming, multimedia streaming, file sharing and voice over Internet protocol (VOIP) applications.

The following devices/software will be used in the design:

1. WLAN security switch (2 numbers): Such a switch provides wireless threat protection and defends against RF attacks. It provides multicast management and delivers high-quality video and multimedia. It minimizes service disruption by providing self-healing and resilient design. The topology is flexible in the sense that APs can connect to it at Layer 2 or 3.
2. WLAN APs (11 numbers): These will be used with the above switches. They have dual radios, dual POE ETH ports, smoke detector, dual band diversity antennae and temperature control features.
3. 1-port 1000 Base GBIC (gigabit interface convertor) (4 numbers): This is used to convert electric signals to serial optical signals and vice versa. It also interfaces a fibre optic system with an Ethernet system. It can be adapted for either optical or copper applications and hot-swappable, and can be upgraded to electro-optical communication networks.
4. WLAN management software: It is an integrated tool for pre- and postdevelopment planning, systemwide configuration and upgrades, ongoing monitoring and reporting. It generates a detailed map showing calculated RF coverage and floor plan. It can calculate WLAN topology, AP placement and configurations, including power level and channel settings.
5. Ethernet routing switch with 24 POE ports: This is an 8-rack unit high switch, supporting up to 384 ports of Gigabit desktop connectivity and providing hardware-based Layer 3 routing at wire speed.

12.7 An application using wireless sensor networ ks

Problem statement: A wireless indoor location and orientation system for a visually impaired (VI) person using wireless sensors and Bluetooth technology

In this project, we design an application that will continuously, and in real time, determine the position of the VI person from a distance, within a confined space. This helps the VI person to move around in an unknown and unfamiliar indoor environment by providing him with the relevant contextual information through auditory cues issued by the system, based on static and dynamic data collected by the obstacle avoidance system.

Thus, in this project we show how to integrate several technologies, including wearable computers, wireless sensors, voice recognition and synthesis systems and Bluetooth. The following hardware and software are used:

1. MOTEkit-5040
2. Wireless microphone
3. Bluetooth adapter and Bluetooth-enabled headphones and webcams
4. Network-embedded system C (NesC) to program the motes
5. TinyOS operating system for communication
6. J2SE1.4.1 using EclipseSDK3.1.2 on windows, and
7. Java Speech API (JASPI) and Java 3D API (J3D)

Design

Four important design steps are as follows:

1. Positioning the VI: We use Mica2 motes as a wireless sensor network; at all places of importance (choke points), like someone's room, corridor and staircase, Mica2 motes are placed

(these are called passive motes or PM). Each of these devices has a unique identification number and is in persistent connection with the base station or central server on which the application runs; the VI also wears a Mica2 mote (called active mote or AM). This underlying network helps to locate the VI's present location.

2. Marking the destination: A wireless collar microphone and speech recognition module is provided to the VI to mark his destination, by speaking into it. The speech is converted into text using JAPI at the base station. The path-finding module is run next, which takes the current location of the VI and destination name as inputs, operates on a database containing the topology of the network, to give the best route using the shortest path algorithm.

3. Information presentation: Information is presented to the VI in a fairly simple manner. The database contains the distances between the choke points in terms of standard steps (of size = 1.75 ft), so when the path has been calculated the VI gets auditory cues like 'move 7 steps ahead', 'turn right', 'staircase ahead 10 steps,' etc., using a Bluetooth headphone worn by him.

4. Contextual information: While the VI is directed to the destination, it is to be ensured that he is given the direction of the nearest choke point only. When he reaches there, he is informed where he is, for example, 'in front of the Head's office'. From there, he gets the directions to the next choke point and so on until the destination is reached.

Figure 12.12 shows the block diagram depicting the proposed orientation system. For details, the reader is referred to Gupta and Chaturvedi (2006).

