Project Report

Mobile Computing - 2012/13

Course: MERC

Group: 5

Name: Bernardo Simões **Number:** 63503 **E-mail:** bernardo.simoes@ist.utl.pt

Name: David Miguel dos Santos Dias **Number:** 65963 **E-mail:** david.dias@ist.utl.pt

Name: Artur Balanuta **Number:** 68206 **E-mail:** artur.balanuta@ist.utl.pt

1. Achievements

In the development of this project, all the projected features functionally was achieved in a portable way capable of being used in the centralized and distributed model.

Features

Authentication

It is possible to authenticate a user on the near tweet app in two ways. The first one requires the user to login to twitter. The second one is based on the user announcing his name on the network. User credentials are stored locally and it is not necessary for user to type again his credentials every time he logs himself on neartweet.

Tweet

Users are able to broadcast tweets over the network to share information with other peers

Reply

Users can reply directly to another users, in *direct reply* mode, meaning only one user will see the message or in *reply all* mode, where every user present in the network will be able to see the message

Integration with other social networks

NearTweet takes advantage of other social networks for authentication, profile enrichment (fetch of user photo) and sharing abilities

Retweet

Once a user is logged with his twitter account, he can retweet the conversation to twitter and continue it from there

Pool

Neartweet enables more types of interaction between users other than only text, one of these is creating a **Pool**, so we can get feedback from our peers with Q&A model

Caching

Clients are able to read previous conversations without having to fetch everything again from the server

Sensor data sharing

For rich user interaction we enable the usability of two common sensors in Android phones

- **Camera, Photo Gallery & Image Url Share**

  Users are able to take photos from the camera, photo gallery or an internet url to share in the network.

- **GPS**

  If user desires to do so, he can share the location of his tweet once it is

written. Localization will find the area belonging to the gps coordinates without the need to access the internet

Spam detection and eradication

To avoid abusive behaviours in the network, we give the user the possibility to report spam another user and if there is detected a pattern of bad behaviour by our users, the user is banned from the network.

View tweets on a map

Neartweet is also a great place to find where our friends are, we've enabled a tweets map view so the user can know where the tweets came from.

Communication

Centralized (client-server model)

Although Neartweet was thought for the mobile world where internet access is inconstant, it's possible to use as a client to a server in the cloud, enabling communications between peers in geographically distant areas, out of range of a local Wifi network

Distributed (P2P network over Wifi Direct)

To enable Neartweer in every conditions, it's used Wifi Direct API to establish connections between peers that find themselves in each other proximity zones

2. Specification

NearTweet app was designed from the beginning to enable communications between peers without access to the world wide web, this service can be very useful in scenarios where connectivity is limited, such as remote villages or festivals where 3G/4G become overloaded.

Where NearTweet excels

NearTweet solves the problem of applications relying on an internet connection in order to be possible to connect users. A spontaneous NearTweet network can be the equivalent to a twitter post with a hashtag describing a specific event or location. Since NearTweet does not rely on the use of internet it removes the need to have a 3G/4G limitation from the users.

With the increase of intaking information people are driven to not wanting to save information for the future. This totally describes NearTweet since once you are off the network all information about it will disappear, this way privacy is assured, just like a normal human conversation.

On the other hand it also possible to record the tweets that you have most liked for the future by sharing them on twitter.

Where it could be enhanced

Since networks are spontaneously the first user to arrive at a place will be using a one man social network until more people arrive, this might be a drawback for users to use the application in comparison to centralised social networks with network access where it feels like

that there is something happening.

Extra functionality

It would be interesting for NearTweet to have a way to stream local networks to the world. For example if you could not go to that concert that you love so much but you really want to know what is happening in there, you could connect yourself to your browser/mobile app and see what spontaneous networks are being created during that event, see photos people are taking and know what the people that are voting on the pools.

It could even be possible for a user that would like to know what happened on a determined event in time to go back to where he was and what he was saying.

"NearTweet against the cloud"

With the increase of computational power on mobile devices, and the arriving of new standards like *"Wifi Direct"* and *"NFC"* it is predictable that we will start to see an emerging new market of apps based on local device discovery and communication instead of the cloud apps

3. Design

The communication between nodes is done using sockets. Each socket has two handlers, one for the input data and another one for sending data. In the Client-Server model the Server works as a HUB and routes the data to all the Nodes, all the processing and filtering is done at the edges of the Network. This makes the routing process faster. The Nodes in our case the smartphones running Android, have a service running that handles the Socket and the buffering of the messages. Every Time a message arrives the service informs the running activity, the activity then takes care of displaying its contents to the user. The routing is done via the Device ID that every Phone have, this ensures the difference between the nodes and is used as an IP for delivering NearTweets.

