Short Message Service in a Grid-Enabled Computing Environment

Fenglian Xu, Hakki Eres, and Simon Cox

School of Engineering Sciences,
University of Southampton,
Highfield, Southampton, SO17 1BJ, UK
{F.Xu, Hakki.Eres, sjc}@soton.ac.uk
http://www.geodise.org/

Abstract. Pervasive computing provides an attractive vision for the future of computing. Mobile computing devices such as mobile phones together with a land-based and wireless communication network infrastructure are the existing technical prerequisites for continuous access to networked services. The security of the system is a high concern in this environment, as well as its usability. This paper presents a network infrastructure for using the Short Message Service (SMS) to communicate with mobile phones via a grid-enabled service. The network system includes: a messenger server, a messenger client, Globus Servers and the Short Message Service Centre (SMSC) tied together with XML-RPC, TCP/IP and GRAM protocols for communication. A Matlab tool to use the grid-enabled SMS has been implemented and can be used in any grid-enabled environment.

1 Introduction

Pervasive computing can deliver access to information with no limits. IBM has focused its resource on new technologies that connect to any application, from any device over any network, using any style of interface [1]. The idea is to provide an interface environment via a mobile device for mobile workers to access their office. The protocol for the connection between the mobile device and applications such as Email, Excel or other applications is via an Internet access IP address. Here we choose a widely used mobile phone short message service (SMS) to communicate to a service or an application in a grid-enabled environment. The great feature of this protocol is its ease of use and that it does not require access to the Internet from your mobile phone and it is a real-time communication. The GEODISE project [2] aims to aid engineers in the design process by making available a suite of design optimisation and search tools and Computational Fluent Dynamics (CFD) analysis packages integrated with distributed grid-enabled computing, data, and knowledge resources. Design optimisation and search is a long and repetitive process. The process begins with a problem definition or a tool to generate geometry which is meshed; a CFD analysis is then performed from which an objective function is calculated. Design Search is the process by

which the geometry is systematically modified to improve the value of the chosen objective function. This process often takes a long time (up to weeks) and may be fraught with problems caused by the compute infrastructure, meshing, or the actual CFD runs, or with the optimisation failing to make much progress in improving the design. It is therefore important to monitor the ongoing design run to ensure that it is progressing and hasn't stalled or terminated anomalously. Users sit in front their computers to monitor the results or they log-in to check results regularly. This can be very frustrating. Mobility addresses solutions that help to cross time, geographic, media and service boundaries. This paper demonstrates a case of how to use a mobile text message function in a grid-enabled computing environment [3]. The process is done remotely via Globus 2.2 [4] middleware software integrated into Matlab [5] which we use as a scripting/hosting environment. The communication between the service application and the end user is via SMS and it is a two-way channel. One way is that the user submits a job and the job sends text messages, for example, the progress of an optimisation to the mobile phone. The other way is that the mobile user can send a message to kill a job or change a design parameter with a unique job handle if it is necessary or desirable to terminate.

The short message can be up to 160 characters long due to the memory limitation of mobile phones, where each character is 7 bits according to the 7-bit default alphabet [6]. Eight-bit messages (max 140 characters) are usually not viewable by the phones as text messages, instead they are used for data in images and ring tones.

The SMS system in the Geodise project should be:

- able to communicate in two ways,
- able to schedule messages immediately,
- efficient and economic,
- able to cover most mobile network operators,
- able to send both text message and binary image,
- be robust, reusable and platform independent.

In this paper, the SMS has been designed and implemented, as well a tool that enables users to send a text message from the Matlab environment. The service is then integrated to the entire grid-enabled environment [7]. Related work is discussed in Sect. 2. The architecture and the implementation of the message system are described in Sect. 3. An exemplar of using the SMS is depicted in section Sect. 4. The conclusions and future work are summarised in Sect. 5.