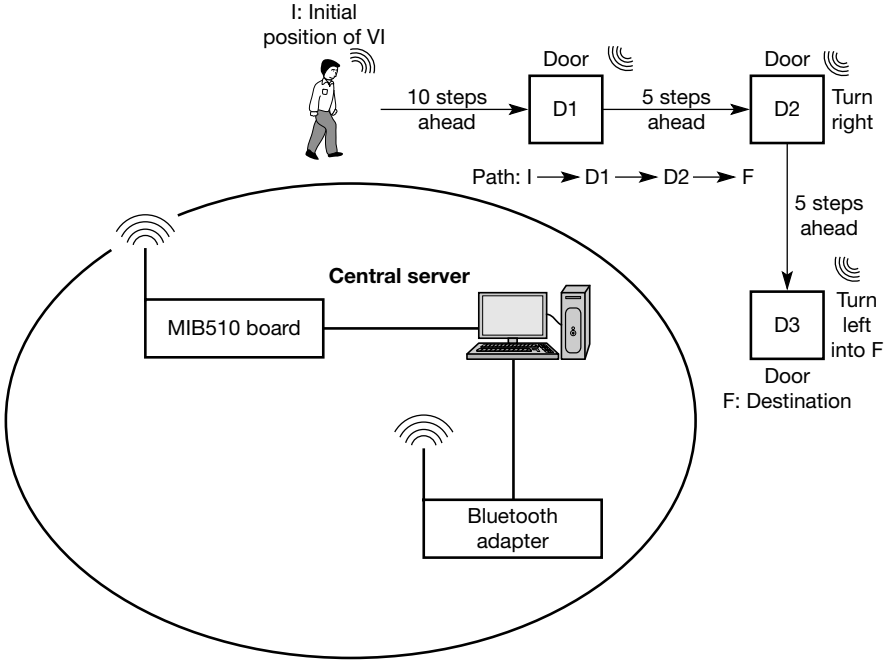


Figure 12.12 The VI System

12.8 Summary

No book on mobile computing can be considered complete without a provision for hands-on experience in the simulation, design and/or implementation of various areas/topics discussed in the chapters in the book. Many hardware and software projects can be thought of in these areas and need to be taken up to thoroughly understand the related topics. Some of these topics are Mobile IP and its implementation; design of IEEE 802.11 LAN for buildings like hostels, restaurants, etc.; use of wireless sensor networks in environment sensing; simulation and comparison of MANET protocols; chatting and other related applications using Bluetooth; gateway design for WAP; mobile agent applications in network monitoring and information retrieval, etc. These are only some examples of projects that can be carried out by students. Many variations and other extension projects are possible to carry out, as mentioned in the problems given below, but details of these cannot be given here. Readers are advised to try implementing these by themselves.

Problems

1. Extensions for TCP in the wireless environment have been discussed in Chapter 5. Simulate these using the Qualnet simulator.
2. Design Mobile IP by incorporating optimization in routing, that is, without triangular routing.
3. Give a paper design for providing Wi-Fi in a small hotel, on the lines of the design for a typical hostel as given in this chapter.
4. Many new protocols have been proposed recently for Manets, for example, TORA. Simulate TORA on Qualnet and compare its performance with that of AODV.
5. JADE is a new, open source mobile agent platform in Java. Design and implement a software distribution system using JADE mobile agents.
6. Design a two-player chess program to be played on handheld devices using Bluetooth.
7. Design a data compression application using TinyOS on a wireless sensor network of mica motes.
8. Design a security system for Manets using the IP address of the nodes to generate the symmetric keys for encryption/decryption.
9. JSim is another simulator available for wireless networks. Simulate the impact of routing attacks in unicast and multicast routing protocols in Manets.
10. Design a Personal Information Manager on a typical PDA using an emulator.

Multiple-choice questions

1. Which one of the following is used by the mobile node for registration in Mobile IP?
   (a) Solicitation
   (b) Advertisement
   (c) Visitor list
   (d) Binding record

2. Which one of the following is used to ensure that packets form the mobile node are not rejected in Mobile IP?
   (a) IP forwarding
   (b) IP masquerading
   (c) Spoof protection
   (d) Proxy ARP

3. Which one of the following parameters is normally used to simulate the traffic type in Manet simulation?
   (a) Constant bit rate
   (b) Available bit rate
   (c) Variable bit rate
   (d) Real-time variable bit rate

4. Which one of the following is used for providing portable Java applications in Bluetooth?
   (a) JABT
   (b) JABWT
   (c) JBT
   (d) JABA

5. Which one of the following is the facility provided by the SDDB?
   (a) Abstract database of records
   (b) Collection of attributes for Bluetooth services
   (c) Update and retrieval of service records
   (d) All of the above

6. Which one of the following port numbers is used to create a server in the WAP Gateway design?
   (a) 9000
   (b) 9100
   (c) 9200
   (d) 9300

7. To determine whether a push message is valid, we use DTD. Which one of the following is the full form of DTD?
   (a) Database Type Definition
   (b) Document Type Definition
   (c) Database Type Data
   (d) Document Type Data

