CHAPTER 1

Short Message Service and IP Network Integration

Short Message Service (SMS) is a mature wireless communication service [Lin 01b, 3GP04l, Nok 97]. Most modern digital cellular phone systems offer SMS, which is considered a profitable added-value service. A natural extension is to integrate SMS with electronic mail services, which provides linkage between mobile networks and IP networks. Furthermore, several Internet applications over SMS can be implemented on similar platforms. This chapter uses the Global System for Mobile Communications (GSMC) SMS as an example to illustrate SMS-Internet integration.

GSM SMS provides a connectionless transfer of messages with low-capacity and low-time performance. Each message can contain up to 140 octets or 160 characters of the GSM default alphabet [ETS97a]. The short messages are transported on the GSM Stand-alone Dedicated Control Channel (SDDC). Since a voice session utilizes GSM radio traffic channels, short messages can be received and sent while the mobile users are in conversation.

The GSM SMS network architecture is illustrated in Figure 1.1. In this architecture, when a Mobile Station (MS) sends a short message, this message is delivered to the GSM radio system, that is, a Base Transceiver Station (BTS) and then a Base Station Controller (BSC). The radio system then forwards the message to a Mobile Switching Center (MSC) called SMS
2 Short Message Service and IP Network Integration

Inter-Working MSC (IWMSC). Details for BTS, BSC, and MSC are given in Chapter 2.

The IWMSC passes this message to a Short Message Service Center (SM-SC). Upon receipt of a short message, the SM-SC may send an acknowledgment back to the originating MS (sender) if the acknowledgment request is specified in the short message. The SM-SC then forwards the message to the destination GSM network through a specific GSM MSC called the SMS Gateway MSC (SMS GMSC). Following the GSM roaming protocol (see Section 9.2.1), the SMS GMSC locates the serving MSC of the message receiver and forwards the message to that MSC. This MSC broadcasts the message to the BTSs, and the BTSs page the destination MS (receiver). Every short message contains a header in addition to the body. The header includes the originating MS address, the terminating MS address, the serving SM-SC address, a time stamp, and the length of the message body. Mobile Station ISDN Numbers (MSISDNs; the GSM telephone numbers), are used for addressing.

As a wireless data service, SMS has distinct features, such as handset alert capability and support for ME-specific and SIM-specific data. The MS is considered an “always-on” device that facilitates instant information exchange. No dial-up modem connection is required to access SMS. Furthermore, SMS provides message storage when the recipient is not available. It also allows simultaneous transmission with GSM voice, data, and fax services. On the other hand, SMS has drawbacks, such as narrow bandwidth and long latency of end-to-end transmission. To design an SMS-IP system, the aforementioned SMS strengths and limitations cannot be ignored. In particular, the IP server itself may also be mobile (for example, the iSMS gateway described in this chapter can move when it provides services), and usage of the limited
wireless bandwidth must be carefully addressed. To facilitate the development of various SMS applications over the Internet, we need a generic gateway to interwork the GSM network and IP network, and many specific data format converters between IP application contents and the SMS byte codes.

In most commercial implementations, SMS and IP networks are integrated through SM-SC, as illustrated in Figure 1.2. In this figure, a gateway interworks the SM-SC to the IP network, where a specific protocol is essential for communication between the SM-SC and the gateway. Since SM-SC implementation is vendor-specific, the SM-SC-based SMS-IP integration solution depends heavily on the SM-SC vendors. Furthermore, the SMS-IP gateway is maintained and controlled by GSM operators. It is difficult for a third party to deploy new services via SMS without having full cooperation from GSM operators. Also, from the GSM operator’s viewpoint, maintaining a reliable, secure, scalable interconnection platform between individual service providers and the SMS-IP gateway will not be an easy task. The SMS-IP approach typically utilizes a centralized gateway, where performance, scalability, and reliability issues must be carefully considered. To address these issues and to further support an environment for quick prototyping and hosting wireless data service, an endpoint SMS-IP integration solution called iSMS was proposed in [Rao01a]. This solution is transparent to an existing SM-SC and GSM network. This chapter first describes a SM-SC-based SMS-IP system called NCTU-SMS. Then we introduce iSMS, a non-SM-SC based SMS-IP system, and elaborate on the designs and implementations of several iSMS applications.

1.1 SMS-IP Integration with SM-SC

In a joint project between the FarEasTone Telecommunications Corporation and National Chiao Tung University (NCTU), we developed a web-based short message system called NCTU-SMS [Hun 04]. NCTU-SMS integrates
4 1 Short Message Service and IP Network Integration

Figure 1.3 The Web-based Graphical User Interface for NCTU-SMS

SMS with IP through SM-SC. With a user-friendly Graphical User Interface (GUI; Figure 1.3), NCTU-SMS allows users to input short messages through a web site. Several commercial web-based SMS services are now available. Among them, a version of NCTU-SMS was the first system utilized by the Taiwanese government (specifically, E-Land County) for civilian services. This system can, for example, notify a citizen that his/her driver’s license should be renewed. At NCTU, students who take the personal communication course receive their test scores and final grades through short messages generated by NCTU-SMS.

1.1.1 NCTU Short Message System

The NCTU-SMS architecture is illustrated in Figure 1.4, and consists of an Application Client (AC), an Application Server (AS), and the SM-SC. A delivery state tree (see Figure 1.5) is used to indicate the status of every short message delivery. The delivery state of a short message is recorded in the Application Client. To send a short message, the following steps are executed:
1.1 SMS-IP Integration with SM-SC

Figure 1.4 The NCTU-SMS Architecture

Step 1. A user (Figure 1.4 (1)) issues a short message through the web-based GUI. Specifically, the user inputs the destination phone number (MSISDN) and types the message text. When the user presses the OK button, the Application Client (Figure 1.4 (2)) generates a message delivery record that records
- the destination MSISDN
- the time $T_i$ when the message is issued,
- the current delivery state,
- the time $T_s$ associated with the delivery state, and
- other parameters.

At this step, the delivery state is **Init** and $T_s = T_i$.

Steps 2 and 3. The Application Client forwards the short message to the Application Server (Figure 1.4 (3)), and the delivery state is changed to **Send-To-AS**. If the communications link between the Application Client and the Application Server is disconnected, the Application Client sets the delivery state to **AS-Failure**. Otherwise, the Application Server receives the message delivery request from the Application Client and performs format checking (for example, the destination phone number format, the message text format, and so on). If the format check fails, an error message is sent back to the Application Client. In this case, the Application Client updates $T_s$ to the current time, the delivery state is set to **AS-Failure**, and the procedure exits. If the format check succeeds, then an acknowledgment message is sent back to the Application Client, and the Application Server forwards the short message to the SM-SC.
Figure 1.5  The Delivery State Tree (AC: Application Client; AS: Application Server)

(Figure 1.4 (4)). When the Application Client receives the acknowledgment, it updates $T_s$ and sets the delivery state to SMS-Delivery.

**Step 4.** When the SM-SC receives the short message, it delivers the message following the standard SMS procedure illustrated in Figure 1.1. Note that the message may not be actually delivered if errors occur, for example, the mobility database *Home Location Register (HLR)* (see Chapter 2 and Section 9.2.1) cannot identify the destination MS. In this case, an error message is sent back to the Application Client. The Application Client updates $T_s$ to the current time, the delivery state is set to Delivery-Failure, and the procedure exits. In the normal case, the message will be sent to the destination MS, and the next step is executed.

