A JINI Based Framework for Nomadic Computing

Jinhua Guo and Sudha Ramaswamy
Department of Computer and Information Science
University of Michigan-Dearborn
4901 Evergreen Rd, Dearborn, MI 48128
{jinhua, sudhar}@engin.umd.umich.edu

Guangming Xing
Department of Computer Science
Western Kentucky University
Bowling Green, KY 42101
Guangming.Xing@wku.edu

Keywords: Mobile agent, nomadic computing, telematics, middleware

Abstract
The growth of relevant telematics technology has enabled OEM companies to equip today’s vehicles with embedded computing platforms and advance wireless technology. As a result, general purpose telematics applications such as automatic parking space search, automated toll collection and collaborative collision avoidance are highly promising. However, the deployment of such telematics application on a rapid and dependable scale over nomadic computing and ad-hoc networks is proving to be extremely difficult. In this paper, the authors introduce a new framework that integrates JINI technology with mobile agents to enable aforementioned telematics applications to meet those deployment challenges.

1. INTRODUCTION
The potent combination of powerful mobile computers and advanced wireless communication technology has enabled a new computing environment called ‘nomadic computing’ [1, 2]. Today’s Users are considered ‘nomads’ because they carry mobile computers and communication devices when they travel between offices, hotels, airplanes and automobiles. The advent of advanced wireless communication technologies has provided with improved connectivity for such mobile computers (laptops, PDA’s or phone organizers). Furthermore, application-level protocols have allowed the proliferation of convenient applications on these devices. In summary, these developments undoubtedly propel us towards the widely expressed goal of accessing and processing information almost ‘anywhere and anytime’.

Though the capability of such nomadic computing technologies is unquestioned, mobility introduces a key technical challenge into nomadic-computing because the available resources may vary widely and unpredictably. Resources such as communication-bandwidth and error-rates can change dynamically within a wireless communication network. Also, a mobile system’s battery may deteriorate over time or such mobile devices may be temporarily switched off or unreachable due to network partitions. Apart from the above bottlenecks, one other nomadic computing issue, location-awareness, requires that mobile computer applications know their operating environment at all times for providing context-dependent activities. For instance, context-dependent activities may involve giving directions or employing stringent security mechanisms.

Existing client-server middleware is not capable of addressing the aforementioned issues in a mobile setting [4]. By imposing a tight coupling between clients and servers, and also relying heavily on the permanent availability of the server, a client-server middleware does not accommodate well with the flexibility imposed by the new distributed scenarios. In addition, the existing client-server middleware is impractical in a nomadic scenario where physical and logical structure of the network is made extremely fluid by connectivity changes. JINI [3] is a new technology that is fast becoming the de-facto paradigm of dynamic distributed systems. JINI technology has demonstrated service focus for abstracting hardware-software distinctions, the ability to support partial failures, the ability to abstract distributed computing protocols besides being completely self-configurable and self-manageable. With many of its inherent advantages, JINI, which is essentially a client-server middleware, has the technology to achieve the flexibility that is required of distributed computing scenarios.

Combining JINI with mobile agents presents a new significance in nomadic computing environment. Unlike existing client-server middleware, which needs permanent connectivity to the server, mobile agents do not require continuous network connectivity. Rather, the connection is required to last only for the duration of time needed to inject an agent from mobile clients to the fixed network and vice-versa. Mobile agents are autonomous and can carry forward services even after the originating user or device disconnects. These mobile agents can then deliver results when the user or device reconnects to the network. Though very significant for the nomadic computing environment, mobile agent system faces tremendous challenges due to the
absence of applications, security framework, infrastructure and uniform standards.

Even with such challenges, combining JINI technology and mobile agents presents enormous benefits to nomadic computing. In this paper, the authors introduce a new framework that integrates JINI technology and mobile agents to enable a whole array of operations such as support for disconnected-operations, increasing availability of services, improving host and user mobility, providing dynamic service discovery and demonstrating location awareness in nomadic computing environments.

The remainder of this paper is organized as follows. Section 2 describes the salient features of mobile agents and the JINI technology. Section 3 discusses the three service paradigms: client-server model, service-on-demand model and mobile agent-based model. Section 4 takes a detailed look at the proposed mobile agent based framework and its implementation. Section 5 presents an application scenario in the nomadic computing environment based on the proposed mobile agent based framework. Section 6 presents related work and, finally, Section 7 concludes and outlines future work.

