INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH & DEVELOPMENT

ISSN: 2278 – 0211 (Online)

Jini Surrogate Architecture: Middleware Platform For Social Networking

Prof. Prashant P. Rewagag
Head,Department Of CSE,G.H.R.I.E.M,Jalgaon, N.M.U. Jalgaon
Harish Prakash Patil
Department Of CSE,G.H.R.I.E.M,Jalgaon, N.M.U. Jalgaon

Abstract:

This paper deals with various issues which arise in case of smart device use for social networking. In recent years both the popularity of social networking applications and the adoption of mobile devices, notably smart phones, are growing very fast. The current generation of smart phones is pocket computers that, compared to their predecessors, are relatively well resourced. This paper deals with the study of various issues for connecting different devices independent of their architecture over a single network. The concept "surrogate" plays important role for such connectivity.

Keywords: *surrogate, social networking, Jini architecture, java platform, etc.*

1.Introduction

In this paper, we are concerned with describing a network middleware that allows applications of a social networking style to run on smart phones. Without additional infrastructure, smart phone features are redundant since users fall back on conventional calling and texting. While calling offers real-time communication, it is necessarily synchronous - requiring participation of two users at the same time. Both calling and texting offer limited forms of communication that do not exploit

the potential of smart-phones as pocket computers.

The paper is organised into seven sections. of the The second section deals with social networking and

Potential of mobile services, the background with the motivations and challenges is incuded in third section. The forth section overviews the Jini network technology. The related work is included in fifth chapter. The sixth section consist of implementation of network connection. The summery is included in seventh section.

2. Social Networking And Potential Of Mobile Services

Social networking is concerned with building online communities of people engaged in common interests and activities. Well-known social network services, for example Facebook, Flickr and Twitter, tend to be Web-based and accessed using desktop technology. Using such services, people have focus on is user location with friends and increasingly the world at large. Mobile devices are becoming increasingly integrated with our daily lives. Many of today's mid-range mobile devices are equipped with features like cameras, GPS, and multiple network interfaces. Such emerging platforms offer a rich set of resources for hosting new mobile applications.

Intermediary-based infrastructure [1]-[4] has emerged to help facilitate mobile service provisioning by ensuring reachability of mobile services by their clients. We have realised such an intermediary using middleware that is applicable to a broad range of application domains, supporting service delivery to businesses and industries such as healthcare, journalism and logistics [5]. At present, however, the majority of applications running on mobile devices act as service requesters that consume services - such as news and weather forecasts - provided by servers connected to fixed network infrastructures. Use of mobile devices for hosting services, and hence acting as service providers, is an emerging area of research. Mobile devices have the advantage of utilising mobile- specific features, such as connectivity with auxiliary devices and knowledge of location, to provide new kinds of

services. They lend to become much more open to sharing information .

Sr	Applications	Details Issues	
1	Google Latitude	Website	offer support for so social networking features
2	Loopt	US-based mobile application	
3	Whrl	based on a whereabouts Journal based on a whereabouts journal	lack mechanisms for propagating updates and the type of context information

Table 1: Overview of existing applications for Social networking

Google Latitude is a that provides users with a Google Maps view that is annotated with markers. A marker indicates the last known location of a user's friend. Loopt allows users to see friends' locations on a map or as a list. Loopt maintains a diary-like "whereabouts" journal [1] that tracks users' locations over time. Loopt also has features for synchronising users' locations with Facebook and Twitter With Whrrl, all journal entries are manual; users can set their location, take a picture, write a note, and set a privacy level. Whrrl allows context data other than location to be made available.

3.Background: Motivation And Challenges

Widespread use of mobile devices. Smart phones have as much processing power as a PC of a few years ago. Smart phones can also be connected to auxilari devices using e.g. Bluetooth Multithoming- auxiliary devices using, e.g. Bluetooth.

the ability for a device to communicate using multiple network interfaces. Interest in social networking and smart phones has largely been orthogonal.

The motivation for middleware platform lies in various social networking issues such as Mobile media, Patient monitoring, and Social Networking. The Mobile media is tagged with Context data which can be either accessed from anywhere by the consumer or can be Synchronized with external repository. In case of social media friends can stay connected with each other by this networking mechanism. The various challenges that middleware platform has to face includes mobility, rechability, scalability, availability.

