An Enhanced Service Framework In MANETs For Application In Emergency Services

Monideepa Roy
Dept. of Computer Science & Engineering
Jadavpur University
Kolkata-700032
monideepa_roy@in.com

Pushpendu Kar,
IBM, Kolkata
pushpendukar@rediffmail.com

Nandini Mukherjee
Dept. of Computer Science & Engineering
Jadavpur University
Kolkata-700032
nandinmukherjee@yahoo.co.uk

***Abstract*—MANETs are increasingly being perceived as the preferred type of networks for emergency situations. In order to overcome the constraints of resources frequently faced by mobile devices in a MANET, a Jini based framework for an enhanced two phase service searching algorithm has been proposed in this paper. This enables even thin clients to perform advanced searches and find more accurate matches in local and remote locations in a MANET. The system also handles service delivery and instances of when the service providers or the leader nodes go out of range during an operation. The feasibility of implementation for this system is demonstrated by applying the framework to two instances of emergency services.**

Keywords-MANETs;Jini service matching; emergency services

I. INTRODUCTION

Well organized responses to emergency situations e.g. earthquakes, tsunamis, floods etc. involves efficient coordination and collaboration among public safety organizations and sharing of emergency alerts and incident related data between disparate systems. But with most wired and mobile communication systems requiring static/wired infrastructure support and interconnecting backbone networks for their smooth functioning, a disruption or failure of such services becomes unavoidable during such emergency situations. So if rescue groups from different locations and departments gather together to implement their respective rescue operations it is pertinent that these groups quickly establish networks among themselves, and exchange data, so as to complete their missions speedily and efficiently. So, it is essential to restore communication in those areas without depending on the conventional communication infrastructure. Under such circumstances MANETs have found wide applicability due to their ease of deployment.

MANETs are a special type of wireless mobile networks which do not require any infrastructure or central control nodes. Because of their unique characteristics of independent network organization, dynamic topology and with no central nodes, they find wide applications in the fields of emergency situations where temporary network establishment is a primary requirement. But the participating mobile devices in a MANET suffer frequently from constraints of resources.

In this paper we propose an enhanced Jini based service registration framework and a two-phase service matching algorithm for providing service users with a customized GUI and flexible matching for desired services that enables even the thin clients in MANETs to participate in complex searches or operations in collaboration with each other. The enhanced registration method enables the service providers to re-register themselves with the lookup service in case of temporary disconnections, along with their service related attributes. A preliminary filtering match is done by the lookup service on behalf of the client and the operation is performed on the services hosted on it, as well as on the nearby lookup services within range, after which a more refined matching is done by the client.

Jini is a programming model/infrastructure that enables dynamic deployment and configuration of distributed systems. The components may be physical devices or software objects (written in Java) that provide services over published interfaces. In Jini terminology such objects and devices are called services. Since Jini is basically intended for networks where the clients and the service providers come and go, therefore they are useful in a mobile computing environment, but it can also be used in any network where there is some degree of dynamic changes.

The rest of the paper is organized as follows: Section 2 describes the application scenarios of the Jini technology in emergency situations, Section 3 presents the system design and overview and Section 4 presents the implementation details along with a discussion on the efficiency of the two

978-1-4244-8270-2/10/$26.00©2010 IEEE

phase matching. Section 5 contains the conclusion and future work.

II. JINI TECHNOLOGY IN EMERGENCY SITUATIONS

The devices in a MANET often suffer from constraints of resources which prevent them from performing complex functions on their own. However, an environment where the nodes share their resources in a cooperative manner enables thin nodes to overcome these constraints and perform complex functions. Here we present a Jini based framework for a MANET where the service providers and service users come together in a collaborative manner. Jini [1] [2] is a service-oriented framework where the devices in a distributed environment can define, discover and advertise their services in an ad hoc manner. It provides a self healing environment where the services and the service consumers can enter and leave a network seamlessly.

