Fall 2008

PrESerD - Privacy ensured service discovery in mobile peer-to-peer environment

Santhosh Muthyapu

PrEServD – PRIVACY ENSURED SERVICE DISCOVERY IN MOBILE PEER-TO-PEER ENVIRONMENT

by

SANTHOSH MUTHYAPU

A THESIS

Presented to the Faculty of the Graduate School of the

MISSOURI UNIVERSITY OF SCIENCE AND TECHNOLOGY

In Partial Fulfillment of the Requirements for the Degree

MASTER OF SCIENCE IN COMPUTER SCIENCE

2008

Approved by

Sanjay Madria, Advisor
Maggie Cheng
Jagannathan Sarangapani

PUBLICATION THESIS OPTION

This thesis consists of the following article that has been submitted for publication as follows:

Pages 1 - 39 are intended for submission to the DATA & KNOWLEDGE ENGINEERING journal.

ABSTRACT

In mobile peer-to-peer networks many s*ervice discovery* protocols have been proposed. Most of these protocols disregard the exposure of the participating peers' privacy details, although they consider the security issues. In these methods, the participating peers must provide their identities, during the service discovery process, to be authorized to utilize the service. However, a peer might not be willing to reveal its identity until it identifies the service providing peer. So these peers face a problem; should the requesting peer or the service providing peer reveal the identity first, and hence, this is similar to the chicken-and-egg problem. The protocol presented in Private and Secure Service Discovery via Progressive and Probabilistic Exposure, solves this problem to some extent and works considerably to discover the services available in the user's vicinity in a single-hop time sync peers only. In this paper, we propose a privacy-preserving model based on challenge/response idea to discover the services available in the mobile peer-to-peer network even when the moving user and the service provider are at a multi-hop distance away. The performance studies shows that our protocol does this in a communication efficient way with reduced false positives while preserving the privacy details of the user and service provider.

ACKNOWLEDGMENTS

I am extremely grateful to my advisor, Dr. Sanjay Madria, for the encouragement and guidance he has given me and the extreme patience he has shown in my research work. He has also given me sufficient freedom to explore avenues of research while correcting my course and guiding me at all times.

I thank Dr. Maggie Cheng and Dr. Jagannathan Sarangapani, my committee members, for the help and support they have provided throughout my Master's degree program. I also thank Intelligent System Center (ISC) for funding the project.

I'm grateful to Anil Jade for sharing his knowledge and providing help during my research. Without his help, this work and its successful completion would not have been possible.

Last, but, at the top of my list, I thank my father Ramanaiah, my brothers Devender, Dr. Mahender and my sister-in-law, Dr. Rama Devi, for the tremendous encouragement and support I have received from them throughout my life which has enabled me to face the challenges and achieve success.

TABLE OF CONTENTS

Page

LIST OF ILLUSTRATIONS

Figure Page

LIST OF TABLES

PrEServD - Privacy Ensured Service Discovery in Mobile Peer-to-Peer Environment

Santhosh Muthyapu and Sanjay Madria

Department of Computer Science, Missouri University of Science and Technology, Rolla, MO 65409

Email: {smv6b@mst.edu} {madrias@mst.edu}

ABSTRACT

In mobile peer-to-peer networks many s*ervice discovery* protocols have been proposed. Most of these protocols disregard the exposure of the participating peers' privacy details, although they consider the security issues. In these methods, the participating peers must provide their identities, during the service discovery process, to be authorized to utilize the service. However, a peer might not be willing to reveal its identity until it identifies the service providing peer. So these peers face a problem; should the requesting peer or the service providing peer reveal the identity first, and hence, this is similar to the chicken-and-egg problem. The protocol presented in [12] solves this problem to some extent and works considerably to discover the services available in the user's vicinity in a single-hop time sync peers only. In this paper, we propose a privacy-preserving model based on challenged/response idea to discover the services available in the mobile peer-to-peer network even when the moving user and the service provider are at a multi-hop distance away. The performance studies shows that our protocol does this in a communication efficient way with reduced false positives while preserving the privacy details of the user and service provider.

1. INTRODUCTION

A Mobile Peer-to-Peer Network (M-P2P) is a decentralized network in which the mobile peers form an arbitrary network topology. The network is ad hoc because each peer though willing to forward the data to others can move out of the network and hence the determination of which peers forward data depends on the network connectivity.

M-P2Ps are becoming increasingly popular in the present day world, though theoretical and practical limits to the overall capacity of the M-P2Ps have been identified. This is because of the self-configuring capabilities of the participating peers to form an arbitrary topology and provide a broad range of services. M-P2Ps are widely used during emergency situations or military conflicts, because they can be deployed quickly and requires minimal configuration.

M-P2P provides various services based on the type of mobile peers and the number of peers present. Any peer in the network can either provide a service or utilize the service or does both. The number of services available in the network increases with the increase in the number of peers. *Service discovery* protocols will ease the process of manual configurations, by the user, when many services are present in the network. The service discovery protocols compel the participating peers to expose their private details like identity. However, the participants might not be willing to reveal their private details during this process. Even though both the (user and service provider) peers are legitimate, neither of them wants to reveal their details before the other does, thus can cause a deadlock situation, similar to a chicken-and-egg problem.

To analyze the problem, consider a service discovery process in M-P2P network, formed by several peers present in a shopping mall. Assume that a patient and a doctor

are present in this network and a patient tries to discover a doctor. The patient, while trying to discover the doctor present in the network, might not be willing to broadcast his/her identity and health problems in the service request. The patient would like to authenticate the doctor before revealing his private details. In other words, the patient will share his/her identity and health problems (private details) only when he/she finds a legitimate doctor. At the same time, the doctor is not willing to share his/her presence and identity (private details) in the shopping mall. The doctor wants to respond only to genuine clients/patients and not to the intruders. In other words, the doctor responds only to the patients who are authenticated and authorized at his end. In this scenario, both the patient and the doctor are not willing to share their details until they can authenticate and authorize the other. In our protocol, these peers (the patient and the doctor) play a game to authorize themselves without sharing the actual information (identities). Note that privacy preserving techniques can also be used to authenticate moving UAVs (Unmanned Aerial Vehicle) where due to security restrictions and communication timeout, secret keys were not shared and instead, some privacy-preserving techniques with stored states can be used to establish the authorization.

Protocol presented in [12] works considerably well to discover the services available in the user's vicinity. In this approach, users and service providers exchange partial identities and service information in each round of message transfer. During each round of message transfers, both the user and service provider verify the partial information provided by the other. This is done until either a mismatch occurs or legitimacy reaches a high probability. Identities of the user and service provider are exchanged in the form of a code word, which is generated by a secret shared between the

user and service provider. The parts of the code word are exchanged in each round of message transfer. To generate this code word, both the peers must have synchronized clocks. Also, the service requests and service information are encrypted before being exchanged among these two peers.

Though the protocol in [12] solves the chicken-and-egg problem to some extent, it has some limitations. A user can gain knowledge about the services available at the service provider's end causing the privacy breach. Since, partial information is exchanged among the user and service provider, service provider responds with different codes when more than one service matches the user's request. This protocol works considerably well only when service provider is in the user's vicinity. The protocol has high false-positives overhead in the initial stage. Though the process converges, it takes large number of message transfers to converge. For example, when a user's request reaches 500 service providers, it causes an average of 307 replies from these service providers. Most of these replies are due to false-positives occurred at the service provider's end, which is because of less number of bits being exchanged in each message transfer. Note that in case of moving UAVs, the time is very restricted to authorize other UAVs in the vicinity and therefore, we need a technique which has very low false positives and is faster.

In this paper, we propose a protocol to solve the chicken-and-egg problem among the peers participating in the service discovery protocol. The participating peers can be in the vicinity of one another or can be at a multi-hop distance. When the peers are at a multi-hop distance, a broker-based architecture will help each peer by multicasting the request to the highly ranked peers in the network. Then each participating peer plays a

game with the peer, in order to authenticate and authorize themselves, before revealing their private details. The game consists of various message transfers between the user and service provider and their validation. Each message consists of encrypted masked information about user/service provider identity and the service request. The game ends either when a mismatch occurs or when the user has been authorized at the service provider's end. Even when a mismatch occur, since all the messages are masked and encrypted, privacy details of the participating nodes have not been revealed. Our simulation results show that our approach has high throughput, and requires fewer number of messages than the Progressive and Probabilistic Exposure approach [12]. Since the complete identities are being exchanged, the probability of false-positives is less. Our broker architecture also helps in finding the reliable peers who possibly could provide the service.