8. Which one of the following is the function of the Tahiti Server in the IBM Workbench?
   (a) Creating and loading aglets
   (b) Despatching and retracting aglets
   (c) Loading class files and invoking run()
   (d) All of the above

9. Which one of the following standards is best for setting up a Wi-Fi system in a typical hostel?
   (a) IEEE 80 1.a
   (b) IEEE 801.b
   (c) IEEE 801.g
   (d) All of the above

10. Which one of the following is the function of Java Speech API (JASPI)?
- **(a)** Converting speech to text
- **(b)** Converting text to speech
- **(c)** Neither of the above
- **(d)** Both of the above

Further reading

A.M. Law and W.D. Kelton (1991), *Simulation Modeling and Analysis* (New York: McGraw-Hill).

A.S. Tanenbaum (2003), *Computer Networks*, 4th ed. (New Delhi, India: Prentice Hall India).

D. Horvat, D. Cvetkovic and V. Milutinovic, (2000), 'Mobile Agents and Java Aglet Toolkits', in Proceedings of the 33rd Hawaii International Conference on System Sciences (HICSS 2000), Hawaii, USA, January 2000, pp. 3090–3099.

D.B. Lange and M. Oshima (1998), Programming and Deploying Java Mobile Agents with Aglets (Addison-Wesley).

H. Schildt (2001), *Java the Complete Reference*, 4th ed. (McGraw-Hill).

J. Hsu, S. Bhatia, M. Takai and R. Bagrodia (2005), 'Performance of Mobile Ad Hoc Networking Routing Protocols in Realistic Scenarios', Scalable Network Technologies Inc. report.

JavaTM APIs for Bluetooth (JSR-82), www.jcp.org/en/jsr/detail?id=82 (accessed May 2005).

MIDP 1.0: Introduction to MIDlet Programming Forum Nokia, 2004, www.forum.nokia.com (accessed May 2005).

P.K. Gupta and A. Chaturvedi (2006), 'DRISHTI—Indoor orientation for visually impaired', B.Tech. project, IIT Roorkee.

Specification of the Bluetooth System, Version 1.1, Volume 1: Core, www.bluetooth.org/spec (accessed July 2005).

Sun Blueprint, 'Designing Wireless Clients for Enterprise Applications with Java', www.java.sun.com (accessed October 2006).

WAP Forum (2001), 'Wireless Application Protocol Architecture Specification', www.wapforum.org (accessed July 2001).

www.cs.hut.fi/research/Dynamics (accessed May 2006).

Appendix A
Java Networ k Programming

The basis of all mobile computing is distributed process communication. Software processes, whether in the same system or remote to each other, must cooperate and communicate with each other to complete any task. With the advent of networking, processes have become more and more distant from each other, and there is a need to provide efficient communication between them. With fixed systems, this is not much of an issue these days, but as nodes and their processes become mobile, initiation and continuance of such interprocess communication has become important. Security is also a major issue in mobile and wireless systems, hence the dominance of Java language for programming such systems.

In this appendix, we present an introduction to **Java as a network programming language**. We shall assume here that the reader is familiar with Java as an object-oriented programming language similar to C++ in its general syntax and semantics; also that the Java source code is compiled into a standardized and platform-neutral byte code, which is interpreted and executed on the fly by a Java runtime environment (JRE) in the operating system (OS). The byte code interpreter is called a Java Virtual Machine (JVM) and translates the generic instructions into native commands of the specific OS or processor, the application is currently running on. For a comprehensive introduction to Java programming, the reader is referred to Schildt (2002).

We shall see the ways and means in which the Java language is used for developing platform-independent software and how it supports security in mobile systems. Also discussed are some of the basic techniques of distributed process communication. These include sockets, remote procedure call (RPC) and its Java incarnation, the remote method invocation (RMI). It is assumed that the reader is already familiar with the concepts of client/server computing.

A.1 Java programming language

Although Java originated in 1995 when some people in Sun Microsystems were trying to develop a new language that would be suitable for programming information-oriented consumer applications, today it is the enabling Internet technology and is seen as a secure programming language for mobile and ubiquitous computing.

Java differs from other programming languages like C++ in that, besides being used to create normal applications, it has the ability to create 'applets'. An **applet** is a small application designed to be transmitted over the Internet, automatically installed and executed by a Java compatible Web browser. It is a tiny Java program, dynamically downloaded across the network, just like an image, sound file or video clip, but is an **intelligent program**, not just an animation or media file; that is, it can react to a user input and dynamically change.