The Jini technology consists of three protocols: discover, join and lookup. The discovery protocol is used when a new service provider joins a Jini federation and wants to register with a lookup service, and the service provider uses either unicast or multicast discovery to find a lookup service. The lookup service is the Jini service repository. The join protocol involves the service provider uploading a service object to the lookup service. Now as a result of the discovery protocol, the service is now ready to be discovered. Accordingly the client also searches for a lookup service and its desired service. In the final stage the client communicates directly with the service provider via the proxy object. However the search mechanism in Jini is not very advanced and returns only exact matches. In an instance of an enhancement to the Jini lookup service, XML based service templates have been utilized for the service matching to be flexible and include advanced service descriptors [3]. However XML file matching incurs considerable overheads and makes the system slower. While this may be traded to achieve search flexibility for Jini in wired networks, this will greatly inconvenience the already constrained MANETs even further. To reduce the overheads of XML file matching, the service attributes are stored within the service proxy in an array. To make the search faster, more user-friendly and extensive, in this paper we present an enhanced lookup service that provides the service providers and the clients with separate GUIs for submitting their specifications and performs a two phase search for retrieving the best matches from among multiple lookup services in the MANET. Since the initial filtering search is done by the lookup service, this ensures that even devices with limited capabilities can perform complex queries as the client performs the search here on a smaller data set. In addition to this, the framework also supports seamless re-registration of a service provider, if it temporarily moves out of range and then comes back. When a service provider moves out of wireless network range of a lookup service, it is automatically de-registered. On the re-entry within range of the service provider, it is automatically re-registered with the lookup services in range.

To illustrate the applicability of our framework, here we present an earthquake survivor management service and a doctor information searching service implemented using Jini over MANETs. The systems consist of several mobile nodes building a mobile ad hoc network with peer-to-peer connection between them, where the nodes are classified as service provider nodes, client nodes and leader nodes (which host the lookup services).

A. Doctor Search Application

In the first application, doctors with mobile devices are the service providers who are willing to host information of their professional services e.g. their schedules, on the network. As soon as they enter into a hospital environment, their mobile nodes search for leader nodes that are within network range. The leader nodes may be computational devices of higher configurations positioned at different departments. Once they find one or more such nodes, the service providers will register with their service attributes on a lookup service by randomly choosing any of the leader nodes. The doctors register with their current information through a doctor registration form which is a GUI. When a patient is in need of some medical help, the hospital staff or the patient's relatives can search for information about the location or availability of doctors, including searches for doctors registered on nearby leader nodes in the MANET. The client then submits the search criteria values according to his requirement e.g. specialization of doctor, location, fees, etc. through another GUI form, to the leader node. The lookup service running on the leader node matches the client's requirements with the service provider's information and performs a preliminary filter to send the relevant information to the patient. The patient application then unmarshalls the received data and then performs a more refined search with the rest of the parameters of location, area of specialization, fees etc. to retrieve the relevant doctors' name(s) who match his requirements. The patient then sends a request for booking to the doctor's device and receives a booking token number. The doctor can check for such requests from time to time and accordingly reschedule his operations if a request for a higher priority operation comes in.

B. Disaster Management Applications

The second application is for seeking out survivors or helping rescue operations as part of a disaster management system. When an area is struck by an earthquake, flood etc. essential communication is one of the first things to be affected, which is caused by the breakdown or disruption of the infrastructure support. During such times even the prevalent mobile network services may not work due to damage sustained to the static infrastructure of the base stations. So even if medical teams or relatives of victims physically reach the vicinity of the disaster hit area, it may not be feasible for them to visit each and every point to assess the degree of damages or the nature of help to be dispatched. Moreover, searching or locating survivors also becomes difficult and time consuming under such circumstances. In such cases communication among the various teams is crucial for the rescue operations to be fast and effective. So if the teams can somehow know the nature of help to be disbursed, the positions of the survivors, or

other such vital statistics it helps in gaining precious time. Mobile rescue team units carrying WiFi enabled devices can spread out in groups at the affected areas, acting as leader nodes. A client's application running on the survivor's devices at various locations can discover and register on their nearest leader nodes with their location name and other vital data e.g. name, age, blood group, location, type of aid needed etc. through a GUI. Other relief teams can then share this information as clients to disburse the necessary help to the respective areas. Similarly, relatives of victims who have come for inquiries can use this service network to retrieve information about their family members. The teams can send a response to the survivors stating the approximate time they will be reaching or other such relevant information/instruction.

III. THE SYSTEM ARCHITECTURE AND OVERVIEW

We propose an enhanced Jini based service framework for deployment in the emergency situations described above. In our framework, we classify the nodes of the MANET as *service providers, service clients* and *leader nodes*. The leader nodes host the middleware that comprises of lookup services along with three other modules - mRIM, mJERC, mSRD, and are generally assumed to be of slightly higher configurations than the other two types of nodes.