Rest of the paper covers Related Work in Section 2, System Architecture in Section 3, then detail explanation of the protocol and algorithms in Section 4 and 5 respectively. Section 6 explains about the protocols resistance from replay attacks and MITM attacks while Section 7 proves the convergence of the protocol. The performance evaluation is given in Section 8 and the paper concludes in Section 9.

2. RELATED WORK

Many service discovery protocols have been proposed. In the insecure networks [9], service provider will advertise all the services it has, while the users' multi-cast their request to discover the services. In traditional service discovery protocols [5, 13], user will provide his credentials along with the service provider's address to avail the service.

In these protocols, the service provider needs to publish/reveal their identities like IP addresses and the user reveal his credentials to authenticate itself at the service provider's end.

In service discovery service protocol [3], service providers will register all the services with a centralized server. A user can discover the service, by simply querying the centralized server. However, this protocol depends on a third party (server), and also the service provider's service information and user's service request are revealed during the process. The Prudent Exposure [11] ensures that only legitimate parties gain the sensitive information, but peers will authenticate and query all the service providers they have credentials with during the process. Hence, a peer has to reveal the service request to all of its service providers which might not be accepted.

In Progressive and Probabilistic [12] approach, users and service providers exchange partial identities and service information in each round of message transfer until a mismatch occurs or legitimacy reaches a high probability. As discussed in the Introduction, though this approach solves the chicken-and-egg problem between the user and service provider, it has some limitations.

Our work of ranking the nodes in the network based on the type and number of services a node provides was inspired from the protocols defined in [10, 4]. [10] presents PeerTrust – a reputation-based trust supporting framework, which include a coherent adaptive trust model for quantifying and comparing the trustworthiness of peers based on a transaction-based feedback system and a decentralized implementation of such model over a structured peer-to-peer network. In [4], nodes (servants) can keep track, and share with others, information about the reputation of their peers. Reputation sharing is based

on a distributed polling algorithm by which resource requestors (peers) can access the reliability of perspective providers before initiating the download from the M-P2P network.

3. MOBILE PEER-TO-PEER SYSTEM ARCHITECTURE

In this section, we will define the key terms used and discuss about the Broker-Architecture, Broker-set formation, Bootstrapping phase and the system model.

a) **Service Provider**: A peer that provides a service.

b) **User:** A peer which initializes the service discovery process and tries to utilize the service.

c) **Broker:** A peer that acts as the cluster-head and forwards the service-request queries to the peers present in other clusters.

d) **Intermediate peers**: Peers those are present in between the User and the Service Provider while discovering and utilizing the services.

In an M-P2P network, due to the arbitrary topology and lack of centralized system, it will be difficult to maintain the details of all the nodes at a particular location. In order to maintain the service details of all the peers and to utilize the network features (services) efficiently, we divide the network into a group of *clusters*. Each *cluster* is a collection of peers. All the peers in a cluster can communicate among themselves. Every cluster will have a special peer node called *Broker* which acts as the cluster-head. Cluster-head is selected, based on the reliability and the transmission features among the peers in the cluster. The *Broker* will have the complete knowledge of the cluster. It periodically pings all the peers in the cluster and thereby maintains the information of the

peers which are moving in and out of the cluster. Broker will calculate the ranking information of each peer present in the cluster. It increases the rank of a peer whenever the peer provides a service successfully.

This kind of architecture, known as *Broker-Architecture*, was introduced in [8], in which among a group of brokers present in the network, the one with lowest *broker_id* is selected as the Master Broker. A slight variation of the Broker-Architecture defined in [8] is being used in this paper. In our model, a Broker is selected based on the peers' reliability and the transmission features.

Broker-Architecture (shown in Figure 1) provides some advantages like -

a) *Scalability*: As the network grows, information about newly joined peers is maintained only at the brokers. This will help to manage the peers' information efficiently and incorporates the load balancing in the network.

b) *Reduces Network Traffic*: Broker will multi-cast all the service requests received, to the top ranking nodes, instead of flooding the network.

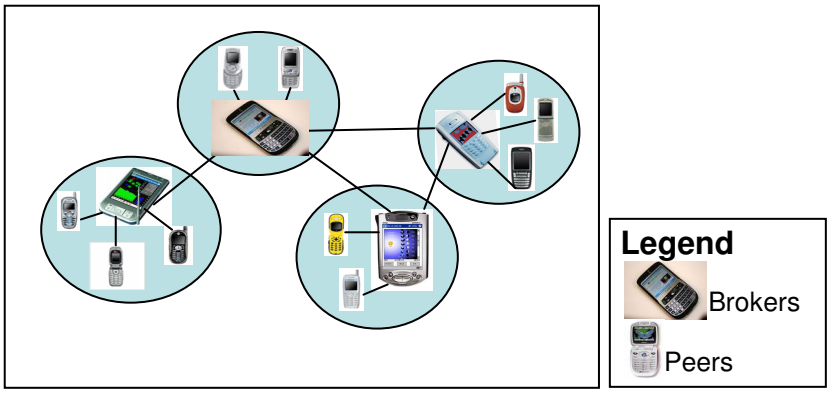

Figure 1: System Architecture

c) *Improves Latency*: The response time for discovering a service in the network reduces because every service request is passed through the Broker and the Broker has some knowledge about the services available in the network.

We assume that all the wireless links are bi-directional and peers can communicate omni-directional. A transmission link exists among the two peers if and only if the peers lie in the transmission range of each other. All the intermediate peers forward the packets selflessly.

3.1 BROKER-SET FORMATION

Initially all the peers in the network are non-brokers. In the process of selecting a peer as the cluster-head, every peer in the cluster will participate in the message transfers. During this process, average ratio of dropped packets to sent packets at each peer (drop-to-send ratio) is calculated. The peer with the least drop-to-send ratio is considered to be more reliable and will be selected as the Broker in that cluster.

Once the broker is selected in the cluster, it will be used as the gateway for all the requests of the peers present in that cluster. A service request is sent to the broker, if and only if the peer cannot find the service provider in its vicinity. If the broker moves out of the cluster, all the peers present in the cluster will again participate in the Broker selection process.

3.2 BOOTSTRAPPING

Figure 2 shows the bootstrapping phase of the protocol, in which, a centralized system is used for registering and querying the services. Service providing peers register periodically their services with the centralized system and users (peers) subscribe for these services by providing their identities. A service provider provides its identity

(ServiceProviderID as SPID) while registering its service. For each service subscribed by the service provider, it will receive the following tuple.

{ServiceID, SPAliasID, SharedKey, UserID_List}

In the above tuple, ServiceID (*SID*) represents the identity of the service and SPAliasID (SP_AID) represents the identity used by the service provider while providing this service. All the peers that provide a particular service will share *SPAliasID*. The advantage of this is that a user (peer) by knowing this unique *SPAliasID* can discover all the service providers, providing that service, present in the Mobile P2P network. Another advantage of using *SPAliasID* is, even when a user discovers a service, it will not have the knowledge of the service provider's actual identity.

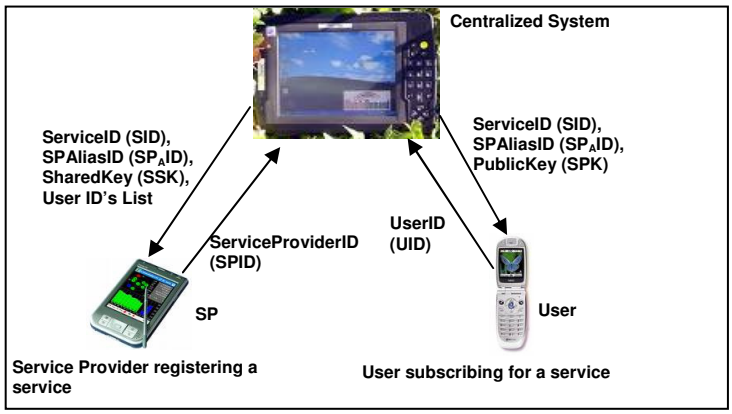

Figure 2: Bootstrapping Phase

A service provider will use the *SharedKey* (Private Key as SSK) to decrypt the messages received from the users. The service providers will get a list of UserID's which helps the service provider to authenticate the users in the service discovery process. However, the service providers must be updated with the latest list of users subscribed to the service.

Users subscribe for the services, with this centralized system, by providing their identity (*UserID as UID*). On subscription, a user will receive the following tuple.