A simple applet for printing 'A Simple Applet' is shown below:

```
import java.awt.*;
import java.applet.*;
public class SimpleApplet extends Applet {
    public void paint (Graphics g) {
        g.drawstring ('A Simple Applet', 20,20);
    }
}
```

Note that the abstract window toolkit (AWT) class shown above contains support for a window-based, graphical interface and applets that interact with the user through the AWT, not through the console-based I/O classes. The applet package contains the class **applet**, and every applet that is created must be a subclass of **applet**.

For downloadable programs like applets to run successfully, two important problems associated with them have to be addressed—portability and security. The importance of Java lies in the fact that it addresses both these problems efficiently and elegantly.

In this book, the emphasis has been on mobile objects and their secure communication. One of the most important aspects of Java for mobile and distributed object computing is its **security** properties. Java is inherently a type-safe language. It has strong typing, true arrays with bounds checking and no pointers. These restrictions make it impossible for a Java program to construct a pointer to read and write arbitrary memory locations.

Java security technology includes a large set of APIs, tools and implementations of commonly used security algorithms, mechanisms and protocols. The Java security APIs include many areas, like cryptography, public key infrastructure, secure communication, authentication and access control. Java security technology provides the developer with a comprehensive security framework for writing applications. It also provides the user or administrator with a set of tools to securely manage applications. Underlying the Java security platform is a dynamic, extensible security architecture, which is standards-based and interoperable. Security features like cryptography, authentication and authorization, public key infrastructure, etc., are built in.

The Java security model was originally based on a customizable 'sandbox' in which Java software programs could run safely, without potential risk to systems or users, from malicious Java code, like viruses and Trojan horses. This original Java sandbox model is shown in Figure A.1. It provided a very restricted environment to run untrusted code. The basic idea was that

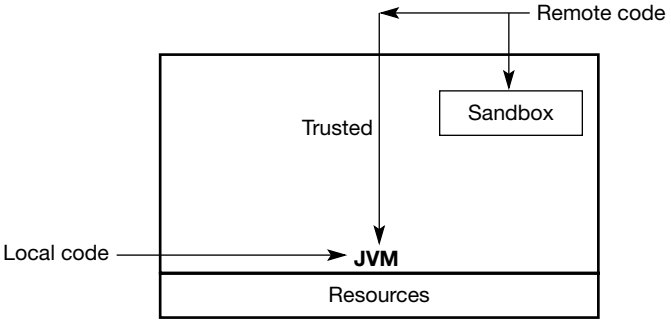


Figure A.1 Java security model

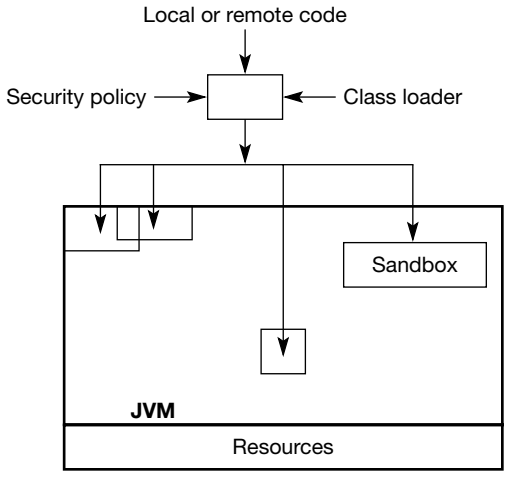


Figure A.2 Java 2 security model

downloaded code such as an applet could not be trusted and therefore should not access vital resources such as the file system. Local code, however, was trusted and had full access to the resources.

The J2SDK v1.4 has enhanced its security and provides a security model that is policy-based. It is easily configurable and provides fine-grained access control. The policy guards a resource and specifies which permissions are available. Unless permission is granted to access a protected resource, the code cannot access it. This access control can be specified for all Java code such as applications, beans and applets. See Figure A.2. The concept that all local code is trusted has been removed.

The classes in the **java.security** package form the basis for authenticating signed Java classes. It is a complex application-programming interface (API) responsible for providing the security provider interface, the message digests, the keys and certificates, the digital signatures and encryption. The **java.policy** file in this package specifies the security policies. A particular policy can be specified within one or more policy configuration files. These files contain a list of entries that map the permissions allowed for code from specific sources. The policy configuration file contains a **keystore** entry to look up the public keys of the signers in the entries.