4.Jini: Network Technology Based On Java Platform

Jini technology is designed to provide simple architecture for devices to deliver services in a network and make themselves available for use. It enables "plug and play" network. Jinni technology will run on any network with at least one Java Virtual Machine (JVM). For those devices which neither support JVM nor have sufficient memory for Jini implementation, third party is introduced.

Jini offers a programming model that leverages Java and extends it to address the "eight fallacies of

distributed computing". The eight fallacies are:

- The network is reliable
- Latency is zero
- Bandwidth is infinite The network is secure
- Topology doesn't change
- There is one administrator
- Transport cost is zero

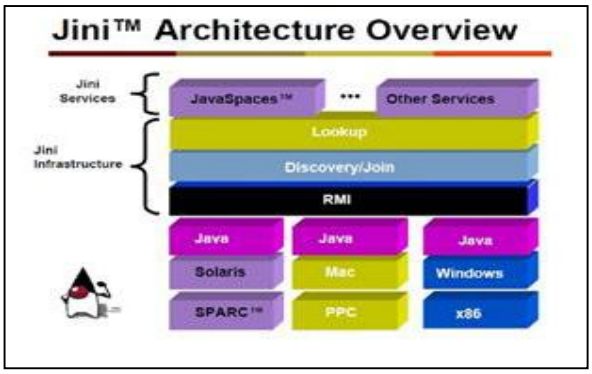

Figure 1: Jini architecture overview

	Infrastructure	*Programming Model*	*Services*
Jini	Discovery Lookup Extended Security	Lease Event Transaction	JavaSpaces TX Manager Other Services
Java	Java VM RMI Security	Java technology-based APIs Beans Swing	Enterprise JavaBeans™ Java Naming and Directory Interface™ JTS

Table 2: Java platform extension by Jini

The network is homogeneous using the JSA (figure 2), a device-service, written in any convenient programming language, can be hosted n a device and exposed to the federation as a regular Jini service. A surrogate host (SH) plays the role of the intermediary and may contain many surrogate (proxy) entities, each representing a particular device-service. A device-service sends its surrogate to a SH as part of the surrogate registration protocol.

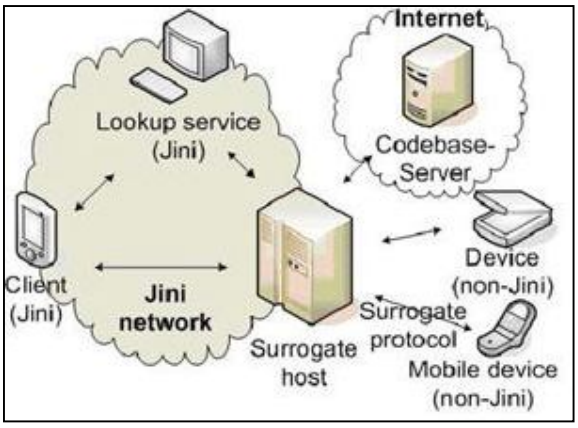

Figure 2: The Jini Technology Surrogate

Once the SH activates the surrogate, it registers a service-object with Jini's decentralised lookup service (LUS). Clients discover the service-object and use this invoke the remote service; requests are actually sent to the surrogate, which may, depending on the request, communicate with the device service.

The Jini Surrogate Architecture (JSA) specification arose to allow devices that cannot run Jini to expose their services to Jini clients. Interconnect protocol includes liveness provisioning. Whatever the protocol is, there must be a way for the surrogate to know whether its device is up-and- running.

5.Related Work

Most of the work related to the Arches project implements some flavor of an intermediary-based architecture. In each case, an intermediary software module assists a thin-client in accessing thick services. There are nevertheless, several interesting variations to this idea. Work such as RPS[19] and JiniCard[13] assumes that the enterprise system is Jini-enabled. ACTS[14] below, AASE[15], and [16] assume the end-client resembles a mobile phone, but do not assume the network service to be Jini-enabled. ACTS and AMASE further assume that the objects implementing the service are autonomous mobile agents that have a veneer of

intelligence to them.

The most interesting related project from an overall goal perspective is [16]. Similarly to Arches, [16] aims to provide complex data services over a cell phone; however, it assumes that the cell phone supports the GSM short message service (SMS) API. SMS is used as a textual mechanism to implement remote procedure calls to enterprise services. The use of a document transport mechanism for transporting programmatic events

resembles the use of SOAP[17] for remote service invocations. The client-server architecture resembles the web, in that specific enterprise application handlers resemble servlets that are invoked by a small, simple set of methods. This scheme takes advantage of some nifty SMS features, such as the ability to define phone shortcuts for frequently invoked applications. The architecture suffers from the underlying limitations of SMS as a "middleware" and from security and scalability limitations.

