Attribute Based Addressing for SIP

Vlasios Tsiatsis †*, Jyh-Cheng Chen ‡, Prathima Agrawal ‡, Mani Srivastava †

† Electrical Engineering Department, University of California at Los Angeles
Los Angeles, CA 90095-1594
{tsiatsis, mbs}@ee.ucla.edu

‡ Applied Research, Telcordia Technologies, Inc.
Morristown, NJ, 07960-6438
{jcchen, pagrawal}@research.telcordia.com

***Abstract*—The Session Initiation Protocol is used for setting up multimedia sessions among users in such a way that personal mobility is handled by the use of one identifier to distinguish among users. This identifier in spite of being convenient and simple does not fully describe the comminicating entities in terms of their capabilities and their characteristics. In this paper we address the problem of describing the communicating parties with multiple attributes in the context of SIP. We present the architecture of the prototype system and its integration with the SIP protocol. The SIP protocol regards the location service as an orthogonal part of the architecture and does not deal with this issue. However the architecture of the location server plays an important role in the operation of the protocol. Based on the experience gained from the operation of prototype system we present some enhancements in terms of the interoperability of the location service and the SIP server.**

***Index terms*—session initiation protocol, attribute based addressing.**

A. INTRODUCTION

The Session Initiation Protocol [8] utilizes a single user identifier similar to an e-mail address in order to distinguish among the different users. The user contact information for the session setup consists of a user name, the machine name and port number where the SIP client runs on. The machine name can be either its IP address or its fully qualified domain name. Thus the SIP contact information is already a collection of attributes with the machine name being the one, which may need a translation from a fully qualified domain name to an IP address. The need for this duality in machine identifiers is a result of the difficulty faced by humans to associate a device with a string of numbers (IP address) and to be able to use this address when referring to this machine. The fully qualified domain name enables a more natural description of a device, which is easy to memorize and associate as well.

This naming scheme greatly reflects the location of the networked entities because the primary concern in the early networks was to route information from one entity to another. And since there were not many networked entities the IP address naming scheme worked perfectly all right.

Moreover, the naming scheme used today although simple[10] gives little information about the services that the networked entities provide. The main concern of a user is to use the network, which is attached to his/her machine to perform a task. As a result the user focuses on the functionality that the network can provide and is not willing to make yet another association of an identifier to a desired service.

With the advent of wireless networked machines the need to keep track of the machine location led to patching the existing naming and routing schemes with location directories. These are essentially associations of the unique machine identifiers with new identifiers, which reflected the current machine location.

As the computers and networked devices became smaller and smaller and they also become wireless, the need for new naming schemes became evident. This is because on the one hand the existing schemes are running out of names, and on the other hand they were designed to facilitate the communication of a relatively small number of machines.

Location is only one attribute of an entity that might change. Several other entity attributes may change over time, but existing naming and addressing schemes do not have efficient ways of addressing this problem.

In this paper we address the naming problem providing a naming scheme and a location service for the Session Initiation Protocol [8]. The proposed scheme enables a more flexible and natural user discovery by taking into account more information about the user and his/her context than only a single identifier. This is done by describing a user with a list of attribute value pairs along with the SIP user and contact information.

The paper is organized as follows: Section B describes the motivation behind the proposed architecture. Section C introduces the architecture for our location server. Section D elaborates on the implementation. Section E describes some related work. Section F includes the future work while section G concludes the paper.

B. MOTIVATION

There are three categories in which we can identify the benefit of using an attribute based naming scheme to describe a user:

1. First of all the number of communication devices that are connected on a network as well as the number of users who tend to be mobile is increasing. As a result the identifiers used to distinguish either the devices or the people will become too large that other people will be unable or unwilling to use them because they don't know them or they don't want to know them. All that people

* Part of the work was performed when the author was with Telcordia Technologies, Inc.

want is to be able to address other people or devices by what they are or represent (e.g. PDAs with audiovisual capabilities, nearest policeman).

2. Often people want to communicate with other individuals or groups of people but they only know some characteristics of either an individual or the group. It is more natural to address these people with a description rather than an identifier (e.g all interns in a company).
3. Finally, what it is desirable to have, is not only a service that plays a role similar to the Domain Name System Service. On the one hand it would be good to have a network of servers, which have the ability to resolve network identifiers from high-level descriptions of entities. This kind of service would be used for name resolution for path setup for call setup protocols over IP networks (e.g. SIP) or it could be used for maintaining profile information for users with changing profiles. On the other hand the need for a service that enables users to push their own data to other users that have some common characteristics is gaining interest in the research community. This is what we call profile-based services. One example of this service is location dependent queries for user discovery. Another example is short messages sent to users that satisfy some criteria.

C. ARCHITECTURE

1st. Overview of SIP

Since our attribute-value based naming system assists the Session Initiation Protocol (SIP) we present a brief introduction to the protocol.

The SIP is an application level protocol that can establish, modify and terminate multimedia sessions or calls. SIP enables the concept of *personal mobility* by providing name mapping and redirection services. Personal mobility is the ability of users to originate and receive calls on any terminal in any location and the ability of the network to identify end users as they move [12].

