Jini connectivity for fieldbus-systems —
a case study for home and building networks

W. Kastner

Jini is a key concept that draws more and more attention among the ever-growing Java community. In most cases the future visions of linking devices without having to configure a network connection are regarded as a challenge that Jini can address in an elegant way. However, there are much more capabilities of Jini than a mere decrease in network configuration complexity. In our opinion Jini is also a good basis for integrating different services of fieldbus-systems (e.g. services of home and building networks) within the Internet. This paper shows how such an integration can be done using a proxy-oriented system architecture.

Keywords: Jini; Java; fieldbus/Internet-connectivity; home and building network

1. Introduction and motivation

Today’s control systems are structured hierarchically. On top we find workstations connected to the Internet running powerful CAD, CAE- and CAM-tools. The layer beyond consists of robust PLCs, architectures based on (fast) microprocessors or simple PCs linked with sensors and actuators in the field by means of well-known fieldbus-systems. In former days it was enough to setup the fieldbus at the lower layer and to control the ongoing technical process from one of the workstations above. Nowadays users demand for real remote controlling, that is to (re-)configure and monitor fieldbus-systems via Internet.

A closer look at software architectures fulfilling the desired job often reveals client/server based approaches. A server running on a dedicated gateway provides read and write access to the fieldbus-devices of interest. Clients connect to the server and communicate using proprietary protocols based upon TCP/IP. If the clients are Java-applets, they can be downloaded from the gateway hosting an HTTP-server and may be executed inside a common Internet browser.

Global connectivity opens increased facilities especially for home and building networks (consider how nice it is to turn on a networked-heating system in your holiday home by means of your WAP-enabled mobile phone). Regarding the connectivity inside our today’s homes and buildings we find several domains that are to some extent already prepared with communication interfaces, such as:

- entertainment electronics with VCR, HIFI chain, video game console or home theatre (concentrated in a few rooms), or carry-on devices such as portable CD players;
- information and communication equipment like plain old telephone and answering machines, but also digital devices, PCs, workstations and terminals;
- additional installation domains including gas and water supply, with the corresponding appliances (and counters);
- security and safety domains providing special comfort services for the sick, the handicapped or the elderly;
- sundry special domains like elevators etc.
If (parts of) those domains can be combined easily with a home and building network, the dream of a real "intelligent" house or building instantly will come true. Keep in mind that the dynamic integration should be done without bothering an end-user to (pre-)configure a connection between the appliances. Presently, such connections have to be made by means of manual network configuration. Hence, to realize our dreams we need "plug and play" features provided by the (fieldbus-)devices, too.

To support plug and play devices Sun Microsystems has introduced Jini (Java Intelligent Network Infrastructure) lately [1]. Jini is a logical layer defined by means of the Java programming language that supports the development of distributed applications by offering Remote Method Invocation (RMI) and a comfortable way of dynamic class loading. Java programs are executed on Java Virtual Machines (JVMs) connected over an arbitrary network as indicated in Fig. 1.

The rest of this paper is structured as follows: the next section will present Jini's main features in a nutshell. Then we explain how to integrate Jini within the fieldbus-systems' world using a proxy-oriented system architecture. Finally, we add our conclusions and finish the paper with a short outlook on ongoing research activities regarding this area.

2. Jini

The Jini vision is simply the following one: any Jini-enabled device plugged into a network is able to automatically see and use the variety of other Jini-enabled devices. Once put into the net any resource accessible in the Jini federation is available for any other Jini-enabled device, as if both of them were connected directly without being explicitly programmed for that connection. A set of Jini-based computing devices is called a Djinn, Jini-federation or Jini-community. In a typical Djinn we find three different kinds of entities: services, service users and (at least one) so-called Lookup Service. A service is an abstract term for a resource that can be used via the network, like specific hardware (in our special case: sensors and/or actuators), software or a communication channel (e.g. a whole home and building network). A service user is the entity that uses one or more services. The protocol between service and service user is not defined at this stage, it is based on Java-based interfaces. Hence, the actual implementation of the communication between the two partners is defined by the service itself (notice: the underlying code is loaded dynamically by the service user!). Last but not least the Lookup Service acts as the middleman between services and service users.

Since the Lookup Service is the central repository for services, Jini has to provide services and service users with appropriate means for dealing with the Lookup Service. This is done by a set of key protocols (Fig. 2):

1. Discovery. Before a service or service user can take advantage of other Jini services it must find one or more Jini-communities. This is done by searching (discovering) for the Lookup Services that keep track of shared resources of that community. The result of the discovery process is that the service or service user are handed one ore more references of the found Lookup Services.

2. Join. Using one Lookup Service reference the service can advertise (publish) its facilities in the Djinn.