A Service-Oriented Middleware for Seamless Nomadic System-Aware (SNA) Servants

Dario Maggiorini, Walter Cazzola
DICo – Università di Milano, Milano. Italy

B. S. Prabhu, Rajit Gadh
University of California at Los Angeles. USA

Introduction

In the last few years there has been a considerable penetration of wireless technology in everyday life. This penetration has also increased the availability of Location Dependent Information Services (LDIS) [1], such as local information access (e.g. traffic reports, news, etc.), nearest-neighbor queries (such as finding the nearest restaurant, gas station, medical facility, ATM, etc.) and others.

New wireless environments and paradigms are continuously evolving and novel LDISs are continuously being deployed [2,3]. Such a growth means the need to deal with:

- *Services without standard interfaces* – same or similar LDISs being offered by different vendors through different APIs but with same standard functional interfaces
- *Services deployed dynamically* – LDIS made available on a need basis or when the scenario dynamically mutates and in addition provides dynamic roaming between services and dynamic service interchangeability
- *Non-classified services* (i.e., novel services).

Seamless Nomadic System-Aware (SNA) Servants

In this research work we enable a mobile device that:

- Is aware (by using introspection provided by reflection [7,9,6]) of and exploits every LDIS available in its range and
- Transparently changes LDIS server when it moves out from the current coverage area without compromising service fruition.

A SNA servant runs on each mobile device looking for LDISs. It is a *seamless* servant, i.e., it provides service without being associated to any specified LDIS; when a service is discovered it can be provided to the user immediately. The servant can offer as many services as have been detected. Through reflection the servant is *aware* of the host system and of the interface provided by LDISs.

Benefits of using a SNA servant are:

1. Clients do not depend on the type and on the implementation of the service they need.
2. LDIS servers do not change their structure/implementation in order to be used in conjunction with SNA servant (they do not need any specific stub or similar technology).
3. Services brokered by an SNA servant can dynamically change; the code necessary to accommodate the change will be downloaded at run-time from the server.
4. Automatic roaming/routing to a needed service.
5. Client request would be dynamically adapted to the service interface and transparently forwarded to the right server. This “application level routing” will be implemented from the infrastructure, needing no cooperation from the network provider (see [8]).

6. Implementation of stateless clients – status of mobile SNA will be kept between connections to different LDISs providing the same service.
7. Connection will be bi-directional :

    - A SNA servant is able to connect to a LDIS to request a service, using a *pull* service model, this approach allows the SNA servants to interact with already existing LDISs and not specially developed for interacting with any particular SNA servant.
    - A LDIS is able to do the reverse and interact with an idle SNA servant; *push* service model, this approach allows the LDISs to efficiently broadcast their services to every SNA servant in their area of coverage when they agree on a common interface.

8. Communication is text based; operations are interpreted as code on the receiving node, this will require lesser bandwidth than many other device-independent technologies.

Implementation technology, especially regarding point 3, is not an issue if we accept to implement each SNA in various technologies accordingly with target platforms requirements (.NET, Java), or if we choose to adopt bytecode adaptation techniques, like [4] or [5] for mixed platforms.

Application Scenarios

Following are some environments that could benefit from our framework.

Location Dependent Services

A compass is an example of this kind of application: an LDIS, providing geographic location based service, will be part of the network infrastructure and knows the position of each agent, an SNA servant on the mobile client will discover this service, create a compass object on the local device and move the needle accordingly to information provided by the LDIS. There is no need for LDIS to know if the compass is text based or graphical.

Other possible services may be data collection or dispensation along a route or visualization of content related to a certain location (e.g. infrastructure monitoring and tourist information).

Remote Systems and Applications Management

Remote management is suffering from the new trend in operating systems - heavy graphical user interfaces. The difficulty of remote management of Microsoft's Windows operating system is a case in point. Even if several remote management tools such as PCAnywhere, ControlIT and VNC, are available on the market, broadband connection and graphical clients are still needed.

As an example the remote management of Internet Information Server (IIS) is a much-desired functionality. Instead of directly logging onto the console it is desired to be able to telnet and connect to the IIS and some utility like SNA servant acting as a stub of the IIS COM component. Making use of this stub to be able to debug, shutdown, restart and control the behavior of the IIS.
Please note that we don't restrict our work to Microsoft Windows and COM architectures. SNA servants are rendered completely platform independent by using Java as host language and CORBA/SOAP as messaging infrastructure; even if we are likely to experience performance degradations for the client and server grows in term of complexity. SNA servants realized mixing Java and C# (by using [4] or [5]) are also under examination in order to transparently interact with CORBA-based as well as .NET-based LDISs.

Graphical Applications "on the fly"

Typically, graphics-oriented applications render data content on a screen. Major problems here are: (*i*) need to transfer a huge amount of data, and (*ii*) need for the data to be adapted to the display of the end-station.
If we disregard a data-oriented approach, we can model a graphical content as a collection of two- or three-dimensional objects. Regardless the number of objects present in the collection, usually they can be classified as belonging to a limited number of classes (point, line, rectangle, box, etc…) and differ by size, placement and orientation.

The code related to a certain shape can be fetched by a SNA servant only once, at startup or on demand. A remote LDIS will be able to connect, create and manipulate as many instances of graphical objects as needed to represent the data. The local SNA servant will connect to a LDIS and represent the data just by asking “what kind of shape it has to draw next”.

