Mobile-to-Grid Middleware: Bridging the gap between mobile and Grid environments

Hassan Jameel[1], Umar Kalim[1], Ali Sajjad[1], Sungyoung Lee[1], and Taewoong Jeon[2]

[1] Department of Computer Engineering, Kyung Hee University
1, Sochen-ri, Giheung-eup, Yongin-si, Gyeonggi-do, 449-701, South Korea
{hassan, umar, ali, sylee}@oslab.khu.ac.kr
[2] Department of Computer & Information Science, Korea University, Korea
jeon@selab.korea.ac.kr

Abstract. Currently, access to Grid services is limited to resourceful devices such as desktop PCs but most mobile devices (with wireless network connections) cannot access the Grid network directly because of their resource limitations. Yet, extending the potential of the Grid to a wider audience promises increase in flexible usage and productivity. In this paper we present a middleware architecture[1] that addresses the issues of job delegation to a Grid service, support for offline processing, secure communication, interaction with heterogeneous mobile devices and presentation of results formatted in accordance with the device specification. This is achieved by outsourcing the resource intensive tasks from the mobile device to the middleware. We also demonstrate through formal modeling using Petri nets that the addition of such a middleware causes minimum overhead and the benefits obtained outweigh this overhead.

1 Introduction

Grid [18] computing permits participating entities connected via networks to dynamically share their resources. Extending this potential of the Grid to a wider audience, promises increase in flexibility and productivity, particularly for the users of mobile devices who are the prospective consumers of this technology.

Consider a teacher who wants to augment his lecture with a heavy simulation test. He uses his PDA to access a Grid service and submit the request. The service after executing the request compiles the results which are then distributed to the mobile devices of the registered students of that course. Similarly a doctor on the way to see his patient, requests a Grid medical service to analyze the MRI or CT scans of the patient from his mobile device. By the time he meets his patient; the results would be compiled and presented on his mobile device to facilitate the treatment.

[1] This research work has been supported in part by the Korea Ministry of Information and Communications' ITRC program in joint collaboration with Information and Communications University, Korea.

The clients that interact with the Grid middleware to accomplish a task are required to use client end libraries. These libraries are relatively resource intensive considering the limitations of mobile devices. Conceiving a distributed system that uses these libraries directly will not be a practical mobile system because of the resource demands.

Moreover, most of the conventional distributed applications are developed with the assumption that the end-systems possess sufficient resources for the task at hand and the communication infrastructure is reliable. For the same reason, the middleware technologies for such distributed systems usually deal with issues such as heterogeneity and distribution (hence allowing the developer to focus his efforts on the functionality rather than the distribution).

The issues that primarily affect the design of a middleware for mobile systems are: *mobile devices*, *network connection*, and *mobility*. Firstly, due to the tremendous progress in development of mobile devices, a wide variety of devices are available which vary from one to another in terms of resource availability. Secondly, in mobile systems, network connections generally have limited bandwidth, high error rate and frequent disconnections due to power limitations, available communication spectrum and user mobility. Lastly, mobile clients usually interact with various networks, services, and security policies as they move from one place to another.

Considering the assumptions and characteristics of conventional middleware technologies it is quite evident that they are not designed to support mobile systems adequately. Instead, they aim at a static execution platform (where the host location is fixed) and the network bandwidth does not vary. Hence, given the highly variable computing environment of mobile systems, it is mandatory that modern middleware systems are designed that can support the requirements of mobile systems such as *dynamic reconfiguration and asynchronous communication.*

In this paper:

- We present an architecture for a middleware (Section 2) enabling heterogeneous mobile devices access to Grid services by providing support for delegation of jobs to the Grid, secure communication between the client and the Grid, off-line processing, adaptation to network connectivity issues and presentation of results in a form that is in keeping with the resources available at the client device.
- We demonstrate (Section 3) that the addition of such a middleware causes minimum overhead and the benefits obtained by it outweigh this overhead.