SMS messages are inherently limited to 160 characters, and requests and responses fit within an SMS message boundary. This limits the nature of interactions allowed in the supported applications.

Finally, the SMS security, sharing and billing schemes don't quite fit the model of its use as a message bus. The paper goes into some detail about the amount of invention it takes to scale this architecture to large numbers of users. RPS and JiniCard resemble Arches in that they assume the enterprise service being targeted to be in a Jini network. RPS proposes a model very similar to the Jini Surrogate architecture, where an intermediary supports the execution of downloadable "handlers" for each enterprise service that needs to be made available to a thin client. Unlike the Jini Surrogate which is interconnect agnostic, RPS proposes a specific protocol (similar to DHCP) and inter- connect architecture for devices to discover intermediaries and connect to them. RPS does not presuppose J2ME enabled devices as Arches does, but therefore supports proportionally less dynamic services.

JiniCard provides a mechanism for a JavaCard to advertise its services to a Jini network. This would allow enterprise applications to access data and applets resident on the smartcard. While Arches focuses on making network services available to the thin-client, JavaCard does the opposite. The JavaCard and RPS intermediaries were designed before Jini Surrogate came into existence, and it is fairly straightforward to map their concepts on to the Jini Surrogate. Data Lockers[18] proposes a protocol and middleware independent approach to provide networked storage and caching for mobile devices. The premise being that mobile devices performing substantial Internet surfing do not have the space to store the resultant data, or the time to wait for the request to be processed. Again the lockers architecture is a

specific instantiation of the Surrogate architecture for storage and caching services. It is unclear why the components of the Lockers architecture that handle storage
couldn't be generalized to do other tasks (as the Surrogate architecture proposes).

ACTS and AMASE propose a mobile, somewhat intelligent, agent architecture over wireless protocols to support adaptive applications that can operate in an intermittently connected environment. They both presume that there are gateways in the wired-wireless network edge that support agent computation as well as content distillation when needed. While the notion of content distillation at the intermediary is interesting, these architectures presuppose the ability to host agents on cell phones, a capability that is beyond that of the capabilities of the current generation of cell phones. It is also unclear whether or when phones will have the capability to support full-blown agents.

6. Implementation

6.1. Discovery Using IP Interconnect

Discovery is the process by which a device and a surrogate host discover each other over an interconnect. In it's simplest form, the device can contact the surrogate host using a known address. More useful is dynamic discovery where the host and the device both use IP multicast. Figure 3. Discovery using an IP interconnect, shows how dynamic discovery works.

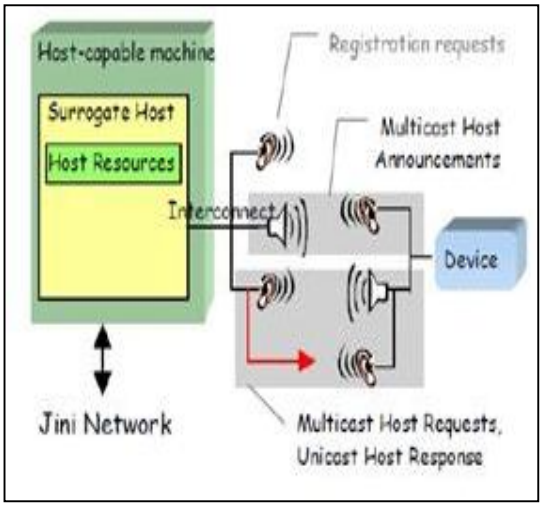

Figure 3: Discovery using an IP interconnect

There are two dynamic discovery protocols, the multicast host announcement protocol and the multicast host request protocol. The multicast host announcement protocol is initiated by the surrogate host which announces its presence on an interconnect by periodically sending IP multicast packets containing its registration request address. When devices establish contact with an interconnect, they begin listening on the host announcement multicast group address and eventually receive surrogate host announcements. The multicast host request protocol is initiated by the device and is run concurrent with the multicast host announcement protocol. When the device establishes contact with an interconnect, it starts sending periodic multicast host requests containing a host response address.

When the surrogate host started, it began listening on the host request multicast group address. When

it receives a request, it immediately sends its registration request address to the device's host response address. As these two protocols run concurrently, either one may result in delivery of the surrogate host's registration request address to the device. Once the device has acquired the registration request address, it can register its surrogate.

