MODELING JINI-UPnP BRIDGE USING RAPIDE ADL

Ahmed Sameh
Dept. of Computer Engineering
The George Washington University
Washington, DC 20052

Rehab El-Kharboutly
Dept. of Computer Science,
The American University in Cairo
P.O.Box 2511, Cairo, Egypt

Abstract

The exploding deployment of network enabled mobile devices, along with the expansion of networked services have created the need for users to easily manage these devices and services and also to coordinate with one another. Service Discovery Protocol (SDP) enables networked devices, applications, and services to seek out and find other complementary networked devices, applications, and services needed to properly complete specified tasks. A number of bridging techniques have been proposed and implemented. In this paper we propose a one directional bridging system (Jini-UPnP Bridge). To validate the proposed system, we model and simulate the bridge using Rapide ADL simulation and analysis toolset. We perform a number of simulation tests and use the Rapide Poset viewer to analyze the simulator's output Poset tree of events.

Key Words: SDP, ADL, Jini, UpnP, Rapide

1- The Proposed Jini-UPnP Bridge

Our work is built on the concept of a Jini network proxy described in Jini Device Architecture and is based on the efforts of Eric Guttman in [1]. A Jini-UPnP Bridge is an entity that enables services that support UPnP protocol to be reachable by Jini clients. For Jini clients, Jini-UPnP is a transparent layer that they are unaware of. The UPnP services that are advertised via the bridge are treated as native Jini services. The proposed Jini-UPnP Bridge is modeled as a special network node that can communicate with other network nodes in both Jini and UPnP protocols. It mainly acts as a *Service User* (i.e. Control Point) in UPnP environment and a *Service Manager* (Service) in Jini environment. It waits for announcements made by UPnP devices and services that are willing to advertise their presence to the Jini clients and acts as a representative, almost a mirror for them in the Jini environment. The proposed bridging process is done through the following steps: The Jini-UPnP bridge searches the UPnP reachable entities to find devices and services that have Jini driver Factory or waits till it receives announcements made by Jini driver Factory services. Once a Jini driver Factory service is found, the Jini-UPnP bridge obtains a complete description of the service including attributes, GUI URL and control URL. The Jini driver factory is downloaded using GET method over HTTP. The Jini-UPnP Bridge performs attributes transformation from UPnP format to Jini format to prepare for service registration. Upon successfully translating the entire service attributes and obtaining the Jini driver factory, the Jini-UPnP Bridge registers the discovered service with **Jini Lookup Service**. Using the Jini driver factory, the bridge creates a service object that is used for registration. Registration is done by sending a join request with all necessary attributes to **Jini Lookup Service** that adds the new service to its cache.

2- Testing and Performance Measures

In each of our tests, we first establish initial conditions by constructing a topology of Jini and UPnP basic entities in addition to the Jini-UPnP Bridge. The following tests have been conducted and proven successful: 1- testing to validate that initial discovery and advertisement activities in our hybrid environment of both Jini and UPnP entities, function correctly, 2- testing a complete scenario of bridging a Jini Service to examine the correctness of the bridging process, 3- testing that the proposed UPnP Jini Bridge successfully propagates changes that occur in the JiniFactory service to the SU Jini clients that have previously discover it, 4- testing to confirm that the JiniFactory service shutdown is propagated successfully to Jini SCM through Jini-UPnP Bridge. Measurements for Jini are done on two stages; first we measure the time taken for Jini SM to register with SCM and the number of messages needed. The time taken for this operation, as shown in the results is **TIME TAKEN 1 = 0.064s** , and the number of messages exchanged is four messages (NUM MSGs 1 :4). The second stage is where the SCM starts matching the newly added service description to the available SU requests. Two messages are exchanged for this operation to complete and the total time needed is **TIME TAKEN 2 = 0.00081s**. Thus the total time for the whole operation starting with SM registration to SU discovery takes **TOTAL TIME = 0.06481s** on average. Bridging a UPnP SM service to be reachable for Jini SUs is done in three stages.

3- Conclusion

The problem we addressed in this research is enabling thin servers and lightweight devices to offer their services to Jini clients through passive and indirect registration using our proposed Jini-UPnP Bridge.

[1] R. El-Kharboutly, "Modeling Jini-UpnP Bridge Using Rapide ADL", M.Sc. thesis in Computer Science, The American University in Cairo, 2002.