Building Distributed Context-Aware Applications

Tore Urnes, Arne S. Hatlen*, Pål S. Malm, and Øystein Myhre
Telenor Research and Development
Postboks 83, N-2027 Kjeller, Norway
+47 63 84 84 00
tore.urnes@telenor.com

Abstract

Context-aware applications gather information about their users and operating environment from sensors. Sensors are physically distributed, need to be shared among many applications, and offer complicated programmatic interfaces. As a result, context-aware applications are hard and time-consuming to develop. To simplify the development task, we have tried using the Jini dynamic discovery protocol as a support infrastructure for context-aware applications. We present several context-aware applications developed using the infrastructure and discuss our experiences.

1 Introduction

Context-aware applications gather contextual knowledge about their users and operating environment. Contextual knowledge is typically obtained from time-varying sensory data—in real time and sometimes after making inferences. Equipped with knowledge about the current situation of usage, context-aware applications are able to automatically perform appropriate actions without the user needing to request them explicitly. Hence, it is believed that context-aware applications are better able to satisfy users' needs than less knowledgeable applications.

Unfortunately, obtaining contextual knowledge is a non-trivial task. Sensors are typically physically distributed and communicate their data using a variety of protocols and formats. Several applications may need to share the same sensor, leading to the notion of sensors as scarce resources. Raw sensory data must often be subject to complicated signal processing and artificial intelligence techniques in order to be suited for use in applications. These and other difficulties result in context-aware applications becoming hard and expensive to develop.

We wish to find ways to simplify the development of context-aware applications. In particular, we are interested in support infrastructures that make it easier to deal with sensors. In this position paper we report on an approach where *dynamic discovery protocols* were used to solve many problems related to sensor distribution, sensor discovery, and delivery of sensory data to applications. Our approach is based on *Jini* [8], a Java-based dynamic discovery protocol from Sun Microsystems.

To evaluate the success of our approach, we have built several context-aware applications using Jini as a support infrastructure. The applications include a smart map, a communication and coordination system for families, and a smart kitchen assistant.

We start by briefly describing Jini before explaining the concept of sensors as Jini services. Next, we present several applications that employ "jinified" sensors to gather contextual knowledge. Finally,

*Currently at WinsoftQSD AS, Oslo, Norway.

we mention briefly some related work before offering concluding remarks and suggesting future work.

2 The Jini Dynamic Discovery Protocol

Dynamic discovery protocols are key to the convergence of consumer electronics and computing. A wide range of new networking technologies connect all kinds of electronic devices. Dynamic discovery protocols allow such devices to be discovered, to offer services, and to interoperate with each other and applications—all without requiring manual installation and configuration.

Jini is a new dynamic discovery protocol from Sun Microsystems. Realized as an extension to Java's mechanism for remote method invocation (RMI), Jini offers a simple and elegant API for both service providers and application developers.

Code mobility is central to Jini. Each Jini service—realized as a hardware device, a software program, or a mixture of both—offers its interface as a serialized Java object. Known as a *proxy*, a serialized object is a representation of an object that can be downloaded and instantiated on any device with a Java virtual machine (JVM). Activation of a proxy's methods typically results in the remote invocation of the corresponding service methods. Jini assumes that JVM's be available on all devices. If a device does not possess a JVM, then necessary information must be mirrored on a Java-enabled machine elsewhere (using some proprietary protocol to maintain consistency between device and mirror).

To allow services to be offered and used, most dynamic discovery protocols provide a service registry where service providers register themselves to signal availability to service consumers. Jini implements the registry as a special Jini service, the *lookup service*. Service registration simply involves entering an appropriate proxy in the lookup service.

The lookup service always contains its own proxy as an entry. Both service providers and consumers must obtain this proxy prior to interacting with the lookup service. A simple multicast protocol is used to locate the lookup service on the local area network and fetch the lookup service proxy. The lookup service proxy is used to register new services, to query the service database, and to subscribe to various kinds of events, among other things.

Jini also consists of a programming model aimed at simplifying many of the tasks required to offer and use Jini services. One of the more difficult tasks is to handle dynamic issues, i.e., the entering, leaving, and partial failure of service providers, applications, and lookup services. Leasing, distributed events, and transactions are concepts and API's which help developers tackle spontaneous changes in the Jini architecture.

