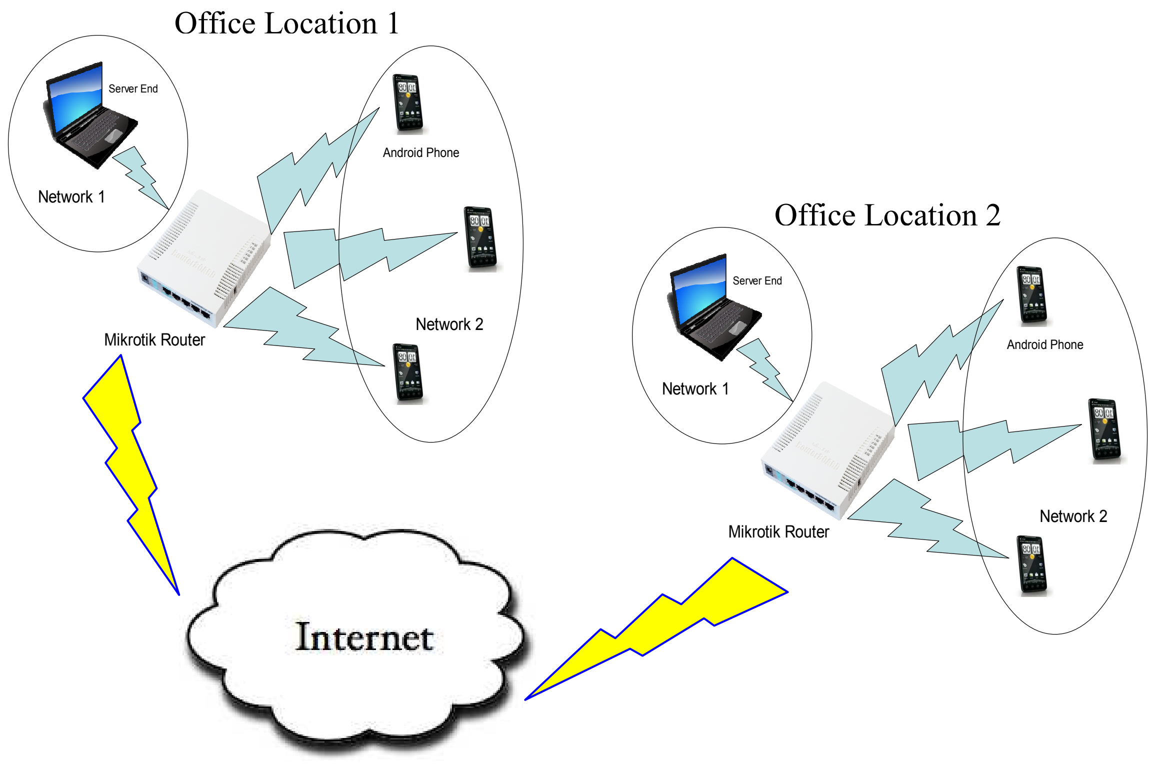
EE 579 Project – User Identification System

Group 6 – Venkatraman Venkatapathy
Abhishek Sharma
Sneha Patil

Project Description:

User Identification System is a system developed to enable communication between invited guests in a meeting carried over at multiple locations. The unique aspect of the project is that it does not use any network provider which is generally expensive. This system involves android phones and Mikrotik router connected to internet as the base setup. We have developed an android phone app which the users can use to communicate with other guests without having their contact number or email ID. This application also allows us to share contact details based on permission from user. The users can send and receives messages and contact information which can be used during the meeting period. The application communicates with the local server for its handling and verification. The Mikrotik routers help us in creating a local WLAN by assigning the devices an IP from a selected pool range. The EOIP Interface built in them helps the two routers connected across the internet to form a Layer 2 broadcast network. The servers located on either locations exchange the list of invited guests and their contact details to enable user interaction between the two sites. Hence this system finds use in office meetings where common data can be handed over to the users on their phone and the users can communicate with one another via messages and share contact Information.

Block Diagram:



Devices Used:

1. Mikrotik Routers -2
2. Android Phones -2

Software Tool:

1. Eclipse with android sdk
2. Mikrotik router OS

Instruction for Operation:

1. Create two files file.txt and rcvdNameList.txt in the project library. Store the list of the invited guest in file.txt.
2. Setup Mikrotik Routers with DHCP Server.
3. Update the following address's
    a. In Client.java:
        i. defaultBroadcatIP (IP address for clients connected to this server)
    b. In MultiThreadedServer.java
        i. clientIP (IP address of the client)
        ii. commServerIp (IP address of the Server in the other location)
4. Run the application UIS on the android phones first.
5. Start MyServer.java on the two servers.
6. Now after verification from the server the users can use the app.

Files Description:

Server Side:

Files used -

1. **MyServer.java**
    a. Runs the Server code.
    b. Used to initiate the client authentication and wait for client requests
    c. Populates hash_map from stored file of invited guests.

2. **Client.java**
    a. Used to start interaction with the clients
    b. The clients receive message from server requesting for their contact information stored in their phones
    c. The IP address of the client can be hard Coded or received from the pool of Mikrotik Router

3. **MultiThreadedServer.java**
    a. Handles and process's different kinds of request from clients and server
    b. Message Types

i. *Server2Server*
   1. attendanceList – To handle request sent from other server to update the List with the received List

ii. *Client2Server*
   1. initialSetup – handle message from client after initial request from server to client for user information. Authenticate user and send confirmation
   2. sendMessage – request from client to send message to another client. Look up in map and send the message.
   3. getContactinfo – handles Request for contact information messages from requester and forwards message to provider.
   4. replyToContact – forwards the reply from the provider to the requester.

Client Side (Android):

Files used:

1. **initialisationClass.java**
   a. Starts the application
   b. Defines the start-up screen
   c. Initiates the communication with the server and waits for authentication
   d. Retrieves the server IP address for future communication
   e. Receives the attendees list

2. **notAuthorized.java**
   a. Handles the alert for a non-authorized attendee

3. **listScreen.java**
   a. Displaying of attendees list
   b. Handles clicking functionality of an attendee's name
      i. Send message
      ii. Get contact info
   c. Viewing the messages and contact information already received

4. **serverService.java**
   a. Background service to receive any packet from the server
   b. Displays the appropriate notification according to the message type received
   c. Upon clicking the notification, the user is guided to the appropriate screen

5. **sendMessage.java**
   a. Opens text box for typing the message
   b. Send button to send the message
   c. Once sent, returns the user back to attendees list screen

6. **messageReading.java**
   a. Handles the message received inbox

b. Displays the list of names of the users who has sent a message
c. Upon clicking the desired name the message is displayed which is defined in readRcvdMessage.java

7. **contactRequest.java**
   a. Handles the alert for notifying the user about a request for his/her contact info
   b. If user selects "Yes" then the contact info is sent else if "No" is clicked then the rejection message is sent

8. **seeContactInfo.java**
   a. Handles the contacts received inbox
   b. Displays the list of names of the users who has sent their contacts
   c. Upon clicking the desired name the contact details is displayed which is defined in readContactInfo.java

9. **negContactRequest.java**
   a. Handles the screen to let the user know about the rejection of his/her request for other party's contact info
   b. Close button will take the user back to the attendees list screen.

Application:

- Automated user verification
- Wireless transfer of data to invitees only
- Permits interaction between invitees without contact information
- Mobile app ensures the phones are in silent mode
- Centralized management of invitees