6.2. Registration Using IP Interconnect

The registration protocol loads a device's surrogate onto a surrogate host. The device registers using the registration request address and one of two ports, one for TCP, and the other for UDP. It is up to the device to choose which protocol to use. Figure 4. Surrogate registration, shows how the device registers its surrogate with the surrogate host.

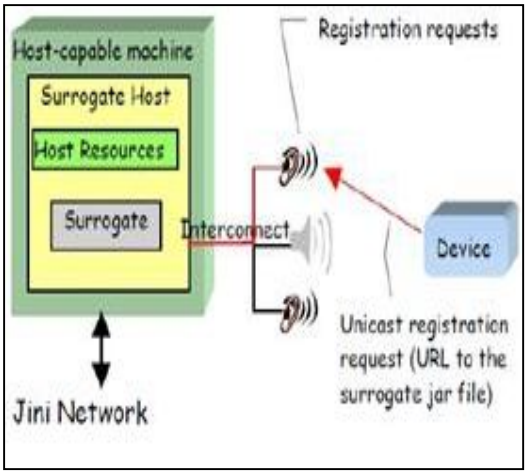

Figure 4: Surrogate registration

The device sends a unicast registration request directly to the surrogate host's registration request address. The request contains the surrogate and the initialization data for the surrogate. The surrogate can be either the bytes of a jar file or a URL specifying the location of the jar file. The surrogate host then converts the stream of bytes into a usable jar file or, in the case of a URL, establishes a connection to the jar file over the network. Either way, the surrogate host establishes a class loader for the surrogate that is responsible for serving class files while isolating the surrogate from other surrogates running on the same host.

6.3.Surrogate Execution

Once a surrogate jar file is loaded, the surrogate host will attempt to activate its surrogate. The surrogate host first examines the surrogate jar's manifest file to determine the class in the jar file that implements the Surrogate interface. This interface enables the surrogate host to activate and deactivate the surrogate. The surrogate host also scans the manifest for any export resources contained in the surrogate jar file. These resources must be made available to clients of the surrogate via the export server. The surrogate host extracts the resources from the jar file and passes them to the export server that generates a unique URL for each resource. These URLs are used to annotate the surrogate's classloader. This way, any object instantiated via this classloader (such as a service proxy), will carry these annotations with it thus allowing them to be loaded. The surrogate host then instantiates a surrogate via its classloader. Finally the surrogate's activate method is called to allocate surrogate resources and start any threads required by the surrogate. Figure 5. A surrogate after activation, shows the state after the activate method has been successfully invoked.

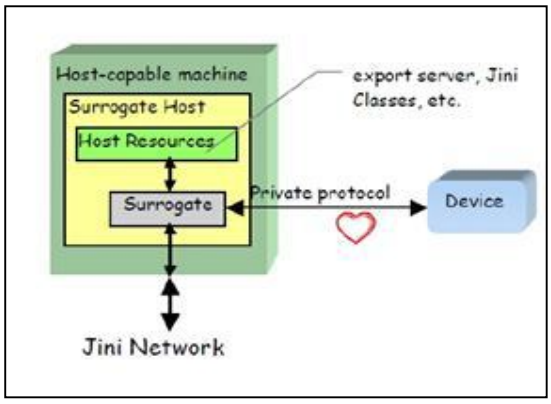

Figure 5: A surrogate after activation

The surrogate is now running within the context of the surrogate host. It has a connection to the Jini network and to various host-local resources, and most importantly, it has a relationship with the interconnect back to the device for which it is the surrogate. The specification defines a means for the surrogate to monitor liveness of the device connection in a way that is independent of the surrogate host implementation

7.Summary

7.1. Java Technology Features that Benefit Jini™ as below:

Feature	Benefit
JVM	Homogeneous network
Portable object code	Architecture independence
Downloadable code	Dynamic environment
Unified type system	No impedance mismatch

7.2.Applications

Plug-and-Play Printer shows how a device can
provide a service; this is the popular way of using
the surrogate architecture

- Urgent Email Service shows a cell phone as a Client
- Tracking the Mobile User shows the Integrating mobile agents.