**Step 5.** When the destination MS receives the short message, an acknowledgment is sent back to the Application Client. The Application Client updates $T_s$ to the current time, the delivery state is set to Delivery-Success, and the procedure exits. Sometimes the short message may
1.1 SMS-IP Integration with SM-SC

not be received by the destination MS (for example, the destination MS is power off). In this case, the short message will be periodically resent until either the destination MS receives the message or the SM-SC gives up. In the latter case, an error message is sent back to the Application Client. The Application Client updates \( T_s \), sets the delivery state to \textbf{MS-Not-Available}, and exits the procedure.

From the above description, if the delivery is successful, the final delivery state is \textbf{Delivery-Success}. If the delivery fails, the final delivery state is either \textbf{AS-Failure}, \textbf{Delivery-Failure}, or \textbf{MS-Not-Available}.

1.1.2 Statistics for SMS Delivery

Several interesting statistics can be derived from the delivery records in the Application Client—for example, the hourly distribution \( T^*_i \) of \( T_i \) (the time when a short message is issued), where

\[
T^*_i = T_i \mod 24.
\]

We are also interested in the distribution of the issued times \( T^*_{i,MNA} \) for those failed SMS deliveries with the final state \textbf{MS-Not-Available} (MNA). Let \( p^*_i(t) \) be the probability that an SMS is issued during the \( t \)th hour (where \( t = 0, 1, 2, \ldots, 23 \)) see Fig.1.6. Similarly, let \( p^*_{i,MNA}(t) \) be the probability that an \textbf{MS-Not-Available} SMS is issued during the \( t \)th hour. Figure 1.6 plots the \( p^*_i \) and \( p^*_{i,MNA} \) curves. The \( p^*_i \) curve indicates that most short messages are issued during normal work hours (8:00 A.M.–5:00 P.M.), with a major peak at 9:00 A.M.. It implies that many SMS users tend to send short messages immediately after they arrive at the office. It is also apparent that the number of issued short messages drops during the lunch break. The \( p^*_{i,MNA} \) curve follows the same trend as the \( p^*_i \) curve. It is interesting to observe that \( p^*_{i,MNA} > p^*_i \) during 8:00 P.M.–1:00 A.M.. This phenomenon can be explained by the following human behavior: We first note that in NCTU-SMS, the short messages are dropped after a resent period of 8 hours. Therefore, the \textbf{MS-Not-Available} short messages are dropped around the eighth hour after they are issued. Our experience indicates that many people turn off their MSs during 8:00 P.M.–9:00 A.M. Therefore, many short messages issued during 8:00 P.M.–1:00 A.M. will be dropped. At 9:00 A.M., a large number of short messages are issued. We observe a nontrivial phenomenon that \( p^*_{i,MNA} > p^*_i \).

From the measured data we collected, Table 1.1 shows the numbers of deliveries, with various final states. For the deliveries with the \textbf{AS-Failure} state, they failed due to sender mistakes (for example, inputting the incorrect
phone number formats). For the deliveries with the **MS-Not-Available** state, they failed due to unavailability of the receivers. Therefore, to consider the failures resulting from the SMS delivery system, we should only consider the **Delivery-Failure** short messages, and this system failure probability is 3.57%.

The most important statistics are the round-trip transmission delays $T_D$ of short messages, which are defined as

$$T_D = T_s - T_i$$

**Table 1.1** Numbers of deliveries, with various final states

<table>
<thead>
<tr>
<th>STATE</th>
<th>AS-FAILURE</th>
<th>DELIVERY-FAILURE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number</td>
<td>423</td>
<td>1649</td>
</tr>
<tr>
<td>Percentage</td>
<td>0.92 %</td>
<td>3.57 %</td>
</tr>
<tr>
<td><strong>STATE</strong></td>
<td><strong>MS-Not-Available</strong></td>
<td><strong>DELIVERY-SUCCESS</strong></td>
</tr>
<tr>
<td>Number</td>
<td>4278</td>
<td>39,780</td>
</tr>
<tr>
<td>Percentage</td>
<td>9.27 %</td>
<td>86.23 %</td>
</tr>
</tbody>
</table>
1.1 SMS-IP Integration with SM-SC

Figure 1.7 The $T_D$ Distribution (Delivery-Success; the normalized factor is 1.1)

Figure 1.7 illustrates the $T_D$ probability distribution for successful deliveries, which is normalized in the range [1, 10] (the normalization factor is 1.1). Figures 1.8 and 1.9 illustrate the $T_D$ probability distribution for failed deliveries. That is, consider the Delivery-Failure and MS-Not-Available short messages as the sample space. Figure 1.8 plots the Delivery-Failure part of the $T_D$ distribution, and Figure 1.9 plots the MS-Not-Available part. Figure 1.8 indicates that most messages in the Delivery-Failure state will be detected within 10 seconds. Figure 1.9 indicates that most messages in the MS-Not-Available state are reported within 15 minutes after the 8-hour timeout.

Figure 1.8 The $T_D$ Distribution for Delivery-Failure
1.2 iSMS System Architecture

Figure 1.10 illustrates the architecture for iSMS, a non-SM-SC based SMS-IP integration solution. In this architecture, an iSMS gateway is introduced. No components in the GSM and IP networks are modified. The MS is a commercial handset product that does not require installing any new software.

A major difference between the iSMS architecture and the SM-SC based architecture is that the iSMS gateway connects to an MS instead of the SM-SC. This MS serves as the GSM-compliant modem that provides iSMS wireless access to the GSM network. We refer to this MS as the MS modem. In our implementation, the iSMS gateway is a PC running on Windows or UNIX operating systems. The PC-based gateway can be a desktop with high...
reliability and availability. The PC can be replaced by a notebook. In this case, the gateway becomes a mobile server, which may move around just like a GSM MS. The MS modem and the iSMS gateway can be connected by the serial port, the Infrared port (for example, IrDA), the USB interface, or the PCMCIA interface. In iSMS, the data sent from the IP network to the GSM network is automatically packaged into short messages. The short messages can be multicast up to 65,535 receivers. Similarly, an MS can broadcast short messages to several servers connected to the iSMS gateway. The iSMS system is identified by the IP network through the IP address assigned to the gateway, and is addressed by the GSM network through the MSISDN of the MS modem attached to the gateway.

In the iSMS model, a user of an iSMS application is also a GSM customer. There are several alternatives to access iSMS services. For example, a user may make a request by sending a short message. This short message is terminated at the MS modem connected to the iSMS gateway. Upon receipt of the short message, the iSMS gateway executes the corresponding routines and returns the results to the user. The iSMS gateway may also be triggered by pre-defined, customized events. In a home security application, for example, if someone rings the doorbell when the user is not at home, the iSMS gateway may send an alerting short message to the GSM MS of the customer. The iSMS gateway consists of two parts (see Figure 1.10):

- **iSMS servers**: Responsible for service provisioning
- **Short message driver**: Responsible for communication between the GSM network and iSMS servers in the IP networks

The communication protocol between the MS modem and the short message driver (the reference point A in Figure 1.10) is implemented by using the SMS AT Command Set [ETS 98c, ETS98b]. The communication functions between the iSMS servers and the short message driver (the reference point B) is implemented through the iSMS communication Application Programming Interface (API) based on the TCP socket. These communication mechanisms are elaborated in Section 1.3.

