A PEER-TO-PEER JINI ARCHITECTURE FOR PERVASIVE MULTIMEDIA

Pan Hui, Onshun Chau, Xiaoshan Liu, Victor O.K. Li

Department of Electrical and Electronic Engineering, University of Hong Kong, Hong Kong, China
Email: {panhui, h0118759, xsliu, vli}@eee.hku.hk

Abstract - Wireless technologies enable a mobile device to connect and access resources available in the environment, enabling pervasive multimedia. In this paper we first introduce a Bluetooth Jini Profile which allows Bluetooth devices to integrate with and discover service in a Jini environment. Then a peer-to-peer architecture, Pervasive Multimedia Overlay Network (PMON), connecting Jini networks for media sharing and streaming service, is developed. This overall structure enhances pervasive multimedia for mobile devices.

Keywords – Bluetooth, Jini, P2P, pervasive multimedia, service discovery, media streaming.

I. INTRODUCTION

Mobile devices such as PDAs and mobile phones are increasingly popular. Even with the improvement in wireless bandwidth, due to their low processing power and unsatisfactory display, mobile devices are still not yet ideal for multimedia services. But the wireless connectivity enables these mobile devices to connect to any local network. This feature allows mobile devices to make use of the resources and services in the local environments instead of having to bring their own resources. For example, when a mobile device enters a room, it can operate the Hi-Fi of the room to play music, the TV to play video to enjoy multimedia service. But this needs well developed service discovery protocols for effectively locating and accessing the services available in the environment. Java Intelligent Network Infrastructure (Jini) [1] is a middleware technology built on the TCP/IP layer to implement a service discovery protocol. Jini technology allows a client to discover services in a local network without actual knowledge of the locations or characteristics of the services. In our work, we assume many Jini environments, each of which allows mobile devices to connect to it, search and use services inside it. Bluetooth [2] is a low cost and low power consumption wireless media and is expected to be found on every mobile device in the future. Jini and Bluetooth SDP are two common service discovery technologies. They may meet each other in many environments. But there is not a general architecture to bring them together and hence Bluetooth cannot discover Jini. We first design a Bluetooth Jini Profile which runs in three modes of operations, namely, Surrogate, Bridge, and Client mode, and allows a multihop Bluetooth connection to a Jini environment. From the success of P2P [3] system in recent years for multimedia sharing and media streaming, we further extend the power of a local Jini environment by building a peer-to-peer architecture and defining the communication protocol to connect individual Jini networks to enable searching and instant streaming of multimedia from other Jini networks. Each Jini network is treated as an individual peer node in the whole P2P overlay network. So our contributions include designing a Jini Profile for Bluetooth/Jini interoperability, and introducing a peer-to-peer Jini architecture for media streaming, thus enhancing pervasive multimedia for mobile devices.

The rest of the paper is organized as follows: First we introduce Jini Middleware Technology, and then we describe Bluetooth technology. Section IV shows the Bluetooth Jini. Section V introduces the Pervasive Multimedia Overlay Network (PMON) and section VI shows one application scenario of PMON. Section VII is the conclusion.

II. JINI MIDDLEWARE TECHNOLOGY

Java Intelligent Network Infrastructure (Jini) is a middleware technology built on the TCP/IP layer to implement a service discovery protocol.

The service provider may be a computer or a hardware device with a controllable interface. As the name "Jini" implies, it is implemented in Java programming Language. If the device does not have a Java Programmable interface, the solution is to add another middleware called COBRA to bridge the two languages.

The core component of the Jini Service Lookup Protocol is the Lookup Service Register (LUS). The service provider advertises its existence and availability by registering itself to an LUS. Normally the LUS is the first component to be up in a Jini community and stays in the Jini community rather statically. The LUS advertises its existence by sending out UDP packets with a well-known multicast address. The interested entities will listen for the packets to determine the existence of the LUS. Then the client will communicate with the LUS to search for the services using an unicast discovery protocol. When a service provider do query the LUS, the LUS returns a registrar object (an object which acts as a proxy for the service provider). A copy of the service object will be placed at the LUS.

When a client wants to search for a service, it first creates a ServiceTemplate. Service templates are used by both service

providers and the clients for service request matching. It accepts three important parameters.

ServiceTemplate(ServiceID serviceID,
java.lang.Class[] serviceTypes, Entry[] attrSetTemplates)

ServiceID: An UUID which uniquely identifies the service.

Class[]: An array of "Class" objects that defines the Class type of the service provider.