Many advantages will accrue from this approach:

1. Objects will draw themselves on the end-station hardware; LDIS does not need to care about their visual properties;
2. Changes in the logic of a service (or the service provider itself) never involves changes in the end-station objects; and
3. Clients may choose to get partial or reduced-cost services by not implementing/accepting some object types.

Competing Technologies in the Market

Other technologies offering solutions for wireless services provisioning can be found on the market.
Modern service provisioning models features small vendors providing independent and modularized service components. Those modularized reusable software components are called Web services. Wireless Web services applications use a wireless front-end for pervasive human interfaces and Web services on the back end to connect to Internet resources.
Web services belonging to different service providers must interoperate in order to provide transparent services to customers.
Distributed computing architectures have long been able operate using remote procedure calls (RPC) and remote object-oriented frameworks (CORBA). Building a large scale interoperable network requires, in addition, standardization on communication protocols. We have XML-based protocols to standardize web services' dynamic discovery processes like (UDDI); service interfaces (WSDL); and asynchronous and synchronous messaging (SOAP and XML-RPC).

Wireless Portal Network

In a wireless portal network, devices connect to classic Internet services trough entry points adopting a “portal” philosophy. Portals create a *protected zone* providing (controlling) access to outside contents. Wireless portal networks support widely deployed thin-client wireless technology, such as WAP (Wireless Application Protocol).
A WAP-based services portal will receive requests embedded in WML (Wireless Markup Language) forms, after checking user’s privileges it translates the request in a SOAP/XML-RPC message that will be routed to a partner web service. On the way back the portal will take care to translate the response in to a WML document again. In this way, the portal works as a proxy for wireless users. The portal operator provides user authorization and management services.
Compared to our approach a thin client-based portal network requires usage of pre-defined partner services and adopts an asymmetrical computation model, for only server-side computers will handle application functionalities. Furthermore, the portal architecture requires a central entity to control all user data. Back-end service providers must rely on the portal architecture to access users.

Wireless Extended Internet

Wireless extended Internet is the wired Internet's expansion to wireless devices. Wireless information devices can get their own IP addresses (through Internet Protocol 6) and full network functionalities. Those devices usually run smart, fat clients that interact with multiple backend services simultaneously and store/process application data on the device.
Like the Internet itself, this architecture is decentralized (i.e. without a controlling entity), however, centralized Web services hubs are still required. Unlike the portal architecture, the hubs themselves can be decentralized: different service providers can provide similar hub services that can interoperate with each other.
Compared to our approach extending the Internet to wireless require each client to know how to interact with the various interfaces of offered web services and imply the knowledge about where a service is located.

Prototype - Implementation Platform

A prototype has already been implemented using the Microsoft .NET framework and C#.

Class metadata is exploited for service self-description. Reflection [6] has been used to make object properties available at runtime with no restrictions to classes explicitly designed for the framework (we are not imposing any interface to implement for an object to be managed). Custom attributes will allow us to insert PGP-like keys in metadata in order to obtain service signing.

A host can use the ordinary telnet application to connect to a remote SNA, and then, using a C#-like syntax it is possible to:

- inspect the object pool;
- create new objects;
- perform operations on contained objects.

Where objects can be instances of .NET framework classes (custom made) as well as COM stubs (excel, word or explorer).

The next steps will be implementation of the broking system, messaging support architecture and service discovery. Finally we are exploring the implementation of an hybrid SNA servant able of interacting with CORBA-based, Java-based and .NET-based LDISs.

Schedule for completion of demonstration

A demonstration based on the above-described prototype may be available by summer 2003.

References

[1] Dik Lun Lee, Wang-Chien Lee, Jianliang Xu, and Baihua Zheng. Data Management in Location-Dependent Information Services. *IEEE Pervasive Computing*, 1(3):65–72, 2002.

[2] Keith Cheverst, Nigel Davies, Keith Mitchell, Adrian Friday, Experiences of Developing and Deploying a Context-Aware Tourist Guide: The Lancaster Guide Project, in: Proceedings of the 6th Annual International Conference on Mobile Computing and Networking (Mobicom 2000), ACM Press, New York, USA, 2000, pp. 20-31.

[3] Massimo Ancona, Walter Cazzola, and Daniele D'Agostino. Smart Data Caching in Archeological Wireless Applications: the PAST Solution. In Proceedings of the 11th Euromicro Conference on Parallel, Distributed and Network-Based Processing (Euromicro PDP 2003), Genova, Italy, February 2003, pp. 532-536. IEEE Computer Society Press.

[4] Jndridge (http://www.jnbridge.com/) : True interoperability between Java and .NET

[5] Stryon inc. (http://www.stryon.com/): Byte code conversion between java and .NET.

[6] Walter Cazzola. Evaluation of Object-Oriented Reflective Models. In Proceedings of ECOOP Workshop on Reflective Object-Oriented Programming and Systems (EWROOPS'98). LNCS 1543. July 1998.

[7] SUN Microsystems. Java Core Reflection API and Specification. Technical report, SUN Microsystems, February 1997.

[8] Fabio Kon, Fábio Costa, Gordon Blair and Roy H. Campbell, The Case for Reflective Middleware. In Communication of ACM 45(6):33-38. June 2002.

[9] .NET Reflection Overview - http://msdn.microsoft.com/library/en-us/cpguide/html/cpconreflectionoverview.asp . MSDN on-line documentation.