Most mobile computing applications require security to be implemented. The above built-in facilities in Java make it easier to write secure applications, as compared to code written in other languages, where security features are afterthought add-ons only.

A.2 Socket programming

Network programming is tightly integrated in Java. Java provides **socket-based communication**, which enables programs to communicate through designated sockets. A **socket** is an abstraction that facilitates communication between the client and the server in a client-server architecture system. Java treats socket communication as it treats I/O operations, in that a program can read from or write to a socket just as it reads from or writes to a file. The classes defined in the **java.net** package make Java a good language for networking. This package needs to be imported when writing Java network programs.

On the Internet, we refer to a TCP or a UDP socket (Java **stream socket** or **datagram socket**), which is made up of an Internet protocol (IP) address concatenated with a port number. The IP address may be a 32-bit IPv4 address or a 128-bit IPv6 address. A port is a 16-bit number local to the host.

Example of a socket: (149.164.29.27, 8000)

To obtain TCP service, a connection must be explicitly established between a socket on the sending machine and a socket on the receiving machine. For example, a server waits for incoming requests by listening to a specified port. When a request is received, the server accepts a connection from the client socket to complete the connection.

Some specific or standard service servers have been defined, which listen to well-known ports. All such ports are numbered below 1024. For example, the telnet port is 23, the file transfer port (ftp) is 21, the Web (http) port is 80, the e-mail port is 25 and the time-of-day port is 13. All user-defined or arbitrary ports have a number greater than 1024.

While writing a program for a typical client-server communicating through sockets, the following two sockets must be created:

The server class creates a **server-socket s** and attaches it to port 8000 using the statement:

```
ServerSocket s = new ServerSocket (8000).
```

The server then starts to listen for connection requests using the statement:

```
Socket connectToClient = s.accept().
```

The server waits until a client requests a connection. Once connected, it reads the data sent from the client through an input stream, processes it as required and sends the result to the client through the output stream.

The client uses the following statement to create a socket that will request a connection to the server at port 8000.

```
Socket connectToServer = new Socket ('149.164.29.27', 8000).
```

Serving multiple clients

It is quite common to have multiple clients that connect to a single server at the same time. A server may run constantly on a server computer, and clients from all over the Internet may want to connect to it. Threads are used to handle the server's multiple clients simultaneously. One thread is created for each connection. The server establishes a connection in the following way:

```
While (true)
{ Socket connectToClient = s.accept();
   Thread t = new ThreadClass(connectYoClient);
   t.start();
}
```

The above server socket can have many connections. Each iteration of the **while** loop creates a new connection. When the connection is established, a new thread is created to handle the communication between the server and this new client, which allows multiple connections to run at the same time.

For practice in programming with sockets, the reader is referred to Liang (1998), Schildt (2002) and Horstmann and Cornell (2007).

Table A.1 Socket primitives for TCP

Primitive	Meaning
SOCKET	create a ne w communication endpoint
BIND	attach a local address to a sock et
LISTEN	announce willingness to accept connections
ACCEPT	block the caller till a connection attempt ar rives
CONNECT	actively attempt to establish a connection
SEND	send some data o ver the connection
RECEIVE	receive some data from the connection
CLOSE	release the connection

There are many socket primitives/calls that can be used. Sockets calls and their meanings and functions are listed in Table A.1 above.

A.3 Remote procedure call (RPC)

The procedure call is one of the oldest techniques used in programming languages and is a simple and efficient mechanism for processes to call procedures or functions in the same processor or system. Its variations include the call back and the mailbox techniques. See Figures A.3–A.5.

Procedure call

Assume that entity A needs the services of entity B in order to carry out its purpose, and B in turn needs the services of entity C in order to fulfill its obligation to A.

The procedure call technique is fast and, because of its sequential nature, makes it easy to follow a program's flow of execution. However, it is also inherently synchronous and difficult to parallelize. Java's RMI system is based on the procedure call technique.

Callback technique

This scenario is similar to what we saw with the procedure call technique; however, instead of asking and then waiting for something, entities A and B ask for something and then continue on with their tasks. When entities B and C have done whatever they were asked to do, they call back the original caller and give it what it asked for. The original caller must stop what it is doing and deal with the callback.