2 Related Work

Several recent efforts have been proposed to address the issue of incorporating SMS capabilities for the remote monitoring and control of automation systems. The OPC-SMS gateway [8] is a service which enables mobile users to send a SMS message to request specific information from OPC servers and to be notified by OPC servers. With recent advances in mobile computing technology, it is possible

to support wireless messaging services which enable mobile users to access their message at anytime and anywhere [9]. This has enabled mobile users to send and receive messages beyond geographic boundaries, however, users have to access Internet to fetch the message. The message received is not a real time message. Actually it is not substantially different from using computers and the usage is more complicated. There are many SMS service providers which offer an API so that they can be used to communicate with mobile phones. However each of them has different programming environments and different functions. Different service providers have been studied and discussed in this section.

Redcoal SMS [10] uses Simple Mail Transfer Protocol (SMTP) protocol so users have to access the Internet to read and send messages. An initial evaluation has shown that only mobile phones with Vodafone operator can receive text messages. It was either caused by the roaming function of the mobile operator or by the functionality of the SMS. In addition, it is located in Australia which introduces a delay into the messages and takes a longer time than a local service. Lucin SMS [11] has a web service with an engine to enable users to send text messages, ringtones, logos and retrieve a list of messages which have been sent. However it is expensive to use. Level 9 [12] provides a SMSC which has two-way communication. It has a programming environment in Perl on Linux, C/C++ and Java. Java programming is object-oriented and platform independent which meets the system requirements mentioned earlier. Level 9 has provided free access to its service for the purpose of this research project. The message from a computer program can be received by a mobile phone in seconds.

3 System Architecture and Implementation

The system analysis has been done and the system requirements have been defined in the previous section. The system architecture, as well as the design and implementation are presented in this section.

3.1 System Architecture

The infrastructure of the SMS for computing applications and mobile users is crossing the computer network and the mobile network. The entire system network is shown in Fig. 1.

The Messenger Service (MS) is a main entity which enables computer applications to send/receive messages to/from mobile phones. The MS includes a stack of services: Globus 2.2 Server, Messenger Client and Messenger Server.

The three of them have provided the functions of security, sending and receiving the messages over the Internet crossing the firewall. The Globus server enables the message client to be run remotely and safely. The protocol between the Globus server and the remote application is via Grid Resource Allocation Manager (GRAM) and the protocol between the messenger client and the messenger server is TCP/IP. The messenger server waits for connections from clients and it then forwards the messages from the clients to the Short Message Service

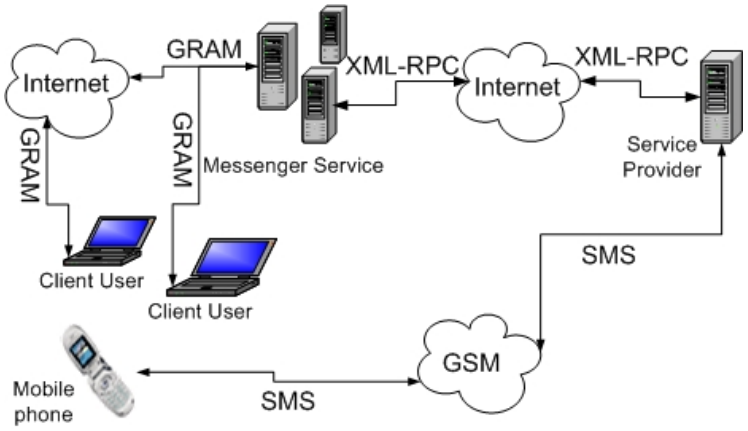


Fig. 1. System network

Centre(SMSC). SMSC is an intermediate entity, which enables communication between GSM networks and the Internet for sending and delivering text messages/binary images for current/3G mobile phones. The SMSC is a commercial software. The protocol between the messenger server and the SMSC software is via XML-RPC [13] over the Internet. In addition to the above software and network protocols, the applications and the mobile devices/mobile phones are the two end entities. The applications are implemented in the Matlab environment which is platform independent.