The main SIP architecture includes two major components. The first component is the SIP client, which is an application that provides the user with the ability to initiate (INVITE requests) and terminate (CANCEL requests) calls to other SIP clients as well as to register (REGISTER requests) to a SIP server. A SIP Server maintains the SIP registration records for a limited amount of time (soft state registrations) and can setup connections between clients. When a SIP client registers with a SIP server it indicates the user name with which it is going to be known to the SIP protocol along with contact information used for setting up the connection. This contact information includes the user name that runs the SIP client, the machine name and the port number that the SIP client application runs on.

Although two SIP clients can communicate directly without the interference of a SIP server, the server is needed for user registration and call setup. A SIP Server can be configured to operate in one of two modes:

a) Redirect Server Mode (Figure 1)

In this mode a client john@hol.gr sends an invitation request to the server indicating the SIP identifier URI (uniform resource identifier) of the other communicating party helen@ucla.edu. The SIP server contacts a location server and determines if there is a registered user under the requested SIP identifier. If such a user exists, the SIP server returns the contact information of client helen@ucla.edu to client john@hol.gr. In general the SIP identifier and the contact information are not identical. This is due to the fact that a user may want to be known under the SIP protocol by one name while his/her contact information is changing according to his/her current location. When finally the client john@hol.gr receives this information it initiate an invitation request to client helen@veria.ee.ucla.edu directly. If the other party accepts the call an acknowledgement is sent back to the originating client and a real-time session can begin between the two parties. The type of initiated session depends on the device or the SIP capable software. For hardware SIP phones [18] only audio sessions can be initiated while the Columbia software SIP client [19] supports the initiation of real-time audio and video sessions.

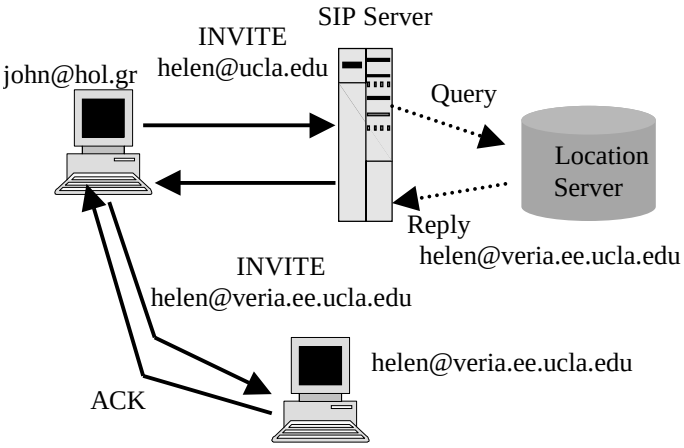

Figure 1. SIP Redirect Server

b) Proxy Server Mode (Figure 2)

In this mode the SIP server does not return the contact information of client helen@ucla.edu to client john@hol.gr but instead sends an INVITE request to client helen@veria.ee.ucla.edu on behalf of client john@hol.gr after contacting the location server. When client helen@veria.ee.ucla.edu accepts the invitation for a connection an acknowledgement of the acceptance is sent to client john@hol.gr. After that, the two communicating parties are ready to initiate a real-time session between each other.

One of the main operations that SIP supports for multimedia real-time setup is location discovery for users. The SIP documentation explicitly mentions that the interaction of the SIP server with the location server is not in the scope of the protocol. However this does not mean that the location server is not an important part of the SIP architecture. We judge that enhancing the intelligence of the location server is crucial for locating users with specific characteristics and this is main the focus of this paper.

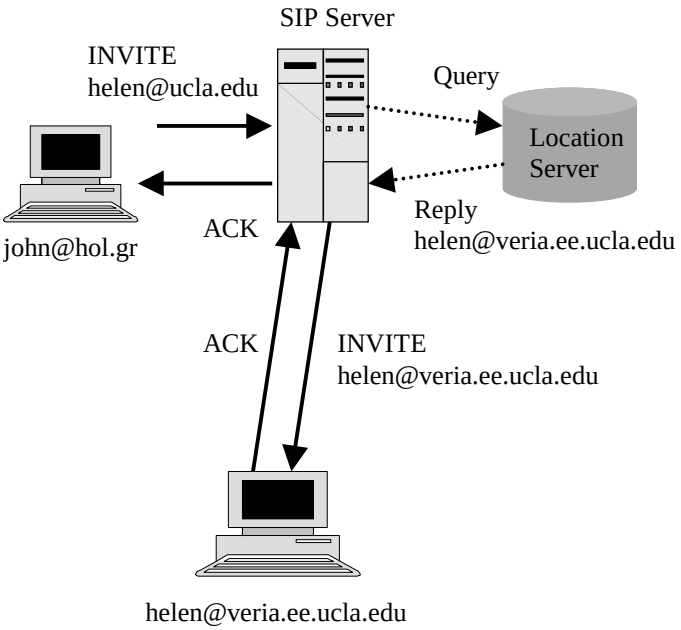

Figure 2. SIP Proxy Server

2nd.Attribute Based Location Server

We considered the implementation of our attribute based location server in two forms. A centralized one, which is used by all the SIP servers to register the SIP clients, and the distributed version, which consists of multiple location servers interconnected together. Each SIP server receives location services from only one distinct location server.