2 Architecture Details

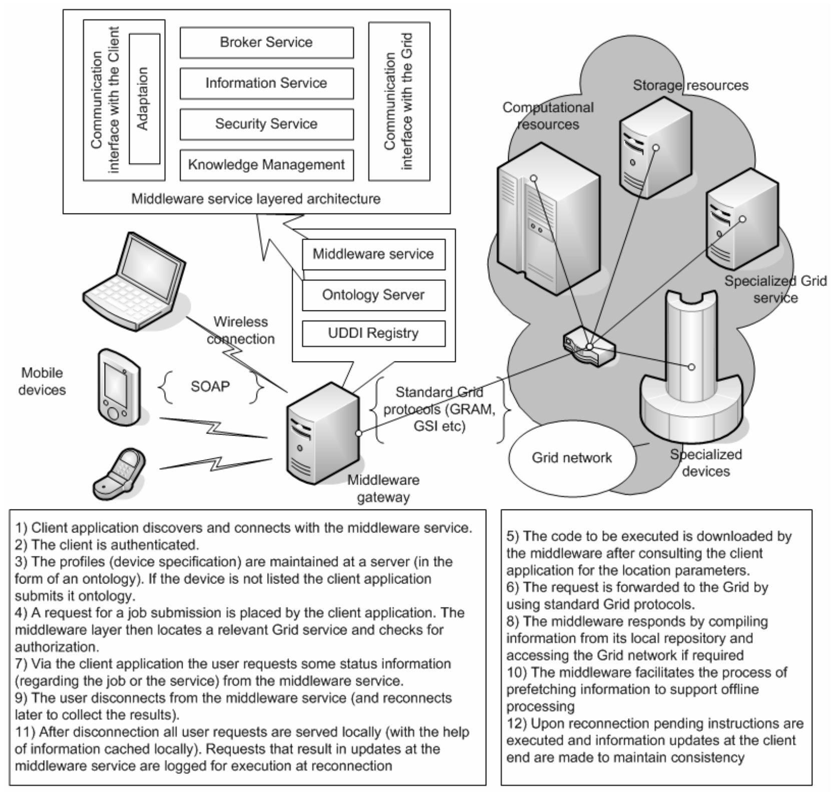
The middleware service is exposed as a web service to the client applications. The components of the middleware service (as shown in Figure 1) are discussed succinctly as follows:

2.1 Discovery Service

The discovery of the middleware by mobile devices is managed by employing a UDDI registry [6], [7]. Once the middleware service is deployed and registered, other applications/devices would be able to discover and invoke it.

2.2 Communication Interface with the Client Application

The interface advertised to the client application is the communication layer between the mobile device and the middleware. This layer enables the middleware to operate as a web service and communicate via the SOAP framework [8].



1) Client application discovers and connects with the middleware service.
2) The client is authenticated.
3) The profiles (device specification) are maintained at a server (in the form of an ontology). If the device is not listed the client application submits it ontology.
4) A request for a job submission is placed by the client application. The middleware layer then locates a relevant Grid service and checks for authorization.
7) Via the client application the user requests some status information (regarding the job or the service) from the middleware service.
9) The user disconnects from the middleware service (and reconnects later to collect the results).
11) After disconnection all user requests are served locally (with the help of information cached locally). Requests that result in updates at the middleware service are logged for execution at reconnection

5) The code to be executed is downloaded by the middleware after consulting the client application for the location parameters.
6) The request is forwarded to the Grid by using standard Grid protocols.
8) The middleware responds by compiling information from its local repository and accessing the Grid network if required
10) The middleware facilitates the process of prefetching information to support offline processing
12) Upon reconnection pending instructions are executed and information updates at the client end are made to maintain consistency

Fig. 1. Deployment model and the architecture

Adaptation to Disconnected Operations The advertisement of the mobile-to-Grid middleware as a web service permits the development of the architecture in a manner that does not make it mandatory for the client application to remain connected to the middleware at all times while the request is being served.

We focus on providing software support for offline processing at the client device. For this we consider two kinds of disconnections; intentional disconnection, where the user decides to discontinue the wireless connection and unintentional disconnection, which might occur due to variation in bandwidth, noise, lack of power etc. This is made possible by pre-fetching information or meta-data only from the middleware service. This facilitates in locally serving the client application at the device. However, requests that would result in updates at the middleware service are logged (so that they may be executed upon reconnection).

To establish the mode of operation for the client application, a connection monitor is used to determine the network bandwidth and consequently the connection state (connected or disconnected). Moreover, during execution, checkpoints are maintained at the client and the middleware in order to optimize reintegration after disconnection.