The details of the three components: a. *mRIM* (mobile Resource Information Manager) b. *mJERC* (mobile Job Execution and Resource Controller) c. *mSRD* (mobile Service Registration Directory) are explained later in this section.

A *service provider* is a node which is capable of (and is willing to) providing a service. A JINI service can be a computation, storage, a search, a hardware etc.

A *client* is a thin node which wants to execute a particular job but does not have the necessary resources to do so.

A *lookup service* acts as repository of services, where a proxy object of each registered service is stored with a set of attributes which define the service. The attributes used to describe the service vary depending on the type of the service provided. The entire system behaviour can be divided into the following stages:

A. Lookup Discovery and Service Registration

To make a service available to other devices on a MANET, a service provider has to register it on a lookup service. For this, the service provider performs multicast discovery to search for nearby leader nodes in a network [4][9][10]. When it finds one or more leader nodes, it sends a request to a randomly chosen node for registration. The choice is made randomly, because otherwise there is a chance that the leader node which has a stronger signal strength than the others will be discovered by most of the service providers every time thereby leading to a resource drain on that node. If the request is approved, then the service provider registers its services on that node along with its service attributes for some duration of time decided by the leader node, which is called the 'lease time' [7][8]. The entire process of discovery and registration by the service provider is done through a GUI. The service attributes are stored within a copy of the service proxy on the lookup service. The *mRIM* acts as the intermediary for storing information of the service provider attributes which it forwards to the *mJERC*. When a service provider submits its service specifications through a GUI, the *mRIM* component receives the information and stores it in the *SST* (Service Specification Table).

B. Leasing

The duration of the lease to be granted to a service provider is decided by the leader node. Before the expiry of the lease, the service provider has to request the lookup service to renew the lease. If the lease is renewed for an additional duration of time, then the service provider can continue to be hosted on the leader node. Otherwise the service attributes are flushed out from the lookup service. The service provider can also opt to cancel a lease at any time if it wants to.

C. Client side registration and query submission

The client nodes also subsequently search for nearby leader nodes and register with them to place their service requests. The mSRD maintains the client requests for services for the matching. The clients enter their service requests through another GUI and the details are stored by the mSRD in the *SRT* (Service Request Table). The mJERC receives the client requests and the service provider specification information from the mSRD and the mRIM respectively.

D. Two Phase Service Matching

A service is defined by a set of k attributes, $A_1,A_2,A_3,\ldots,A_k$. The service attributes of all the services are marshalled and stored in the SST. For each request in the SRT, the mJERC performs an initial filtering of the services from the available ones by matching based on only one attribute, usually A_1 which specifies the type of the service. These filtered services are stored in the *FSST* (Filtered SST). The data from the FSST is now unmarshalled and sent to the client for storing in the *ESST* (Expanded SST). The client now performs a second phase matching to get the final matching services in the *MRT* (Matching Requests Table). The client then accesses the services of its choice. If no matching service is found, it displays a "no matching services found" message. Fig. 1 represents the complete system architecture.

IV. IMPLEMENTATION DETAILS

Our proposed framework has the following deliverables:

i. When a service provider moves out of range of a leader node after initially registering on it, it is de-registered, but is automatically registered again on its subsequent returns within range of a leader node.

ii. The service provider registration and the client side service request submission entries can be made through GUIs.

iii. A two phase service matching strategy has been implemented for matching service requests with the services available.
iv. The network wide search is performed and results from among all the leader nodes within range in the MANET that are hosting services are retrieved.
v. When a client moves out of range after submitting a request, it receives the response when it comes back within range again.

This system was implemented in a mobile ad hoc network. There is no central base station like an access point or a router. In this network the mobile nodes communicate with each other via peer-to-peer connections. To construct this network for the system each mobile node needs to be equipped with a wireless Ethernet cards. The operating system of each node needs to be configured in such a manner that the nodes can establish peer-to-peer connections between them. In this system the Windows XP operating system was used on every node for configuration and establishment of a mobile ad hoc network.

Since the basic framework is the same for both the proposed applications, so we describe the implementation details of both of them together along with screenshots of both the applications: The devices communicate with each other using D-Link AirPlus DWL-520+ Wireless PCI Adaptor cards. (Intel® 82566DC Gigabit – Network connection, 100Mbps/full duplex). The agent diagram of the overall system is shown in Fig. 2.