2. MOBILE AGENTS AND JINI

2.1. Mobile Agents

A mobile agent is a running program that can move between various hosts within a network when and where it chooses. Mobile agents offer many potential advantages over traditional approaches. By moving the computation to another host, it is often possible to collate the computation with an important database allowing high throughput and low latency access. Compared to more traditional client-server models, mobile agent model can avoid transmitting large amounts of data across the network, which is of enormous value when the network is slow or unreliable. A mobile agent can move from one server to another, with partial results, until it has accomplished its task before returning to the originating host.

In addition to improvements in speed and reliability, mobile agents can also help bring structure to the distributed applications. For instance, a service model designed to relocate itself within a network due to changing network conditions or due to changing locations of its clients can be easily implemented as a mobile agent [5].

There are many existing mobile agent systems, but few, if any, will fully meet the needs of those large and complex applications. In fact, it is very unrealistic to build large and complex applications that are purely based on the agent technology. It is the authors view that mobile agent is just another design tool that must be available to the programmer in combination with other tools to design and develop mobile applications.

2.2. JINI

JINI [3] is a completely self-diagnosing and self-configurable distributed network architecture that can outlive its services nodes. It allows services to dynamically join or leave with no impact on the network users. It also attempts to overcome the common pitfalls in distributed computing.

JINI is a java-based solution. Using the Java programming language, JINI solves most programming model and language issues involved with distributed computing. Java is more than a programming language – it is a platform. By adopting the Java Virtual Machine (JVM), Java overcomes the issues related to heterogeneity in machine platforms and operating systems. Thus, any application can run on any platform or operating system without the need for building platform specific applications.

JINI has provided a new infrastructure built from the ground up for existing distributed architecture. JINI provides a new set of protocols – discovery and join – to enable spontaneous networking of services. It also provides a kind of naming service called a lookup service, through which a generic service may register and advertise its availability to others. The service providers are allowed to advertise their specialty or uniqueness in the form of attributes. Furthermore, JINI provides a distributed event mechanism through which a service requester can register interest in a service provider. The service requester will be notified when the service becomes available, without the need for polling. Moreover, JINI handles partial failures using a leasing mechanism. In this environment, leasing enables the service provider to be listed on the lookup service for a specified period of time, beyond which the service provider is required to renegotiate a new lease or stand removed from the lookup service. This type of mechanism enables the formation of a dynamic network through which a service provider can exit and enter without bringing down other systems.

In summary, JINI addresses the pervasive, ubiquitous and dynamic distributed computing requirements within a single architecture. As a result, JINI technology is becoming the de facto paradigm of dynamic distributed systems.

2.3. Integrating Mobile Agent with JINI

Architecture

As discussed in previous sections, combining JINI with mobile agents presents a new significance in nomadic computing environment. Integrating JINI architecture with mobile agents provides a new framework that enables a whole array of operations such as support for disconnected-operations, increasing availability of services, improving host and user mobility, providing dynamic service discovery and demonstrating location awareness in nomadic computing environments. In the following sections, the
authors introduce the proposed framework and discuss in detail its significance in the nomadic computing environment.

3. THE EXTENDED JINI SERVICE MODEL

The basic concept of JINI involves clients communicating with services through the use of proxies downloaded from JINI lookup service. This concept is akin to a traditional client-server model. However, the challenges posed by constraints of mobility and frequent disconnections require us to rethink this model. For instance, a service may need to relocate itself to mobile devices to accommodate the uncertain connectivity. On the contrary, a client may be written as a mobile agent and allowed to migrate to the server side for purposes of performance and availability. These new models are presented in Fig. 1.

3.1 Traditional Client-Server Model

In other distributed computing technologies, such as the common object request broker architecture (CORBA) and the Distributed Component Object Model (DCOM), a client side stub and server-side skeleton support communications between the client-entity that requests a particular service and the server entity that provides the particular service. In the case of JINI technology, this stub is the service proxy object that the server uploads to the lookup service. Hence, the server implementation can be altered automatically and transparently to the client. This concept of downloading the code at runtime offers JINI significant advantages over CORBA and DCOM. This concept is similar to the dynamic binding in object-oriented programming.