2.3 Communication Interface with the Grid

The communication interface with the Grid provides access to the Grid services by creating wrappers for the API advertised by the Grid. These wrappers include standard Grid protocols such as GRAM [9], MDS [10], GSI [11] etc which are mandatory for any client application trying to communicate with the Grid services. This enables the middleware to communicate with the Grid, in order to accomplish the job assigned by the client.

2.4 Broker Service

The broker service deals with initiating the job request and steering it on behalf of the client application. Firstly the client application places a request for a job submission. After determining the availability of the Grid service and authorization of the client, the middleware downloads the code (from the mobile device or from a location specified by the client e.g. an FTP/web server). Once the code is available, the broker service communicates with Grid's GRAM service to delegate the job.

A status monitor service (a subset of the broker service) interacts with GRAM to determine the status of the job. The status monitor service then communicates with the Knowledge Management module to store the results. The mobile client may reconnect and ask for the (intermediate/final) results of its job from the status monitor service.

2.5 Knowledge Management

The knowledge management layer of the system is used to manage the relevant information regarding both the client and Grid applications and services. The main function of this layer is to connect the client and Grid seamlessly as well as to introduce the capability of taking intelligent decisions such as downscaling the results according to device profile, based on the information available to the system.

2.6 Information Service

This module interacts with the wrapper of the GLOBUS toolkit's API for information services (MDS [10]). It facilitates the client application by managing the process of determining which services and resources are available in the Grid and also monitors resources such as CPU load, free memory etc.

2.7 Security

The Grid Security Infrastructure is based on public key scheme mainly deployed using the RSA algorithm [12]. However key sizes in the RSA scheme are large and thus computationally heavy on handheld devices such as PDA's, mobile phone's, smart phones etc [13]. We employ the Web Services Security Model [14] to provide secure mobile access to the Grid. This web services model supports multiple cryptographic technologies.

The Elliptic Curve Cryptography (ECC) based public key scheme can be used in conjunction with Advanced Encryption Standard(AES) for mobile access to Grid which provide the same level of security as RSA and yet the key sizes are a lot smaller [13].

Communication between the user and middleware is based on security policies specified in the user profile. According to this policy different levels of security can be used e.g. some users might just require authentication, and need not want privacy or integrity of messages.

3 Petri Net Model of the System and its Analysis

In this section we model the interaction between the mobile client and the mobile grid middleware service. Our goal is to estimate the delay caused by the communication between the client and middleware service as well as the additional processing done by it. This delay should be within acceptable limits so that the mobile client user is not at a disadvantage as compared to a normal Grid user as far as time is concerned. We use the time to completion of the whole process as an index of performance of our middleware communication architecture. We keep the time taken by Grid processing constant as our results will be bench marked against it. The communication is modeled by using non-Markovian Stochastic Petri nets [2] [3]. We follow an approach similar to the work done by Antonio Puliafito et al [1].

To make the Petri Net model in Figure 2, we modeled the following sequence of operations between the middleware and mobile client:

A mobile client first sends a request(*send_req_uddi*) to a UDDI registry to discover an existing middleware service. The UDDI registry returns(*send_resp_uddi*) the URI(Uniform Resource Indicator) of the middleware service to the mobile client. The client then sends a request(*send_req*) to the middleware service which includes a URI of its ontology file and a URI of the code to be executed on its behalf. The middleware retrieves the code (*retrieve_code*) and the ontology file (*retrieve_ont*) at the same time and then executes the code (*code_exec*). The code execution includes requests to the Grid, and thus the rest of the job is done by the Grid. We just simulate it as an immediate transition , as this time would be the same as in the case of a normal Grid user. Upon receiving the results, the middleware scales down the results accord-ing to the device profile in the ontology file (*result_downscaling*) and sends the results to the mobile device (*send_result*). This concludes the communication session. Initially , the place Ready contains a token and at the end of the session the token is in place (*end_session*).