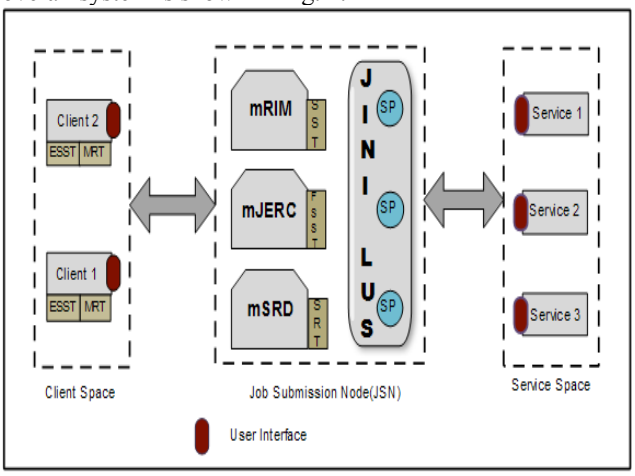

Figure 1. The system architecture

A Service Provider Implementation

Any person wanting to participate in the system and willing to offer some service has to carry a mobile device (e.g. laptop, Pocket PC, Palmtop etc.) which is Java and Jini enabled. When a service provider enters within a MANET and wants to register his service, he/she needs to fill up a service registration form and click on the 'Register' button to register his service with the leader node. The service attributes for the doctor service are name, specialization, location, schedule, address and fees. The service attributes for the disaster management service are name, age, sex, blood_group, location and aid-type. Fig. 3 and 4 show the screenshots of the forms for the two applications.

The registration process involves the following steps: *Discovery***:** The Service Provider (SP) searches throughout the MANET for a leader node through multicasting to receive the location of a leader node.

- *Marshalling:* The SP marshals the service attributes of the doctor's information with the service proxy.
- *Join:* The SP then sends the service proxy to the lookup service and requests for a required amount of lease period. On expiry of the initial lease at a later stage the service provider may request for a renewal of the lease if it finds necessary.

Since all the nodes in the network are mobile, it may often happen that the participating nodes may move out of range of each other. So the MANET should be equipped to handle such cases.

If at any time a service provider wants to register with a network but is not within network range of any leader node, the console will show a message "Outside the lookup server range or lookup server is down".

If after some time the SP enters within the network range again, then the service program will search for leader nodes and if it finds one or more such nodes, then the SP will become automatically registered on a leader node randomly chosen from them.

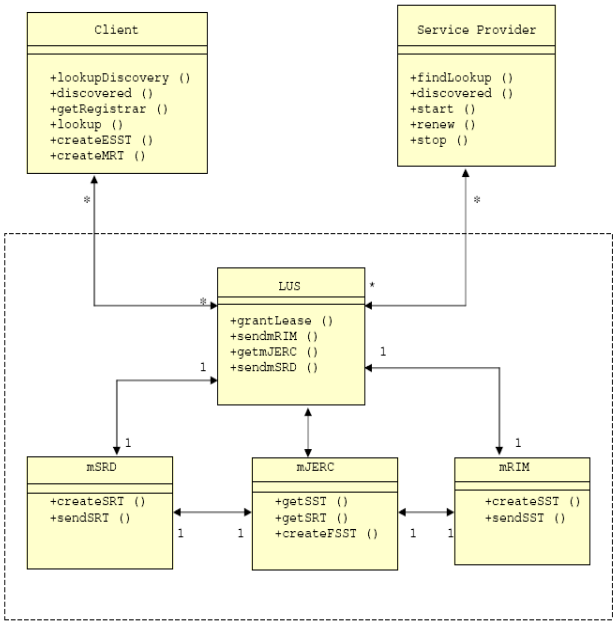

Figure 2. The agent class diagram

When the service provider is moving around inside the network and if he/she goes outside the communication range of a lookup service, the system will automatically detect this and unregister that service provider from the lookup service.

Accordingly if that service provider moves back into the range of communication of a lookup service again, he/she

will automatically become registered to the lookup service as before.

During the process of registration of a service provider, it acquires a lease from the leader node for a specified period of time, during which the service provider is expected to remain registered to that leader node and a service proxy object of the service provider is created and hosted on the lookup server during that time.