3.2 Security

The text messages/data are transferred over the Internet and over the wireless network, hence security is important to control access to the system, which consists of authentication and authorisation. The authentication shows who you are and the authorisation grants what you can do with a shared software resource. Globus Toolkit 2.2 has three components: resource management, information service and data management. They all use Grid Security Infrastructure (GSI) [14] for enabling authentication and communication over an open network. GSI provides a number of useful services for Grids. GSI is based on public key encryption, X.509 certificates [15], and the Secure Sockets Layer (SSL) communication protocol. Extensions to these standards have been added for single sign-on and delegation. The GRAM allows different clients to run a job remotely. The clients have to request a personal certificate from the Globus organisation, or from any other trusted certificate authority, such as the UK-eScience Certificate Service [16], to use GRAM. The certificate together with a private key is saved to the client side in a directory called `.globus` under the home directory depending on the host environment. Each certificate has its own subject in the format of: `/C=UK/O=eScience/OU=Southampton/L=SeSC/CN=fenglian xuchen`. This sub-

ject can be obtained via a command `grid-cert-info -subject`. Entries of the certificate subject and the user name are saved in a file called `grid-mapfile` located in the directory `/etc/grid-security` on the Globus server. This file is owned by the system administrator and has permissions 600. If a client presents a certificate which matches one of entries in a `grid-mapfile`, then the client can use any software resource which is permitted.

The Globus 2.2 GSI prevents users without a trusted certificate from using resources. This means that once users have an account on the server side and a personal certificate on the client side, they can use any software resource under the group of users on the server side remotely from the client. Therefore an extra feature of the security beyond Globus 2.2 GSI is necessary to protect a certain software resource; namely authorization which plays an important role in the security layer. The protected software has to be installed under a directory which belongs to a certain group. Users who wish to join this group have to obtain permission from the system administrator. The software is assigned with read and execution permissions. This would only allow the administrator and the group users to use the software. The messenger server can be located either on the same machine as the messenger client or a different machine. The messenger client has to be located on the same machine as the Globus server where it can benefit from the Globus Security. Each message has a unique ID, an associated job ID, the job status information etc. This information is stored in a relational database and can be retrieved later for terminating a job.

In the current system, we have not implemented any encryption or signing of the messages intransit, though this is, in principle, possible.

3.3 System Design

By combining the system security and the architecture, the system has been designed as shown in Fig. 2. The system includes three nodes: (i) the Globus server together with the messenger client and the messenger server, which runs on a Linux box; (ii) the SMSC service, which runs on a remote machine over the Internet; (iii) and the platform independent Matlab tool, which runs on either Windows or Linux machine. The Matlab tool shall be able to submit a job to the Globus server and query a job status from a relational database. The submitted job starts the messenger client and passes a message to the messenger server. The messenger server sends the message to a mobile phone via SMSC and creates an initial entry in the database. When the messenger server receives a message from the mobile user via SMSC, the entry in the database is updated with an action in the message. The action is then retrieved by a tool in the Matlab application.

The communication between the Matlab tool and the messenger client is via Resource Specification Language (RSL) [17] which is understood by GRAM. The communication between the messenger client and the messenger server is via TCP/IP. The protocol between the messenger server and SMSC is XML/RPC. XML/RPC is a simple, portable way to make remote procedure calls over HTTP. The SMSC has its own security system which is embedded in each message with user name and password over the network. The message server acts as a SMSC