In the Distributed Model we migrated the Server and made it a part of the service. When the application wants to create a new Network, it just informs the Service to create a new local server. Then the Server creates a handler Thread to manage client connections that listens for incoming connections. When using WiFi-Direct the GO has to provide the Server, this ensures that only one server is running at a given time, and also minimizes resource utilization.

"Activities"

While developing our application we took in consideration the Android Developers guidelines[1]. You can see our *"Class Diagram Interaction"* on the Attachments, section 7.1.

Our navigation and content select is done on an Split Action Bar[2] that renders diferently according to the device screen size and offers navigation to the previous screen.

We divided our interface in 6 activities instead of hiding/removing elements, this gives the application a sense of fluidity between the launching of intents and gives the proper use to the

[1] http://developer.android.com/training/index.html

[2] How to use android action bar: http://developer.android.com/guide/topics/ui/actionbar.html

Android back key[3]. All our application ScreenShots can be found on section 7.2 on the Attatchments. The MainActivity is where most of the app logic resides, it implements location event listener, and ConnectionInfo Listener to listen to Wifi Direct events such as pears leaving the network and so on. Here reside also some user preferences like associate a twitter account and enable or disable location sharing

The LoginActivity is launched in case the user is not logged and will provide methods for the user to authenticate himself on the NearTweet app using twitter or just locally.

The NewTweetActivity is used for the user to create a new tweet, it lets the user chose if he wants to add photos from his gallery, camera or url.

The NewTweetPoolActivity is similar to NewTweetActivity but it lets the user create new pools.

The TweetDetailsActivity is used to see more information about a tweet specified on the MainActivity, here you can launch an intent that will show you where in the map the tweet was created and it is also possible to reply publicly or privately to a tweet by calling the NewCommentActivity which I will describe next.

On the NewCommentActivity you will be able to create either a public or private comment on a tweet.

Finally we have the TweetDetailsPool which is similar to the TweetDetailsActivity but is optimized to show TweetPools, and gives the user ways to select their options by executing a long-press.

4. Implementations Choices

AsyncTasks and Services

To make the interface fluid we used a combination of a Service running in background, and AsyncTasks running in the Activities. The Activities that need to have access to the network layer, use the AsyncTask that binds with the service. From that moment on the AsyncTask updates the UI when it is necessary. The Service is in charge of connecting with the Server and maintaining the connection. It also stores the receiving data, for easy access from the AsyncTasks, this way we don't need to transfer data between the Activities and it is super updated.

Authentication

In order to authenticate users on a decentralized network we need to be able to identify the user device, this could be done in numerous ways, like read the wifi mac address, reading the serial number or reading the ANDROID_ID. As is described on the android web site[4] the best way to track a device installation is by reading his ANDROID_ID so we used it to identify each user in the NearTweet app.

On our application we choose to let the user pick two different kinds of authentication, by logging in through twitter where we automatically retrieve his profile picture and user name and

[3] How to navigate on an Android application: http://developer.android.com/design/patterns/navigation.html
[4] http://android-developers.blogspot.pt/2011/03/identifying-app-installations.html

add them to our application. The second way is local where the user picks his own name.

After the authentication process all data that the user inputs is saved in a SharedPreferences objects and can be accessed on all activities and even after the phone is turned off or the app is closed. The user preferences are only cleared when the user uninstalls the app.

Android Version

We choose to use Android 4.2 with API 16 in order to be able to use the wifi direct on our devices and be able to actually see this app working in real life.

Wifi Direct

We use Wifi-Direct to establish the communication between peers, when peers launch the app, a discover peers is launched and the app will connect to the peers that he finds, creating a group where one assumes the role of group owner(or GO). The GO will be the rendezvous point for all the peers in the spontaneous network, having a server socket always listening, propagating the messages sent by each peer through client sockets.

5. Experimental Evaluation

All the feature functionality was tested in the both environments, centralized and distributed with more than 2 peers to validate the ability the interaction between several users.

For the experimental evaluation, we used two Nexus 7 running Android 4.2.2 and one Galaxy Nexus running also Android 4.2.2.