{ServiceID, SPAliasID, PublicKey}

User encrypts the service request using the *PublicKey* (SPK) and tries to discover the service with *ServiceID* in the network at a service provider with *SPAliasID*. User and the service provider use the information gained in the subscription phase during the service discovery process. A peer (User or Service Provider) will be identified with a different identity (ID) at different situations, based on the type of service it is providing or utilizing.

3.3 OVERVIEW OF THE SERVICE DISCOVERY PROCESS

The ranking information of the peers is present at its broker. Initially all the peers will have the same rank (say zero). The rank of a peer is increased, at its broker, when it successfully serves a request. The model works on the assumption that, a peer with higher rank (successfully served many requests) will have the higher probability to serve the current request.

A user initiates the service discovery process by broadcasting an encrypted service request. All the peers who receive this request will try to decrypt the message, using all the SharedKey's they have. If none of them succeeds in doing so, the user will send the request to its broker. When a service request is arrived at the broker, it sends the request to all the peers in the same cluster. If there is no reply from the peers, then broker sends the request to the highest ranked peer set. If the service is not yet found, then broker resends the request to the next highest ranking peer set. This process is done until

the service is found. Brokers will send the request to the highest ranking nodes by finding the routes (between user and the service provider) using the DSR [6] protocol.

Instead of flooding the service request, multicasting is being used in the above process which reduces the network traffic. By multicasting the request to the highly ranked peers, Brokers increase the probability of finding the requested service. When a peer (service provider) is able to decrypt the service request, it starts playing a game (discussed in next Section) with the user by sending a reply to the service request. Both the user and the service provider will play the game to authorize the other. The game will be continued until both of them are authorized or when at least one of them recognizes that the other node cannot be authorized.

Once the service is discovered, the rank of the node providing the service will be increased at its broker and this information is shared among the other brokers. Also, all the brokers present in the route will cache the route information, which will help in the future service discovery process.

Peers in the mobile ad hoc network move randomly causing the wireless connections, between the intermediate peers (peers present on the route between user and the service provider), to be disconnected. So there is a probability of route failure between the user and the service provider, during the authentication process (i.e., while the peers exchange the masked identities). When a route failure occurs, the Broker peer of the user will find a new route between the user and the service provider, using the DSR [6] protocol. The Broker, which holds the current masked identities (generation and usage of masked identities is discussed in Section 4) being used in the game, will resend the last packet, sent between the user and service provider. By doing so, the peers can still

continue the game and this keeps the route failure transparent to the user and the service provider.

4. PrEServD PROTOCOL

In the bootstrapping phase, the centralized system runs the algorithm *GENERATE_KEYS* (discussed in the Section 5.1) to generate the tuple shown in the previous section. After the bootstrapping phase, a peer in the M-P2P network will send a service-request packet to discover the service, it subscribed for. The service request will be encrypted with the *PublicKey* (SPK) obtained from the centralized system. Service request packet will be generated using the *GENERATE_MASKED_IDENTITY* (this algorithm generates the initial masked identities and is discussed in Section 5.2) and will be of the form$[UPK,\ m_1, m_2]_{SPK}$, where *UPK* is the User's Public Key, m_1 is the masked *ServiceID* and m_2 is the masked *UserID***.**

$$m_1 = GENERATE\_MASKED\_IDENTITY\ (SID)$$

$$m_2 = GENERATE\_MASKED\_IDENTITY\ (UID)$$

When a peer receives a service request it will try to decrypt the message with all the available *SharedKey's*. If a service providing peer can decrypt the message successfully, then it can identify the service for which the request has been received. This is possible because the tuple, received from the centralized system, has a *SharedKey* that is paired with a unique service. Once the message is decrypted, the service provider will have

$$m_1^{'} = SSK\ (m_1)$$

$$m_2^{'} = SSK\ (m_2)$$

Now the service provider will use the *COMPARE* method to compare $m_1^{'}$ with its service (the service which is paired with the *SharedKey* used for decrypting the message) and to compare $m_2^{'}$ with the list of UserID's. Results from both the comparisons ($R(m_1^{'})and\ R(m_2^{'})$) are sent along with newly masked *ServiceID* (n_1) and SP_AID (n_2).

n_1 = *GENERATE_MASKED_IDENTITY (SID)*

n_2 = *GENERATE_MASKED_IDENTITY* (SP_AID)

Masking is done using the *MASK* algorithm, which is discussed later. The packet is encrypted using the User's Public key (*UPK*) obtained from the service request packet. The reply message will look like:

$$[R(m_1^{'}), R(m_2^{'}), n_1, n_2]_{UPK}$$

The service provider will compare $m_2^{'}$ with the list of UserID's (paired with the service) and a reply message is sent for each comparison. So, for the initial service request, a service provider will send multiple replies.

At the user end, the received message will be decrypted with the User's Private Key. The user will then validate the first two parts of the message ($R(m_1^{'}), R(m_2^{'})$) by equating them with its expected values. If there is any mismatch the user will ignore the received packet. Only if all the values match then the user will compare $n_1^{'}$ with the requested ServiceID and $n_2^{'}$ with the Service Provider's ID (SP_AID) using the *COMPARE* method, where

$$n_1^{'} = USK\ (n_1) and\ n_2^{'} = USK\ (n_2).$$

User will send the results of the compare method ($R(n_1^{'})$ and $R(n_2^{'})$) along with the newly generated masked identities m_3, m_4 (using the *MASK* algorithm, discussed in Section 5.4). The packet will be encrypted as

$$[R(n_1'), R(n_2'), m_3, m_4]_{SPK}$$

where m_3 = *Mask (SID, m_1, 2) and* m_4 = *Mask (UID, m_2, 2)*

where SID, UID are the actual identities of the Service and User respectively; m_1, m_2 are the masked identities used in the service request (acts as the current masked identity) for Service and User respectively and the third parameter "2" represents the number of iterations i.e., the number of packets exchanged between these two peers.

In this way the user and service provider exchange messages until each one is authorized by the other or when the reply messages does not have the expected values. The sequence diagram for the authorization process is shown in the Figure 3.

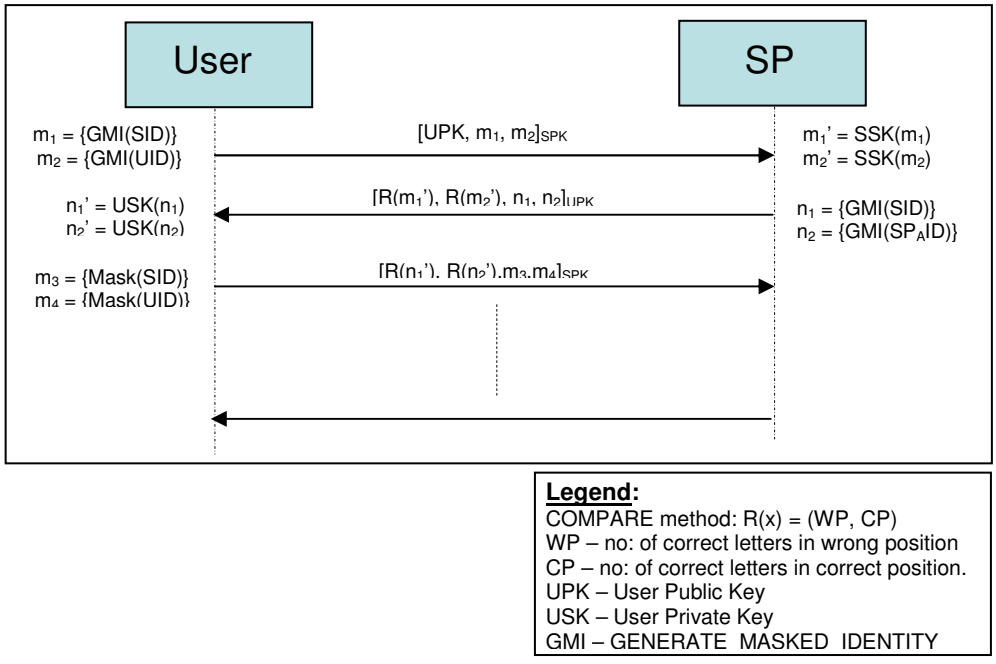

Figure 3: Sequence Diagram for the Authorization Process

5. ALGORITHM FOR PrEServD PROTOCOL

In this section, we will discuss algorithms used in our protocol to discover the services in a network. *GENERATE_KEYS* algorithm is run during the bootstrapping phase at the centralized system to generate the tuples (discussed in Section *3.2* Bootstrapping).