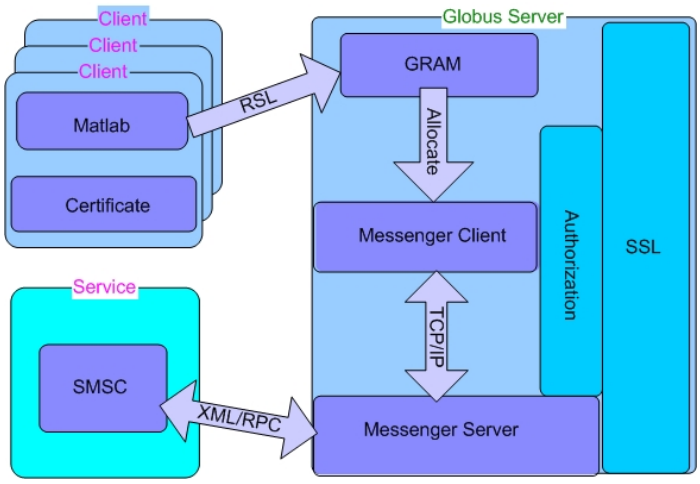


Fig. 2. Architecture of the messenger

client to connect with the SMSC and makes a remote procedure call with arguments. The arguments include the following fields: user name, password, mobile number and text message. Additionally, there are three other reserved fields. The authorization and the reserved information are embedded in the messenger server. This makes the system secure.

3.4 Implementation

The SMS system is service-oriented and it consists of a: messenger server, messenger client, Globus server and SMSC. The messenger client and messenger server play a key role to bridge other services. The architecture of these two components is based on the client-server design pattern. This design pattern allows multiple users to use the messenger server without modification to the software. They are implemented in Java socket programming.

The messenger client is responsible for requesting a connection to the messenger server, sending a short text message and closing the connection. The messenger server creates a server socket for incoming connections and listens for clients constantly for accepting a new client connection and starting a message received thread. This process would enable multiple clients to connect to the server and to send messages simultaneously. Each thread processes one message based on one client. The thread process checks if the message is sent by the messenger client and forwards the right message together with authentication information to the SMSC.

The two-way message processes is shown in Fig. 3. In Fig. 3, a top level work environment is an application in Matlab and the end entry is a mobile phone. The system includes two message flows: Messages from an application to a mobile

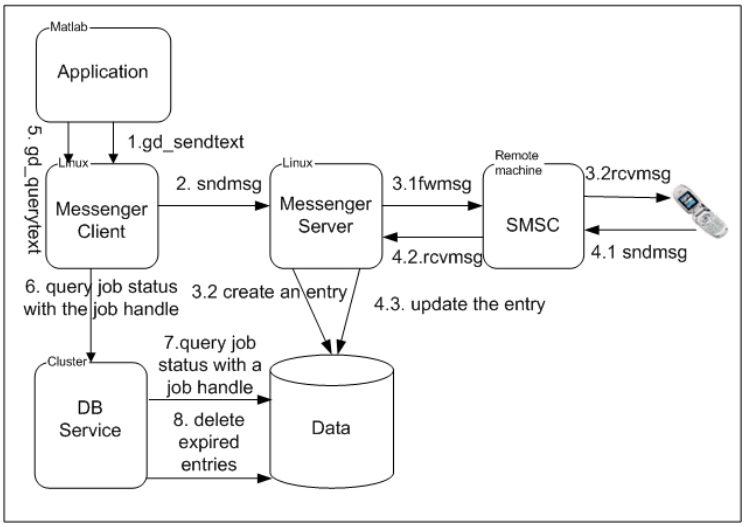


Fig. 3. Two-way message processes

phone(down-flow) and messages from the mobile phone to the application(up-flow).

The down-flow sends notifications to the mobile user and the up-flow sends control messages to the application. A notification includes a result of a running job and a job handle. The job handle is used as a reference and it is stored in the database for security and usability reasons. The job is submitted by the application to a remote machine with the Globus server. The messenger client sends the message to the messenger server. The messenger server processes the message and forwards it with a unique message ID to the SMSC and creates an entry in the database. The down-flow ends with a mobile phone which receives the message from the SMSC via SMS. The up-flow consists of two sub-flows: one begins with the mobile user sending a reply message with the message ID to the SMSC. The SMSC then forwards the message back to the messenger server which updates the entry with the specified message ID. The other one is that the application starts `gd_querytext` with a job handle to the messenger client via the Globus Server. The messenger client passes the message to the DB Service which is able to query a job status from the database. The job status is returned back to the Matlab. The job status is regularly queried to decide if the job needs to be terminated or not.

The database is designed to store the information about each message being sent to SMSC. The messenger server creates an entry when a text message is forwarded to the SMSC. The entry is updated when the messenger server receives a reply message. The expired entries need to be deleted automatically by the DB Service or whenever a job state with “to abort” is received. The entry information is shown in Table 1.