The GPS functionalities were tested using different geographic points, to verify the identification of the address.

6. Conclusions

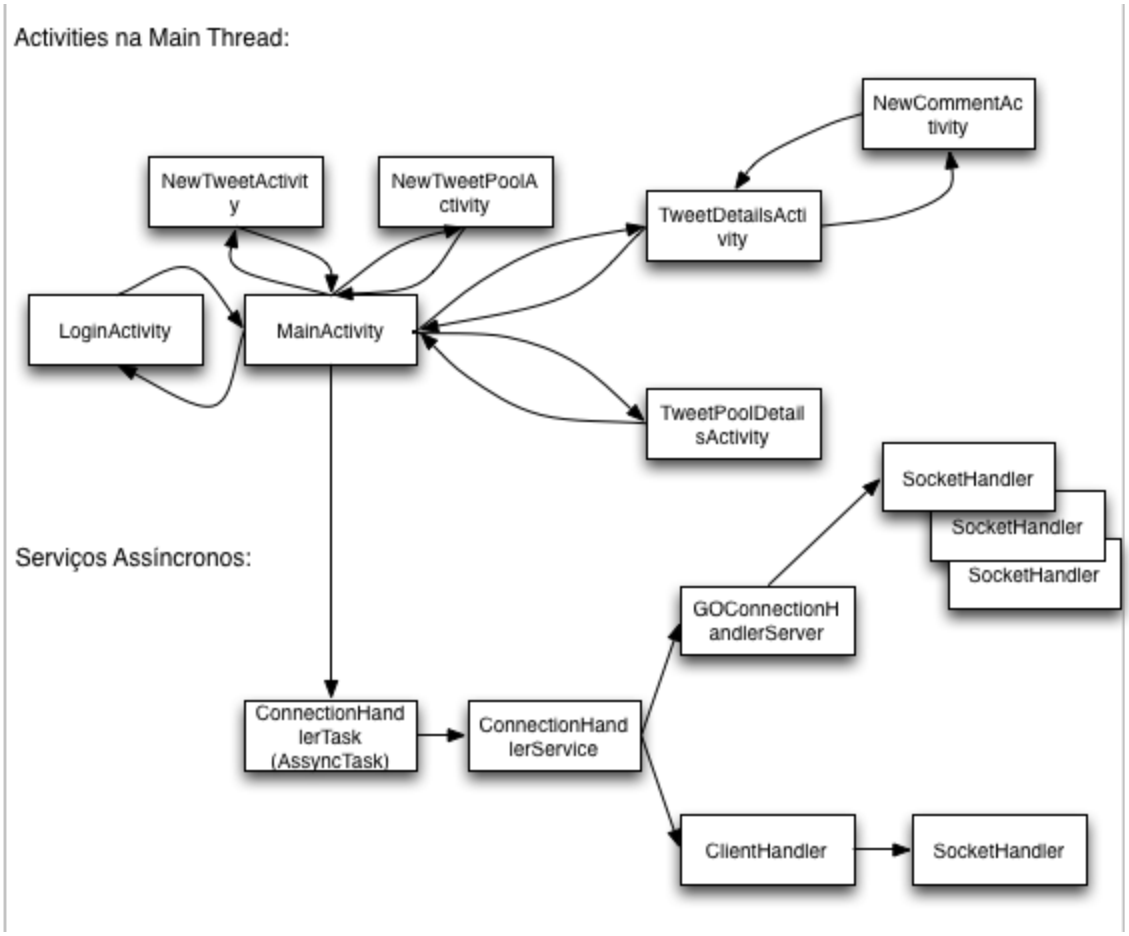
Our group got very excited with this project because it was one of the first courses that enabled us to actually deliver a real app, usable in the real world, and Android was the right choice to do it, it's by far the most used mobile operating system and the one with the best community supporting it.

One of the main challenges in building NearTweet was using State of the Art technologies such as Wifi-Direct, it's a very recent technology implemented on the Android operation system, making the documentation and the community around very scarce, making extremely difficult to achieve good results in a short amount of time. We believe however that this will change with time since it enables new type of applications.

Regarding course improvements, we felt, since developing for Android and mobile applications in general is a completely different approach than it was done so far during Bachelor and Masters degree, thanks to Application life cycle and machine resource limitations, students could greatly benefit if the Android labs started from week 0, we didn't find useful 'scratching' J2ME or HTML5, since the knowledge was used.