GENERATE_MASKED_IDENTITY is used by both the user and the service provider to generate the initial masked identity of the service, the user and the service provider. *COMPARE* is used to check the distance between the identity, present in the message received at a node, and the corresponding actual identity. If the results of the *COMPARE* algorithm are same as the expected values at a node, then *MASK* algorithm is run at that node to generate new masked identity, which is used in the further message transfers.

5.1 GENERATE KEYS PROTOCOL

This algorithm in Figure 4 is run by the centralized system for every subscription of a user or for every service registry by a service provider. The input for this algorithm is the identity of user/service provider along with the subscription or registry information. As discussed in the bootstrapping phase, the service provider will receive the following tuple as an output from the *GENERATE_KEYS* algorithm:

{ServiceID, SPAliasID, SharedKey, UserID_List}

A user subscribed for a service will receive the following tuple as an output from the *GENERATE_KEYS* algorithm:

{ServiceID, SPAliasID, PublicKey}

These tuples are used by the service provider and the user during the service discovery process to authenticate one another.

```
GENERATE_KEYS: This algorithm runs at a centralized system to subscribe users for a service or to
register a  service provider's service.
Input: Node's identity and whether the peer registers a service or subscribes for a service.
(nodeID, type)
Output: {ServiceID, SPAliasID, SharedKey, UserID_List} or {ServiceID, SPAliasID, PublicKey}

GENERATE_KEYS (nodeID, type)
{
        IF (type = registry)
        {
                IF (newService)
                {
                   Generate a random identity for the service (ServiceID)
                   Generate a random identity for the Service Provider (SP_AID)
                   Select a random Public-Private (SharedKey) key pair
                }
                ELSE
                {
                   Retrieve the ServiceID
                   Retrieve the SP_AID
                   Retrieve the SharedKey
                   Retrieve the List of User's subscribed
                }
        }
        ELSEIF (type = subscription)
        {
                Retrieve the ServiceID
                Retrieve the SP_AID
                Retrieve the PublicKey
        }
}
```

Figure 4: Generate Keys Algorithm

5.2 GENERATE MASKED IDENTITY PROTOCOL

This algorithm in Figure 5 is run at the User end to generate the masked identity of the service and the user. These masked identities are used in the initial service request packet. This algorithm is also run at the service provider's end to generate its masked identity which is used in the first reply of the service request.

Example: Assuming the length of the identities (peer and service) to be 6, let's say user picks two random numbers for Expected_CP and Expected_WP between 1 and 3.

Expected_CP = Random (1, 3), Expected_WP = Random (1, 3)

Here CP and WP stand for "number of correct letters in correct position" and "number of correct letters in wrong position" respectively. CP and WP can be understood clearly from the COMPARE method discussed in Section 5.3. Now to generate the initial masked identity, let's pick the positions of the letters randomly.

Loop 'Expected_CP' times

randomPositions = Random (1, 6)

Loop 'Expected_WP' times

randomLetters = Random (1, 6)

Using these values the user will generate the initial mask identity for a service. For example, let **AICH5K** be the actual id of a service, Expected_CP = 2, Expected_WP = 1, randomPositions = {3, 5} and randomLetters = {6}. We can observe that the degree of set (array) randomPositions is 2 which is equal to Expected_CP and the degree of set (array) randomLetters is 1 which is equal to Expected_WP, as expected from the algorithm.

GENERATE_MASKED_IDENTITY: This algorithm runs at the user and the service provider to create the initial masked identities.

Input: Identity of a User or a Service or a Service Provider to be masked. (ID)

Output: Masked identity of the input. (MaskedIdentity)

```
GENERATE_MASKED_IDENTITY (ID)
{
        Expected_CP = Random (1, i) // "i" is a random number less than "n"
        Expected_WP = Random (1, i)

        LOOP Expected_CP times
                randomPositions = Random (1, n) // "n" is the length of the Identity
        LOOP Expected_WP times
                randomLetters = Random (1, n)

        ActualIdentity = ID
        MaskedIdentity = "" // initially empty string

        FOREACH position in randomPositions
                MaskedIdentity[position] = ActualIdentity[position]

        FOREACH letterPosition in randomLetters
                MaskedIdentity[Random(1,n) – randomPositions] =  ActualIdentity[letterPosition]

        LOOP index = 0 to MaskedIdentity.Length
                IF (MaskedIdentity[index] == " ")
                        MaskedIdentity[index] = Random(0-9,A-Z)

        RETURN MaskedIdentity
}
```

Glossary of terms and functions:

1) Random (x,y) is a function that returns a random integer between 'x' and 'y'.
2) All identities are Strings and Identity[i] represents the character at i^{th} index in the string "Identity"
3) "randomPositions" and "randomLetters" are the Integer Arrays
4) "Random(1,n) – randomPositions" returns an integer between 1,n and which is not present in the array 'randomPositions'
5) "Random(0-9,A-Z)" returns either an integer between 0 and 9 or an alphabet between A and Z.

Figure 5: Generate Masked Identity Algorithm

Since ActualIdentity is **AICH5K,** the number of correct letters in correct positions is 2 (Expected_CP) and their positions are 3 and 5. So MaskedIdentity is **--C-5-.**

The number of correct letters in wrong positions must be 1 (Expected_WP) and the letter at the actual position is 6. Next pick a random number from 1 to 6 except 3 and 5 (Random(1,n) – randomPositions) to place the letter at position 6 of the ActualIdentity. Say the random number is 1 and so the position 1 in MaskedIdentity must be filled with the letter at position 6 in ActualIdentity. Therefore, the MaskedIdentity is **K-C-5-.** Now fill the empty positions with some random alphanumeric letter to get the initial masked identity for the service, as ***KZC*X54.**

Similarly the initial masked identities are generated for UserID and ServiceProviderID at the user and service provider respectively. The initial service request packet from the user will contain the masked identities of the requested service and the user along with the user's public key.

5.3 COMPARE PROTOCOL

The *COMPARE* algorithm in Figure 6 basically checks the distances between the provided identity and available identities. For example, if the service ID (SID) of a particular service is "FLWRE9". The user masks this particular ID into "LABRE0". When we compare the actual ID and masked ID, we can see that "L" is present in the masked ID but in the wrong position where as "R and E" are present in the correct position and rest of the letters in the masked ID are not matching with the actual ID. So the user expects a 2 for CP and a 1 for WP (CP – the number of correct letters in correct position, WP – the number of correct letters in wrong position).

COMPARE: Returns the distance between the identity received in the message and the corresponding actual identity.
Input: Actual identity and the received identity (ReceivedIdentity, ActualIdentity)
Output: CP (number of correct letters in the correct position) and WP (number of correct letters in the wrong position) in the received identity.

```
COMPARE (ReceivedIdentity, ActualIdentity)
{
        LOOP index_RI = 1 to ReceivedIdentity.Length
        {
                LOOP index_AI = 1 to ActualIdentity.Length
                {
                        IF (ReceivedIdentity [index_RI] = ActualIdentity [index_AI])
                        {
                                IF (index_RI = = index_AI )
                                        CP++
                                ELSE
                                        WP++
                        }
                }
        }
        RETURN (CP, WP)
}
```

Glossary:
"Identity[i]" represents the character at i^{th} index in the string "Identity"

Figure 6: Compare Algorithm

At the service provider's end, when it receives a message, after decrypting the packet it calculates the distances (WP and CP) for both SID and SP_AID. The results are sent in the reply to the user. When the user gets the reply from the service provider, it compares the values of WP and CP in the reply packet with that of its expected values. If they are equal, then the communication continues until the user is authorized or a mismatch (of WP/CP) occurs. The user is said to be authorized when WP = 0 and CP = 6 (assuming the length of the SID is 6).

5.4 MASK PROTOCOL

At a peer, if the values in the received packet and the values expected for CP, WP are same then the MASK algorithm in Figure 7 is run to generate new masked identities. At a peer, MASK algorithm uses the knowledge gained from the previous masked identities it encountered during the service discovery process as explained in the following example.

In the example discussed in GENERATE_MASKED_IDENTITY, for the ServiceID **A1CH5K**, we have generated **KZCX54** as the initial masked identity, which is sent in the service request by the peer. Now, the peer will expect the reply message with a (2, 1), for (Expected_CP, Expected_WP). If the reply message has the same values as (Expected_CP, Expected_WP) then the user will generate new masked identities (m_3, m_4) using the MASK algorithm. For example, if the user receives (2, 1) as the reply then the peer will jumble the positions of "C" and "5" and might also include another new letter in correct position, so the node will have a new masked id m_3 = ***AC*X4Y*5*.** The MASK method is used by both the user and service provider to generate the new masked identities in every message transfer.