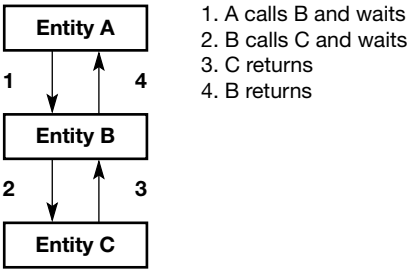


Figure A.3 Traditional procedure call

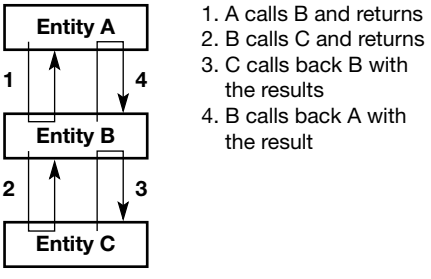
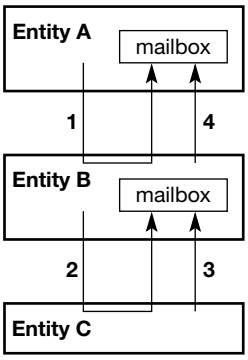


Figure A.4 Callback technique



1. A calls B and returns with a mailbox
2. B calls C and returns with a mailbox
3. C updates the mailbox
4. B updates the mailbox

Figure A.5 Mailbox technique

The callback technique is useful because it permits truly asynchronous processing. However, it is also complicated, and makes the program's flow of execution more difficult to follow. Java's GUI toolkits (AWT—Advanced Windowing Tool Kit and Swing—a package which provides platform independent programming) are based on the callback technique.

Mailbox technique

Here, entity A asks entity B to do something and tells entity B to put the finished result in its mailbox. Entity A then goes about its business, checking the mailbox periodically to see if entity B is finished. Or entity A may simply stop and wait for entity B to finish its part, as in the procedure call technique. Entities B and C interact in the same way.

The mailbox technique is more difficult to implement than either of the other two techniques; however, it allows for asynchronous processing, while at the same time avoiding the problem of the confusing flow of execution. These two factors turn out to be very important in distributed systems.

However, when processes are remote to each other, as in a network, they use a **remote procedure call** (RPC) mechanism based on the message-passing primitives **send** and **reply** to communicate with each other. RPC message traffic takes the form of a request message from a client to a server followed by a reply message from the server back to the client, very much like calling a procedure. This is shown in Figure A.6.

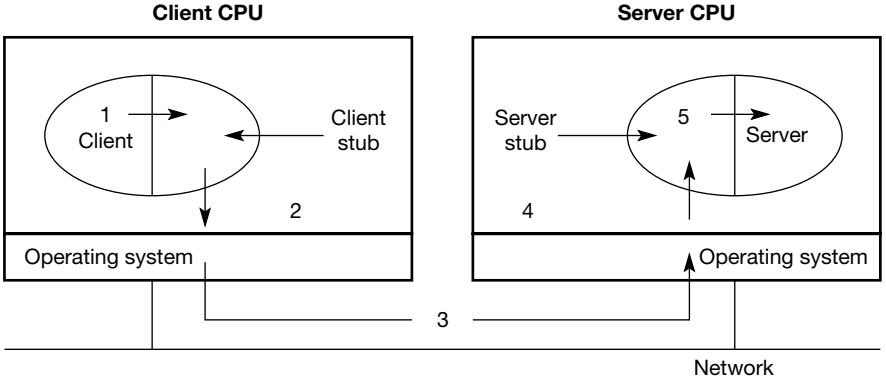


Figure A.6 Remote procedure call mechanism

Here, 1 and 5 are procedure calls, 2 and 4 are procedure returns and 3 is a message.

RPC is a better mechanism for communication than sockets. It manages the channel better in the sense that the application program can be written such that the location of the procedure, whether local or remote, is transparent to the client and the server. It is also a higher-level communication method when compared to sockets.

A.4 Remote method in vocation (RMI)

This is a Java feature similar to RPCs. It has been designed to provide a distributed object development tool for Java. It allows a thread to invoke a method on a remote object on different JVMs.

Although they are quite similar, there are two basic differences between RMI and RPC. RPCs support only procedural programming where only remote procedures/functions may be called, whereas RMI is object-based and supports invocation of methods on remote objects. Furthermore, in RPC, parameters to remote procedures are ordinary data structures, whereas in RMI, objects can be passed as parameters to remote methods.

To make remote methods transparent to both the client and the server, RMI implements special remote objects called **stubs** and **skeletons**, as shown in Figure A.7.

The RMI protocol interface lets Java objects on different hosts communicate with each other in a transparent way. Clients can invoke methods of a remote object as if they were local methods. This preserves the object-oriented paradigm in distributed computing.