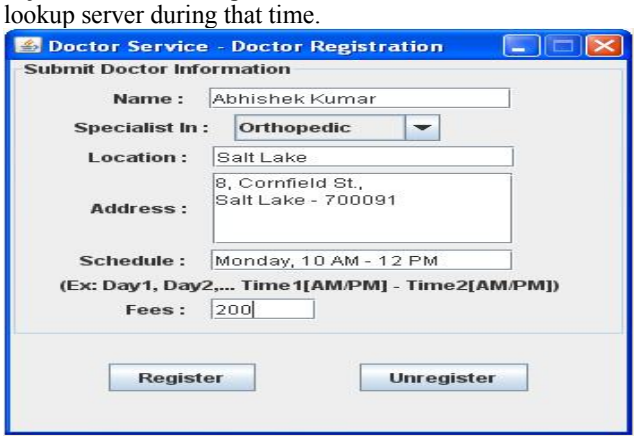

Figure 3. Service Provider Registration form for doctor application

But the service provider may decide to stay on in the network for a longer time than the original lease time. So to remain registered on the network the service provider has to renew the lease before the expiry of the previous lease.

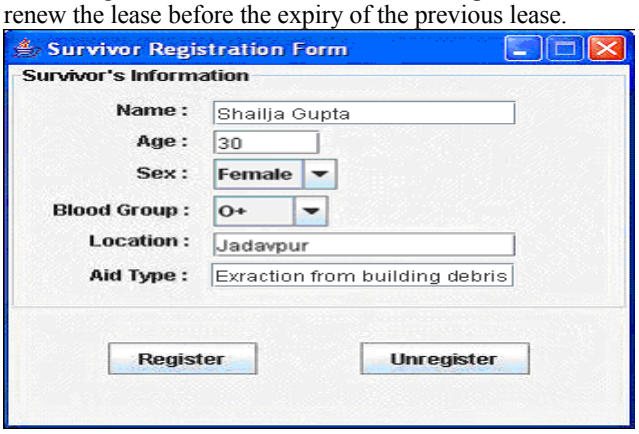

Figure 4. Service Provider Registration form for disaster management application

In both the applications, the leader nodes are also expected to move around from time to time. If a leader node moves out of range while a service provider is registered with its lookup service, then the service provider begins a new search for alternative leader nodes within range. When it finds such nodes within range, it will automatically register again on another randomly chosen leader node.

B. Client Implementation

Persons searching for doctor information and the rescue teams trying to retrieve disaster statistics are the clients in the system. The clients are also equipped with Java and Jini enabled handsets. When a client wants to search for any information, he/she fills up a query form through a client GUI and clicks on the 'Search' button to submit the query, Fig. 5, 6. The information retrieval involves the following steps.

- **Discovery:** The client also searches the network for a leader node for sending a service request to it.
- **Receive:** The lookup service of the leader node sends a registered service proxy to the networked client.
- **Unmarshalling:** The clients' unmarshall the service proxy for retrieving the service provider's information.
- **Matching:** The matching of the client request with service provider's attributes is done through a two phase matching procedure which is explained as follows:

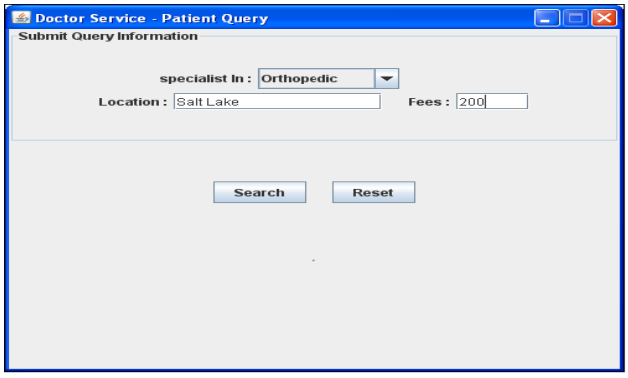

Figure 5. Client query submission form for doctor service

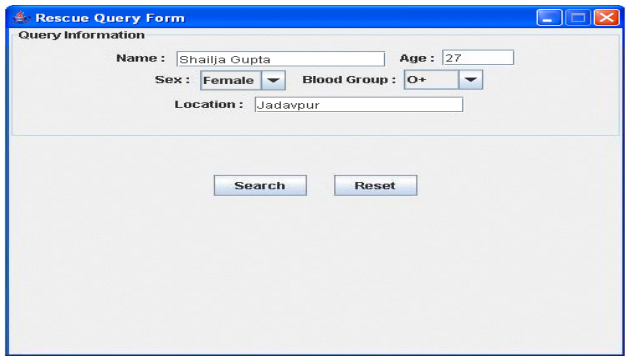

Figure 6. Client query submission form for disaster management service.