During iSMS system initialization, the short message driver opens the serial port driver (reference point A) for sending/receiving short messages to/from the GSM network (via the MS modem). The short message driver also opens and listens on a pre-defined TCP port (reference point B) for server connection requests. For each connection request, the server will register the MSISDNs of its customers to the short message driver. Messages from registered senders will then be forwarded to the server.
The short message driver performs conversion between the TCP socket API (interface to the iSMS server) and the SMS AT Command Set (interface to the MS modem). The short message driver receives incoming short messages from the serial port and passes these messages to the iSMS servers according to a registration table. Depending on the registration status, the short message driver may forward a message to several iSMS servers or drop the message if no server has registered the sender of the message. For outgoing short messages, the driver receives messages from servers, transforms them into a short message format, and then sends them out to the GSM network via the serial port.

An iSMS server may run on the same host as the short message driver or on a remote site. For security reasons, the driver authenticates the servers for communication sessions. A command sent from the MS is pure text in the short message. The short message driver dispatches the message to the appropriate iSMS server based on the caller Id (i.e., the MSISDN of the requesting MS) and the mapping in the registration table. The server parses the command in the message body and then invokes corresponding internal functions or external agents to execute the messages. Functions are executed on the same address space as the server, while agents are running on different processes. The caller Id is used to identify the user in the current iSMS version. iSMS security is implemented at the application level. For example, a password may be required as a parameter of every command sent from the MS to the iSMS gateway.

To develop a new service, one implements iSMS servers that communicate with the short message driver by using the functions defined in the iSMS communication API. In the current iSMS version, we have implemented two kinds of servers:

- Email Forwarding Daemon
- Agent Dispatching Server

The iSMS platform allows developers to implement new server types. The relationship among various server types and the short message driver is illustrated in Figure 1.11. In this figure, the Email Forwarding Daemon ((1) in Figure 1.11) relays messages between the MSs and email systems on the IP domain. It supports an interface to Microsoft Exchange Server as well as to standard Simple Mail Transfer Protocol (SMTP) and Post Office Protocol Version 3 (POP3) [Mye 96].

The daemon converts a short message sent from an MS to email and forwards it to a SMTP server for delivery. The daemon may periodically query a mail server (for example, via a POP3 interface), pick up important
emails (according to user profiles), and send out SMS notifications to users’ mobile stations. The daemon software can be easily generalized to support other mail systems, such as AOL Instant Message.

A server of an agent-dispatching type ((2) in Figure 1.11) consists of an agent dispatcher and several agents, where each of the agents is an iSMS server. The dispatcher invokes the agent corresponding to the SMS message header and passes the message body as the parameter to the agent. Each agent implements one function. When the agent finishes processing of a message, the agent dispatcher collects the results and sends them back to the short message driver.

Depending on the implemented services, each server of an agent-dispatching type may implement its own message parsing rules and maintain a command table with function/agent pairs. The current iSMS version has implemented a general-purpose agent dispatcher platform. In this platform, details of communication between the short message driver and the server are hidden from service developers. The developer only needs to specify the agent dispatching rules and implement the agents that carry out the services.

The iSMS platform allows service developers to implement new server types ((3) in Figure 1.11). In this case, the developer needs to implement the interaction between the short message driver and the servers. We provide a communication API that allows the developer to quickly deploy new servers. This communication API is elaborated in Section 1.3.2.

The system also supports group broadcasting and smart message delivery, including the ring tone, music, and icons. The smart messaging protocol defines the format of an ASCII stream that can be passed via different transport protocols. Smart messaging is considered in iSMS because it has
been adopted by major GSM handset suppliers for SMS as well as Infrared among PDA and even for Bluetooth [Blu 04].

If all iSMS applications and the iSMS gateway are implemented in a portable notebook that is not connected to any IP network, then the iSMS system becomes a mobile server, as illustrated in Figure 1.12 (a). In this configuration, mobility management of the iSMS server is automatically maintained by the GSM Mobile Application Part (see Chapter 8). In other words, the existing GSM mechanism will transparently track the moving iSMS server, and iSMS does not need to implement any location tracking mechanism. A mobile iSMS server can be used, for example, in a mobile library application in which the library truck moves around a city. The iSMS server connects to the library database in the truck. An iSMS customer can use the MS to check the status of a book from the library database and the location of the book mobile.

Figure 1.12 (b) shows a different iSMS configuration in which the iSMS systems are connected to different isolated Intranets (e.g., homes), which can query each other through a mobile network.

### 1.3 iSMS Communication Protocols

Two communication protocols have been developed in iSMS. A communication protocol between the MS modem and the short message driver is implemented using the SMS AT Command Set [ETS 98c]. iSMS also implements a Communication API that provides functions to support interaction
between the short message driver and the iSMS servers through TCP connections.

With the above communication protocols, several types of MS (customer) and Agent (service) interaction are implemented in iSMS:

**Query-Response Services:** A query (in SMS format) from an MS is sent to the iSMS server, which in turn invokes corresponding agents and sends back results to the original MS. Examples of query-response services include querying stock (see Section 1.4.1), train schedules (see Section 1.4.3), and UPS package status.

**Relaying Services:** The iSMS server forwards messages and information (such as emails and AOL Instant Messages) to the corresponding mobile users whenever messages are available (see Section 1.4.4). The server maintains the user profiles and actively collects information from different Internet servers.

**Notification Services:** The iSMS server delivers a notification to the user’s MS. Notifications are triggered by events specified by the users. For example, a user may specify information (headline news, stock quotes, and weather information) that he/she wants to receive every day to a scheduling server in iSMS (such as the cron service in UNIX). As another example, an iSMS agent (for example, a stock/auction monitor agent) running on behalf of a user monitors information on the Internet, and delivers alert messages to the user.

**Group Communication Service:** The iSMS server allows specifying groups, each of which contains more than one MS member. Messages to a group (via SMS or from other services) will be forwarded to all members in the group (see Section 1.4.4).

This section shows how iSMS communication protocols are implemented based on GSM. To accommodate iSMS for non-GSM systems, if the systems support AT commands, then we only need to replace the GSM modem with an appropriate wireless modem (for example, a cdmaOne card phone). If the mobile systems do not support AT commands, the short message driver must be modified. With the communication API structure, the iSMS server and all iSMS service applications need not be changed.