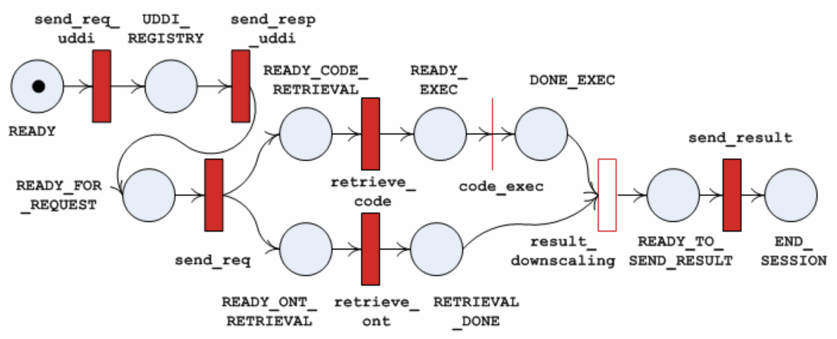


Fig. 2. Petri net model; communication between the client and the middleware

3.1 Parameters used in the Petri Net Model

In order to make our model simple, we do not consider secure communication as well as disconnected operation. To evaluate the Petri Net model in Figure 2, we used the following numerical parameters which are consistent with the ones used in [1]. We give a description of the parameters as follows:

Size of a request (D_{req}): The mobile client sends two types of requests, one to the UDDI (service type request) and one to the middleware service (URIs for the ontology and code files) . The size of these requests can safely be assumed to be small.

Size of the reply (D_{min},D_{max}): This depends upon the type of data requested by the mobile client which could be merely a small numerical value (as large as D_{req}) or a little bigger image file. So we have two extremes of data sizes.

Size of the ontology (D_{ont}) and code (D_{code}): The ontology file is an XML document and we can safely assume its size to be $\leq$ 10 KB.

Mean processing time for downscaling the results ($1/\lambda_{scale}$): We fix this time as an exponentially distributed random time whose exact value depends upon the specific application.

Throughput of the communication network (Th_{low},Th_{high}): We assume two kinds of transmission rates in the network. The wireless network has been assumed to have lower throughput (Th_{low}), where as the Grid and the middleware service are assumed to be connected with high speed link indicated by (Th_{high}).

We use the above mentioned parameters to describe the distributions associated with the transitions in the Petri Net model, depicted in Table 1.

Table 1. Parameters used in the Petri Net model

Transition Name	Type	Expression
send_req_uddi	Deterministic	(D_{req}/Th_{low})
send_resp_uddi	Deterministic	(D_{min}/Th_{low})
send_req	Deterministic	(D_{req}/Th_{low})
retrieve_ont	Deterministic	(D_{ont}/Th_{high})
retrieve_code	Deterministic	(D_{code}/Th_{high})
code_exec	Immediate	-
results_downscaling	Exponential	(λ_{scale})
send_result	Uniform	$[D_{min},D_{max}]/Th_{low}$

3.2 Numerical Evaluation of the Petri Net

We assigned the following numerical values to these parameters as shown in Table 2 for the evaluation of the Petri Net. These values are consistent with the ones used by [1].

Table 2. Numerical Values used for the Parameters

Parameter	Description	Value
D_{req}	Dimension of client request	1 KB
D_{min}	Minimum amount of Data	1 KB
D_{max}	Maximum amount of Data	30 KB
D_{ont}	Dimension of ontology	10 KB
D_{code}	Dimension of code	40 KB
λ_{scale}	Results scale down rate	4 req/s
Th_{high}	Throughput of wired network	1 Mbps

The firing rate of the *results_downscaling* transition has been fixed to λ_{scale}=4 requests/sec. This factor is not only application dependent but also dependent on the computational power of the computer containing the middleware. However a value of 4 req/sec is a reasonable approximation as used in [1]. We assume a high speed link in the wired network as it constitutes the Grid network and assign a value of Th_{high}=1 Mbps.

Based on these values, we evaluated the Petri Net described in Figure 2 by using the WebSPN [4] [5] tool with which we can associate exponential as well as non-exponential firing rates to the transitions. Figure 3a shows a graph of time to completion (t)versus the throughput (Th_{low}) of the wireless network ranging from 10Kbps to 1Mbps. The values of the transitions are shown in Table 3. The values for the *send_result* transition has been depicted as $[a,b]$ to show the minimum (a) and maximum (b) value of the uniform distribution.