Entry[]: An Entry object will conceptually represent the attributes of the service in an object format. The type of the object passed can be regarded as the type of attribute of the service, and the value parameters that follow the Object type are the attribute values.

If there is service match, the client will get copies of matched service object from the LUS. The client can communicate with the service provider through the Remote Method Invocation (RMI) and the service provider will tell the client where to download the service implementation codebase.

III. BLUETOOTH TECHNOLOGY

Bluetooth is a short range wireless technology developed by the Bluetooth Special Interest Group (SIG). It operates in the unlicensed Industrial, Scientific and Medical (ISM) band, which is centred around 2.45 GHz. It is originally designed for the replacement of cable. Bluetooth uses fast (1600 hops/sec) frequency hopping (FH) technique with 79 channels centered at (2,402 + k)MHz where K = 0 , 1 , 2 ... 78. FH technique is employed for the sharing of the transmission medium and for security. The maximum asynchronous data rate in Bluetooth v1.1 is 732 kbits/s. Each Bluetooth time slot lasts for 625μs.

Bluetooth devices can operate in one of two modes: Master or Slave mode. Bluetooth devices are organized into Piconets. In a particular Piconet, one Bluetooth device acts as Master, and the others as Slaves. It is the Master that schedules the data traffic over the Piconet. A collection of Slave devices operating with one common Master is referred to as a Piconet. The maximum allowable number of active Bluetooth devices which may actively participate in a particular Piconet is eight (one Master and seven Slaves). Each active member is indicated by a 3-bit number called Active Member address (AM_ADDR). A Slave can send packets to the Master only if the Master has sent it a data packet. Thus, the Slaves cannot send packets to each other directly. Like TCP and UDP in the TCP/IP protocol, there are two kinds of links in Bluetooth communication. They are Synchronous Connection-oriented (SCO) and Asynchronous Connectionless (ACL) links. An SCO link can be used for transmission of voice packets which are never retransmitted. An ACL link is used for transmitting data packets. An ACL link supports broadcast and if there is a packet loss (may be due to collision), an ACL packet can be retransmitted.

A Scatternet can be formed by linking several Piconets together in an ad hoc fashion to accomodate more Bluetooth devices. The discussion of Scatternet is outside the scope of this article but our proposed architecture will assume a Scatternet environment.

Bluetooth implements Service Discovery Protocol (SDP). The service provided by a bluetooth device is called a profile. Bluetooth v1.1 standardizes several profiles. In the upcoming v1.2 more profiles will be standardized. The service registers itself to the SDP Database as a service record. A service record is represented by attribute-value pairs. Here are some important attributes for a service.

ServiceRecordHandle (0x0000) -- acts as a primary key to identify a service in the SDP server.

ServiceID (0x0003) -- UUID that uniquely identifies a service.

ProtocolDescriptorList(0x0004) -- lists the protocol required to support the profile.

The service request procedure is done by a request-respond scheme, through SDP Protocol Data Units (PDU).

IV. BLUETOOTH JINI PROFILE

Jini is built on a TCP/IP network. One of the critical points for Bluetooth devices to participate in a Jini community is that the access point (AP) should be able to route multicast packets in and out of the Bluetooth network. There is some work on Jini/Bluetooth interoperability [4]. But again the range of the Bluetooth client to the Jini surrogate is only limited to one hop and there is no public message exchange protocol such as XML for platform independency. Our goal is to develop an architecture which enables Bluetooth devices to look for Jini services up to multiple hops away.

Serveral factors affect our architecture design:

1. The limit on the range of Bluetooth radio:

Life would be easy if every Bluetooth device is just one hop from the access point. One of the design features of the Bluetooth device is power conservation. To limit power consumption, the power of the bluetooth radio is rather small. Currently most of the Bluetooth modules use Class 3 radio (1mW). The maximum range is 10m, but due to interference, the typical range of the Class 3 radio is just 5-6m. In the case of the PAN or LAN profiles, if the access point is not within range of the client device, multicast routing of packets has to be peformed over multiple hops. This increases system complexity. In this article, we will propose a bridge architecture for the routing of Jini service requests in a Bluetooth network.

2. The ad hoc characteristics of Bluetooth network

In a Bluetooth network, it is assumed that all the nodes have a rather high mobility; they can freely join or leave arbitrarily. Our proposed architecture should be able to deal with possible changes of the Bluetooth network topology.

3. The constraint of Bluetooth device

Bluetooth is rather lightweight and simple. It is designed for cable replacement and for embedded systems. Likewise, any protocol built on the Bluetooth network should also have the same quality.