3.2 Service Code-on-Demand Model

In this model, a client contacts a server to download a class. Once the class is downloaded, the client creates an instance and a thread to run the code. This service code-on-demand can be easily implemented using JINI technology. During service registrations, service providers may upload the entire object directly to the lookup service rather than to the proxies. The clients query the lookup service(s) for particular services that might be of interest and download the service object. The clients may invoke methods or call different functions offered by the service locally. The service can then be continuously accessed even when the clients are disconnected from the nomadic computing network. Such a model provides interactive services to the users while they are still disconnected from network. This type of service model is particularly useful for small and interactive services such as tour guides and local navigations.

3.3 Mobile-agent based Model

The mobile agent idea has its origin in process migration though one doesn’t see operating systems supporting it. While Java provides a whole host of features to build agent-based systems, perhaps the most significant feature is the recently released JINI connectivity technology.

In the mobile agent-based model, a client is written as a mobile agent. The mobile agent can move within a network from one server to another, with partial results, until the client has accomplished its task. In this model, the JINI service provides a special proxy during the registration process. When the service method is invoked, the service method automatically moves the agent from the client to the server and creates a new thread to activate the agent. The agent runs on the server side until the work is completed. After that, the agent may continue to move to other services or come back to the originating mobile host. This service model improves both the performance and availability of available services. Also, such a model is particularly useful especially in a slow and unreliable network or when the final answer is much smaller than the amount of information needed to be accessed. In the next section, we take a closer look at the details of the mobile agent-based framework and its implementation.

![Figure 1. Three Types of Service Models](image-url)
4. MOBILE AGENT-BASED FRAMEWORK AND ITS IMPLEMENTATION

The mobile agent-based framework consists of three main components. The first component comprises the mobile agents, i.e., entities that are expected to perform some tasks. The second component is the mobile agent host(s), i.e., the service that provides the platform for the mobile agents to execute. The third and final component is made up of lookup and location services to provide dynamic service discovery and location updates. To be an active agent platform, a given node in the system must have at least one active agent host. The components of the framework are illustrated in Figure 2. The figure describes a typical sequence of interactions among different components within the framework.

4.1 Agent Host

Both mobile and static nodes can be active agent platforms as long as those nodes have at least one active agent host. The agent host implements one method, acceptAgent(), which the agents call to travel to the implementing agent host. The agent host then binds an incoming agent to a thread.

4.2 Mobile Agent

An agent is a serializable java object. It has a doWork() method that is called when an agent arrives on a given host. The agent uses the lookup service, if and when it decides, to look for new service providers such as when the agent wants to travel to a new agent host. Similarly, the agent uses the location service to locate the originating host when it accomplishes its task, before returning to its originating host.

4.3 Lookup Service

In general, portable devices usually move among localities in unpredictable ways without any static knowledge of the locally available resource and services. Therefore, mobile computing platforms should support the dynamic discovery of resources and services by imposing limited knowledge on the client side. This is usually referred to as service discovery.

In JINI, the lookup service is one of the core infrastructure interfaces within the JINI architecture. It provides the mechanism for devices or services or users to discover or join or detach from a JINI community. In the agent-based framework, mobile agents use lookup services to dynamically look for service providers or mobile hosts or other mobile agents. In addition, using the lookup service, mobile agents can download agent proxies that allow agents to migrate from host-to-host.

4.4 Location Service and Location Awareness

In nomadic computing, it is imperative to trace mobile entities, which include users, hosts and agents. The location services keep track of mobile hosts, users and agents. In the proposed framework, a location service is basically an integration of the lookup service and the leasing service. In this environment, leasing service enables the service provider to be listed on the lookup service for a specified period of time, beyond which the service provider is required to renegotiate a new lease or stand removed from the lookup service. When a mobile host is disconnected without announcement, the mobile host’s lease will eventually expire and will be removed from the lookup service. This feature automates the process of disconnection and reconnection among mobile hosts.