A stub is a proxy or surrogate for the remote object and resides with the client. It uses Java object serialization. Serialization is the process of writing the state of an object to a byte stream. Serialization creates a **parcel** consisting of the name of the method to be invoked on the server and that method's parameters. This is called **marshalling** the parameters. It then sends this parcel to the server where it is received by the skeleton for the remote object.

The skeleton is the proxy for the object on the server side. It uses deserialization to **unmarshal** the parameters and invokes the desired server method. It then marshals the return value (or exception if any) into a parcel and returns to the client. The stub unmarshals the return value and passes it to the client.

A simple client/ser ver application using RMI

We shall again take the example of designing a simple client/server application using RMI. The **java.rmi** package must be imported for this purpose. In this program, we develop an RMI-based client-server system, which would read the server's date and time and modify it according to

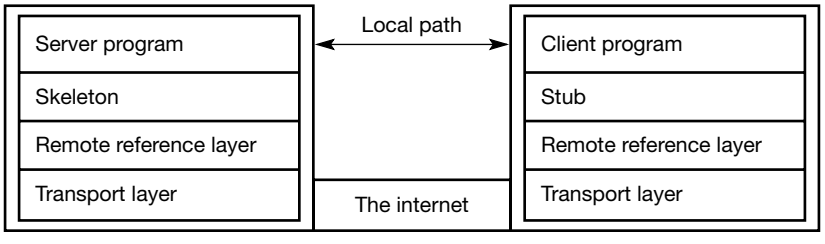


Figure A.7 The RMI technique

user's input at client. To create this application, the first step is to define a remote interface between the client and server objects, as follows:

```
public interface DateServer extends Remote
{
public String getTime()throws RemoteException;
.......
}
```

Then, we develop the remote object and its interface. The server is a simple unicast remote server. We create server by extending java.rmi.server.UnicastRemoteObject as follows:

```
public class DateServerImpl extends UnicastRemoteObject implements DateServer
{
   DateServerImpl() throws RemoteException
   {
   super();
   }
```

We implement all the required remote methods in this class

```
public String getTime() throws RemoteException
{
.........
}
```

In the main method of the server object, the RMI security manager is created and installed. It protects access to system resources from untrusted downloaded code.

```
System.setSecurityManager(new RMISecurityManager());
```

The server will use the **java.rmi.Naming** class to bind its name to the registry. We call it 'DATE-SERVER'.

```
Naming.rebind('DATE-SERVER', Server);
```

In the next step, we develop a client program (DateClient.java). In order for the client object to invoke methods on the server, it must first look up the name of server in the registry. We use the java.rmi.Naming class to lookup the server name.

```
DateServer remoteObject = (DateServer) Naming.lookup
                          ('//localhost/DATE-SERVER');
```

The remote method is then invoked using the remote interface name (remoteObject) as prefix and the remote method name (getTime) as suffix.

```
System.out.println('Time at Server' + remoteObject.getTime());
```

This completes our RMI Client-Server system implementation.

// ADDITIONAL: about compilation and running the program

Once the implementation is complete, we need to generate the stub and skeleton code. The RMI system provides an RMI compiler (rmic) that takes the generated interface class and

procedures stub code on itself. Here, we also have to include a simple file security.policy with contents as follows:

```
grant {
permission java.security.AllPermission;
};
```

Put all Java files inside a folder and execute the following commands from inside that folder.

```
start rmiregistry
javac DateServer.java
javac DateServerImpl.java
rmic DateServerImpl
java -Djava.security.policy=security.policy DateServerImpl
```

Then, open another command prompt window and run

```
javac DateClient.java
java -Djava.security.policy=security.policy DateClient
```

One basic disadvantage with RMI is in not providing **interoperability**. Since RMI uses object serialization for marshalling, it provides a Java-only solution. RMI cannot be used to communicate between Java applications and those written in other languages like C++. This disadvantage is removed in CORBA (common object request broker architecture).

CORBA is a middleware that allows interoperability between the heterogeneous client and server applications for communication. Together with OLE (object linking and embedding), it provides a language-neutral, distributed object framework for network access. For example, a C++ program could use CORBA to access a database service written in COBOL. CORBA was created and is managed by the Object Management Group (OMG), an open standards body with more than 700 companies as members. For more information on CORBA, the reader is referred to Orfali and Harkey (1998).

Further reading

A.S. Tanenbaum (2003), *Computer Networks*, 4th ed. (New Delhi, India: Prentice Hall India).