### 1.3.1 SMS AT Command Set

The short message driver connects to the MS modem by serial ports or a PCMCIA card. In the current iSMS version, the messages are delivered through serial ports. Two serial ports are used to test the driver in the
16 1 Short Message Service and IP Network Integration

<table>
<thead>
<tr>
<th>Table 1.2</th>
<th>AT commands used in iSMS (a partial list)</th>
</tr>
</thead>
<tbody>
<tr>
<td>AT COMMAND</td>
<td>DESCRIPTION</td>
</tr>
<tr>
<td>+CNMI</td>
<td>New Message Indications to TE</td>
</tr>
<tr>
<td>+CSCA</td>
<td>Service Center Address</td>
</tr>
<tr>
<td>+CMGD</td>
<td>Delete Message</td>
</tr>
<tr>
<td>+CMGS</td>
<td>Send Message</td>
</tr>
<tr>
<td>+CMGL</td>
<td>List Message</td>
</tr>
<tr>
<td>+CSMP</td>
<td>Set Text Mode Parameters</td>
</tr>
<tr>
<td>+CMGR</td>
<td>Read Message</td>
</tr>
<tr>
<td>+CMGW</td>
<td>Write Message to Memory</td>
</tr>
<tr>
<td>+CMSS</td>
<td>Send Message from Storage</td>
</tr>
<tr>
<td>+CSMP</td>
<td>Set Text Mode Parameters</td>
</tr>
<tr>
<td>+CMTI</td>
<td>SMS Message Received Indication</td>
</tr>
<tr>
<td>+CBMI</td>
<td>New CBM (Cell-Broadcast Message) Indication</td>
</tr>
<tr>
<td>+CDSI</td>
<td>New SMS-STATUS-REPORT Indication</td>
</tr>
</tbody>
</table>

ME: Mobile Equipment  
TA: Terminal Adaptor

simulation mode. The NULL port always accepts outgoing short messages successfully. The LOOPBACK port always sends back outgoing messages as incoming short messages. A variable MOBILE_COM_PORT is used in the driver to indicate which port is connected.

The communication protocol between the MS modem and the short message driver is based on the SMS AT Command Set [ETS 98c]. To run this protocol for a specific MS model, one should specify the type of the MS modem. The MS modem setup is achieved by two variables: MOBILE_TYPE and MOBILE_INIT_STRING. Some of the AT commands used in the short message driver are listed in Table 1.2. Every command sent from the short message driver begins with “AT” (for example, “AT+CMGS”). The MS modem’s responses are without “AT” (for example, “+CMGS”).

When the short message driver receives a message from an iSMS server, the driver divides the message into several segments, with length not exceeding 140 octets. For each receiver, the driver generates a set of SMS packets from the message segments. For example, if the message is divided into four segments and there are three receivers, then the drivers will generate 12 SMS packets. The driver pushes these SMS packets into a FIFO queue, and
transmits them sequentially. For every SMS packet, the driver issues the SMS AT command that instructs the MS modem to submit a short message. There are two command modes for the MS: *text mode* and *Packet Data Unit (PDU) mode*. The parameters of AT commands for these two modes are different. In PDU mode, the parameter for sending short message is the entire short message packet. In Table 1.2, the AT commend is `+CMGS`, with the following format (in PDU mode):

\[
\text{AT+CMGS=} <\text{length}> <\text{CR}> <\text{pdu}>
\]

where \(<\text{length}>,\) is the length of the actual data unit in octets. The \(<\text{pdu}>\) is the short message to be delivered. Details of other AT commands can be found in [ETS 98c].

### 1.3.2 iSMS Communication API

The iSMS API was implemented by using VC++. Through this API, servers and agents for specific applications can be easily developed. For every application, there is an iSMS server that communicates with the short message driver through a TCP connection (TCP port number 1122), and several agents may be created to interact with the iSMS server through command execution. The relationships among the short message driver, iSMS servers, and agents are illustrated in Figures 1.10 and 1.11.

In the iSMS API, a class `CsmsdServer` implements the following communication functions between a server and the short message driver:

- **Connect()** sets up a communication link from a server to the short message driver. This function takes two arguments: the IP address of the short message driver and the TCP port number of the driver. This function returns the status of connection establishment.

- **Disconnect()** terminates the TCP link between the server and the short message driver.

- **SetTimeout()** sets a timeout period. When a server issues an operation to the short message driver, a timeout period is set. If the socket to the driver is not ready before the specified timer expires, then the operation fails.

- **Register()** specifies the customers of a server with their GSM MSISDNs. The function takes two arguments: an array of phone numbers and the size of the array.

- **Status()** returns the communication status between the server and the driver, as described here:
1 Short Message Service and IP Network Integration

- **SMCMD_READABLE** indicates that the server is ready for retrieving a short message from the driver.
- **SMCMD_WRITABLE** indicates that the server is ready for sending a message to the driver.
- **SMCMD_ACK** indicates that the message from the server to the driver is successful.
- **SMCMD_CLOSED** indicates that the connection is closed.

**Send()** sends data to one or more customers (that is, GSM MSs).

**Recv()** is invoked by a server for receiving data from a GSM MS.

**RecvACK()** returns acknowledgment from the driver about the status of message transmission.

Besides the **Send** function, the **CsmsdServer** provides two additional functions for facilitating text messages and unstructured binary data delivery:

- **SendText()** is used to send a message with a null-terminated string of ISO-8859-1 characters or traditional Chinese (BIG 5) characters.
- **SendData()** is used to send the unstructured binary data using GSM 8-bit coding.

### 1.3.3 Implementation of an Echo Server

This section illustrates how to implement an echo server using the class **CSmsdServer** described in the previous subsection. This demo server simply echoes the incoming short message. The C program is given in Figure 1.13, and is described as follows:

**Line 1.** A list of MSs in **phone_list** is saved.

**Lines 3–5.** The variables (to be elaborated later) are declared.

**Lines 6 and 7.** The IP address of the short message driver is retrieved, and is stored in the variable **lhost**.

**Lines 8 and 9.** The TCP port number is set, and the connection timeout is set to 3 seconds.

**Lines 10 and 11 (exception case).** If the connection cannot be set up before the timer expires, then exit.

**Lines 12–14 (normal case).** The connection between the server and the short message driver is established, and the server registers the valid
1.3 iSMS Communication Protocols

```c
#include <stdlib.h>
#include <smsio.h>
char *phone_list[]={"+886936000001", "0931000001"};
int main() {
    class CSmsdServer server; octet data[1024],dcs,option;
    char sender[22]; char* da[1]; u_long host;
    int port, ret, length; LPHOSTENT lphost;
    lphost = gethostbyname("localhost");
    if(lphost!=NULL) host=((LPIN_ADDR)lphost->h_addr)->s_addr;
    port = 1122;
    server.SetTimeout(3, 0);
    if (server.Connect(host, port) != INET_SUCCESS)
        { printf("Failure: connect to smsd
"); _exit(1); }
    server.SetTimeout(0, 50);
    if (server.Register(phone_list, 2) != INET_SUCCESS)
        { printf("Failure: register valid users
"); exit(1); }
    while (1) {
        Sleep(1000);
        ret = server.Status();
        if (ret & SMCMD_CLOSED) break;
        if (!(ret & SMCMD_READABLE)) continue;
        ret = server.Recv(sender, data, &length, &dcs, &option);
        if (ret != INET_SUCCESS) break;
        printf("Sender: %s
Message: %s
", sender, (char*)data);
        while (!(server.Status() & SMCMD_WRITABLE)) ;
        da[0] = sender;
        server.SendText(da, 1, (const char*)data);
        while (!(server.Status() & SMCMD_ACK)) ;
        ret = server.RecvACK();
        if (ret == SMCMD_NACK_SENDSM) break;
        printf("\n Sending SMS....successful \n");
    }
    return 0;
}
```

Figure 1.13  A Simple Echo Server Program

MSs specified in phone_list. The registration procedure should be complete in 50 ms, or it is considered a failure.

For successful registration, the server enters a loop (Lines 15–29).

**Lines 16–20.** In this loop, the server waits to receive an incoming short message for 1,000 seconds (Line 16) and then it reads the short message at Line 20.

**Lines 21–25.** The server waits until it is allowed to transmit (Line 23). Then it returns the message it just received to the MS (Line 25).
20 1 Short Message Service and IP Network Integration

**Lines 26–29.** If the echo message has been forwarded to the MS, the short message driver acknowledges this operation (Lines 26 and 27).

