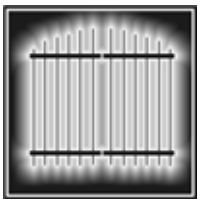
Crossroads
The ACM Student Magazine



Experimental Mobile Gateways

by ***Premshree Pillai***

Introduction

ShortMessaging Service (SMS) services are becoming near-ubiquitous. If you have ever used your mobile device to check your email, the latest sports scores, or check the news, you have used a Short Messaging Service (SMS). This paper takes you through the inner-workings of these services, finally demonstrating how to SMS-enable your weblog.

In this paper, a low-level implementation will be described to demonstrate how amobile-to-Web gateway works. The example network uses the Web, butcarriers and service providers may use their own networks.

What Is a Gateway?

One very popular mobile service available throughmost carriers, is Yahoo's 8243 service [**8**]. This service allows you tocheck your Yahoo! Mail, use Yahoo! Instant Messenger, or check scores,news, and so on. When you access this type of service, you are using both thecarrier's network and the service provider's network, in this caseYahoo!. There are also many carrier-dependent services, in which caseyou make use of the carrier's network alone.

To provide you with these services, these networks implement something known as a *gateway*.A gateway is nothing but, as the word suggests, an entrance to thenetwork. In the case of mobile-based services, these gateways arecalled mobile-to-network gateways, or simply *mobile gateways*.

Typically, a service has many users. The gateway, depicted as a thick line in the service network in **Figure 1**, performs the following tasks:

- Implements a queuing mechanism to service each incoming request.
- Determines whether the service requested is acceptable.
- Sends a response granting the service.

Thus, a gateway acts as a filter between the service network to the service requester's (user's) network.

The sections that follow provide an overview of an SMS-to-Web gatewayand an MMS-to-Web gateway. For simplicity's sake, the examples assumeonly one user. This leaves the task of implementing mobile gatewaysthat service incoming mobile requests, in the form of SMSes and MMSes,on a continuous basis.

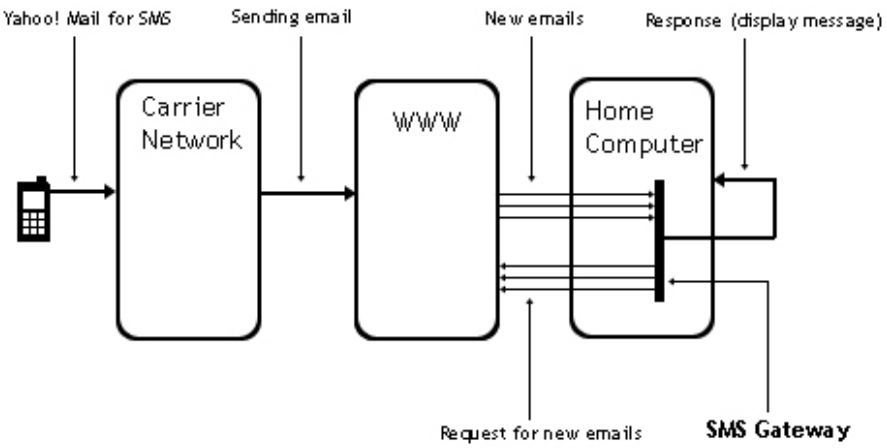


Figure 1: An Experimental Gateway Configuration.

The SMS-to-Web Gateway

At a minimum, the following are required in order to send a service request to a gateway:

1. An incoming service request from a mobile (in this case in the form of SMSes).
2. A number to send the service request.
3. A service network, typically.

To satisfy the first of the above requirements, you may use anycell-phone (note that the network type, whether GSM or CDMA, determinesavailability of service numbers).

The main task is to provide your own service, using your own gateway.Given that you do not have your own service number or your own network,you will need to use an intermediary service such as Yahoo! Mail for SMS [**8**] as a way to send requests. For your network, you will simply use the Web.

Before you implement your gateway, you will need to decide what serviceyou want to offer. This example uses a simple model: a user sends SMSeswhich are displayed as plain text on the sender's home computer.

The user will send messages using a service like Yahoo! Mail for SMS.This service is offered by Yahoo! to select carriers (for example, Hutch andOrange). Users are able to send emails from their Yahoo! mailaccount using SMS. Here is how to send an email using Yahoo! Mail:

Logging in to your account is done by SMSing the following to 8243 (this is Yahoo!'s service number, more commonly called the Yahoo! Gateway):

```
in username password
```

Sending an email is achieved similarly, by SMSing the following to 8243:

```
to recipient@host.com mail_body
```

In this case, there is a single user, so the recipient email address will be one of the user's email addresses.

Because the Web is your intermediate network, the user's home machineshould be connected to the internet at all times. Your gateway, which issoftware, will be implemented on this machine.