1) Centralized Approach

The overall architecture for a centralized name resolution server of this system is shown in Figure 3. The name server consists of a Database Engine, and a number of Client Communication Components. The number of client components depends on the number of active clients since each time a new connection occurs, a new client component is spawned to provide attribute based naming resolution services to the requesting client. The clients request services from the server using messages formatted in a specific way described later in the paper. The client components communicate with the database engine to update and query the attribute value pair database.

The database engine consists of a Client Request Engine, which is responsible for client queries and a Soft State Manager, which is responsible for maintaining the soft state of each database record. For each record in the database, along with the attribute value pairs that characterize a network entity, there are additional record fields, used by the entities to communicate with other entities. In the case of the SIP protocol the additional information is the SIP contact identifier.

The centralized approach is suitable for isolated environments where the communication traffic to the server is light; otherwise the server becomes the bottleneck of the SIP protocol.

The more interesting approach to architect the location server is the distributed case.

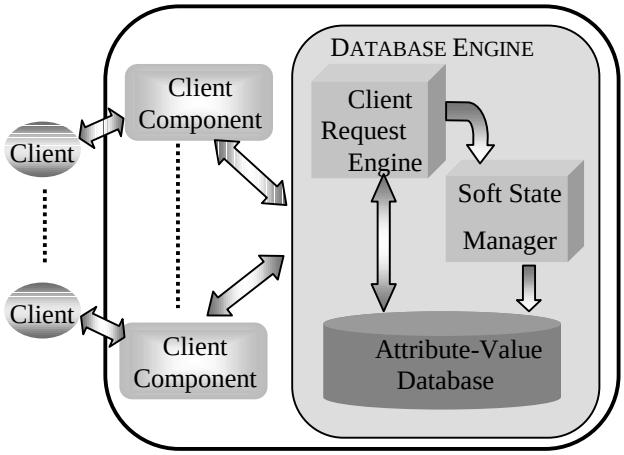

Figure 3. Centralized Server

2) Distributed Approach

In the distributed version of the name resolver different resolvers are connected with each other to provide distribution of the database information and higher availability to the user. The database information can be either replicated in every server or in a set of servers. The replication occurs when an entity registers and updates the registration information. The registration and update messages are then forwarded to every eligible server and result in changes in the database of each server. An eligible server is the one within the scope of a registration or update message. The types of messages as well as their formats are described later in the paper. The way the different servers are connected together depends on the configuration that each server obtains by contacting a server that maintains neighborhood information.

When each server is started, it contacts another server that maintains the connectivity list of the distributed network. The connectivity list is used for forwarding the subsequent messages to the rest of the network. The internal server architecture is depicted in Figure 4.

As in the centralized server case, there is a client component, which is responsible for servicing client requests and a database engine responsible for accessing the local database of attribute value pairs. The additional components shown in Figure 4 enable the server-to-server communication.

There is one packet receiving and one packet sending component for each server. The messages exchanged between the servers carry the appropriate source and destination identification to assist those two components in their operation.

The Duplicate Elimination Tables are tables that maintain information about the processed packets already received or sent so that duplicate messages are not processed again. Duplicate packets can be received when more than one other name servers can contact one server. The list of the name servers that the current one can contact is maintained in the Neighbor Table.

The Open Client Connections is a table that maintains inoformation about the client query requests in progress. The query processing in the distributed case resembles the

recursive query processing in DNS [11]. All the DNS servers are hierarchically connected together according to the domain name they reside in. When a DNS server cannot resolve the requested domain name, it forwards the query to another DNS server that can handle the request. In the meantime the client waits until the query reply comes back through the reverse path. In our case the query is also recursive and as a result all the information about the waiting client must be stored in the Open Client Connection Table so that the query reply is forwarded to the appropriate client.

More details about the server-to-server communication will be given in the next section.

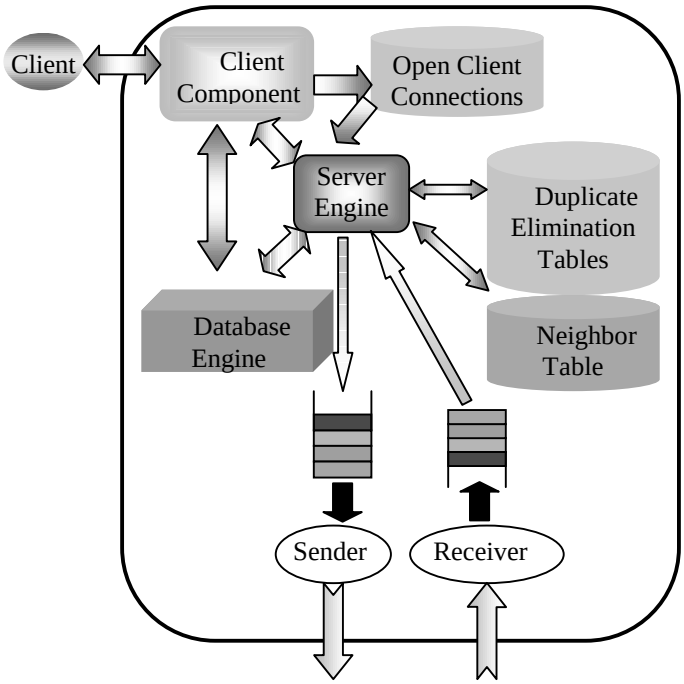

Figure 4. Distributed Server Case

D. IMPLEMENTATION

The implementation of the attribute based naming server was done in Java in order to provide a portable proof of the concept of attribute based naming of networked entities. The integration of the naming server as a location server for SIP was performed using the SIP client and server implementations from Columbia University [19]. Our goal was to provide an implementation that would help us understand the problems faced by introducing such a service not only to SIP but other environments as well[14].