3 Sensors as Jini Services

Sensors may be viewed as electronic devices offering context services. Many of the problems related to distribution of sensors, heterogeneous sensor hardware, and lack of interoperability between sensors and applications are not unique to context-aware applications, they also appear in other device networks such as home networks. Dynamic discovery protocols were designed to solve such problems. It would therefore likely be desirable to register sensors as Jini services, hence relieving developers from dealing with the said problems.

Many sensors generate data at varying rates making it cumbersome for applications to detect context changes through sensor polling. Jini's distributed event system allows sensors to instead notify applications when certain pre-defined changes occur. Sometimes, a piece of contextual may require combining data from a large number of sensors. In such cases, it may be appropriate to collect the sensory data in a central database and register the database, rather than the sensors, in

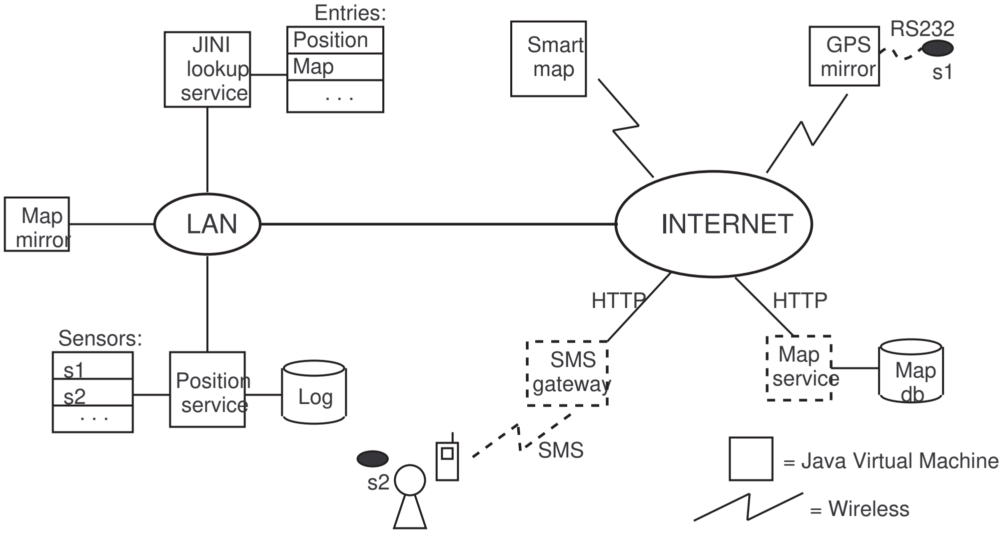

Figure 1: Jini support infrastructure for the smart map application.

the Jini lookup service.

4 Applications

In an effort to enable us to evaluate the usefulness of a Jini-based support infrastructure, we have employed it in several interesting context-aware applications: a smart map application, a communication and coordination system for the family, and a smart kitchen assistant. We now describe the smart map application in some detail before briefly presenting the other two.

A smart map knows the user's position and is therefore able to automatically select and update the displayed region of the map so as to always include the user's current position. Running on portable, Java-enabled devices with Java RMI support, the smart map receives all contextual information (GPS coordinates) and data (maps in the form of GIF images) from the network (in our case, the wireless GSM network). The application can “follow” any available position sensor; maps are supplied by a commercial, third-party map server (`www.totalkart.no`).

Figure 1 shows the implementation architecture of the smart map application. Position and map services are available as Jini services in a lookup service. We use GPS sensors to supply position information. Rather than registering each position sensor as a separate Jini service, they are instead collected in a position service database. To facilitate the use of non-connected GPS sensors, coordinates can be communicated to the position service database as SMS messages. The Map mirror entity is responsible for maintaining the map service registration, including a proxy that provides a Java RMI front-end to the HTTP protocol of the third-party map service.

The other two applications are realizations of home automation scenaria. The communication and coordination system also uses the position service database in addition to sensors for detecting the identity of family members. The kitchen assistant uses numerous kitchen sensors, including a scale, an egg sensor, and a fridge door sensor to help suggest food recipes and offer guidance in food preparation. All three applications of this section share the same Jini-based support infrastructure.