6. RESISTANCE TO REPLAY ATTACKS AND MITM ATTACKS

In this section, we will explain in detail how our protocol prevents the replay attacks [14] and man-in-the-middle attacks [16].

6.1 REPLAY ATTACK

Consider peers Alice, Bob and Mallory (an intruder) present in the network. Assume that, Alice is subscribed for a service (say service X) at Bob and is trying to discover this

MASK: At a node, if the values received in the packet and the values expected for CP and WP are same then the MASK algorithm is run to generate new masked identities.
Input: Actual identity, corresponding current masked identity and the number of iterations done so far. (ActualID, currentMaskedID, noOfIterations)
Output: Newly masked identity. (newMaskedID)

```
MASK ( ActualID, currentMaskedID, noOfIterations) {
    newMaskedID = "" //Initially empty string
    IF (noOfIterations < 3) {
        LOOP index = 0 to ActualID.Length
            newMaskedID[index] = currentMaskedID[Random(1, ActualID.Length)]
    } ELSE {
        newLetter // represents the letter added from ActualID
        randomNumber // represents the position of the "newLetter"
        usedLetters [] = RetrieveCPWP(ActualID, currentMaskedID)

        LOOP index = 0 to ActualID.Length
            randomNumber = Random(1, ActualID.Length)
            IF (ActualID[randomNumber] NOT IN usedLetters)
                newLetter = ActualID[randomNumber]
                BREAK

        newMaskedID[randomNumber] = newLetter

        FOREACH position in usedLetters
            newMaskedID[position] = ActualID[position]
        LOOP index = 0 to ActualID.Length
            IF ( newMaskedID[index] == " " )
                newMaskedID[index] = Random(0-9,A-Z)
    }
    RETURN newMaskedID
}
```

Glossary of terms and functions:

1) "Random(x,y)" returns a random integer between 'x' and 'y'
2) "usedLetters" is a character array that holds the characters that are common between ActualID and currentMaskedID.
3) "RetrieveCPWP(ID1, ID2)" is a function that returns the common characters between ID1 and ID2.
4) "Identity[i]" represents the character at the i^{th} index in the string 'Identity'.
5) "x NOT IN Y" returns true if character 'x' is not present in the array 'Y'.
6) "Random(0-9,A-Z)" returns either an integer between 0 and 9 or an alphabet between A and Z.

Figure 7: Mask Algorithm

service X. In order to discover the service X, Alice will create a request packet (which contains the masked identities of SID and UID) and broadcasts the same. Bob processes the received packet, creates a reply packet and sends it to Alice. Alice processes the received packet (checks for the match in (CP, WP)), creates a new reply packet and sends it to Bob. This continues until both the peers are authorized at one another (from the assumption, they will be authorized).

Assume that, Mallory eavesdrops the conversation between Alice and Bob and tries (in the future) to get authorized at Bob using the packets stored from this session. So, Mallory starts broadcasting the service request packet, then Bob will process it and sends the reply (this reply packet consists of newly generated masked identities and so Bob's expected values changes). Mallory sends the next packet it stored (from the Alice-Bob session) as a reply. But, when Bob processes this packet, there will be a mismatch between the values received in the packet and the values Bob expects. So, Bob will stop communicating with Mallory and Mallory will not be authorized for the service X.

From the above example, we conclude that an intruder can never be authorized if it tries to retransmit the packets it obtained by eavesdropping because a peer generates a new reply packet (with new masked identities and expected values) for every packet received.

6.2 MAN-IN-THE-MIDDLE ATTACK

The man-in-the-middle attack [16] is a form of active eavesdropping in which the attacker makes independent connections with the victims and relays messages between them, making them believe that they are talking directly to each other over a private connection when in fact the entire conversation is controlled by the attacker. The attacker

can control the conversation if it can intercept the public-keys of the nodes participating in the conversation.

In PrEServD protocol, peers will obtain the keys during the bootstrapping phase and these keys are used in order to authenticate one another. Also, all packets are encrypted and decrypted only at the User/SP (endpoints) restricting the intermediate nodes to intercept the packets. So, when an intruder in the network tries to eavesdrop during a service discovery process, it cannot intercept the packets because it cannot decrypt them. An intruder can only forward the received packet or it can try to retransmit the packet at a later point of time. But as discussed in Section 6.1, the intruder will not be authorized for using these services.

7. PrEServD PROTOCOL CONVERGES

In this section, we discuss the transfer of messages among the peers and mathematically prove the convergence of our protocol.

In the service discovery process, a peer (user) initiates a service request message. In response to this request message, a user will receive many replies from other peers in the network. These peers and the user play a game (by exchanging messages) in order to authenticate each other to utilize the service, provided each of them are authorized to do so. Let us consider replies from only one peer (say service provider) in order to keep the convergence proof simple.

In response to the service request message from the user, a service provider might reply with multiple messages (say n_1) and the user receives these n_1 messages. The user processes the received messages and responds to the messages that have the expected CP

and WP values. Suppose the user replies back with n_2 ($n_2 \leq n_1$) messages in response to the received messages, then the service provider will process these received n_2 messages and responds back to the messages that have the expected CP and WP values. If the service provider replies with n_3 messages (again $n_3 \leq n_2$) and say it receives n_4 messages so on and so forth until both the service provider and the user are authenticated. When the user is authenticated at the service provider, they stop playing the game and the number of messages being transferred among these peers will become zero. Also, whenever the values of the CP and WP are not same as that of the expected, the peers stop communicating for that particular reply.

Formally, if n_i represents the number of messages exchanged in i^{th} iteration and n_j represents the number of messages in j^{th} iteration between the two peers, then

$$n_i \geq n_j \ \forall \ i < j$$

where i, j ∈ N (N is the set of Natural Numbers)

If *f(x)* is a function, which represents the number of replies a peer sends, it is clear from the above argument that *f(x)* is a monotonically decreasing function and is non-negative. So, we can state that

$$\frac{f(x)}{f(x-1)} \leq 1 \ \text{ i.e., } \frac{f(x)}{f(x-1)} = k$$

where 'k' is a constant such that $0 < k \leq 1$. Hence, we have $f(x) = k\, f(x-1)$.

By using the Integral Test for Convergence [15], "A non-negative monotonic decreasing function *f* defined on an unbounded interval [Z, ∞) converges iff the integral on *f* is finite, where Z is an integer", we can prove that *f(x)* converges. In our case, Z = 1 and integral of *f* is

$$\int_1^{\infty} f(x)dx = k \int_1^{\infty} f(x-1)dx$$

which can be deduced to $\int_1^{\infty} f(x)dx = \frac{k}{1-k} f(1)$, (where $0 < k \leq 1$) which is a finite value. Hence, from the Integral Test for Convergence, the PrEServD protocol converges.

8. SIMULATION

We built a simulation environment to study the experiments conducted using the PrEServD protocol described in Section 4 and Progressive Approach [12] protocol.

The simulation area is approximately 1000 X 600 m^2 and it can afford a range of 30 to 120 peers in the network. The maximum connection distance between any two peers is 100 m. List of the simulation parameters are provided in Table 8.1.

Table 8.1: Simulation Parameters

Parameter	**Range**
Simulation area	1000 X 600 m^2
Number of peers	30 ~ 120
Maximum Connection distance	100 m

The simulation environment is built with varying number of nodes in the network. The movement of the nodes is handled by implementing the random way point model (RWP) [1]. In RWP, each node moves along a zig zag line from one way point to the

other. The random way points are uniformly distributed over the given area and all the nodes tend to converge at the center. But this type of model has some common problems. When we take the average speed of a node, it tends to decay over a period of time and eventually approaches zero. RWP chooses a destination and speed for a node randomly and independently, and the node will keep moving at that speed until it reaches that destination. A common problem arises when a node moves very slowly for a given long destination which it reaches after a very long time, which increases simulation time. To overcome such a problem, we have used a slight variation of RWP in which we consider a new parameter, time. To overcome the average speed decay problem, we randomly choose speed which is uniformly distributed in the interval [1, V_{max}) instead of (0, V_{max}) used previously. This ensures that the average speed does not tend to zero.

The formation of the node cluster (Broker architecture) is handled by implementing a Connected Dominating Set (CDS) [7] model. Several algorithms for the CDS formation have been discussed in [2], we have used Steiner tree based CDS construction to define Broker nodes in the network.