The gateway is like a daemon: whenever a new request comes in, thegateway services it. More

specifically, the gateway is an endless loop that sleeps in between events for a pre-defined time,meaning it is not a tight infinite loop. In this case, the service issimply displaying the SMS.

A more detailed look at how the gateway grants service requests follows:

- The user sends an SMS to one of his own email addresses. The emailaddress could be a Web mail account or a POP3 account, which is evenbetter.
- The gateway is running in the background. It uses the Internet toaccess the user's mail account to check for new emails. New emails arefiltered to get only those emails that have been sent via SMS. Thesimplest way to do is to check for the subject, which is typically setto '[none]' when using Yahoo! Mail for SMS.
- The gateway parses the raw email source, and extracts only the body of the email, simply displaying it in the console.
- After one round of processing, the gateway "sleeps" for a while.
- After the sleep period is over, it springs back into action.

It is obvious that our program has to poll for incoming messages (POP3 mailboxes, etc.) continuously. The simplest way to do this is to run the main function within an infinite `while` loop (with some "sleep" period). However, using an infinite while loop is not a very good idea. Instead, when actually using the script, the main function should be run only once in the script; and the script must be scheduled to run at regular intervals using a scheduler like cron.

Using a scheduler makes more sense. If you use a while loop instead, and if for some reason an error occurs, the script would halt. Thus, your gateway will no longer work until you re-run it. Errors in scripts are very common. To avoid such problems, it is best to use a scheduler. Even if an error occurs at a particular run, the script will be scheduled to run again.

A typical instruction within a crontab file would be as follows:

```
* * * * * /usr/home/premshree/gateway.rb
```

Now, an SMS gateway has thus been implemented. The gateway fits into the scheme of things as shown in **Figure 2**.

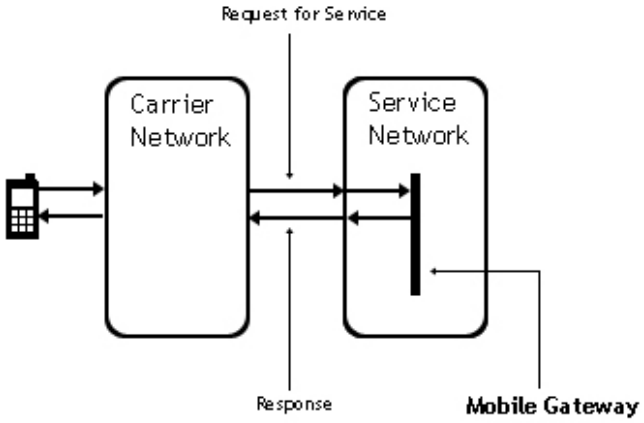


Figure 2: A Typical Gateway.

The following pseudocode shows the overall logic flow for building the gateway:

```
/* SMS Gateway pseudocode */use poplib;// Config variablesPOP3_HOST = "pop3.host.
com";POP3_USER = "xyz";POP3_PASS = "*****";// Function to get
mailsfunction getReqestsFromWeb() {     // connect to POP3 server       poplib = new
poplib();       poplib.setHost(POP3_HOST);      poplib.setUser(POP3_USER);
```

```
poplib.setPass(POP3_PASS);      // retrieve mails       mails = poplib.getMails
();     // process mails        for mail in mails {             mail_body = parse
(mail).getBody();               display(mail_body);     }}function display
(what) {        console.print(what);}getRequestsFromWeb();
```

The MMS-to-Web Gateway

Just as the SMS gateway was implemented, an MMS(Multimedia Messaging Service) gateway can be implemented. The difference between an SMS and an MMS is that SMS messages can only contain text/plain messages, while emails sentusing MMSes have textual as well as multimedia data (typically images).Thus, MMS messages are "multipart" messages—the MIME [2] type part of theemail could be text/plain, and the other part could be an image/jpeg.

So the only difference between an SMS gateway and an MMS gateway is inhow each handles the various MIME extensions of the email.

Note: In an SMS message, the MIME type can be text/plain only, so you do not have to explicitly handle such email messages.

The MMS gateway is identical to the scheme shown above for the SMS gateway.

The following pseudocode shows how to implement the MMS gateway:

```
/* MMS Gateway pseudocode */use poplib;use EmailParser;// Config variablesPOP3_HOST =
"pop3.host.com";POP3_USER = "xyz";POP3_PASS = "******";// Function to get
mailsfunction getReqestsFromWeb() {     // connect to POP3 server       poplib = new
poplib();       poplib.setHost(POP3_HOST);      poplib.setUser(POP3_USER);
poplib.setPass(POP3_PASS);      // retrieve mails       mails = poplib.getMails
();     // process mails        for mail in mails {             mail_body = parse
(mail).getBody();               mail_parts = EmailParser.
getEmailByMIMETypes(mail_body);         result = "";            for mail_part in
mail_parts {                    if (typeOf(mail_part) == "text/
plain") {                               result = result +
process_text(mail_part);                        }                       if
(typeOf(mail_part) == "image/jpeg") {                           result = result
+ process_image(mail_part);                    }               }
display(result);        }}function process_text(text) { return text;}function
process_image(image) {  BINMODE;        return image;}function display(what)
{       console.print(what);}getRequestsFromWeb();
```

SMS-to-LiveJournal: An SMS-to-Web Gateway

LiveJournal [3] is a very popular blogging website that runs on opensource software. A practically useful example of a mobile gateway isSMS-to-LiveJournal gateway, simply called SMS2LJ [5].The source, written in Python, is available under the terms of theGNU Public License (GPL) [1].