5 Related Work

This section briefly mentions other support infrastructures for context-aware applications. Early work on location-aware applications at Olivetti Research resulted in an infrastructure for active badges [4] exhibiting many of the concepts underlying dynamic discovery protocols such as Jini. Inspired by the successes of user interface toolkits at providing abstractions that simplify input handling, researchers at Georgia Institute of Technology developed a context toolkit [6, 2] offering components for building context widgets. Somewhat more broadly focused than our work, University of Kent researchers sum up lessons learnt from building numerous context-aware applications by outlining design issues for support infrastructures [5]. Schmidt et al. proposes a four-layer architecture [7] that simplifies the construction of mappings from raw sensor data to high-level events. Finally, researchers at the University of Queensland offer a messaging infrastructure [1] allowing sensors to deliver data to subscribing applications through a tuple space-based notification server (the SLP dynamic discovery protocol is used to get in touch with the notification server).

6 Discussion and Future Work

The task of dealing with sensors is a tough problem for developers of context-aware applications. We feel that dynamic discovery protocols greatly simplify those problems that are related to distribution and dynamic aspects. Unfortunately, incorporating sensors into a dynamic discovery protocol such as Jini is not trivial. In fact, a considerable amount of the effort behind the applications presented here went into "jinifying" all the sensors. It appears, though, that there is substantial momentum behind Jini and that the future might bring devices and sensors that are ready to participate fully in systems such as Jini.

Our work with Jini as a support infrastructure for context-aware applications will continue and we have started to investigate dynamic aspects more closely. Currently we are building Jini-based sensor networks on top of the Bluetooth protocol for wireless communication [3].

References

[1] D. Arnold, B. Segall, J. Boot, A. Bond, M. Lloyd, and S. Kaplan. Discourse with Disposable Computers: How and Why You Will Talk to Your Tomatoes. In *Proceedings of the Workshop on Embedded Systems* (Cambridge, MA, USA, Mar. 29–31). The USENIX Association, 1999.

[2] A.K. Dey, G.D. Abowd, and D. Salber. A Context-Based Infrastructure for Smart Environments. In *Proceedings of the First International Workshop on Managing Interactions in Smart Environments* (MANSE '99, Dublin, Ireland, Dec. 13–14), Published as Lecture Notes in Computer Science, Springer Verlag, 1999.

[3] J. Haartsen, M. Naghshineh, J. Inouye, O.J. Joeressen, and W. Allen. Bluetooth: Vision, Goals, and Architecture. *ACM Mobile Computing and Communications Review*, 2(4):38–47, October 1998.

[4] A. Harter and A. Hopper. A Distributed Location System for the Active Office. *IEEE Network*, 8(1):62–70, January/February 1994.

[5] J. Pascoe, N. Ryan, and D. Morse. Issues in Developing Context-Aware Computing. In *Proceedings of the First International Symposium on Handheld and Ubiquitous Computing* (HUC '99, Karlsruhe, Germany, Sept. 27–29), Published as Lecture Notes in Computer Science, Vol. 1707, Springer Verlag, pages 208–221, 1999.

[6] D. Salber, A.K. Dey, and G.D. Abowd. The Context Toolkit: Aiding the Development of Context-Enabled Applications. In *Human Factors in Computing Systems: CHI '99 Conference Proceedings* (Pittsburgh, PA, USA, May 15–20), pages 434–441. ACM Press, 1999.

[7] A. Schmidt, K.A. Aidoo, A. Takaluoma, U. Tuomela, K. Van Laerhoven, and W. Van de Velde. Advanced Interaction in Context. In *Proceedings of the First International Symposium on Handheld and Ubiquitous Computing* (HUC '99, Karlsruhe, Germany, Sept. 27–29), Published as Lecture Notes in Computer Science, Vol. 1707, Springer Verlag, pages 89–101, 1999.

[8] J. Waldo. The Jini Architecture for Network-Centric Computing. *Communications of the ACM*, 42(7):76–82, July 1999.