The Australian Centre for Jini Technologies

Mandate

The Australian Centre for Jini Technologies has the following mandate:

> To perform applied research and development into the use and dissemination of Jini technologies, and to act as a showcase and clearing-house for these technologies in Australia.

Introduction

Jini is a new distributed technology from Sun Microsystems based on the Java programming language. It allows software and hardware to be written that is ‘‘Jini aware’’, and to capture and bridge between non-Jini applications, hardware and software services. This allows new and existing systems to be built or brought into a Jini environment.

A Jini-aware system is able to take advantage of the Jini ‘‘network plug and work’’ environment by being able to find and use services without prior knowledge of the implementation of these services. on the other hand, new services and new ways of doing things, can be started and announce themselves to clients.

Jini-enabled networks are more flexible than traditional networks, and allow mobile users and applications to join and take advantage of network services without prior knowledge of these services. The mobile user can bring his laptop within a local network and start using the local printers directly. On the other hand, when she returns home her laptop will re-establish contact with the game she was playing and continue.

Issues

Jini technology is new. The framework has been in place for less than a year. The number of Jini services is trivially small right now. For Jini to succeed, the number of services must radically increase, but also a transition path from current technology to Jini-enabled systems must be clearly available. What is missing is two factors:

- Where can Jini take us?
- how do we get there from here?

These involve a number of issues, of a short-term and of a medium-term nature (deliverable within a couple of years). The short-term problems involve creating and delivering Jini services within an almost totally non-Jini environment. The longer term issues involve the types of systems we will be using in a few years (home networks, smart devices, high-bandwidth fixed connections, mobile systems within both good and poor quality networks) and what we should be doing with these systems. Migration strategies will be needed to move between these.

Demonstration Projects

Static laboratory

This project will be to set a laboratory of computers and devices running in a Jini environment. It will be used to demonstrate interoperability between computers with differing Operating Systems and hardware configurations. The services available in this laboratory will also be made available on the Internet, so that they may be used and demonstrated from remote sites.

Mobile computers

This project will use some of the facilities of the static laboratory, but will add mobile computing devices (such as the PalmPilot) to this to demonstrate configuration capabilities in a dynamic manner.

Research Projects

Jini services in C

A Jini service needs at least some elements of a Java Virtual Machine in order to run. It needs to understand the Jini multicast and unicast protocols, download, instatiate and call methods on various objects in the Jini implementation, such as Lease and ServiceRegistrar objects. The subset of Java required to do this is ‘‘close to’’ Embedded Java plus RMI.

Every Java implementation has substantial overheads, and even this subset is no exception. In addition, it requires substantial knowledge of the *implementation* of classes such as `com.sun.jini.reggie.RegistrarProxy`, in order to ensure that the implementation wil in fact work under the restricted JVM. An alternative is to go intead to the protocol level in the implementation between the Sun private classes and implement this by a low-overhead language such as C.

This project is to implement the documented and standardised parts of the Jini protocol in C, determine the protocol used by private classes in the Sun implementation, ascertain how stable this will be in future versions and implement that. This will allow devices to implement lightweight discovery and join mechanisms that can be used in devices with very restricted memory, such as Lego MindStorms.

Constrained memory user interface

User interfaces built with Java are moving from the AWT toolkit to Swing (also known as the Java Foundation Classes). The AWT made heavy use of user interface (GUI) objects belonging to the underlying windowing toolkit such as Motif. Swing makes little use of the structured objects of the underlying GUI, and could be implemented using a far lighter system. For example, the AWT on Unix machines requires the Motif toolkit, whereas Swing would only need some basic capabilities of the lower-level X Window library.

This project is to provide window support for the Swing toolkit using minimum necessary support from underlying window systems. In the first instance this will be to implement Swing using Xlib only. This will allow lightweight window systems to be used on palm devices such as the ‘‘Itsy’’ which are currently implemented with Unix and X.

The project could be extended to look at the various Microsoft Windows lightweight window systems for alternative implementations.

User interfaces for non-keyboard devices

Bridges between CORBA and Jini

Standard desktop

Bridges between Windows and Jini

Jini in a Bluetooth environment

Legacy wrappers

Bridges between Jini and the IETF Service Location Protocol

Jan Newmarch (http://pandonia.canberra.edu.au)
jan@ise.canberra.edu.au
Last modified: Sun Oct 24 22:49:02 EST 1999