Table 3. Numerical Values used for the Parameters

Th_{low}	*send_*	*send_*	*send_req*	*retrieve*	*retrieve*	*results_*	*send_*
	req_uddi	*resp_uddi*	*send_req*	*_ont*	*_code*	*downscaling*	*result*
K bits/sec	sec	sec	sec	sec	sec	req/sec	sec
10	0.8	0.8	0.8	0.08	0.32	4	[0.4, 24.0]
20	0.4	0.4	0.4	0.08	0.32	4	[0.4, 12.0]
50	0.16	0.16	0.16	0.08	0.32	4	[0.16, 4.8]
100	0.08	0.08	0.08	0.08	0.32	4	[0.08, 2.4]
200	0.04	0.04	0.04	0.08	0.32	4	[0.04, 1.2]
500	0.016	0.016	0.016	0.08	0.32	4	[0.016, 0.48]
1000	0.008	0.008	0.008	0.08	0.32	4	[0.008, 0.24]

Let's see the affect if we fix the result size to 1KB and the other values the same as in Table 3. We can do that by making *send_result* a deterministic transition with firing rate D_{min}/Th_{low}. After evaluating the Petri Net with this value, we obtain a graph shown in Figure 3b. The graph shows no notable distinction with varying Th_{low}.

We can conclude by studying the two graphs that except for the two obvious parameters, namely the wireless network throughput and the result size, the time to completion is not severely affected by the middleware to client communication and even with low throughput and considerably large result set, the time taken by the middle-ware to mobile device communication is within acceptable limits, only in the order of a few seconds in the worst case.

4 Related Work

Various efforts have been made to solve the problem of mobile-to-Grid middleware. Signal [15] proposes a mobile proxy-based architecture that can execute jobs submitted to mobile devices, so in-effect making a grid of mobile devices. A proxy interacts with the Globus Toolkit's Monitoring and Discovery Service to communicate resource availability in the nodes it represents. The proxy server and mobile device communicate via SOAP and authenticate each other via the generic security service (GSS) API. The proxy server analyzes code and checks

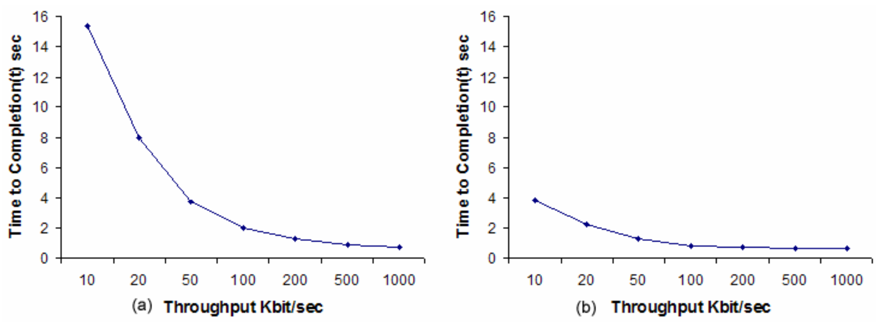


Fig. 3. (a) Time to completion (t) vs Throughput of the wireless network Th_{low}. (b) Time to completion (t) vs Throughput of the wireless network Th_{low} with result size = 1 KB.

for resource allocation through the monitoring and discovery service (MDS). After the proxy server deter-mines resource availability, the adaptation middleware layer component in the server sends the job request to remote locations. Because of this distributed and remote execution, the mobile device consumes very little power and uses bandwidth effectively. Also their efforts are more inclined towards QoS issues such as management of allocated resources, support for QoS guarantees at application, middleware and network layer and support of resource and service discoveries based on QoS properties.

In [16] a mobile agent paradigm is used to develop a middleware to allow mobile users' access to the Grid and it focus's on providing this access transparently and keeping the mobile host connected to the service. Though they have to improve upon the system's security, fault tolerance and QoS, their architecture is sufficiently scalable. GridBlocks [17] builds a Grid application framework with standardized inter-faces facilitating the creation of end user services. They advocate the use of propriety protocol communication protocol and state that SOAP usage on mobile devices maybe 2-3 times slower as compared to a proprietary protocol. For security, they are inclined towards the MIDP specification version 2 which includes security features on Trans-port layer.