There are two interesting issues when we perform location dependent queries, such as “where is the nearest gas station,” in nomadic environments. First, the change of user location affects the path along which the answer is returned. Second, if the user’s location has changed when an answer arrives, the answer might not be valid. With the support of mobile agents and location services, these two issues can be easily solved. When the agent obtains the answer, it can automatically detect the user’s location change through location services, then download the newly updated proxy, and return to the user’s new location to provide the requested service. Furthermore, the agent may also check the validity of the answer using the updated user location information. If the answer is not valid, the agent then discards it and performs another query based on the user’s current location.
5. AN APPLICATION SCENARIO

In this section, we describe the application scenario of automatic parking space search and reservation. This application scenario takes advantage of the agent mobility in a highly nomadic environment. Parking is a nightmare in most situations, especially when attending a large event such as a football game or a large conference center. Using the proposed framework, such nightmares may be avoided through coordination between the in-vehicle system and local parking service systems. When approaching the desired destination, the first thing that you are interested in is knowing the parking services around the destination. Thus, while passing a billboard that advertises parking lots, the in-vehicle system may send out an intelligent agent requesting information.

After considering a whole host of criteria such as available parking services, price and proximity to the desired destination, the agent makes the reservation and saves the driving direction for the in-vehicle system. Since there is no need for continuous connection in the mobile agent based framework, the mobile hosts can simply acquire a new IP address when roaming to a new subnet, and then update its address or location or proxy using the lookup services. Upon downloading the updated proxy, the mobile agent then communicates with the mobile host directly instead of going through a triangle.

Figure 3. Automatic Parking Space Search and Reservation
(a) the in-vehicle host sends out an agent to search for the parking services around the destination; 
(b) after making a reservation, the agent returns to the host when the host is reconnected.

6. RELATED WORK

Mobile IP [6] is a widely accepted standard that supports seamless handoffs making it possible for mobile hosts to roam among subnets without changing IP addresses. In a mobile IP environment, a mobile host or router can change its point of attachment from subnet to subnet. If a mobile host is away from home when an Internet host sends an IP datagram for delivery to the mobile host’s home network, the datagram will be tunneled to the host’s current foreign network. The home agent will encapsulate the datagram with an IP header carrying either the foreign agent’s IP address or the mobile host’s co-located care-of address. Other mobile IP extensions include smooth handoff [7] and an extension for IPv6 [8]. Mobile IP suffers from the “triangle routing problem” since the path from the sender to mobile host via home agent takes two sides of a triangle, rather than the third side that is the direct path. Since there is no need for continuous connection in the mobile agent based framework, the mobile hosts can simply acquire a new IP address when roaming to a new subnet, and then update its address or location or proxy using the lookup services. Upon downloading the updated proxy, the mobile agent then communicates with the mobile host directly instead of going through a triangle.

There are also several well known systems that support nomadic applications. Coda is a distributed file system that supports disconnected-operations through hoarding (user-assisted cache management), which updates logging with extensive optimizations while disconnected and reintegrates upon reconnection [9]. Bayou provides support for sharing data among mobile users [10]. Rover combines relocatable dynamic objects and queued remote procedure calls to provide a uniform distributed object architecture for code shipping, object caching and asynchronous object invocations [11]. Traveler focuses on developing models, prototyping systems software and running experiments in support of those capabilities needed by a mobile user [12].

A lot of work is being done in the area of mobile agents. Mobile agents offer many potential advantages over traditional approaches. Recently, several robust and efficient, mobile agent systems, such as AgentSpace [13], Ara [14], Concordia [15] and Nomads [16], were proposed. However, all such systems are monolithic and such an approach to mobile agent systems is harming the spread of acceptance of mobile code. Developers are hesitant to create applications that require use of a new, large and monolithic system. Based on the widely accepted middleware system, the proposed framework provides several mobility paradigms, including code on demand and mobile agents.
7. CONCLUSIONS
In this paper, the authors introduce a new framework to enable the integration of JINI technology with mobile agents. The new framework alleviates a lot of the issues and bottlenecks faced by existing client-server model. The new framework is essentially a client-server model but one that taps into the inherent advantages of the JINI architecture and the mobile agents model. Using the JINI architecture, virtually any type of service (software component or hardware device) may freely interact in a network without the need for complex protocols, messaging driver, operating systems and cabling.

Using the proposed JINI based framework, nomadic computing can greatly benefit in providing new telematics applications. This new framework supports disconnected-operations, increases host and user mobility, improves reliability and speed, provides autonomous service discovery and location awareness in a truly challenging nomadic computing environment. The authors also present an application scenario to describe the working of the framework. The proposed framework can be readily adapted to other challenging applications in the realm of nomadic environment.

ACKNOWLEDGMENTS
This work is supported by the Center for Engineering Education & Practice (CEEP) at the University of Michigan-Dearborn.

REFERENCES