8.1 PERFORMANCE COMPARISON

Simulation is performed to evaluate the performance of the PrEServD protocol. We compare throughputs, latency, number of messages transferred and false-positives in discovering services available in Mobile P2P network using the protocol defined in [12] and PrEServD protocol. We also study the performance of our protocol when the states (information of the current masked identities) are not stored.

8.1.1 Throughput: We define, *Throughput* as the ratio of the number of requests satisfied to the number of requests initiated in the network. Figure 8 shows the

comparison between the throughputs of both the protocols. The readings are taken in a network with 60 nodes and each node subscribed to a maximum of 30 services from the available 100 services. The number of requests initiated in the network varies from 5 to 25.

In both the protocols, when a user broadcasts a service request packet, all the neighboring nodes will process the packet independently and concurrently. So, a single service might be found at multiple neighbors, which might result in discovering more number of services than requested. Also, if the service is not available in the vicinity, PrEServD protocol will try to discover the service in the entire network, unlike Progressive Approach. Hence, throughput of the PrEServD protocol is better than the Progressive Approach.

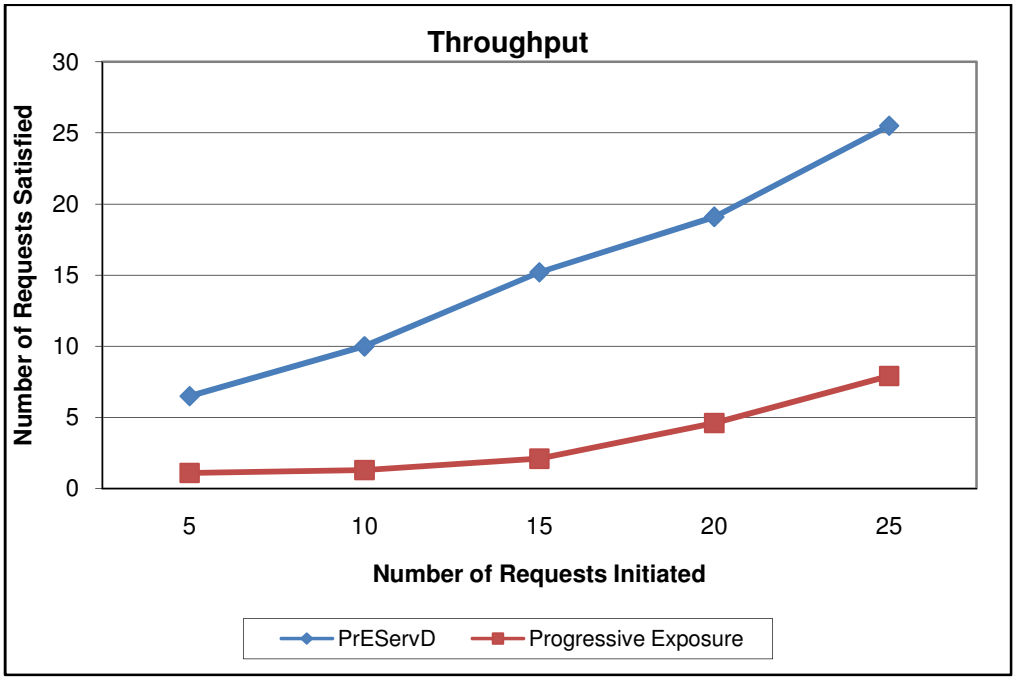

Figure 8: Throughput for PrEServD and Progressive Approach

8.1.2 Messages Broadcasted during Service Discovery: We compare the number of messages transferred/broadcasted in each protocol during the service discovery process. Figure 9, shows comparison between the number of messages broadcasted in PrEServD protocol and Progressive Approach in a network with 60 nodes while increasing the number of requests from 5 to 25.

With increasing number of requests, the number of messages broadcasted in the network increases for both the protocols. But the number of messages broadcasted in the Progressive Approach is very high, compared to PrEServD protocol, because of the lesser numbers of bits being transmitted in each message resulting in a large number of false-positives. In the next section, we discuss the percentage of false-positives and compare the same for both the protocols.

8.1.3 Percentage of False-positives: In PrEServD protocol, a participating peer compares the Expected(CP, WP) values with the received(CP, WP) values and replies for a match. A match can be either a true-match or a false-positive. A match is defined as a true-match, when it occurs between the received(CP, WP) values (of the actual UserID or ServiceID) and the Expected(CP, WP) values. All other matches are considered as false-positives.

As explained in Progressive Approach [12], the number of false-positives decreases exponentially for a single service request but the total number of messages broadcasted in the network is very high due to these false-positives which can be seen in Figure 9. Figure 10 compares the percentage of False-positives for both the protocols. We can observe that most of the messages broadcasted in the Progressive Approach are because of the False-positives (it is close to 90%).