5 Conclusion and Future Work

In this paper we identified the potential of enabling mobile devices access to the Grid. We focused on providing solutions related to distributed computing in wireless environments, particularly when mobile devices intend to interact with grid services. An architecture for a middleware layer is presented which facilitates implicit interaction of mobile devices with grid services. This middleware is based on the web services communication paradigm. It handles secure communication between the client and the middleware service, provides software support for offline processing, manages the presentation of results to heterogeneous devices (i.e. considering the device specification) and deals with the delegation of job requests from the client to the Grid. We also demonstrated that the addition of such a middleware causes minimum overhead and the benefits obtained by it outweigh this overhead.

In future we intend to provide multi-protocol support to extend the same facilities to devices that are unable to process SOAP messages. Moreover, we will continue to focus on handling security, improving support for offline processing and presentation of results depending upon the device. Along with this implementation we intend to continue validating our approach by experimental results.

References

1. Puliafito, A., Riccobene, S., Scarpa, M.: Which paradigm should I use?: An analytical comparison of the client-server, remote evaluation and mobile agents paradigms', IEEE Concurrency and Computation: Practice & Experience, vol. 13, pp. 71-94, 2001.
2. Bobbio, A., Puliafito, A., Telek, M.: A modeling framework to implement preemption policies in non-Markovian SPNs. IEEE Transactions on Software Engineering, vol. 26, pp. 36-54, Jan. 2000.
3. Telek, M., Bobbio, A.: Markov regenerative stochastic Petri nets with age type general transitions. Application and Theory of Petri Nets, 16th International Conference (Lecture Notes in Computer Science 935). Springer-Verlag, pp. 471-489, 1995.
4. Bobbio, A., Puliafito, A., Scarpa, M., Telek, M.: WebSPN: A WEB-accessible Petri Net Tool. International Conference on WEB based Modeling and Simulation, San Diego, California, pp. 137-142, 11-14 January 1998.
5. WebSPN 3.2: http://ing-inf.unime.it/webspn/
6. Hoschek, W.: Web service discovery processing steps. http://www-itg.lbl.gov/ hoschek/publications/icwi2002.pdf
7. UDDI specification: www.oasis-open.org/committees/uddi-spec/doc/tcspecs.htm
8. SOAP Framework: W3C Simple Object Access Protocol ver 1.1, World Wide Web Consortium recommendation, 8 May 2000; http://www.w3.org/TR/SOAP/
9. GT3 GRAM Architecture: www-unix.globus.org/developer/gram-architecture.html
10. Czajkowski, K., Fitzgerald, S., et al.: Grid Information Services for Distributed Resource Sharing. Proceedings of the Tenth IEEE International Symposium on High-Performance Distributed Computing (HPDC-10), IEEE Press, August 2001.
11. Welch, V., Siebenlist, F., et al.: Security for Grid services. HPDC, 2003.
12. Welch, V., Foster, I., et al.: X.509 Proxy Certificates for dynamic delegation. Proceedings of the 3rd Annual PKI R&D Workshop, 2004.
13. Gupta, V., Gupta, S. et al.: Performance Analysis of Elliptic Curve Cryptogra-phy for SSL. Proceedings of ACM Workshop on Wireless Security - WiSe 2002 pages 87-94, Atlanta, GA, USA, September 2002, ACM Press.
14. Giovanni, D., Maryann, H., et al.: Security in a Web Services World; A Proposed Architecture and Roadmap, 2002, IBM and Microsoft Corp. A joint security whitepaper from IBM Corporation and Microsoft Corp. April 7, 2002, Version 1.0.
15. Hwang, J., Aravamudham, P.: Middleware Services for P2P Computing in Wireless Grid Networks. IEEE Internet Computing vol. 8, no. 4, July/August 2004, pp. 40-46.
16. Bruneo, D., Scarpa, M., et al.: Communication Paradigms for Mobile Grid Users. Proceedings 10th IEEE International Symposium in HPDC, (2001).
17. Gridblocks project (CERN) http://gridblocks.sourceforge.net/docs.htm
18. 18. Foster, I., Kesselman, C., Tuecke, S.: The Anatomy of the Grid: Enabling Scalable Virtual Organizations, Int'l J. Supercomputer Applications, vol. 15, no. 3, 2001, pp.200-222.