The Jini™ profile defines the architectural components and necessary procedures cooperating with the Jini Surrogate architecture to allow a client in a Bluetooth network to perform Jini service lookup. Jini™ Profile runs in any one of the three operation modes: Surrogate, Bridge, and Client .

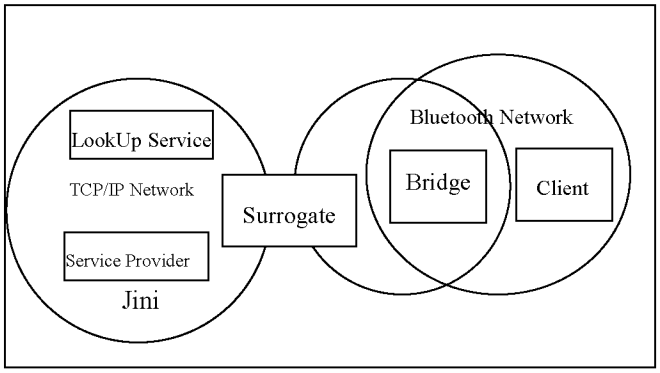

Fig. 1. Bluetooth Jini™ Profile Architecture Overview

Surrogate performs session management and parses Jini service lookup requests from an XML document sent by a device in a Bluetooth network. The Bridge routes Jini service lookup requests initiated by a Client in a Bluetooth network to enable multihop service discovery. The Client initiates Jini service lookup requests. The request description is put into an XML file and transmitted through an OBEX-FTP profile. Fig. 3 shows the Bluetooth protocol stacks needed for Jini Service Lookup.

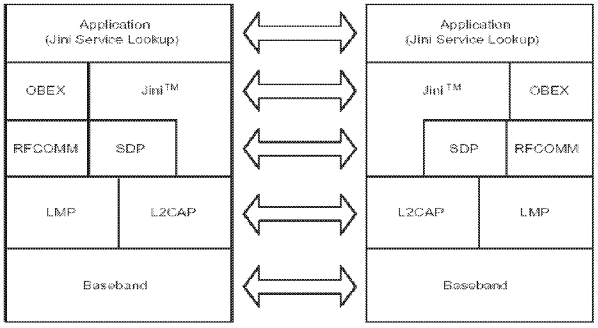

Fig.2. The Bluetooth Protocol Stack for Jini Service Lookup

V. PERVASIVE MULTIMEDIA OVERLAY NETWORK

The system we propose is called Pervasive Multimedia Overlay Network (PMON). In this network architecture, each Jini network/environment is treated as a single peer and connected by P2P protocol to form a P2P network. Each peer (individual Jini Network) contains four main components: Jini LookUp Service Register (LUS), Multimedia Gateway (GW), Media Streaming Server (MSS) and the Jini/Bluetooth Surrogate. Fig. 3 shows the PMON architecture overview.

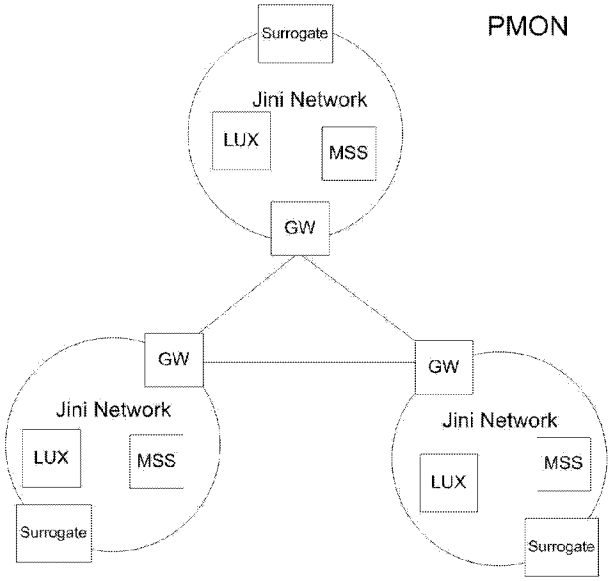

Fig. 3. PMON Architecture Overview

1. Jini LookUp Service Register (LUS)

Jini LookUp Service Register is the most important component in a Jini community. It acts as a database for service providers to advertise their services in a Jini community. A copy of the service object is stored in the LUS.