C. The Two Phase Matching Procedure

We explain the two phase matching procedure with reference to the doctor information matching application and compare it with a traditional attribute matching method. Attributes are used to specify the particulars of the service. In Jini, attributes are not name/value pairs; they are classes that implement the *net.jini.core.entry.Entry* interface. While the lookup service will support any attribute that is an Entry object, the following attributes are provided by the Jini package: *Address, Comment, Location, Name, ServiceInfo and ServiceTypeStatus*. We have used two attributes, *Name* and *Comment*. The name attribute contains the service name and the comment attribute contains the name, schedule,

location, address, and fees, Fig. 7. Using these attributes helps us for not having to create a new class type. The method is explained in more detail later in this section.

Now we proceed to show that our proposed two phase matching method performs better than a traditional matching. First we consider and calculate the total time elapsed if a regular matching is performed on all the services with the full set of k service attributes, represented as the k-tuple $T = (A_1, A_2, A_3, \ldots, A_k)$, i.e. $t_1 = (a_1(t_1), a_2(t_1), a_3(t_1), \ldots, a_k(t_1))$, $t_2 = (a_1(t_2), a_2(t_2), \ldots, a_k(t_2), \ldots, t_n = (a_1(t_n), a_2(t_n), \ldots, a_k(t_n))$.

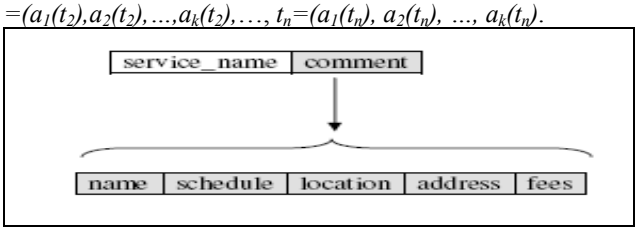

Figure 7. Parsing the comment field

The client query is represented by the k-tuple $Q = (q_1, q_2, q_3 \ldots q_k)$ [5] [6]. We consider the code snippet performing the regular search for this. Let the total number of services available in a federation at any time be n. The maximum number of fields to be matched is k for a service, (for our applications we have $k=6$), and we assume that the time for performing a single match is t seconds, then the maximum time for completing the entire matching will be

$T_R = k \times n \times t$ seconds.

Now we consider the two phase matching algorithm. For the two phase method, in the initial stage the service attributes are stored as a 2-tuple T = (*service_name, comment)* in the service proxy. The lookup server performs a preliminary matching based on the service_name field only and filters out the comment field data. The client then unmarshalls the comment field from this table to get the (k-1)-tuple C = $(A_1, A_2, A_3, \ldots, A_{k-1})$, and performs the second matching on these fields to retrieve the final matching result, Fig-9.

```
...
for i = 1 to n
   {
     if (( q1 = a1(ti )) && ((q2 = a2(ti ))...
&& (qk = ak(ti )) then
       filter and retrieve all matches;
...
   }
```

Figure 8. Code snippet for the regular matching

We now represent the service attributes as the 2-tuple $T = (S, C)$ i.e. $t_1 = (s_1, c_1)$, $t_2 = (s_2, c_2) \ldots t_n = (s_n, c_n)$ where the first component S represents the name of the service and the second component C represents a comment. The comment component is later parsed and represented as an (k-1)-tuple $C = (A_1, A_2, A_3, \ldots, A_{k-1})$, i.e. $c_1 = (a_1(c_1), a_2(c_1), a_3(c_1), \ldots, a_{k-1}(c_1))$, $c_2 = (a_1(c_2), a_2(c_2), \ldots, a_{k-1}(c_2), \ldots c_n = (a_1(c_n), a_2(c_n), \ldots, a_{k-1}(c_n))$. For our application, there are 5 attributes i.e. the name of the doctor, his area of specialization, location of the doctor, his schedule, his address and consultation fees respectively.