Table 1. Message information in SMS

messageID	jobHandle	sendTime	expireTime	receivedTime	jobStatus
1	1213131334	31/12/02 12:00	01/01/03 12:00	0:00	running
2	1234567487	31/12/02 13:00	01/01/03 13:00	01/01/03 12:45	to abort
3	1234567488	02/01/02 13:00	03/01/03 13:00	03/01/03 12:45	running

The `messageID` is a primary key and has a unique ID. The `jobHandle` is the handle of the current running job which sends the results to the mobile user. The `sendTime` is the date and time when the outbound message was sent. The `receivedTime` is the date and time that any reply message related to this job was received. The `jobStatus` is initialised when the message is sent and it is updated to reflect any request (such as to abort) from a reply message. The `expireTime` is the `sendTime` plus a specified time which is defined by the user, as the lifetime validity of the message in the system by analogy with the lifetime of a proxy certificate. After this expiry time the entry is deleted from the database and messages requesting that this job to be aborted will be ignored.

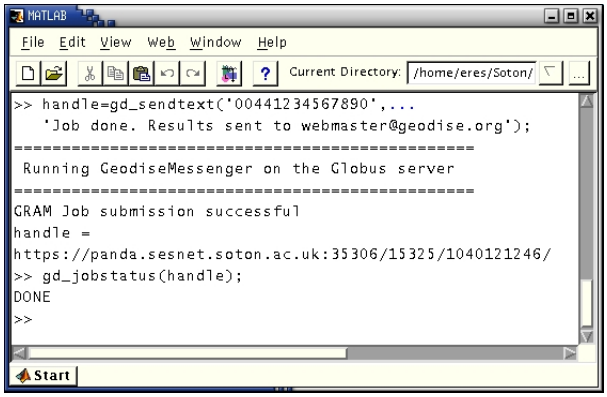
4 Application Exemplar

This section demonstrates how to use the gd_sendtext tool in the Matlab environment. In general this function does the following processes:

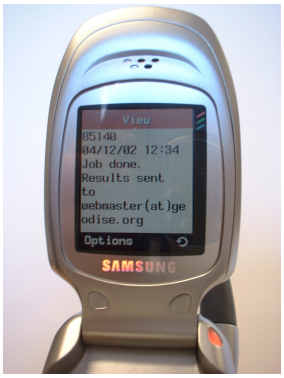
- checks if the user has a valid proxy,
- generates a grid proxy by requesting a password if there is no valid proxy,
- runs the messenger client remotely on the Globus server via GRAM to submit a job,
- checks the status of the process by using a unique handle returned by the Globus server during job submission.

The Geodise toolkit [18] provids a Matlab tool called `gd_proxyquery` which enables users to obtain Grid proxy information. The grid proxy can be generated by using `gd_createproxy` command. This command invokes the Java CoG [19], which in turn pops a window where the user can type a password. After the user enters their password and presses the "Create" button, a proxy certificate is generated for the user. The job is submitted by `gd_jobsubmit` which takes a RSL string and a remote host name as arguments. The RSL string combines the remote directory, the executable program name and the arguments to execute the program. An example of using `gd_sendtext` is shown in Fig. 4.

As it can be seen from Fig. 4(a), once the job has been submitted, a response message to the submission will be received and displayed on the window; and a unique job handle is also returned and displayed on the screen. The job status can be obtained by using `gd_jobstatus`. The job status is specified by ACTIVE and DONE. The DONE means that the job has been finished. On the other end,



(a) Using `gd_sendtext` to send a message



(b) Message received from `gd_sendtext`

Fig. 4. Example of using `gd_sendtext`

a mobile user will receive the message in a few seconds. The result is shown in Fig. 4(b). By comparing with Fig. 4(a), the contents of the message shown on the mobile phone are same as the second argument of the `gd_sendtext` function apart from the special character @ due to the parser of the SMSC software.