The whole process repeats until one of the following conditions is met: the connection is closed (Line 18), the server fails to read the text (Line 21), or the server receives a negative acknowledgment (Line 28).

### 1.4 Examples of Services

This section describes several services implemented in iSMS. Special commands are defined for these services. An online menu can be built in the MS so that a customer can check the menu to figure out the meaning of a special command. Note that we have also built a platform to test the implemented services without actually involving the GSM network (the platform simulates the interaction of the user MS and the iSMS wireless modem).

#### 1.4.1 Accessing the Web from GSM MSs

iSMS supports a service that allows users to surf the web using standard GSM MSs. The web accessing architecture is illustrated in Figure 1.14. In this architecture, a customer (that is, a GSM MS holder) sends out a web query in SMS format to a proxy server connected to the iSMS server. The proxy server and the iSMS server may be running on the same host or on different hosts. Based on the customer’s preference, the proxy maps the input query to a proper Hypertext Transfer Protocol (HTTP) call and forwards the call to the corresponding web server. The proxy then filters and converts the data returned from the web server to SMS format, and returns the SMS

![Figure 1.14 iSMS Web Accessing Architecture](image-url)
message to the original caller. This implementation has several interesting characteristics:

- Standard GSM MSs are used as terminals for surfing the web. No special devices or software are needed.

- The communication medium is SMS. Both queries sent from the MSs (the GSM customers) and data received by the MSs are in SMS format. A single Uniform Resource Locator (URL) access is performed by two SMS messages: one from the MS to the proxy containing the URL query and another one from the proxy to the MS including the resulting data. The “instant transmission” property of SMS allows the customer to send an instant web query without long “machine” setup time experienced in the dial-up service to PC.

- Functionality of mapping SMS queries to HTTP calls and converting the data in HTML format to that in SMS format is implemented in the proxy. This application allows customers to tailor the call mapping and data conversion functions. The proxy picks the proper mapping and filter functions based on the customer’s profile. The customers may also utilize the default mapping and conversion functions supported by the proxy.

iSMS implements functions for querying Internet information, such as stock quotes, currency exchange rates, and delivery status of FedEx packages:

**Stock Quotes**

The stock quote query command is of the form

\[
\text{QUO} \{\text{symbol1}\} \{\text{symbol2}\} \ldots
\]

The QUO command is explained with the following example:

\[
\text{QUO} \ http://investor.msn.com/quotes/quotes.asp?Symbol=$1\ /bin/quotefilter
\]

The first field, QUO, is the keyword for query. The second field, \{symbol1\}, specifies HTTP call mapping. The third field, \{symbol2\}, defines the filter/conversion function. If \{symbol1\} and \{symbol2\} are not specified, the default call mapping and conversion functions are used.

With the above setup, a customer can enjoy web querying through his/her GSM MS. Note that the customer can turn off and then turn on
1 Short Message Service and IP Network Integration

the MS and the setup is still valid. Suppose that the customer has a GSM MSISDN +886936105401, and the MS modem in the iSMS gateway has the phone number +886-931-144-401. The customer sends out an SMS message to +886-931-144-401 as follows:

QUO T

This command queries AT&T stock. When the iSMS server (running on the phone number +886-931-144-401) receives the message, it forwards the message to the proxy server, which in turn converts the SMS query into an HTTP call as follows:


The proxy sends the HTTP call to the destination web site. When receiving data from the web site, the proxy invokes a filter function (i.e., /bin/quotefilter) that re-formats the data received from the web site. The proxy returns the output of the filter function to the customer’s MS 0936-105-401. The SMS data looks like

T Last 85 7/8
Change +1 9/16 (+1.85%) Volume 7.708 M

where the first line indicates the last price of AT&T stock, the second line shows the price change since the last closing, and the third line gives the transaction amount.

Currency Exchange Rates

Consider a currency exchange rate query as another example. The currency rate query command issued by the customer is of the form

CUR from to

The system returns the currency exchange rate between the currency of the country from and the currency of the country to. For example, the short message

CUR USD TWD

queries the currency exchange rate between, the American dollar and the Taiwan dollar.
1.4.2 Handset Music Service

Existing MSs can store several music tones in memory. An MS may play various alert tones for incoming calls sent from different caller IDs (either mobile telephones or fixed telephones). Teenage MS holders are particularly interested in having fancy music tones such as the latest pop songs. Nokia, for example, has developed a messaging protocol for this purpose. Based on SMS, the protocol, called *Smart Messaging* [Nok 97], specifies a set of pre-defined message headers for sending music tones, business cards, etc. Each smart message contains a header for the message type. Nokia MSs recognize the header and perform different actions on the message body. Two issues regarding smart messaging deserve attention:

- Each smart message contains a Non-ASCII, binary header, and it is not trivial to input a smart message from regular MSs. It may not be convenient to send a music tone from one MS to another MS.
- Only Nokia MSs understand how to interpret short messages in smart messaging format.

To address these issues, iSMS developed a *Simple Tone Language (STL)* to represent the music notes that are translated into smart message format by the iSMS music agent. More precisely, a GSM customer inputs a short message as a music tone request, containing a music agent name, a receiver of the music tone, and a music tone body in STL. The message is sent to the iSMS server. Upon receipt of the request, the music agent encodes the music notes into a short message and sends it to the receiver. The handset can then store and play the received music tone. The regular expressions [Hop 74] for the STL grammar are described as follows. A STL music tone is expressed as

\[
tone = [style] [tempo] [volume] [repeat] (note-expression)+
\]

The components of *tone* are elaborated on as follows:

- The *style* value ranges from S00 to S02. In Nokia’s smart messaging specification, S00 represents natural style (rest between notes), S01 represents continuous style (no rest between notes), and S02 represents staccato style (shorter notes and a longer rest period). The default value of *style* is S00.
1 Short Message Service and IP Network Integration

Table 1.3 Lengths of 1/4 note

<table>
<thead>
<tr>
<th>SYMBOL</th>
<th>LENGTH OF 1/4 NOTE</th>
<th>SYMBOL</th>
<th>LENGTH OF 1/4 NOTE</th>
</tr>
</thead>
<tbody>
<tr>
<td>T00</td>
<td>2.40 sec.</td>
<td>T08</td>
<td>0.95 sec.</td>
</tr>
<tr>
<td>T01</td>
<td>2.14 sec.</td>
<td>T09</td>
<td>0.85 sec.</td>
</tr>
<tr>
<td>T02</td>
<td>1.90 sec.</td>
<td>T10</td>
<td>0.76 sec.</td>
</tr>
<tr>
<td>T03</td>
<td>1.70 sec.</td>
<td>T11</td>
<td>0.67 sec.</td>
</tr>
<tr>
<td>T04</td>
<td>1.51 sec.</td>
<td>T12</td>
<td>0.60 sec.</td>
</tr>
<tr>
<td>T05</td>
<td>1.35 sec.</td>
<td>T13</td>
<td>0.54 sec.</td>
</tr>
<tr>
<td>T06</td>
<td>1.20 sec.</td>
<td>T14</td>
<td>0.48 sec.</td>
</tr>
<tr>
<td>T07</td>
<td>1.07 sec.</td>
<td>T15</td>
<td>0.43 sec.</td>
</tr>
</tbody>
</table>