7.3. Surrogate Life-Cycle Phases

- Discovery: Between surrogate host and device
- Retrieval and loading of surrogate: Surrogate gets retrieved and loaded into execution environment .
- Execution: Surrogate participates in the activities of Jin federation on behalf of a device
- Liveness monitoring: Connectivity between surrogate and device gets monitored

8.Reference

1. V. Halteren and P. Pawar, "Mobile Service Platform: A Middleware for Nomadic

Mobile Service Provisioning," in Proc IEEE International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob' 06). Montreal, Que: IEEE Computer Society, June 2006, pp. 292-299.

2. S. N. Srirama, "Publishing and Discovery of Mobile Web Services in Peer to Peer Networks," in Proc. 1st International Workshop on Mobile services and Personalized Environments (MSPE' 06). Gesellschaft fur Informatik, November 2006, pp. 99-112.
3. Radovanovi, A. Ray, J. Lukkien, and M. Chaudron, "Facilitating Mobile Service Provisioning in IP Multimedia Subsystem (IMS) Using Service Oriented Architecture," in Proc. 5th International conference on Service-Oriented Computing. Springer-Verlag, 2007, pp. 383-390.
4. J. Wikman and F. Dosa, "Providing HTTP Access to Web Servers Running on Mobile Phones," Nokia Research Center, Technical Report, May 2006.
5. T. Weerasinghe and I. Warren, "Towards Mobile Service Provisioning," in New Zealand Computer Science Research Student Conference. Auckland, New Zealand: University of Auckland, April 2009.
6. N. Bicocchi, G. Castelli, M. Mamei, A. Rosi, and F. Zambonelli, "Supporting location-aware services for mobile users with the whereabouts diary," in MOBILWARE '08: Proceedings of the 1st international conference on MOBILe Wireless MiddleWARE, Operating Systems, and Applications. ICST, 2007, pp. 1-6.
7. A. Gupta, A. Kalra, D. Boston, and C. Borcea, "Mobisoc: a middleware for mobile social computing applications," Mob. Netw. Appl., vol. 14, no. 1, pp. 35-52, 2009
8. A. Acharya, N. Banerjee, D. Chakraborty, K. Dasgupta, A. Misra, S. Sharma, X. Wang, and C. P. Wright, "Programmable presence virtualization for next-generation context- based applications," Pervasive Computing and Communications, IEEE International Conference on, vol. 0, pp. 1-10, 2009.
9. Jini Technology Surrogate Architecture Specification, Sun Microsystems Std., Rev. 1.0, 2001. [Online]. Available https://surrogate.dev.java.net/doc/sa.pdf
10. J. Wikman and F. Dosa, "Providing HTTP Access to Web Servers Running on Mobile Phones," Nokia Research Center, Technical Report, May 2006.
11. E. Ferro and F. Potorti, "Bluetooth and Wi-Fi wireless protocols: A survey and a comparison," IEEE Wireless Communications, vol. 12, no. 1, pp. 12-26, 2005.
12. S. N. Srirama, M. Jarke, and W. Prinz, MWSMF: a mediation framework realizing scalable mobile web service provisioning," in MOBILWARE '08: Proceedings of the 1st international conference on MOBILe Wireless MiddleWARE, Operating

Systems, and Applications. ICST, 2007, pp. 1-7.

13. Kehr, R., Rohs, M., Vogt, H., "Mobile Code as an Enabling Technology for Service-oriented Smartcard Middleware," http://citeseer.nj.nec.com/kehr00mobile.html.
14. Baumgarten, H., Borrmann, L., Köhler, T., Pink, S., Lacoste, G., "Middleware for a New Generation of Mobile Networks: The ACTS On The Move Project,"http://www.isoc.org/isoc/whatis/conferences/in et/96/proceedings/a6/a6_3.htm.
15. Pascotto, R., "AMASE: Agent-based Mobile Access to Information Services," T-Nova Deutsche Telekom Innovationsgesellschaft mbH Berkom,
16. http://www.infowin.org/ACTS/ANALYS YS/PRODUCTS/THEMATIC/AGENTS/ch3/a mase.htm.
17. Stajano, F., Jones, A., "The Thinnest of Clients: Controlling It All Via Cell phone," Mobile Computing and Communications Review, Vol. 2, Number 4.
18. W3C, "Simple Object Access Protocol SOAP) 1.1", http://www.w3.org/TR/SOAP/.
19. Villate, Y., Pitoura, E., Illarramendi, A., lmagarmid, A., "Extending the data services of mobile computers by external data lockers," Proceedings of the Third International Workshop on Mobility in Databases and Distributed Systems, IEEE Computer Society Press, September 200