Paul Laskorski

CSCI E-65

Final Project Writeup

5/2/2016

**Project Overview**

My app uses the MultiPeerConnectivity (MPC) module to create a proximity based social network. The user creates a profile consisting of some text fields and a profile image. The profile is publicly visible to other users. Users can detect other users with the running application and send messages to each other.

**Editor Tab**

The user first should create a profile by going to the editor tab. There are fields for a screen-name, age, short headline, and longer about me section. Clicking on the profile image brings up the image picker, where the user can change his/her profile image. Clicking the save button will update the user's profile by changing the data model and restarting the MPC services. Other users will see the user briefly go offline and then back online.

**Profiles Tab**

The profile tab shows all of the available peers. When a peer comes online the peer will automatically appear in the table with its associated data. The app will send a request to the user for its profile image, and update the UI accordingly when the image data is received. When a peer goes offline, the status will change and the profile image will change to grayscale.

Clicking on a profile row in the table will bring up a full page view of the profile, with the longer text fields fully visible, and a larger version of the image. The user can tap the chat button to chat with that profile, or the back button to return to the Profiles tab.

**Chat tab**

The Chat tab shows all active chats in a brief 3 message view. The user can tap a row in the table to go to the detailed chat view where messages are sent and received. The message history is automatically updated as messages are sent and received in the main text box. The user can type a message into the text field at the bottom and tap the Send button to send the message, or the Back button to return to the tab view.

**Improvements**

I spent most of my time on the backend, creating a pretty abstract and extensible framework for using MPC to do a variety of things for a basic social network. Adding the ability to send files or do live streams shouldn't take much additional work for the back end. The UI needs a lot of work to make it look better. Additionally, I wasn't able to get the data to persist to disk, so that needs to be done to prevent the user from losing chat history and the profiles. I added the “appID” and stored everything based on that ID instead of the MCPeerID so that everything could be saved to disk since the MCPeerID changes with each app startup. Unfortunately I ran out of time.