- The tempo value ranges from T00 to T31, which represent the lengths of 1/4 note, as given in Table 1.3. The default value is T08.
- The volume value ranges from V00 to V15. The default value is V07. Although the smart messaging specification defines the programmable volume, some handsets, such as Nokia 6150, do not support this feature. In this case, the music volume is adjusted by the handset user, not the message received by the handset.
- The repeat value ranges from R00 to R15, where R00 means repeating infinite times, R01 means repeating the tone for one time, and so on. The default value is R01.
- The note-expression component is of the format

  \[
  \text{note-expression} = \text{note} \ [\text{scale}] [\text{duration}] [\text{duration-specifier}]
  \]

  where a music note is expressed as

  \[
  \text{note} = 0 | 1 | 1# | 2 | 2# | 3 | 4 | 4# | 5 | 5# | 6
  | 6# | 7
  \]

  The scale component of the corresponding music note is expressed as

  \[
  \text{scale} = "L" \ ; \ \text{note}-1 \text{ is 440 Hz}
  \]

  \[
  "M" \ ; \ \text{note}-1 \text{ is 880 Hz}
  \]

  \[
  "N" \ ; \ \text{note}-1 \text{ is 1760 Hz}
  \]

  \[
  "O" \ ; \ \text{note}-1 \text{ is 3520 Hz}
  \]

  where the default value is M. The duration of the corresponding music note is expressed as
1.4 Examples of Services

Notes:

STL: 

\[
\begin{align*}
  & A \; ; \text{full-note} \\
  & B \; ; \text{reserved} \\
  & C \; ; \text{1/2-note} \\
  & D \; ; \text{1/4-note} \\
  & E \; ; \text{1/8-note} \\
  & F \; ; \text{1/16-note} \\
  & G \; ; \text{1/32-note}
\end{align*}
\]

where the default value is D. The duration specifier of the corresponding music note is expressed as

\[
\begin{align*}
  & X \; ; \text{dotted note} \\
  & Y \; ; \text{double dotted note} \\
  & Z \; \text{2/3 length}
\end{align*}
\]

Figure 1.15 illustrates the STL representation of a Taiwanese Song.

\begin{align*}
\text{duration} &= "A" \; ; \text{full-note} \\
&| "B" \; ; \text{reserved} \\
&| "C" \; ; \text{1/2-note} \\
&| "D" \; ; \text{1/4-note} \\
&| "E" \; ; \text{1/8-note} \\
&| "F" \; ; \text{1/16-note} \\
&| "G" \; ; \text{1/32-note}
\end{align*}

1.4.3 Train Schedule System

Consider the iSMS train service that provides train schedule query information and ticket reservations. This train schedule server consists of an agent dispatcher and two agents:

- The query agent allows the customers to query particular train schedules.
- The reservation agent allows the customers to reserve train tickets.

At system initialization, the train schedule server registers the customers to the short message driver. This server should allow adding/deleting customers during normal operation. A customer may send a train schedule query (a short message) to the iSMS gateway. Based on the caller ID (the MSISDN of the customer who issues this query), the short message driver forwards
the message to the train schedule server. In iSMS, the train schedule query command is of the form

\[ \text{tra} \{\text{FromStation}\} \{\text{ToStation}\} \{\text{options}\} \]

where \{FromStation\} and \{ToStation\} represent the departure station and arrival station, respectively. The field \{options\} is of the format

\[ \{\text{option}\}: \{\text{Time1}\} - \{\text{Time2}\} (S|A) (F|E) \]

where \{Time1\} - \{Time2\} represents the time range (default = CurrentTime-2300), S and A represent the time range for the departure or the arrival time range (the default value is S), and F and E represent Numbered Express (default) and Express, respectively.

From the header of the short message, tra, the agent dispatcher invokes the train schedule query agent using the short message content

Taipei - Taichung, 2:00pm - 3:00pm

as the parameters. In this query, the customer would like to know the train numbers for the trains from Taipei to Taichung, which depart between 2:00 P.M. and 3:00 P.M. After execution of the request, the agent returns the train numbers to the agent dispatcher. This result is formatted as a short message and is sent back to the original customer. Upon receipt of the result, the customer may issue another short message to reserve tickets for specific train numbers. In this case, the ticket reservation agent will be invoked to handle the request.

1.4.4 Other iSMS Services

In addition to the services described in the previous subsections, other examples of services running on the current version of iSMS include the following:

**Online Help Service:** An iSMS user types the \textit{h} command to query the services available in the iSMS system.

**Personal Profile:** iSMS maintains a personal profile repository for individual registered users. Profiles are organized on a per-user basis (according to the phone numbers). Each entry has the format “\textit{key-word=\textit{value}}”, where \textit{value} can be a phone number, an address, a personal note, and so on. A mobile user can add an entry to his/her personal profile by sending a short message to iSMS. For example, a mobile user may send the following short message to iSMS

PB Robin +19179075010
This message instructs iSMS to add a new entry

Robin = +19179075010

to his/her profile. Then the user can query the entry with a keyword by sending the following message to iSMS

PQ Robin

In response, iSMS returns the following message

Robin = +19179075010

A web interface has been provided, and users may update their profile using regular browsers as well. This repository serves not only as a mobile user’s repository on the network, but also as the user profile for other services. For example, email services retrieve the user profiles to locate senders’ email addresses by names.

**Broadcast Message Service:** An iSMS user can broadcast a message to several destinations with the following command:

```
bc {receivers} {message}
```

In this command, `{receivers}` is a list of MSISDNs of the receivers and `{message}` is the message to be delivered. In a similar group communication service, the MS can broadcast a message in a group of phone numbers (with command `b`), create a new group (with command `bn`), add/delete phone numbers to/from a specific group (with commands `ba` and `bd`, respectively), delete a specific group (with command `bk`), and list all phone numbers in a specific group (with command `bl`).

**Smart Messaging Services:** A GSM user uses the following commands to send smart messages. The command

```
rt {receivers} {song}
```

sends a ringtone `{song}` to the receivers in the list `{receivers}.` The command

```
lg {receivers} {image name}
```

sends an operator logo `{image name}` to the receivers. The command

```
ic {receivers} {image name}
```

is used to send a group logo.
Group Communications: iSMS implements a group communication mechanism. A group is identified by a unique name and contains a set of phone numbers. A message sent to a group will be forwarded to all members in the group. iSMS supports the following group communication commands:

- Creating a group with founding members
- Querying members in a named group
- Adding/deleting members to/from an existing group
- Sending messages to a named group

This group mechanism can also be used by other services. For example, a multi-user game based on SMS may communicate with its mobile players using this group communication mechanism.

Mobile Dictionary Service: The command

```
dic {English word}
```

returns the Chinese meaning of the English word `{English word}`.

Email Service: iSMS implements an agent to SMTP/POP3 servers for relaying email and SMS. A short message from mobile users to iSMS with the format

```
ema {email-address} {message}
```

is transferred to an email and sent to a SMTP server for delivery. Conversely, the agent consults the mail server via the POP3 interface on behalf of a mobile user periodically, and delivers notifications of new messages to a user’s MS via SMS. The user reads an email just as if he/she were reading a short message.