1st. Database structure

The database of the entity descriptions consists of a list of records, which contain the following information: an entity identifier which acts as a handle for update operations, a list of attribute value pairs, some entries for database maintenance (record number, timestamp, expiration time), the IP address and the port number of the entity and a user name, which is associated with this entity (in our case the entity is a SIP user). The structure of a record is shown in Figure 5. The list of attribute value pairs consists of a list of triples (AttributeName, Operator, Value). In the current implementation every triple has the equality operator ("=") as its operator. However an entity can also have an attribute described by using other operators. An example is the range operator used to describe that a user's work hours are from nine to five.

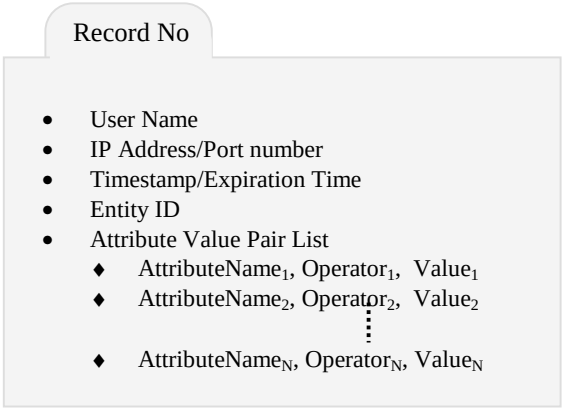

Figure 5. Database Record

2nd.Message Types

A client connecting to an attribute based name server can perform the following operations:

1. Registration
2. Signoff
3. Description Modification
4. Query
5. Application Level Forwarding

Clients request each of the above operations by connecting to the name server and sending a message formatted in a specific way. In the current implementation the message is send as a string of characters without any coding, for compression or encryption purposes. The first part of the message contains the identification of the source of the message in terms of a user name, an IP address, and a port number and a keyword that distinguishes the type of the message. The next part is a field (time-to-live field) that facilitates the scoping of the messages in terms of number of server hops that the message can travel. The structure of the rest of the message depends on the message type.

1) Registration

A sample registration message is shown in Figure 6(a). This type of message is characterized by the keyword **register,** which is followed by the list of attribute value pairs that the requesting entity possesses. When the registration of the entity is completed the name server gives the client a handle for that entity. This handle acts as a certificate that the client shows to the name server when the former requests an update of the registration record. This handle is a string of hashes concatenated together. These hashes are produced by the fully qualified domain name of the server, the originating user information, the register request string and a server sequence number. The possible types of updates are the signoff of an entity from the database, and the modification or deletion of a set of the entity attributes.

2) Update Operations

Examples of a signoff message and attribute deletion and modification messages are also shown in Figure 6(b)-(d). The only information that a signoff message has to carry is the record handle returned by the registration request. An attribute deletion message carries the identification of the user whose description needs to be changed and the names of the attributes that need to be removed. An attribute modification message contains the user record handle and a list of attribute value pairs. The attribute names of the pairs may or may not exist as part of the original user description. In the former case the attribute value is changed, while in the latter case the new attribute value pair is inserted in the user description.

```
(a) src(vlasios@veria.ee.ucla.edu:4443)
    ttl(2)
    register
    #a=1#b=2#c=3#
(b) src(vlasios@veria.ee.ucla.edu:4443)
    ttl(255)
    signoff userid (1345_345_224_1)
(c) src(vlasios@veria.ee.ucla.edu:4443)
    ttl(255)
    del_attr userid(1345_345_224_1)
    #a#c#
(d) src (vlasios@veria.ee.ucla.edu:4443)
    ttl(255)
    mod_attr userid (1345_345_224_1)
    #a=3#d=4#
```

Figure 6. Message examples

3) Query

A client can issue a query, which is, in general, a combination of primary queries. A primary query consists of an attribute name, a matching operator and an attribute value. An attribute value can be either a number or a string of characters.

The primary query operators are:

1. exact match operators(Attribute = Value),
2. range operators ($A_1>V_1$, $A_2>=V_2$, $A_3<V_3$, $A_4<=V_4$, A_5 @$[V_5$-$V_6]$),
3. set operators (A @$[V_1, V_2, .., V_N]$) and
4. the wildcard operator (=*)

The notation A_5 @$[V_5$-$V_6]$ means that the primary query will be satisfied for those entities that have the attribute A_5 in the range $[V_5, V_6]$. The notation A @ $[V_1, V_2,.., V_N]$ means that the primary query will be satisfied for those entities that have an attribute A belonging into the set $[V_1, V_2, .., V_N]$. A number of primary queries can be combined together, using AND/OR operators together with parentheses to construct composite queries. An example of a composite query is presented in Figure 7. A client that issues this query is looking for a policeman or a firefighter at Zimbabwe.