C.S. Horstmann and G. Cornell (2007), *Core Java 2, Vol. 1—Fundamentals*, 7th ed. (New Delhi, India: Pearson Education).

C.S. Horstmann and G. Cornell (2007), *Core Java 2, Vol. 2—Advanced Features*, 7th ed. (New Delhi, India: Pearson Education).

H. Schildt (2002), *Java 2: The Complete Reference*, 5th ed. (New Delhi, India: Tata McGraw-Hill).

http://java.sun.com/javase/technologies/security/index.jsp (accessed July 2007).

R. Helton and J. Helton (2002), *Mastering Java Security: Cryptography, Algorithms and Architecture* (Wiley Dreamtech).

R. Orfali and D. Harkey (1998), *Client-Server Programming with Java and CORBA* (John Wiley & Sons).

S. Oaks (2001), *Java Security* (O'Reilly Publishers).

W.R. Stevens (1998), *UNIX Network Programming* (New Delhi, India: Prentice Hall India).

Y.D. Liang (1998), *An Introduction to Java Programming* (New Delhi, India: Prentice Hall India).

This page intentionally left blank

Appendix B

Comparison Between Qualnet and NS2

In this appendix, we provide a comparison of the two simulators Qualnet and NS2 used in networking. While NS2 is a freely downloadable software, Qualnet is proprietary. The comparison is given in Table B.1.

Table B.1 Comparison of Qualnet and NS2 network simulators

Qualnet	NS2
1. Commercial simulator, based upon GloMoSim simulator.	**1.** Freely available for research and educational purposes.
2. Uses the parallel simulation environment for complex systems (PARSEC) for basic operations, hence can run on distributed machines.	**2.** Runs on a single machine, no parallel execution support in NS2.
3. Mainly developed for wireless scenario simulations, but wired networks also supported.	**3.** Mainly developed for wired networks, but its CMU extension facilitates the wireless network simulation.
4. Qualnet includes a graphical user interface for creating the model and its specification. So, it is very easy to specify small to medium networks by using the GUI.	**4.** To create and simulate a model, we have to specify all connections in a special model file manually. Uses OTcl for model file specifications.
5. Since it uses primarily Java for the GUI, it is available for Linux as well as for Windows. The simulator itself is the specified target system optimized C program.	**5.** It is designed for Unix systems but runs under Windows CygWin as well.
6. Faster simulation speeds and greater scalability are achievable through smart architecture and optimized memory management of Qualnet.	**6.** Not as fast and scalable.
7. Not used much in research as it is not freely available, hence lesser support (code samples etc) available on Web.	**7.** Widely used for research, hence large number of resources available on Web.
8. Simulation of wireless sensor networks is supported in Qualnet 4.5 (using ZigBee library).	**8.** No such support available as of now.

(*Continued*)

Table B.1 Comparison of Qualnet and NS2 network simulators (*Continued*)

Qualnet	NS2
9. Simulation of GSM mobile networks also supported.	**9.** GSM not supported in NS2.
10. Includes a variety of advanced libraries such as mesh networking, battery models, network security toolkit, and a large number of protocols at different layers.	**10.** These advanced libraries for wireless support are not available in NS2.
11. Includes a 3D visualizer for better visualization of a scenario.	**11.** 2D animator (NAM) is used with NS2.
12. Much easier for beginners who want to evaluate and test different existing routing protocols, as it can be done with GUI, without writing even a single line of code.	**12.** Requires knowledge of Tcl scripting before you can begin with NS2.
13. For implementing new protocols, Qualnet uses C/C++ and follows a procedural paradigm.	**13.** NS2 also uses C++ for new protocol/model development, but it uses object-oriented paradigm (usage of different classes) for programming.

Overall, in the current research on networks, NS2 is preferred as it is open source and freely available. There are plenty of resources freely available for NS2 (in terms of protocol code, mobility models, etc) on the Web, which have been developed by researchers. On the other hand, support for Qualnet is available only from its official forums.

For wireless networks simulation, Qualnet seems to be a good choice due to its rich libraries for wireless.

For those who just want to evaluate the performance of various existing protocols, Qualnet is undoubtedly a better choice, due to its ease of usage and better visualization tools.

Further reading

J. Hsu, S. Bhatia, M. Takai and R. Bagrodia (2005), 'Performance of Mobile Ad Hoc Networking Routing Protocols in Realistic Scenarios', Scalable Network Technologies Inc. report.

Index

Italicized page numbers indicate illustrations.