Here the client query is represented by the tuple $Q' = (q_1', q_2')$. In the first phase of the proposed two phase method, an initial filtering is done on the n services based on only the first field, the *service_name*. The code snippet for this is shown in Fig. 8, and the time for its completion is $T_1 = n \times t$ seconds. The number of the relevant services on which the rest of the (k-1) fields are to be matched, now reduces to m, where $m \leq n$.

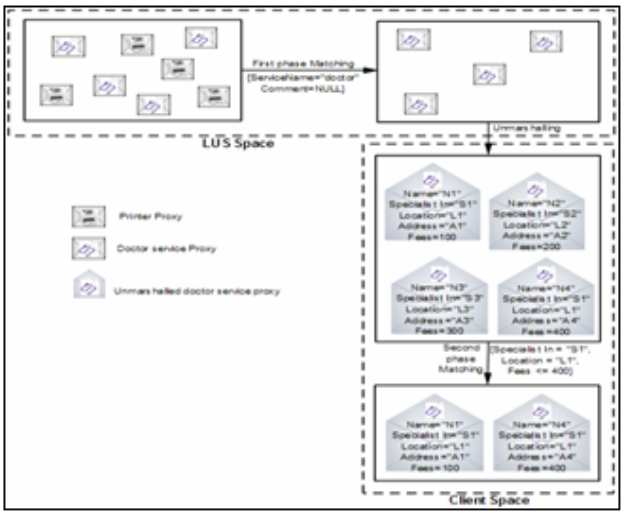

Figure 9. The two phase matching procedure

```
...
for i = 1 to n
  {
    if ( q1' = si ) then
      filter and retrieve all ci;
  }
...
```

```
...
for i = 1 to m
   {
     if (( q1 = a1(ti )) && ((q2 =
a2(ti ))... && (qk-1 = ak-1(ti )) then
       filter and retrieve all matches;
   ...
   }
```

Figure 10. Code snippets for the two phase matching

In the second phase the matching is now done on the remaining *(k-1)* fields. The query Q is represented by the (k-1)-tuple $Q = (q_1, q_2, q_3 \ldots q_{k-1})$. As shown in Fig. 10, the time for completion of the second phase is $T_2 = (k-1) \times m \times t$ seconds.

So the total time for the completion of the entire search is:

$T_{TP} = T_1 + T_2 = n \times t + (k-1) \times m \times t$ seconds.

We proceed to show that the proposed two phase matching method performs better than the regular method.

$$T_R / T_{TP} = \frac{k}{1+(k-1)\times(m/n)}$$, where $$\frac{k}{1+(k-1)\times(m/n)} \geq 1$$

since m is always $\leq n$, so m/n is always ≤ 1. In the extreme case that $m = n$, T_R will be equal to T_{TP}. For all other cases, T_R will essentially be always $\geq T_{TP}$.

In the query submission process if client does not specify any values for the 'Specialist In', 'Location', 'Fees' fields, then he/she will get the information of all the registered doctor's throughout the network, Fig-11, 12. He/she can also narrow down the number of retrieved matches by setting the values to all or some of these query fields to find out only the information inquired for.

If the query information does not match any of the currently registered doctor's information, it will show the message "No matching doctor with the requirement". Otherwise it will show the matching doctors' information. A client can participate in the system and access the services only when he/she resides within the range of communication of at least one lookup service, otherwise it will show a message indicating that he/she is outside the range of communication of the lookup service.

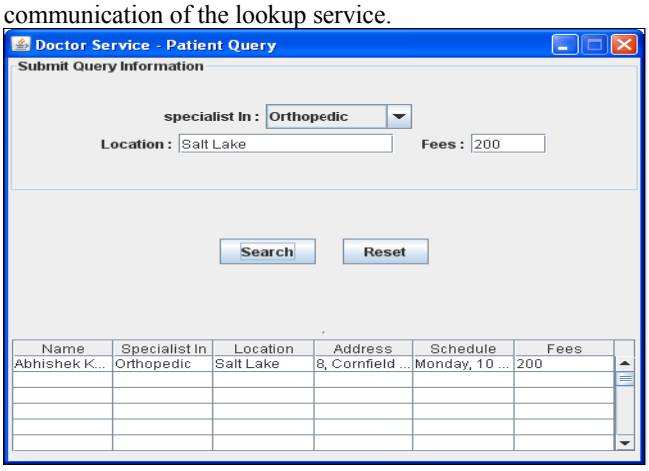

Figure 11. A snapshot of query submission form showing Doctor's information matches with the requirement

The mJERC performs a first phase filtering on the SST based on the service_name field of the SRT and stores the matched data in the *FSST* (Filtered SST). The data from the FSST is now unmarshalled and sent to the client and stored in the *ESST* (Expanded SST). The client now performs a second phase matching of the ESST with the rest of the attributes of SRT to get the final matching services in the *MRT* (Matching Requests Table).