A query message has the keyword **query** followed by a query modifier and the actual composite query. The query modifier modifies the output of the actual query. The output of the actual query is a set of records that satisfy the query. Out of these records the user may choose to filter these records so as to get any of the records, or all the records or the record, which describes an entity, that is nearest to the client that issues the query. The modifier used in each case is ANY, ALL, NEARESTFROM respectively. In the example in Figure 7 the user requests that all the matching records be returned in the query reply.

```
(a) src(vlasios@veria.ee.ucla.edu:4443)
    ttl(255)
    query
    all
    location = Zimbabwe
    AND
        (occupation=firefighter
        OR
        occupation= policeman)
```

Figure 7. Query Example

In order for the NEARESTFROM modifier to have a meaningful output the users must also register their locations. In our prototype implementation for demonstration purposes we assume that the location of a user is given as a triple of the x, y and z coordinates of a coordinate system whose origin is at the center of a city where the SIP network of servers is employed. Without loss of generality the location of a user can be an arbitrary character string (e.g. Room 1045) as long as there is a well-defined ordering function among those strings in terms of the location.

The location attribute is useful when the entities are spatially distributed over an area of interest or the client is interested in having location dependent services as in the case of a query with the NEARESTFROM modifier.

Query → SetModifier BooleanQuery

- SetModifier →
  - any
  - all
  - nearestfrom(x,y,z)
- BooleanQuery → ANDquery (OR ANDquery)*
- ANDquery → PrimaryQuery (AND PrimaryQuery)*
- PrimaryQuery →
  - (BooleanQuery)
  - ID (= | > | >= | < | <=) (ID | NUM)
  - ID @ [(ID |NUM) - (ID |NUM)]
  - ID @ [(ID |NUM) , (ID |NUM), ... , (ID |NUM)]

ID : character string, NUM : number

Figure 8. Query Grammar

3rd. Query Language

The query grammar is depicted in Figure 8. A query is preceeded by a query modifier called SetModifier because it determines the output of the query, which is in general a set of matching records. The actual query can be viewed as an expression tree with the operators AND and OR as the non-leaf nodes and PrimaryQueries as leaf nodes. An example of an expression tree is shown in Figure 9. A PrimaryQuery can be one of the four types of primary queries depicted in Figure 8.

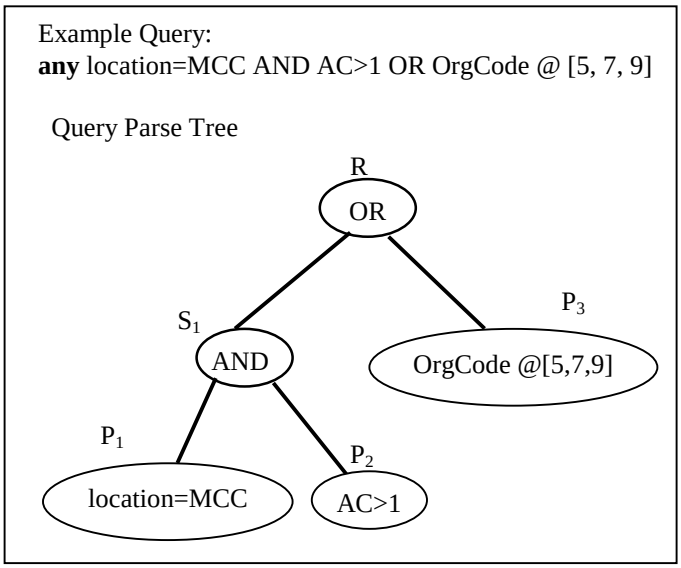

Figure 9. Query Parse Tree

4th. Lookup Algorithm

After the descriptive name server receives a query, a query parse tree is formed based on the query string. An example of a query and a query parse tree is shown in Figure 9. The first part of the message that contains the **query** keyword and the source identification is omitted is Figure 9. The query targets the users that work in location MCC and have authorization code greater than 1 or their organization code lies in the set [5,7,9].

According to the query grammar the parse tree will have a root (R) of an OR operation with two subtrees (S_1 and P_3) which correspond to one AND operation and one primary query. The AND operation will have two primary queries as leaves. As a result, primary queries P_1 and P_2 will produce the result sets Set_1 and Set_2. Because of the fact that P_1 and P_2 are leaves of a subtree which has an AND tag the result of the operation that this subtree corresponds will be the intersection ($Inter_1$) of Set_1 and Set_2. The result of the right subtree query will be Set_3. Finally the result of the tree R will be the union of $Inter_1$ and Set_3 ($Union_1$). Because of the query modifier ANY the result of the original query will be any random record of the $Union_1$ set.

5th. Application Level Forwarding

Application level forwading in our service is the ability of the location service to support forwarding of packets from one user to another using the location server network. The forwarding of these packets is subject to conditions that resemble a query.