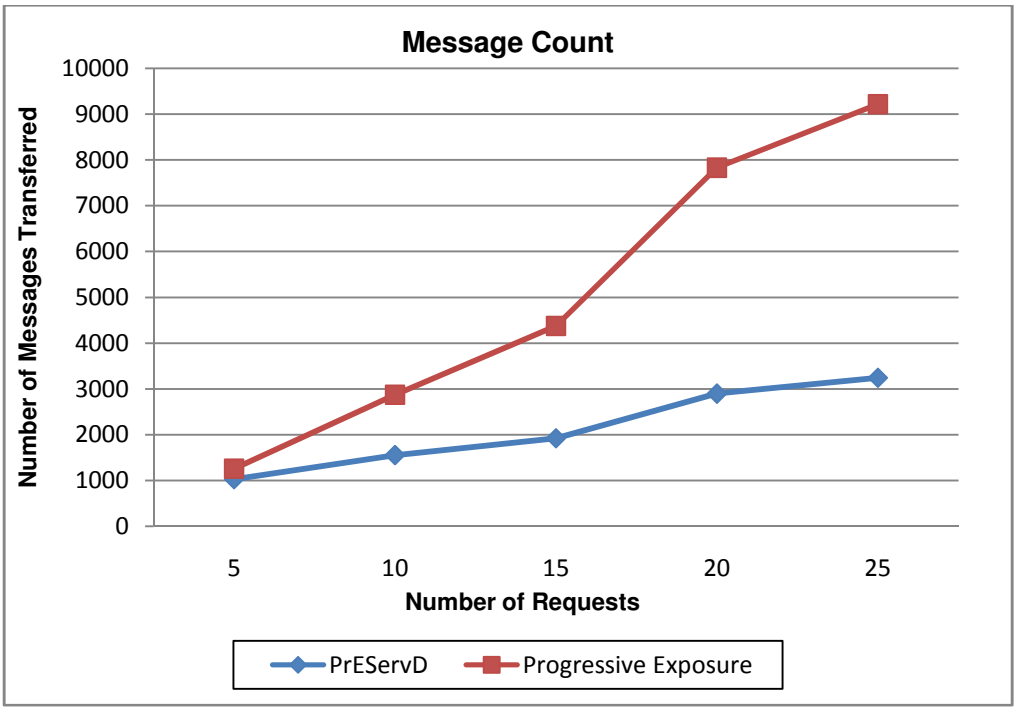

Figure 9: Message Transfers in PrEServD and Progressive Approach

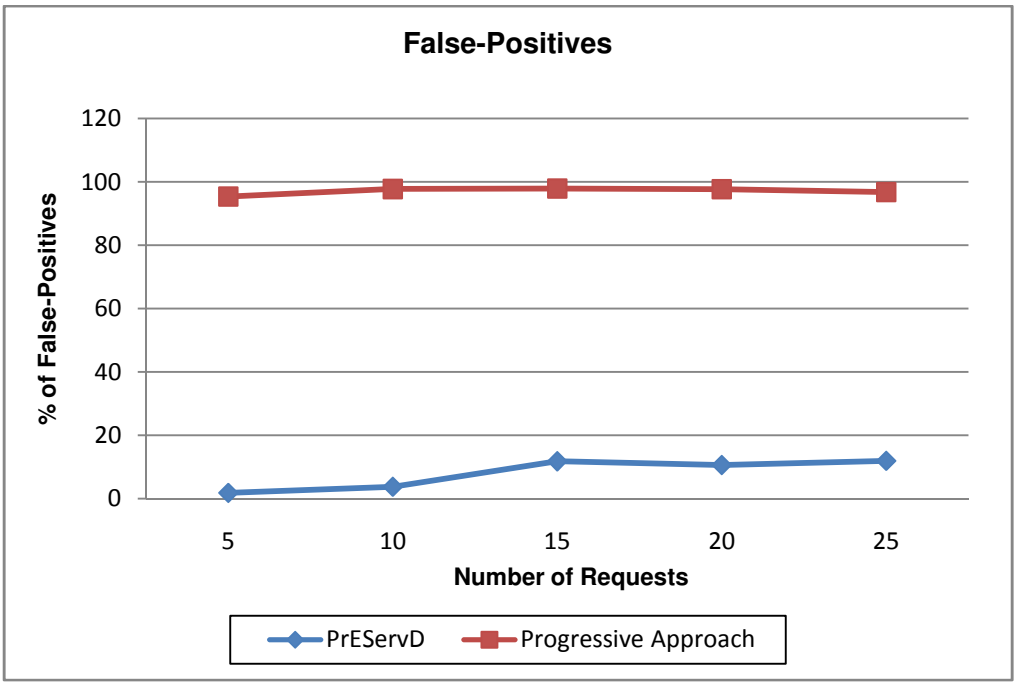

Figure 10: False-Positives in PrEServD Protocol and Progressive Approach

From Figure 10, we can also see that very less number of false-positives occur in PrEServD protocol. The reasons are twofold. One is that the information in PrEServD protocol is sent with complete masked identities instead of partial identities. Two, more levels of comparisons are required in PrEServD protocol to identify a match which results in lesser number of false-positives, unlike the Progressive Approach.

8.1.4 Energy Consumption: The energy consumed for a service discovery process is the cumulative energy consumed at all the participating peers. The energy consumed at a peer depends on factors like reception power (the number of messages it receives), processing power (power consumed for computations) and transmitting power (number of messages it transmits). So, the Energy Consumed is directly proportional to the number of messages being broadcasted in the network and the number of nodes participating.

The readings are taken in a network of size 60 and the energy consumed for discovering a service is calculated using the following equation:

$$EC = [n * (t + r + p)] + [m * n (t + r)]$$

where 'n' represents the number of messages broadcasted,

't' represents the energy consumed for transmitting a single message,

'r' represents the energy consumed for receiving a message,

'p' represents the energy consumed for processing a message.

'm' represents the number of intermediate nodes (nodes participating in the discovery process other than user and service provider).

The first part of the equation can be explained by the following argument: The user and the service provider will receive process and transmit exactly half the total

number of messages broadcasted. The second part of the equation explains the energy consumed by the intermediate nodes and each intermediate node is involved in transmitting and receiving messages.

For PrEServD protocol, the readings are taken for both the regular approach and by restricting the discovery process to single-hop (similar to Progressive Approach). In the regular approach 'm' is greater than zero while 'm' becomes zero for the restricted approach and Progressive Approach. Figure 11 shows that the energy consumed for discovering the services is high for PrEServD protocol compared to that of the Progressive Approach. This is because more number of peers' participate in the service discovery process and each peer contributes to the total energy consumed. The energy consumed by the PrEServD protocol (restricted to single-hop) is comparatively less due to the fact that lesser number of messages are transferred in the service discovery process.

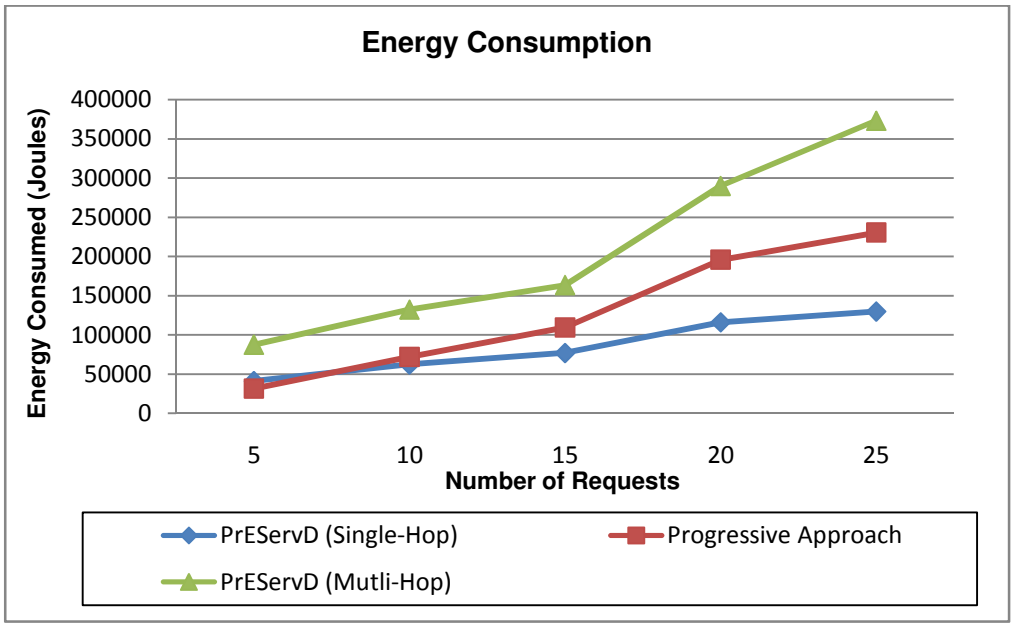

Figure 11: Energy Consumption for PrEServD (single-hop, multi-hop) and Progressive Approach

8.1.5 Latency: We compare latency, the average-time taken (in milliseconds) for discovering a service, for the two protocols. Figure 12 compares the average-time taken by the two protocols for discovering five services while increasing the number of peers in the network ranging 20 to 100. Figure shows that, the latency for the PrEServD protocol is little high, this is because PrEServD tries to discover the service in the whole network (services which are multi-hop away) when the service is not available in the user's vicinity whereas the other protocol only finds services within a single hope.

The results from Figure 12 signify that the average-time taken for discovering a service is less in PrEServD protocol, when restricted to a single-hop. This is because lesser number of messages are transferred in PrEServD protocol compared to Progressive Approach.

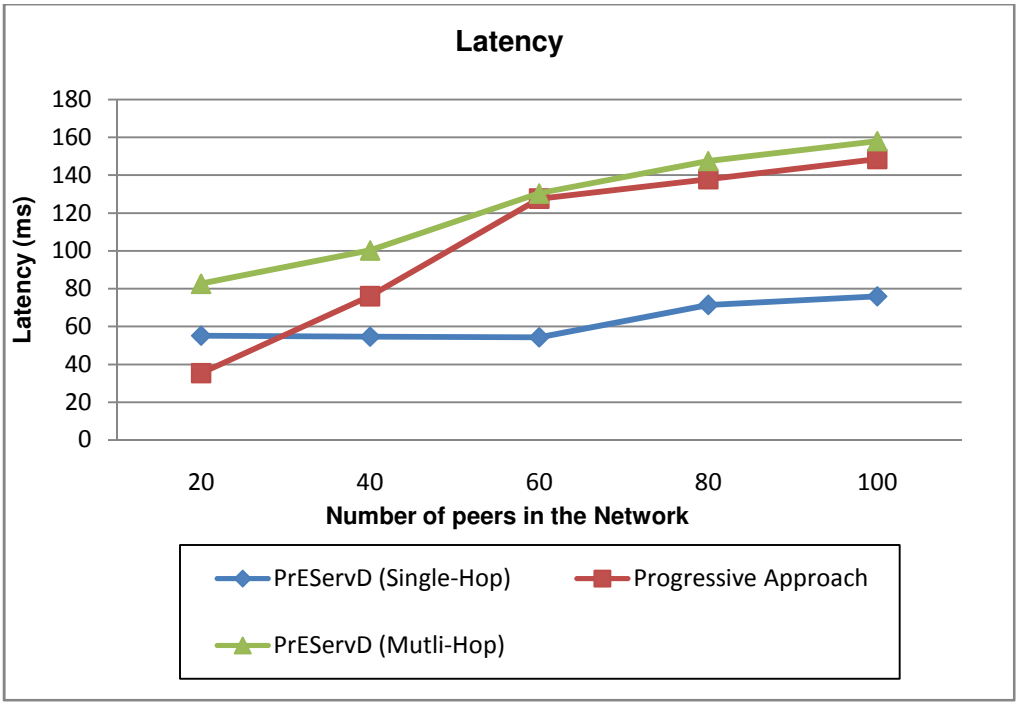

Figure 12: Latency for PrEServD Protocol and Progressive Approach

8.1.6 Restart Rate: In M-P2P network, peers move randomly causing the wireless connections, between the intermediate peers (peers present on the route between user and the service provider), to disconnect. Though, we keep these disconnections transparent to the user and the service provider (as discussed in Section 3.3) by storing the states (information of the current masked identities) in PrEServD protocol, we also studied our protocol without storing the states. That is, we restart a service request whenever the wireless link between the user and the service provider disconnects during the service discovery process and we define, Restart Rate as the number of requests restarted over the total number of requests.

