Obstacles to Implementing Jini on the Control Area Network (CAN)

Meredith Beveridge

Objective: A run-time infrastructure on a real-time embedded network

Jini promises:
- Spontaneous networking
- Automatic reconfiguration
- Platform-independence

CAN provides:
- Reliable data transmission
- Real-time guarantees

Problem: Jini’s design assumes TCP, which is incompatible with CAN

Original TCP-based Jini:
- Distinction between unicast and multicast
- Message sizes constrained by UDP packet size

Ideal Solution:
- Remove unicast/multicast distinction
- Leave message fragmentation to implementation

Approximated Solution:
- Multicast all messages (CAN is multicast)
- Implement CAN-specific fragmentation, ignoring Jini message size

Sender/Receiver identification fields designed for TCP’s needs

Registration and Service Discovery:
- Registration and service discovery implemented with TCP-based RMI

New Jini messages replace platform-dependent RMI

“ID object” includes all relevant sender/receiver ID fields

Use information available to compute unique CAN message IDs; attach to messages

We were able to use Jini without rebuilding an entire new infrastructure, and proved it can be done (but it wasn’t pretty...)

Conclusion: Approximating Jini over CAN was successful, but painful