An application-level-forwarding-request message consists of two parts, apart from the source identification part and the TTL field, which are carried on every request. The first part is a condition expressed in terms of a query and the second part consists of the data that the requesting client sends to other entities that satisfy the query. The data portion of the message is encoded in base 64 encoding since the whole protocol is character string oriented. An example of an application forward message is shown in Figure 10.

```
(b) src(vlasios@veria.ee.ucla.edu:4443)
    ttl(255)
    app_lev_forw
    filter (location = Zimbabwe
        AND
                (occupation = firefighter
                OR
                Occupation = policeman)
     )
    packet (Base64 encoded message)
```

Figure 10. Application level forwarding example

6th. Reply Format

The replies each server sends back to the requesting client are given in the form of XML [2]. XML is the perfect candidate for this kind of a naming system. It is first of all a string-oriented language in order to be compatible, portable and easy to debug. Secondly structured information can be easily represented in XML. On the one hand entity registration and signoff, attribute deletion and modification can result either in success or failure, which can be represented by just one success or error message. On the other hand a typical query reply will result in a list of entities, which in turn have their own lists of attribute value pairs. This kind of information fits perfectly into the XML specification. Figure 11(a) presents a successful registration reply, which includes the entity handle for future updates. Figure 11(b) shows a query reply for which the query was not satisfied, while Figure 11(c) shows a query reply which includes all the entity information in a structured way.

7th. SIP Integration

For the purposes of a proof of concept implementation the Columbia SIP client and server was used while the first author was with Telcordia Technologies, Inc. The SIP client is written in Tcl/Tk while the SIP server is written in C. The operations that the SIP client allows are user registration, user signoff and refresh of user registration because of the soft state registration. The attribute value naming system provides similar operations in addition to attribute value pair modification and attribute value pair addition and deletion operations. With these additional operations an entity can have a variable behavior in terms of the services this entity provides

and causes the queries of the other entities to have variable outcomes. Therefore we integrated these additional operations in the SIP client as well as the SIP server.

SIP provides a mechanism for its extensibility by means of stating in a SIP packet if the packet needs special handling ("Require" packet header). Therefore providing a new user location service is smoothly integrated with the SIP protocol by using the special SIP packet header. Every SIP packet originating from a SIP client to a SIP server carries this special header if the attribute based naming system is used. Along with the "Require" header another header ("Abea-name") was added. The additional header serves the purpose of carrying the information specific to the attribute based naming system. This information is the actual message (registration, signoff, attribute modification/deletion and application level forwarding). In the reply packets the same header operates as the reply from the location server. All the necessary User Interface components for the additional operations were added to the Tcl/Tk SIP client and all the demultiplexing code for the new location service was incorporated in the SIP server. These include components for registration and modification of an entity description, as well as for the query construction. The registration, the signoff and attribute modification/addition/deletion operations are encapsulated in SIP REGISTER requests, while the user query is encapsulated in SIP INVITE requests.

```
<reply>
     <userid>AF134BC_CD975_1</userid>
</reply>
```

(a)

```
<reply>
     <error>User Not Found!</error>
</reply>
```

(b)

```
<reply>
  <list>
    <entity>
      <user>
       helen@veria.ee.ucla.edu:4500
       </user>
       <attribute_value_list>
         <item>a=1</item>
         <item>b=new york</item>
         <item>d=876</item>
       </attribute_value_list>
     </entity>
  </list>
</reply>
```

(c)

Figure 11. XML Reply Examples

After the SIP server receives a SIP packet that requires special handling the attribute based naming server is contacted to take care of the special operation. This server processes the request and returns an XML reply to the SIP server. If the request was a registration/signoff/attribute modification addition and deletion then the reply does not need to be further processed by the SIP server. The XML reply is subsequently sent to the SIP client for being processed. If the request was a query request then the SIP server processes the XML reply, formats the reply according to the SIP protocol (list of contact information) and returns the reply to the SIP client. No further processing is need on behalf of the client because the reply does not contain any special handling directive.

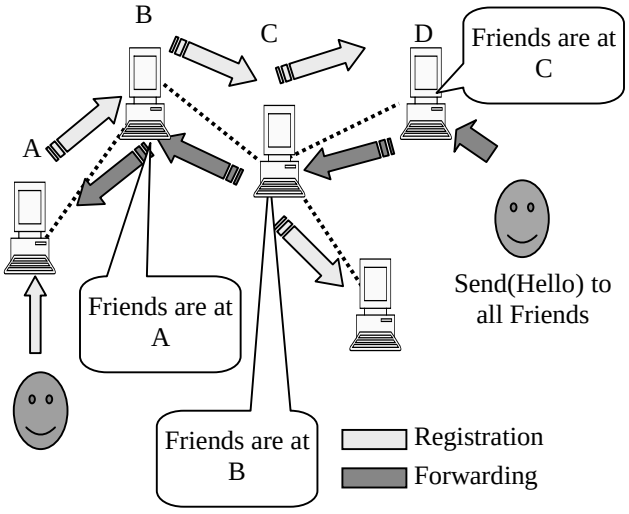

Figure 12. Application level forwarding

8th. Information Distribution and Request Processing

In the case of an entity registration, entity signoff and entity description modification the requests are forwarded to the whole distributed system or to a set of servers. This is accomplished by scoping the messages using the time to live message field.

In the case of registration and update messages the Server Engine updates the local database, determines the neighboring server(s) and forwards the requests to the set of qualifying servers (neighbors with the message scope). They, in turn, update their databases and forward the requests to their neighbors.