This tool is similarto the SMS-to-Web gateway that was discussed earlier. Where it differs is inthe service provided: using this tool, a user can send an SMS, which thenis processed by the gateway, but instead of simply displaying the message onthe computer, the message is posted to your LiveJournal account. This allowsyou to post to your LiveJournal account whenever you are on the move.

Instead of using the `display()` function, you use the`postToLiveJournal()` function:

```
function postToLiveJournal(what) {      server = "http://www.livejournal.
com/";  data = {user: LJ_USER,          password: LJ_PASSWORD,
subject: "",            message: what,          year: time.getYear(),           month:
time.getMonth(),                day: time.getDay(),             time_hrs:time.
getHours(),             time_mins: time.getMinutes() }  XMLRPC.post(data, server);}
```

The above pseuodocode is an illustration of an XML-RPC[**9**] procedure.

Using Mobile-to-Web Gateways

The above example (SMS-to-LiveJournal Gateway), shows how to perform"moblogging," or, blogging on the move. An even better way to achievemoblogging is to send multimedia messages using your mobile device: anMMS-to-LiveJournal Gateway [**4**].

There are plentyof ways Mobile-to-Web Gateways could be used:

- Comment Alert: Mostblogging tools allow readers to leave comments to your posts. You can easilycreate an application that would alert you, by way of SMS or MMS, when youreceive a comment. This can be easily achieved by "parsing" your comments page,or even better using an API (if one is available) to keep track of commentcounts.
- Mail Alert: Another very interesting way to use MobileGateways is to create a mobile mail alert application. This application wouldin essence, poll your POP3 mailbox for incoming mails, and when one arrives, itwould simply send an email to your gateway address, typically of the formyour-mobile-number@your-service-provider. This would make an interestingapplication for those on the move with handheld mobile devices. In fact, youcould even message the entire text of the email, limited, of course, only bythe SMS/MMS character limit imposed by your mobile phone operator.
- Using such gateways, you could provide your own mobile service. For example,you could provide topical news. How? Well, you would have a primary emailaddress that acts as the target "address" to obtain a news service. People whowant the service could simply send a message to that mail address, a typicalmessage being of the form: news whatever-category. Your gateway simply pollsyour POP3 mailbox for such messages, parses them, and accordingly delivers theservices!

Conclusion

This article explained simple, experimental mobile gateways. The focus here was on the software process involved: processing of incoming service requests. These gateways will not scale well for multi-user (typically more than 100,000 users) applications. A scalable mobile gateway would require considerations both on the network as well as on the software:

- A scalable network would typically be distributed [**6**, **7**].
- Handling multiple incoming service requests would involve implementation of a suitable queuing mechanism: this would involve both the network-level protocol as well as software mechanisms.
- Unlike the simple gateways described above, a scalable gateway would have a software protocol in place to handle information exchange.

The services demonstrated in the above examples are simple, but useful applications. What service you would want to provide to a mobile device is only limited by your imagination!

Acknowledgements

The author would like to thank Kiran Jonnalagadda for his MMS to LiveJournal gateway, written in Python. Subsequent "ports" of the gateway have been inspired by his code.

References

1 GNU General Public License. <http://www.gnu.org/copyleft/gpl.html>.

2 IANA MIME Media Types. <http://www.iana.org/assignments/media-types/>.

3 LiveJournal. <http://www.livejournal.com/>.

4 MMS to Livejournal Gateway. <http://premshree.seacrow.com/code/ruby/mms2lj.rb/download>.

5 SMS to LiveJournal Gateway. <http://sourceforge.net/projects/ljtools>.

6 Tanenbaum, A.S. (2002). *Computer Networks*, 4e, Prentice Hall.

7 Tanenbaum, A.S. van Steen, M. (2002). *Distributed Systems: Principles and Paradigms*, Prentice Hall.

8 Yahoo! Mail for SMS. <http://in.mobile.yahoo.com/new/mail/>.

9 XML-RPC Home Page. <http://www.xmlrpc.com/>.

Biography

Premshree Pillai (premshree_pillai@yahoo.co.in) received a B.S. in Information Technology from Mumbai University. He works with the "market innovation" group at Yahoo!, Bangalore. He evangelizes the use of open source languages for development. In his free time, he likes to read books, fiction mostly.