Forwarding Service: An iSMS user can execute a command `{command}` and then forward the results to another user `{receiver}`. The command is

```
fwd {receiver} {command}
```

These services can be combined to yield more powerful services. For example, a boy may order a song and request to forward the song to play on his girlfriend’s MS on her birthday. This action combines the handset music service, forwarding service, and event scheduling service. These services are powerful building blocks when iSMS integrates with applications offered by content providers. Examples of these applications include
mobile banking services, mobile trade services, credit card information, life insurance account information, airline/travel/concert ticket reservations, news/information, entertainment, and so on.

1.5 Caching for iSMS-Based Wireless Data Access

This section uses a business card application as an example to illustrate how cache-based wireless data access can be implemented in an SMS-IP platform such as iSMS. To converge wireless data with the Internet, the wireless application protocols may integrate a lightweight web browser into wireless terminals with limited computing and memory capacities. The wireless application protocols implemented in the wireless application gateway (for example, the iSMS gateway) and the wireless terminal enable a mobile user to access Internet web applications through a client/server model. A client application running on the wireless terminal may repeatedly access a data object received from the application server. To speed up wireless data access, cache mechanisms have been proposed. For example, the Wireless Application Protocol (WAP) [OMA04] user agent caching model tailors the HTTP caching model to support wireless terminals with limited functions. In this model, a cache in the wireless terminal is used to buffer frequently used data objects sent from the WAP gateway. When the data objects in the application server are modified, the cached objects in the wireless terminals are obsolete. To guarantee that the data presented to the user at the terminal is the same as that in the application server, a strongly consistent data access protocol [Yin 99] must be exercised. Furthermore, the cache size in a wireless terminal is typically small, and a cache replacement policy is required to accommodate appropriate data objects in the cache. We use the iSMS business card service as an application for wireless data access study. The business card service is a generalization of the phone book feature in mobile handsets, which is one of the most popular handset features. Unlike the phone book, the business cards are stored and maintained in a business card database in the network (that is, iSMS application server), and the most frequently used business cards are cached in the wireless terminal. This application offers four major advantages over the phone book feature in mobile handsets:

- When a user changes the handset, the phone book may not be conveniently transferred to the new handset. This is particularly true when the old handset is broken and the phone book is lost. With the iSMS
business card service, the user can access his/her “private” phone book from any handset.

Besides the private phone book, the business card service can also maintain a “public” phone book database (just like the yellow pages) in the application server.

After a phone conversation, the business card service allows the call parties to exchange their business cards, which provide much more information than the phone numbers in the phone book feature.

When the information (for example, phone number) of a user changes, it is not automatically reflected in the phone books of other users. With the business card service, a user can update the business card in the database of the business card application server (typically, through the Internet), and other users always access the correct information.

In the iSMS-based business card application, the format of the business card follows the vCard standard as illustrated in Table 1.4 [Daw 98]. In this format, the \texttt{FN} field is used to specify the vCard object. The \texttt{N} field is a single structured text value, which corresponds, in sequence, to the Family Name, Given Name, Additional Names, Honorific Prefixes, and Honorific Suffixes. A person may have several telephone numbers; for example, work phone number, fax number, and cellular telephone number. In vCard, these numbers are included in the \texttt{TEL} field. We also include a \texttt{CALENDAR} field that allows the user to fill in the schedule of public events he/she wants to share with others. In our iSMS implementation, the size of a vCard can be unlimited in the iSMS application server. However, the vCard is tailored to

\begin{table}[h]
\centering
\begin{tabular}{|c|c|c|}
\hline
\textbf{FIELD} & \textbf{DESCRIPTION} & \textbf{LENGTH} \\
\hline
\texttt{VERSION} & Version of vCard & 13 bytes \\
\hline
\texttt{FN} & vCard object name & 30 bytes \\
\hline
\texttt{N} & Name information & 40 bytes \\
\hline
\texttt{ORG} & Organization information & 50 bytes \\
\hline
\texttt{TIT} & Job title & 50 bytes \\
\hline
\texttt{ADR} & Address & 120 bytes \\
\hline
\texttt{TEL} & Phone number & 130 bytes \\
\hline
\texttt{EMAIL} & Email address & 50 bytes \\
\hline
\texttt{URL} & Uniform resource locator & 50 bytes \\
\hline
\texttt{CALENDAR}\textsuperscript{*} & Public calendar event & 100 bytes \\
\hline
\end{tabular}
\caption{The vCard format in the iSMS-based business card service}
\end{table}

\textsuperscript{*}The \texttt{CALENDAR} field is not defined in the vCard standard.
be of fixed size when it is delivered to the wireless terminal. The length of each field in our implementation is also shown in Table 1.4. The appearance of an iSMS business card in a PDA is illustrated in Figure 1.16. An iSMS business card can be updated, added to, or removed from the database by the application server or by the wireless terminal.

We consider two strongly consistent algorithms for wireless data access: Poll-Each-Read (PER) and Call-Back (CB) [Yin 99, Sat 90]. In both algorithms, when a data object (for example, a business card) is updated at the iSMS application server, the valid object is not immediately sent to update the cached copies in the wireless terminal. Instead, the valid object is delivered to a wireless terminal only when a data access operation is actually performed.

In Poll-Each-Read [How 88], at a data access request (see (b) and (e) in Figure 1.17), the wireless terminal always asks the iSMS application server to check whether the cached object is valid. If so, the iSMS application server responds affirmatively (Figure 1.17 (f)) and the user accesses the object.
32 1 Short Message Service and IP Network Integration

Figure 1.17  Data Access in Poll-Each-Read

in the cache of the wireless terminal (Figure 1.17 (g)). If the data object is updated before the access (Figure 1.17 (a) and (b)), the iSMS application server sends the current data object to the wireless terminal (see Figure 1.17 (c)), and this object is stored in the cache of the wireless terminal (see Figure 1.17 (d)). In this approach, when an object \( O_i \) is found in the cache, the wireless terminal still needs to obtain \( O_i \) from the iSMS application server if \( O_i \) has been invalidated. Thus, a cache hit may not be beneficial to PER. Define effective hit ratio as the probability that for an access to object \( O_i \), a cache hit occurs in the wireless terminal and the cached object is valid.

In PER, the cost for accessing an object with an effective cache hit is a cache affirmative request and acknowledgment exchange between the iSMS application server and the wireless terminal (msg3 and msg4 in Figure 1.17). For a cache miss or a cache hit in which the data object is invalidated, the access cost is a request message sent from the wireless terminal (msg1 in Figure 1.17) and a data object transmission from the iSMS application server to the wireless terminal (msg2 in Figure 1.17).

In the Call-Back approach [How 88, Nel 88], whenever a data object is modified, the iSMS application server sends a message to invalidate the corresponding cached object in the wireless terminals (see (a) in Figure 1.18). The cache storage of the invalidated object is reclaimed to accommodate other data objects, and the wireless terminal sends an acknowledgment to inform the application server that the invalidation is successful (see (b) in Figure 1.18). During the period between points (a) and (d) in Figure 1.18, if other updates to this object occur, no invalidation message needs to be sent to the wireless terminal (because no invalidated copy will be found in the cache). In this approach, all objects stored in the cache are valid, and a cache hit is always an effective hit (if the message transmission delay is ignored). In a data access, if a cache hit occurs, the cached object is used without any communication between the application server and the wireless
Data Access in Call-Back

For a cache miss, the access cost is a request message (msg3 in Figure 1.18) sent from the wireless terminal and a data object transmission (msg4 in Figure 1.18) from the application server to the wireless terminal. It is required to invalidate a cached object at the wireless terminal (if it exists) when the object in the application server is updated (msg1 and msg2 in Figure 1.18).