The exemplar of using `gd_sendtext` shows how to send a notification to a mobile phone. The other way is that the recipient has to send a message to a fixed number, where the number to send to can be prepurchased, or obtained from a bulk service provider, who then forwards messages based on keywords in the remainder of the message. The message is consisted of `NUMBER [KEYWORD] ACTION JOB_ID`. For example, `81001 'kill' ID#` and `81000 SOTON 'abort' ID#`. In this example, `81001` is a prepurchased dedicated number, whereas `81000` is a bulk number from which the service provider forwards messages based on the key word (e.g. 'SOTON').

5 Conclusions and Future Work

The power of this application has enabled users to be notified beyond geographic boundaries and to remotely control a system in a grid-enabled environment. The messenger service can be plug-n-played in any grid-enabled environment and it is easy to use. We have demonstrated how the short message service has been integrated with a grid-enabled computation on design search and optimisation system to allow monitoring and simple steering (e.g. termination of a job). The advantage of using the short message service over the email protocol SMTP is that it doesn't requires an Internet connection and it is fast and easy to use. The architecture of two-way communication has been designed and the one way of sending a message from an application to a mobile phone has been implemented and tested. The other way to receive a message from a mobile phone to the

application is to be implemented and tested. Future work will include use of third generation (3G) technology to send output, e.g. graphics, from a code to a 3G mobile. We will also move to using the Open Grid Service Architecture as reference implementation becomes available as an open standard for connecting our service to grid-enabled applications.

Acknowledgements

We are grateful to Level 9 networks for providing SMS service capability for this project.

References

1. IBM Resource. (2002) http://www.research.ibm.com/thinkresearch/pervasive.shtml
2. The Geodise Project. (2002) http://www.geodise.org/
3. Global Grid Forum. (2002) http://www.gridforum.org/
4. The Globus Project. (2002) http://www.globus.org/
5. Matlab 6.5. (2002) http://www.mathworks.com/
6. GSM 03.38 Standard Specification. (2002) http://www.dreamfabric.com/sms/default_alphabet.html
7. Pound, G.E., Eres, M.H., Wason, J.L.and Jiao, Z., Keane, A.J., Cox, S.J.: A grid-enabled problem solving environment (PSE) for design optimisation within matlab. accepted by IPDPS (2003)
8. Kapsalis, V., Koubias, S., Papadopoulos, G.: OPC-SMS: a wireless gateway to OPC-based data sources. Computer Standards & Interfaces **24** (2002) 437–451
9. Tan, D.H.M., Hui, S.C., Lau, C.T.: Wireless messaging services for mobile users. Journal of Network and Computer Applications **24** (2001) 151–166
10. (Redcoal Website) http://www.redcoal.com
11. (Lucin Website) http://www.soapengine.com/lucin/soapenginex/smsx.asmx
12. (Level9 Website) http://www.level9.net:2048/RPC2
13. (XMLRPC Website) http://xmlrpc-c.sourceforge.net/xmlrpc-howto/xmlrpc-howto.html
14. Foster, I., Kesselman, C., Tsudik, G., Tuecke, S.: A security architecture for computational grids. In: 5th ACM Conference on Computers and Communications Security, San Francisco, California (1998)
15. Housley, R., Ford, W., Polk, W., Solo, D.: RFC2459: Internet X.509 public key infrastructure certificate and CRL profile. In: PKIX IETF Working Group, San Francisco, California (1999)
16. UK Grid-Support Center. (2002) http://www.grid-support.ac.uk/
17. The Globus Resource Specification Language (RSL) v1.0. (2002) http://www-fp.globus.org/gram/rsl_spec1.html
18. Eres, M.H., Pound, G.E., Jiao, Z., Wason, J., Xu, F., Keane, A.J., Cox, S.J.: Implementation of a grid-enabled problem solving environment in Matlab. accepted by the Workshop on Complex Problem-Solving Environments for Grid Computing (Held in conjunction with the ICCS) (2003)
19. Commodity Grid Kits. (2002) http://www.globus.org/cog/