Presence and Mobile Instant Messaging

Ajith Kannath
Sandeep Paul
AIM of the project

- Develop a presence and chat messenger similar to g-talk, Y-msgr etc.
- Target platform is Android.
- Using SIP protocol.
- Challenge: SIP stack is supported from Android 2.3 onwards only.
- Only 1 phone in market currently (Google Nexus) supports SIP stack.
- Implemented custom SIP server.
- Client – server based SIP architecture is used.
Client 1

Register, Query for Presence information

SIP Server

Instant Messaging

Register, Query for Presence information

Client 2
SIP Protocol

- Clients register to the SIP server with their IP.
- SIP Server maintains list of all available clients and the 'friends-list' of those clients.
- Client queries the server every 1 minute (configurable) for 'presence' of its friends.
- Server returns a list of clients who are currently present & client updates 'friends-list'.
- The user is allowed to pick any friend who is visible and communicate with him.
SIP Protocol (Contd.)

- SIP server also maintains IPs of the registered clients which is send out in response.
- Clients use this IPs to directly message each other.
- If any clients reboots or changes IP all friends are updated within a time span of maximum one minute.
Major phases in the IM application

- There are three major components or views in this application:
- The authorization phase, which lets the user ping the server with authorization details to sign into the system
- A list view which is displayed to the user upon successful sign in, which displays online friends
- A chat view, with text boxes for sending instant messages along with a log of the conversation so far
Client Sign in Mechanism

- The user's credentials are maintained on the server in the form of a hash map which maps usernames to their passwords.
- This structure has been chosen due to its relative ease of implementation in Java.
- On authorization request, the server verifies the user's credentials against his entry in the hashmap and sends a reply message either confirming or denying authorization.
- If username not present in the map, the server responds with message to sign up first.
Client Registration on the Server

- The device must initially register itself with the server to be able to use the server for instant messaging.
- We implemented a simple sign up mechanism, wherein a first time user will be prompted to sign up by pressing the relevant button.
- This button fires off a message with an encoded keyword to the server, after receiving which the server adds the new user's username and password to its user base (in this case the hash map).
Retrieval of Presence Information

- Once a user signs in, his friends list, which is maintained as another list structure on the server is retrieved.
- The server pings the client device with this list, which is then used to dynamically populate a `listView` (Android View type) with all his/her friends who are online at the time.
- The user can then click on any of the items on this list, which takes him to the next phase of the application with the chat and history boxes.
Limitations of the system

- We have implemented a simplified version of SIP (Session Initiation Protocol).
- This was in part due to the lack of support for the full fledged SIP stack on versions older than Android 2.3.
- This meant we had to hardcode the IP of the server (our laptop in this case) every time we moved to a different subnet.
- Dynamic modification of friends list information is not implemented.