In the case of a client query the Server Engine searches the local database and if any registration records are found, it returns them to the requesting client. Otherwise the necessary information for the open client connection is stored in the Open Client connection table and the query request is forwarded to the neighbors of the current name server. This form of a query request follows the recursive query principle, which is also followed by the DNS servers. If the query reaches a terminal server (no neighbors other than the one that send the request) and no records satisfy the query, no negative reply is sent back. In this way the network does not fill up with unnecessary negative repiles. The purpose of a query is to find some entities satisfying the query and not to receive negative replies. This means that the originating server implements a mechanism to send a negative reply to the client after a timeout

period. If negative replies were tracked, additional state should be preserved in each server that forwards the query, so that it gathers all the replies from the other servers (downstream servers) and send one reply back to the server that it got the request from (upstream server). In this case the timeout mechanism would also be necessary for each upstream server.

When an application-level forwarding message enters a name server, the name server searches the local database to find all the records that satisfy the query. If some records are found, the server determines if the entities described by the records have registered to the current server directly or via a registration forward message from another server. If the entity has registered directly with the current server the data part of the message is sent to the matching entities using UDP packets. If the entity has registered directly with another server the application level forward packet is sent to that server. Thus if the registration message has gone through a series of servers then the application level forward packet will go through the reverse path (Figure 12). If no records were found in the local database the message is forwarded to the neighbors of the current server and the procedure above is repeated.

E. RELATED WORK

1st. Content based routing

In [4] the authors have implemented a content based addressing and routing architecture based on the communication model of an event notification service. This model clearly differs from our approach in the proposed architecture. They base the content routing on the event notification service called SIENA [3], which is in principle, a publish/subscribe service. The users interested in some piece of information subscribe their interests in terms of constraints on attribute values. Each interested user registers a filter similar to our query and every piece of published information (notification) is filtered through the user subscription (filter). If a notification matches the subscription then the notification is forwarded to the interested party. The philosophy of this approach is clearly different from our approach in which users issue the filters that scan the information database of an attribute based location server. However our approach could benefit from the techniques for merging subscriptions when one subscription can be logically inferred from another. These techniques could be used for implementing an indexing scheme for more efficient searching through our distributed system.

2nd. Lightweight Directory Access Protocol, WHOIS and WHOIS++ Service

The Lightweight Directory Access Protocol (LDAP)[17] is a protocol for accessing directory services like X.500 [7]. However it is not extensible and flexible enough to provide a location service that can do more than just binding of user descriptions to low level addresses. The WHOIS and WHOIS++[9][6] services are designed to provide information about the Internet users similar to our scheme but the supported query language is limited. It supports only equalities, regular expressions and wildcards combined with the AND, OR and NOT boolean operator, thus making the matching procedure purely string oriented. Our scheme provides a richer collection of operators and matching mechanisms.

3rd. Service Discovery Protocols

The Service Location Protocol [15] is a protocol for discovering network services with a different query language that firstly indicates the type of service that is being searched and then the attributes of this service in a structure-free manner. The services register with a central directory agent that maintains all the necessary information about the available services and handles the client requests. Our approach is not restricted to only network services and it does not include a centralized repository of user registration information. Moreover our query language has more operators and a query modification machanism that is applied on the result of the query.

The Secure Service Discovery Service [5] proposes an architecture of hierarchically connected directory servers that maintain the descriptions of network services in XML [2]. Additionally the servers exchange indexing information in the form of lossy compressed summary service records. The query language [16] used by SDS is based on the XML, and it is less expressive than our query language since it is oriented to search through XML structured information.

4th. Intentional Name Resolution

Intentional Name Resolution [1] from MIT addresses the descriptive name resolution and application level packet forwarding problem by creating a distributed system of descriptive name resolvers (INRs). These resolvers are also responsible for disseminating routing information about the registered intentional names. Networked entities are described with attribute value pairs that are hierarchically structured in order to narrow down the search space. However this approach has the drawback that the queries must also be structured in the same way that the information is structured. Therefore the entities issuing queries must know the exact information structure so that their queries are correctly resolved. Our flat naming scheme does not impose this limitation and it enables us to also incorporate information similarity indicators that can be used for evaluating the relevance of a descriptive name.

F. FUTURE WORK

The current status of the implementation enables a SIP user to invite another user issuing a query with the following modifiers: a) ANY: return any record that matched the query, b) ALL: return all records that matched the query and c) NEARESTFROM(x,y,z): Return the records that are nearest to the given point d) modify and delete attributes thus changing the behavior of the query matcher. If an entity d changes a type of service, it can use this mechanism to update its registration.

The scope of the registration messages can be selected by the client so as to minimize the amount of the forwarded registration messages and the amount of storage for each server. This solution however calls for an indexing scheme since the unsatisfied queries in one server will be virtually

broadcasted in the whole network of the location servers. This problem can be solved by having the registration and update messages forward the attribute names beyond the scope of the message. In this way each neighbor in the neighbor table will have a list of attribute names that the server knows something about. The server either has at least one record that includes this attribute name or it knows another server that has similar information. In this way the queries are forwarded to the neighbors that have some information about some attribute names. In addition to maintaining attribute names each neighbors can have indexes of attribute names as well as the ranges of the values of attributes. Moreover an indexing scheme similar to the one used in [4] can also be employed.