2. Multimedia Gateway (GW)

A Multimedia Gateway (GW) just acts as a gateway for a Jini network to communicate with other Jini networks. GW talks to other GW by using peer-to-peer protocol descriptors. The protocol used in our system is similar to Gnutella Protocol v0.4 [3].We define five types of descriptors, Ping, Pong, Query, Query Hit, and Push. A GW uses Ping and Pong to maintain connection status to the Pervasive Multimedia Overlay Network (PMON), Query to query media files in the network and QueryHit to return query results. The Push descriptor is a mechanism that allows a firewalled GW to contribute file-based data to the network. Although each Jini network is treated as a virtual node in the PMON, actually GWs are the real nodes connected in the overlay. It keeps a list of the machines and their corresponding media files shared out and hence acts as a proxy to answer whether its local domain has a particular media file during a P2P search. When a requested media file is not in the local network, GW will send out Query messages to all its neighbors in the PMON and this procedure will continue in other GW until the TTL value in the query header drops to zero. If a match is found in a remote GW, that particular GW will return a QueryHit which contains the metadata of the local media file, the address and the network condition of the machines that provide the service. The local GW, upon receiving the QueryHits, will collect them in a list and choose the best one for streaming the data.

3. Media Streaming Server (SS)

The Media Streaming Server can be a Real Time Protocol (RTP) or Real Time Streaming Protocol (RTSP) media server. It shares media files and registers with the LUS to act as a service provider and allows direct connection from foreign Jini domains.

4. Jini/Bluetooth Surrogate

Jini/Bluetooth Surrogate is a Bluetooth access point which runs Bluetooth Jini Profile in the surrogate mode and allows a Bluetooth device to discover a Jini network.

The architecture of the Surrogate is shown in Fig. 4. It consists of two large modules, one interfacing with the Bluetooth network and the other the Jini community. The Surrogate can be decomposed into serveral modules. (The term "Surrogate" in this article is defined as Bluetooth Jini Profile--Surrogate operation mode. This definition is different from that of SUN™ [5]; however our proposed "Surrogate" includes the majority of the functions in SUN™'s definition of surrogate)

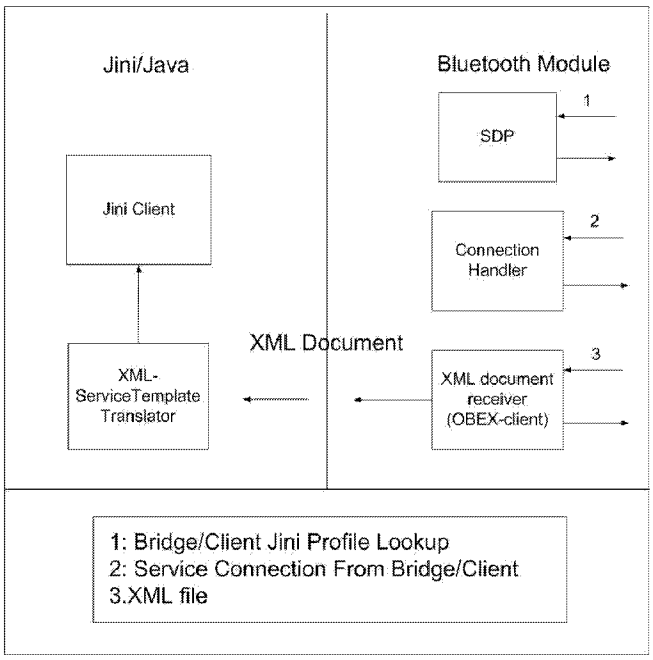

Fig. 4. The internal structure of a Surrogate with Jini Client included

1. Surrogate SDP Database

The Surrogate's SDP Database has a record Jini Profile with "Surrogate" as the mode of operation. It allows its surrogate service to be discovered.

2. Jini Profile Connection Handler

It receives connection requests from Bridge or Client and performs session management. It can also monitor the client connection status.

3. XML document receiver (OBEX-Client)

It acts as the client for OBEX-Client for the XML document. When the file is fully received, the file is stored in a temporary place. OBEX, the Object Exchange Profile, is a protocol allowing object push, pull, and initialization of the Object Exchange Session. It forms the basis for other profiles such as the FTP profile which allows flat file transfer over the Bluetooth network. We will use OBEX-FTP for the transmission of XML documents in our proposed architecture.

4. XML-ServiceTemplate Translator

Converts XML data to Jini ServiceTemplate (a Java object that represent a service query). The translation process is written in the Java Programming language.

5. Jini Client

Uses the ServiceTemplate constructed by XML-ServiceTemplate Translator to query for service in LUS.