We plot a graph for the restart rate (shown in Figure 13); in a network with 60 nodes while increasing the number of service requests from 5 to 40. We compare the results for various node mobility rates. Node mobility rate (X %) is defined as the percentage of total number of nodes moving randomly in the network at any point of time.

From the Figure 13, we observe that as the number of service requests increases the number of requests restarted also increases. We note that when the network has at least 20% of the nodes moving, an average of 11.25 requests are restarted for every 40 requests initiated, which is about 28.12% of the total requests. By restarting these requests, more network resources are being utilized and the latency is also increased while maintaining the throughput. In order to overcome this, we store the information of the states and there by utilize the network resources very efficiently.

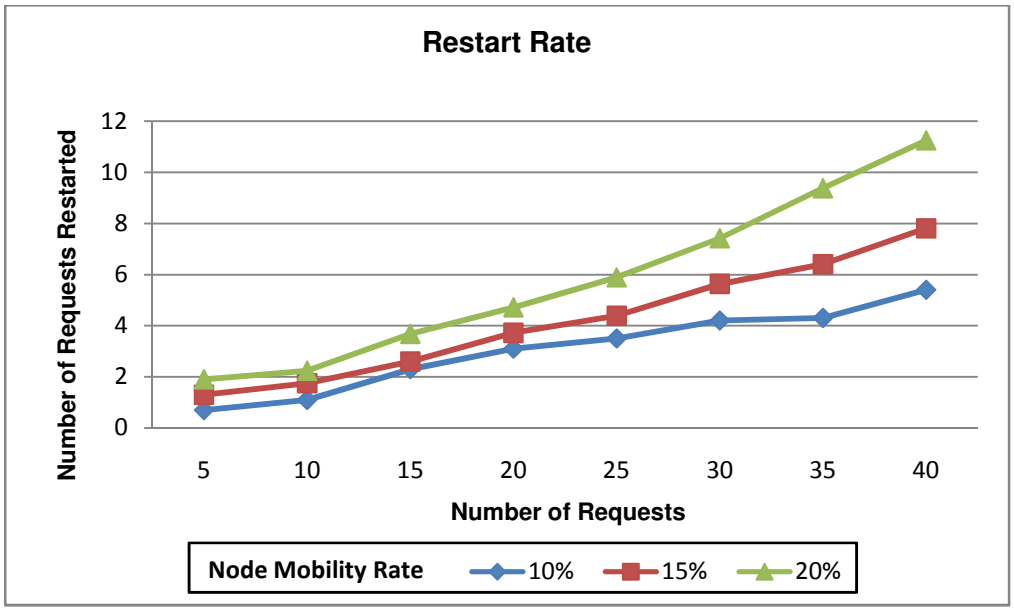

Figure 13: Restart Rate for PrEServD Protocol while Increasing Node Mobility Rate

9. CONCLUSIONS AND FUTURE WORK

In this paper, we proposed a protocol that ensures privacy between the peers participating in the service discovery process. The peers reveal their identity progressively (by playing a game) until they are authorized. We defined the broker architecture that helps in organizing the network and discovering the services that are at a multi-hop distance away from the user. We also discussed how the protocol prevents replay attacks and MITM attacks. This protocol can be applied in any scenario, where the two communicating parties (peers) are not willing to reveal their identities, like blind dating, car pooling etc.

Simulation results prove that the PrEServD protocol is much more efficient than the Progressive Approach [12]. Our algorithm provides much better throughput, and energy consumption and the latency of our protocol is only little high though our protocol

can discover a service which is multiple hops away. Also, our protocol converges and requires very less number of message transfers for the same resulting in very less number of false-positives.

In our protocol, a new user might not be authorized at a service provider though it is subscribed for the service. This is because the service provider might not have the latest/updated list of users subscribed to the service. As a future work, we will provide a new approach to generate tuples at the broker so that the service providers will have the updated list of the users' subscribed. Also, by generating the tuples at the broker, we can renew the keys paired with the service.

10. REFERENCES

1. C. Bettstetter, G. Resta, P. Santi, The Node Distribution of the Random Waypoint Mobility Model for Wireless Ad Hoc Networks, IEEE Trans. on Mobile Computing, Vol. 2, no.3, pp. 257-269, July-Sept. 2003.
2. J. Blum, M. Ding, A. Thaeler, and X. Cheng, Connected Dominating Set in Sensor Networks and MANETs, Handbook of Combinatorial Optimization (Editors D.-Z. Du and P. Pardalos), pp. 329-369, 2004, Kluwer Academic Publisher
3. Steven E. Czerwinski, Ben Y. Zhao, Todd Hodes, Anthony D. Joseph, Randy Katz, An Architecture for a Secure Service Discovery Service, Fifth Annual International Conference on Mobile Computing and Networks (MobiCOM '99) , Seattle, WA, August 1999.

4. E. Damiani, S. De Capitani di Vimercati, S. Paraboschi, P. Samarati, Managing and Sharing Servants' Reputations in P2P Systems, in IEEE Transactions on Knowledge and Data Engineering, vol. 15, no.4, July/August 2003, pp. 840-854.
5. C. Ellison, UPnP Security Ceremonies V1.0, Intel Co., http://www.upnp.org/download/standardizeddcps/UPnPSecurity Ceremonies_1_0secure.pdf, Oct. 2003.
6. David B. Johnson and David A. Maltz, Dynamic Source Routing in Ad Hoc Wireless Networks, in Mobile Computing, edited by Tomasz Imielinski and Hank Korth, Chapter 5, pp. 153-181, Kluwer Academic Publishers, 1996.
7. Y. Li, S. Zhu, M. T. Thai, and D.-Z. Du, Localized Construction of Connected Dominating Set in Wireless Networks, NSF International Workshop on Theoretical Aspects of Wireless Ad Hoc, Sensor and Peer-to-Peer Networks (TAWN04), Jun. 2004.
8. Anirban Mondal, Sanjay Kumar Madria, Masaru Kitsuregawa, ConQuer: A Peer Group-based Incentive Model for Constraint Querying in Mobile-P2P Networks, appeared in 9th IEEE proceedings of MDM'07.
9. M. Nidd, Service discovery in DEAPspace, IEEE Pers. Commun., vol. 8, pp. 39–45, Aug. 2001.
10. L. Xiong, L. Liu. PeerTrust: Supporting Reputation-Based Trust in Peer-to-Peer Communities, in IEEE Transactions on Knowledge and Data Engineering (TKDE), Special Issue on Peer-to-Peer Based Data Management, 2004.

11. F. Zhu, M. Mutka, and L. Ni, A Private, Secure and User-Centric Information Exposure Model for Service Discovery Protocols, IEEE Trans. Mobile Computing, vol. 5, pp. 418-429, 2006.
12. F. Zhu, W. Zhu, M. W. Mutka and L. Ni, Private and Secure Service Discovery via Progressive and Probabilistic Exposure, IEEE Transactions on Parallel and Distributed Systems, vol. 18, no. 11, pp. 1565-1577, Nov. 2007.
13. Sun Microsystems, Jini Technology Core Platform Specification, http://www.sun.com/software/jini/specs/jini1.2html/jini-spec.html, June 2003.
14. "Replay Attack" - http://en.wikipedia.org/wiki/Replay_attack Nov. 2007
15. "Integral test for convergence" - http://en.wikipedia.org/wiki/Maclaurin-Cauchy_test Dec. 2007.
16. "Man-in-the-Middle Attack" - http://en.wikipedia.org/wiki/Man-in-the-middle_attack Nov. 2007.

VITA

Santhosh Muthyapu was born in Hyderabad, India on June 29, 1983. He completed his Bachelor of Technology in Information Technology at Chaitanya Bharati Institute of Technology in Hyderabad, India in May, 2004. Later he worked as a software developer in an MNC for 2 years.

Santhosh started his Master of Science program with the Computer Science Department at Missouri University of Science & Technology (earlier University of Missouri-Rolla) in August, 2006.

He worked with Dr. Sanjay Madria in his area of interest, Mobile Ad hoc Networks. During his research, he developed an efficient protocol called "PrEServD – Privacy Ensured Service Discovery in Mobile Peer-to-peer Environment" and received his Master of Science Degree in December of 2008.