The typical cache size in a wireless terminal is not large. When the cache is full, some cached objects must be removed to accommodate new objects. We consider the Least Recently Used (LRU) replacement policy. This policy is often utilized to manage cache memory in computer architecture [Sto 93], virtual memory in operating systems [Sil 01], and location tracking in mobile phone networks [Lin 94]. LRU uses the recent past as an approximation of the near future, and replaces the cached object that has not been used for the longest period of time. LRU associates with each cached object the time of its last use. When a cached object must be replaced, LRU chooses the object that has not been used for the longest period of time. Therefore, every object in the cache has a rank. If a cached object has the rank 1, it means that the object is most recently used. If an object \( O_i \) has a rank \( k \), it means that \( k - 1 \) objects are more recently used than \( O_i \) is. If \( k > K \) where \( K \) is the size of the cache, then \( O_i \) has been removed from the cache.

The number of data objects in the application server is much larger than the cache size in the wireless terminal. However, our experience in exercising wireless applications [Rao 03] indicates that for an observed period, only a small number \( N \) of data objects in the iSMS application server are potentially accessed by a wireless terminal. Although the objects to be accessed vary from time to time, the number \( N \) is not significantly larger than the cache size of the wireless terminal. That is, the data access pattern of a wireless terminal exhibits temporal locality [Sto 93], which is the tendency for a wireless terminal to access in the near future those data objects referenced...
in the recent past. Temporal locality may not be observed in wireline Internet access because desktop users typically navigate through several web sites at the same time. Conversely, the restricted user interface of wireless terminals only allows a user to access a small region of data objects for instant information acquisition. Thus, caching can effectively reduce the data access time for the wireless terminals. The cache performance for the iSMS-based wireless terminals was investigated in [Lin03c].

1.6 Concluding Remarks

This chapter first introduced a SM-SC SMS-IP integration system called NCTU-SMS. Then we described iSMS, a non-SM-SC-based platform that integrates IP networks with the short message mechanism of mobile networks. This platform was developed by AT&T and FarEasTone. iSMS supports middleware for creating and hosting wireless data services based on SMS. The iSMS hardware architecture can be easily established with standard MSs and personal computers. We have developed agent-based middleware with an API, which results in a lightweight solution that allows quick deployment of added-value data services. Compared with the solutions that integrate IP networks through SM-SC, iSMS has the following advantages:

**Easy installation:** The iSMS hardware components are a standard GSM MS and personal computer. The iSMS software can be installed on the PC without special treatment.

**Generic platform with personalization:** Several applications have been developed for iSMS, including email delivery/forwarding, web access (for example, a stock query or a train schedule query) and handset music service. iSMS users can easily develop new services and tailor the existing services to fit their needs.

**Transparency:** The iSMS services are transparently run on GSM networks without any maintenance effort by GSM operators.

**Immediacy:** Unlike the dial-up service to a PC, iSMS offers instant information exchange between the MS and the servers. This property is desirable for a customer to enable IP computing when he/she is moving.

For a GSM operator, iSMS is targeted for small companies that subscribe to the *Closed User Group (CUG)* service [Lin 01b]. In each of the small
companies, a PC-based iSMS gateway is installed. The iSMS is designed for two types of services:

- **For standard Internet services**, an iSMS gateway serves as an access point for the GSM user to receive existing Internet services (such as stock quotes or Yahoo! surfing).
- **Customized services** (for example, private mails) are tailored for each iSMS gateway. These types of services are exclusive to a closed user group of an iSMS gateway, and cannot be accessed by users outside the gateway (although different CUGs may have the same customized services).

For both types of services, the iSMS gateways do not communicate with each other. Thus, there is no need to manage or administer the distributed iSMS gateways. The iSMS gateway approach has the following advantages:

- **Scalability**: One can easily accommodate new iSMS customers by installing PC-based iSMS gateways at their company sites. Since the radio paths for these iSMS gateways are different, adding new customers does not increase traffic to a specific radio link. It is clear that iSMS is scalable for standard Internet services. To accommodate more users for Internet services, one only needs to add more iSMS gateways. For other approaches such as WAP, all accesses to the Internet should go through a centralized WAP gateway, which may require great maintenance, and the gateway may become a bottleneck. In iSMS, accesses to the Internet are performed through thousands of routes using independent iSMS gateways.

- **Reliability**: When an iSMS gateway at a company fails, one only needs to fix this failed gateway. This failure does not affect other iSMS gateways. To enhance the reliability of iSMS service in a company, one can have duplicated configuration on an iSMS gateway site. In this configuration, an MS (the customer) may register to an operational iSMS gateway and a standby iSMS gateway. This setup is performed at gateway initialization (when the customer subscribes to iSMS). For a standard GSM MS, the phone numbers of both gateways are stored in the phone book of the MS. The MS first tries the operational iSMS gateway for service. If it does not work, then the MS tries the standby iSMS gateway. Message resending can be automatically performed by the GSM divert service. If the GSM MS is equipped with the SIM ToolKit feature, then when the operational gateway fails, the standby gateway
will instruct the MS to switch the gateway over the air (by modifying the gateway phone number stored in the MS). When the operational gateway is recovered, the gateway phone number in the MS is switched back by the operational gateway over the air. Gateway restoration is transparent to users.

**Performance:** Since the customer traffic is distributed to individual iSMS gateways, good performance (for example, short latency time) can be expected. However, the end-to-end short message transmission delay for iSMS is longer than that for NCTU-SMS.

Due to limited SMS bandwidth, most of the iSMS applications are transaction oriented. When high-bandwidth mobile data infrastructures (such as GPRS and UMTS) are available, the data rate of iSMS can be significantly increased, and development of general data services will be essential. The iSMS philosophy has been generalized for iMobile [Rao 01b], a user-friendly environment for mobile Internet. The iMobile platform is described in Chapter 17. Additionally, the reader is referred to Chapter 12 in [Lin 01b] for more details about SMS.

### 1.7 Questions

1. Describe the procedure of SMS delivery from an MS to the other MS.

2. Describe the NCTU-SMS architecture. What is the NCTU-SMS delivery state tree?

3. Describe the SMS architecture. Does delivery of SMS affects voice call setup? Does it affect voice conversation?

4. Compare the two SMS-IP integration approaches: NCTU-SMS (an approach involving SM-SC) and iSMS (an approach that does not involve SM-SC). Which approach will experience longer transmission delay for end-to-end short message delivery?

5. Describe the three service types defined in iSMS. Can you define a new service type for iSMS?

6. In accordance with Section 1.2, design a new user-defined server and an agent.

7. Describe four services for MS-agent integration in iSMS.
8. Section 1.3.2 defines functions in the iSMS Communication API. Do these functions suffice to support all SMS-IP services? Can you add more functions?

9. iSMS supports the `SendText()` and `SendData()` functions. Give an example to show when `SendData()` is used.

10. Design an iSMS GUI for a PDA that allows adding new iSMS applications without modifying the PDA.

11. What is a strongly consistent data access protocol?

12. Describe the the Poll-Each-Read and the Call-Back approaches. Is one approach always better than the other?

13. Evaluate iSMS in terms of scalability, reliability, and performance.