VI. ENJOY PERVASIVE MULTIMEDIA

When a mobile client connects to a Jini network by the Jini Profile, it looks for an LUS and searches for any multimedia services. Through the LUS, it discovers that a device, say a speaker nearby is available for sharing. The client then connects to the service and asks the speaker to play a certain media, which is passed as a parameter along with the call. If the speaker can find the matching media files locally, it will play it. If it cannot find the service, it connects to the GW. If there is no required media data found in GW's local media list, the GW will issue a P2P search. Fig. 5 shows the chronological sequence of a Pervasive Multimedia Streaming Process.

1. The Speaker/ Monitor at the Client Domain registers itself to the Jini Lookup Service Register (LUS1)
2. A Streaming Server at Server Domain (SS2) registers itself to the LUS2.
3. A Multimedia Gateway at Client Domain (GW1) registers itself to the LUS1.
4-5. A Multimedia Gateway at the Server Domain (GW2) is up; it discovers available multimedia server(s).

6-7. GW2 contact the SS2 to cache any available media data.

8-9. A Client (e.g. mobile node) comes in to the Server Domain and searches for the available multimedia service. It finds the Speaker/Monitor through the LUS1.

10. The Client connects to the Speaker/Monitor through Bluetooth and passes in requested media data.

11. The Speaker/Monitor tries to find the requested media data locally. If it finds the service available, just play the media data.

12-14. If the requested media data is not available, it tries to find the data through GW1. GW1 is found through a Jini LUS.

15. GW1 tries to find whether media data is available in the local domain, from its cached memory.

16. If no media data is found, it sends Query messages to its neighbours in the PMON.

17. For each Gateway (GW2) found, it attempts to find the requested media data, from its cache.

18. If matched media data is found, Query Hit will be sent back, with the location (e.g. IP address + port number + protocol) of the Multimedia Server (SS2) to requesting gateway (GW1).

19. GW1 redirects the result to Speaker/Monitor.

20-21. The Speaker/Monitor connects directly to the SS2 with the required result.

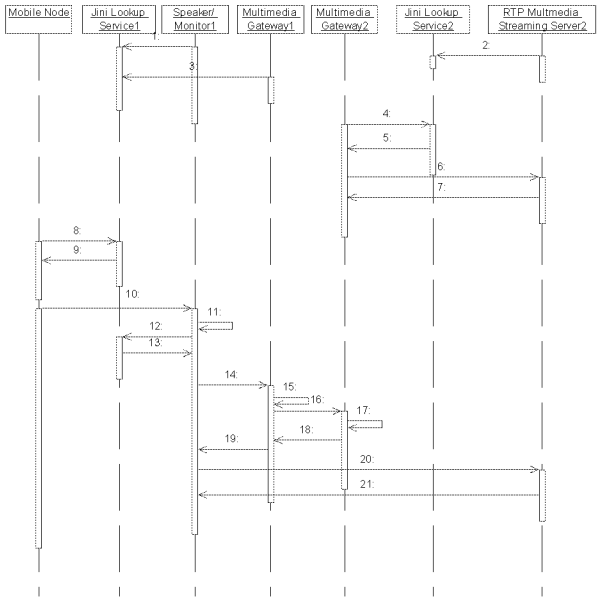

Fig. 5. Chronological Sequence of a Pervasive Multimedia Streaming Process

VII. CONCLUSION

We propose a Bluetooth Jini Profile and a Pervasive Multimedia Overlay Network (PMON) connecting Jini networks together to provide pervasive multimedia service for mobile devices equipped with Bluetooth.

ACKNOWLEDGEMENT

This research is supported in part by the Areas of Excellence Scheme established under the University Grants Committee of the Hong Kong Special Administrative Region, China (Project No. AoE/E-01/99).

REFERENCES

[1] Jini Network Technology, "Jini Specifications V2.0", http://wwws.sun.com/software/jini/specs/index.html, June 2003.

[2] C. Bisdikian, "An Overview of the Bluetooth Wireless Technology", IEEE Communication Magazine, vol. 39, pp. 86-94, December 2001.

[3] Clip2, "The Gnutella Protocol Specification v0.41", http://www9.limewire.com/developer/gnutella_protocol_0.4.pdf.

[4] S. Kasper and L. Buhrer, "Jini Discovers Bluetooth", http://www.tik.ee.ethz.ch/~beutel/projects/sada/2002ss_sa_vincent_bt_jini.pdf, Summer 2002.

[5] The *Jini™ Technology Surrogate Architecture Specification* version 1.0 Standard, http://surrogate.jini.org, October 2003.