D. Leader Node

The Lookup service consists of the crucial part of the Jini network technology. The Lookup service runs on the leader node. There may be more than one leader node in the Jini federation. The Lookup service hosts the service proxies of the registered service providers and filters and delivers responses to the clients within the range of communication of the lookup service. Each service proxy stores the information of one service provider as service attributes. A client needing to access a service first finds out a lookup service by multicasting a discovery message and then accesses the service directly after getting the required service proxy for that service from the lookup service.

V. Conclusion & Future Work

These applications have been implemented at present within a limited network range area consisting of 7-8 nodes. We have used JMatos which is a lightweight Jini lookup service suitable for installing on resource constrained devices. We plan to scale up this framework in the future on a wider network coverage area comprising a greater number of nodes. Secondly, in this framework if the client nodes move out of range after submitting a request then it may not receive the response sent back from the leader node. We propose to handle this by storing the responses of the clients for specified periods of time for later retrieval by the clients.

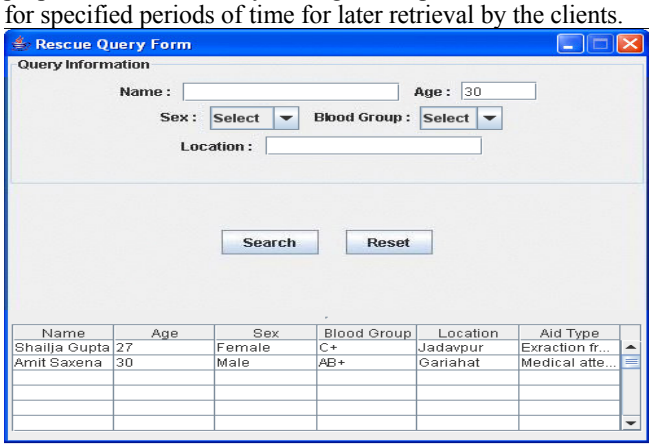

Figure 12. A snapshot of query submission form showing Victim's information

References

[1] Sun Microsystems, "Jini Discovery and Join Specification", version 1 .O. 1, November 1999.

[2] J. Waldo, "The JiniTM architecture for network-centric computing", Communications of the ACM, July 1999.

[3] M.B. Meller, B. N. Jorgensen, "Enhancing Jini's Lookup Service using XML-based Service Templates", Proceedings of the Technology of Object-Oriented Languages and Systems, 2001, ISBN:0-7695-1095-7.

[4] F. Zhu, M.W. Mutka, and L. M. Ni, "Service Discovery in Pervasive Computing Environment", IEEE Pervasive Computing, October-December 2005, pp. 81-90.

[5] V. Srinivasan, S. Suri, G. Varghese, "Packet Classification using Tuple Space Search", ACM SIGCOMM Computer Communication Review Volume 29, Issue 4 Oct.1999.

[6] R. C. Veltkamp, "Shape Matching: Similarity Measures and Algorithms", Proceedings of the International Conference on Shape Modeling & Applications table of contents, Page: 188, 2001, ISBN: 0-7695-0853-7.

[7] K. Bowers, K. Mills and S. Rose, "Self-Adaptive Leasing for JINI", PerCom 2003: 539-542.

[8] S. Rose, K. Bowers, S. Quirolgico, and K. Mills, "Improving Failure Responsiveness in Jini Leasing", DISCEX (2) 2003: 103-105.

[9] Hsu, Kuo-Wei, "Design a Jini-based Service Broker for Dynamic Service Combination Framework", Journal Of Software, VOL. 1, No. 3, September 2006.

[10] T.Erl, Service-Oriented Architecture: Concepts, Technology, and Design, Prentice Hall PTR, August 2005.