With the attribute based naming system in which the queries are an important part (for the INVITE requests and for the application level forwarding), the SIP users can implement some interesting types of services like the following:

1. The users can register with the following attributes: i) DISCLOSE_USER = [userlist] which means that if the user that issues the query is in the user list then and only then a matched record is disclosed and it is taken in account for the filtering process that the modifier imposes, ii) DISCLOSE_MACHINE = [machinelist]; in this case the records are disclosed if the originating client issues the query from a machine listed in the machinelist, iii) DISCLOSE_TIME[FROM-TO]; this attribute is used by a user that requests from the system to disclose her information if the query is issued in the time interval [FROM-TO].
2. A message system can be implemented using the application level forwarding ability of the location server. This can be done by encapsulating the application level forwarding message inside an INVITE request by using the “Require” keyword in the SIP packet. A unified messaging system was proposed by [13], which involves SIP and RTSP (Real Time Streaming Protocol). Our system enables a different architecture for implementing a messaging system using the added functionality of the location service.
3. Since a query can be forwarded in the network of the location servers it would be good if the location server had the ability communicate with SIP clients and servers. The intuition behind this is that a query may reach a destination user’s network and then return to the originating user. After that the originating user sends an INIVTE request. If the location server is able to play the role of a proxy it can initiate a call thus eliminating one roundtrip message exchange.

G. CONCLUSION

The Session Initiation Protocol is used to setup real-time sessions among users that register with a SIP Server using user names that resemble an e-mail address. The location server utilizes the user information in order to find the user’s contact information, which serves as the other communicating party’s SIP address. In this paper we deal with the problem of making the INVITE requests address a user with a more natural way by using attribute value pairs. We have presented the overall architecture for two types of location servers, a centralized and a distributed one. A distributed location server enables a user to register locally while it can be found by any other SIP server that uses another location server connected with the former one. This can be done either with full replication of the location information or with scoped registration and indexing of the attribute names. Along with the ability to locate mobile users a messaging service based on the application level forwarding feature can be implemented.

The attribute based naming system proposed for the location server enables the SIP users to initiate calls to users whose exact SIP addresses may not be known. The only requirement is that the users register with enough attribute value pairs so that other users can discover them using the proposed attribute based location service.

H. REFERENCES

[1] W. Adjie-Winoto, E. Schwartz, H. Balakrishnan, J. Lilley, ”The Design and Implementation of an Intentional Naming System” Operating Systems Review, vol.33, no.5, Dec. 1999. pp. 186-201.

[2] T. Bray, J. Paoli, C. M. Sperberg-McQueen, E. Maler, “Extensible Markup Language (XML) 1.0 (Second Edition)”, W3C Recommendation, October 2000, http://www.w3.org/TR/REC-xml.

[3] A. Carzaniga, D.S. Rosenblum, A.L. Wolf, ”Achieving Expressiveness and Scalability in an Internet-Scale Event Notification Service”, Proceedings of the Nineteenth ACM Symposium on Principles of Distributed Computing (PODC2000), Portland OR. July, 2000.

[4] A. Carzaniga, D. S. Rosenblum,, A. L. Wolf ”Content-Based Addressing and Routing: A General Model and its Application”, Technical Report CU-CS-902-00, Department of Computer Science, University of Colorado, January, 2000.

[5] S. E. Czerwinski, B. Y. Zhao, T. D. Hodes, A. D. Joseph, R. H. Katz, “An Architecture for a Secure Service Discovery Service”, Proceedigs of Fifth Annual International Conference on Mobile Computing and Networks (MobiCom '99), pp. 24-35.

[6] P. Deutsch, R. Schoultz, P. Faltstrom, C. Weider, "Architecture of the WHOIS++ Service", RFC 1835, August 1995.

[7] The Directory: Overview of Concepts, Models and Service. CCITT Recommendation X.500, 1988.

[8] M. Handley, H. Schulzrinne, E. Schooler, J. Rosenberg, “Session Initiation Protocol”, RFC 2543, March 1999.

[9] K. Harrenstein , M. Stahl., E. Feinler, "NICNAME/WHOIS", RFC 954, October 1985.

[10] P. Maniatis, Mary Baker, "IdentiScape: Tackling the Personal Online Identity Crisis.", Technical Report CSL-TR-00-804, Stanford University, June 2000.

[11] P. Mockapetris, K. J. Dunlap, “Development of the Domain Name System”, Proceedings of SIGCOMM ’88, pp. 123-133.

[12] R. Pandya, “Emerging mobile and personal communication systems”, IEEE Communications Magazine, vol. 33 , pp. 44-52, June 1995].

[13] K. Singh, H. Schulzrinne, “Unified Messaging using SIP and RTSP”, IP Telecom Services Workshop, Sept. 11, 2000. Atlanta, Georgia.

[14] D. Tennenhouse,”Proactive Computing” Commun. ACM 43, 5 (May. 2000), pp. 43-50

[15] J. Veizades, E. Guttman, C. Perkins, S. Kaplan, “Service Location Protocol”, RFC 2165, June 1997.

[16] Xset Database and Query Engine, http://www.cs.berkeley.edu/~ravenben/xset/

[17] W. Yeong, T. Howes, S. Kille, “Lightweight Directory Access Protocol”, RFC 1777, March 1995.

[18] http://www.3com.com/products/sip/index.html

[19] http://www.cs.columbia.edu/~hgs/sipc/