My Dream of Jini*

Fabio Kon Jalal Al-Muhtadi Roy H. Campbell M. Dennis Mickunas

Department of Computer Science
University of Illinois at Urbana-Champaign
{f-kon,almuhtad,roy,mickunas}@cs.uiuc.edu
http://choices.cs.uiuc.edu/2k

1 Prelude

Jini provides a set of protocols, conventions, services and facilities to handle dynamic distributed systems. It enables the construction of dynamic distributed object systems and software drivers for network plug-and-play devices. Jini provides protocols to allow services to join a network and discover what services are available in this network. It also defines standard service interfaces for leasing, transactions, and events [Wal98].

The introduction of the Jini technology in 1998 was a very important contribution to the field of dynamic discovery and ad-hoc networking. It also opened the eyes of a large number of people to the ideas of ubiquitous computing earlier proposed by Mark Weiser [Wei92].

Although Jini provides the basic tools for the development of simple, small-scale networks of objects, it does not yet provide a powerful and complete solution to the problem of supporting ubiquitous computing effectively. The Jini of our dreams would provide additional support for (1) dynamic configuration and adaptation, (2) dynamic resource management, (3) security and privacy, and (4) scalability and interoperability.

2 Dynamic Configuration and Resource Management

Jini provides some limited support for dynamically configuring Jini clients by downloading a proxy from the Lookup server. The proxy is automatically linked to the client side and is responsible for communication with the server. Java also provides support for dynamic loading class files, which can be retrieved from locations defined via any URL.

At the moment, dynamic unloading of classes in Java seems not to work. This makes dynamic reconfiguration a little tricky, requiring that an extra level of indirection be added.

We believe that in order to accommodate component-based applications for the highly-dynamic environments of the future, system software must provide powerful mechanisms for automatic configuration and dynamic reconfiguration. Automatic Configuration [Kon00] refers to applications that are assembled at runtime by downloading components from different network-centric repositories, taking special care with user preferences, resource requirements and dynamic resource availability. Dynamic reconfiguration is important for software evolution, adaptation to changes in the environment and for dealing with failures.

Java does a very good job in hiding the details of the underlying operating system and hardware platform. This is a wonderful feature, most of the time. However, there are many cases in which a better understanding of the underlying resources would be desirable. Multimedia and real-time applications, for example, can benefit from knowing the amount and type of memory, CPU, and network resources that are available at runtime. Moreover, some applications are able to select different algorithms and policies according to notifications received from the underlying system. These adaptations are desirable, in some cases, to improve performance but, in other cases they may be critical, like in intercontinental videoconferencing for example.

*This research is supported by the National Science Foundation, grants 98-70736, 99-70139, and EIA99-72884EQ.

3 Security and Privacy

Once the devices in a workspace or a household are automated and connected through a network using a Jini-like technology, it becomes important to consider security issues such as privacy, authentication, and access control. In distributed environments security is critical, as there are more areas that can be breached. When the distributed system start to control our homes, security and privacy becomes essential. In such settings, access to some resources must be restricted, communication links must be secured, and mutual authentication among communicating components is a major requirement. Additionally, any security mechanism must be tailored to accommodate the dynamism and agility that are associated with distributed components, active spaces, and smart devices.

Unfortunately, the current Jini security model is limited and incomplete. As it stands today, the Jini security model is based on JDK 1.2. This security model does not scale well for distributed environments and it lacks support for distributed security services, authentication, and authorization. To compensate for this, a service has to provide its own authentication and authorization model to identify authorized clients.

An ideal security model should help out in providing security services for various Jini services. For example, controlling access to specific Lookup services and restricting the distribution of some services or "group" of services. To meet the objectives above, we need a security model that is based on a widely available security infrastructure, granting us flexibility, scalability, and the ability to achieve secure interoperation between different devices regardless of their platforms and programming environments.

SESAME [VGV97] is an extension of Kerberos that provides support for authentication using digital signatures, preventing off-line dictionary attacks, handling of access control privileges for users, and different key management and distribution schemes. Tiny SESAME [AMAMC00] is a subset of SESAME that supports authentication, simple encryption, integrity checks and RBAC (Role-Based Access Control). It is a lightweight component-based security mechanism. Protocols and security services can be loaded and unloaded dynamically, allowing the security model to adapt to environments with scarce resources and expand once resource become available. In our security research, we constructed a secure active home environment by combining Java with Tiny SESAME.

4 Scalability and Interoperability

Jini has two problems in regard to scalability. First, it is not yet clear how the Jini discovery mechanisms can be efficiently deployed in large-scale systems composed of hundreds of local networks and thousands or millions of devices. Second, Java does not yet scale down to PDAs and embedded devices. And it is not a matter of saying "memory will get cheaper; in the future, every device will have dozens of megabytes of memory". Ubiquitous computing comprises not only mobile computers and PDAs but also books, soda cans, and paper clips with embedded systems on them. Devices with limited memory, CPU, and batteries will become more common, not less common.

It seems, then, that the world of computing will not move towards 100% pure Java solutions for everything. Thus, it is important to investigate mechanisms that facilitate the interoperability among radically different computing devices. CORBA seems to be a good solution since it provides an elegant language-independent distributed object model that can scale up to millions of objects and down to PDAs and embedded systems [RMKC00]. In addition, the CORBA trader, for example, provides a much richer query mechanism than the Jini Lookup service.

Since Java 1.3 comes with CORBA support, interoperability with small devices could be achieved by defining OMG IDL interfaces for the Jini services and developing Jini to CORBA bridges in Java. In this scenario, devices could choose to communicate with the Jini services using either Java RMI or CORBA. Scaling up the lookup and discovery mechanisms, however, is still an open problem.

5 Conclusion

Jini is, perhaps, the most advanced implementation of a technology supporting dynamic ad-hoc networking for ubiquitous computing. However, it still lacks a number of features that are important for its widespread use. Some of these issues will have to be added to Jini itself; others can be provided as separate services in the system. The Java/Jini community must investigate the implementation of these missing features in a way that leaves space for diversity but that guarantees interoperability.

References

[AMAMC00] J. Al-Muhtadi, M. Anand, D. Mickunas, and R. Campbell. Secure Smart Homes using Jini and SESAME. In *Proceedings of the 16th ACSA/ACM Annual Computer Security Applications Conference*, 2000.

[Kon00] Fabio Kon. *Automatic Configuration of Component-Based Distributed Systems*. PhD thesis, Department of Computer Science, University of Illinois at Urbana-Champaign, May 2000.

[RMKC00] Manuel Román, Dennis Mickunas, Fabio Kon, and Roy H. Campbell. LegORB and Ubiquitous CORBA. In *Proceedings of the IFIP/ACM Middleware'2000 Workshop on Reflective Middleware*, pages 1–2, Palisades, NY, April 2000.

[VGV97] M. Vandenwauver, R. Govaerts, and J. Vandewalle. Overview of Authentication Protocols: Kerberos and SESAME. In *Proceedings of the 31st Annual IEEE Carnahan Conference on Security Technology*, pages 108–113, 1997.

[Wal98] Jim Waldo. Jini Architecture Overview. Available at `http://java.sun.com/products/jini/whitepapers`, 1998.

[Wei92] Mark Weiser. The Computer for the 21st Century. *Scientific American*, 265(3):94–104, September 1992.