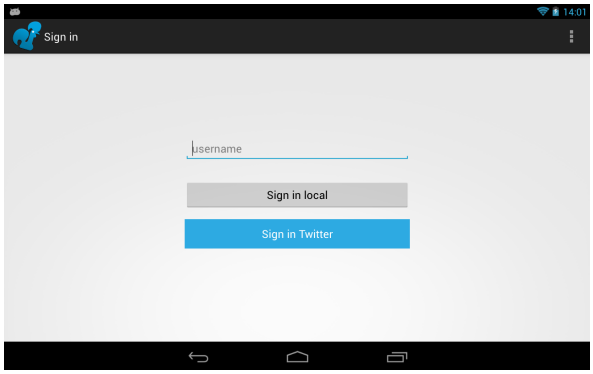
7. Anexos

7.1 Class Diagram Interaction

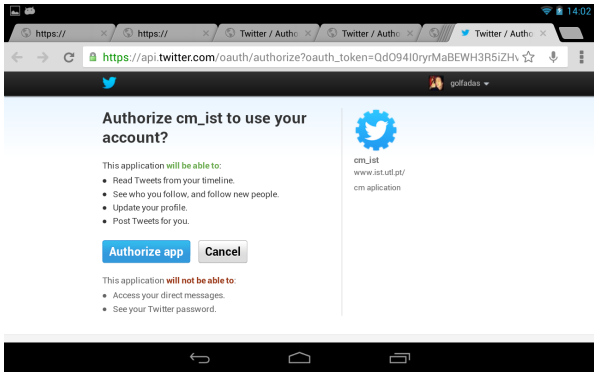


7.2 ScreenShots

Login Activity



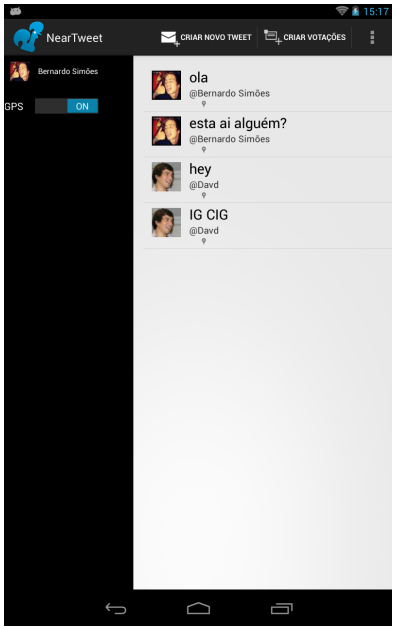
Login by twitter

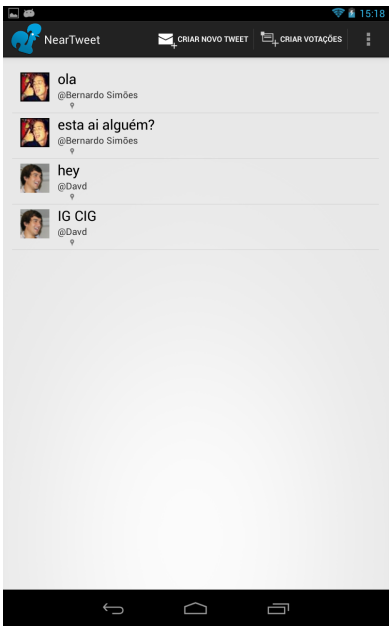


Main Activity

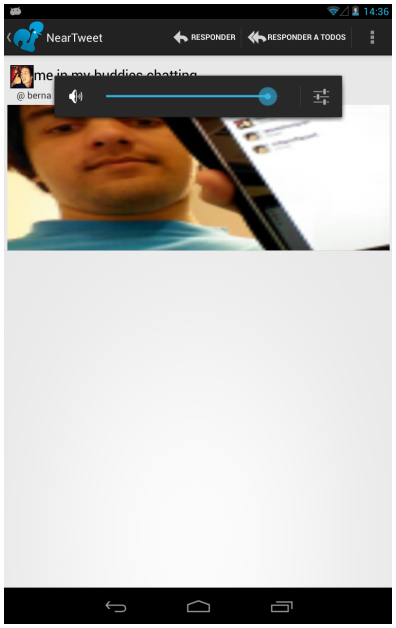
Spam reporting

NewTweet

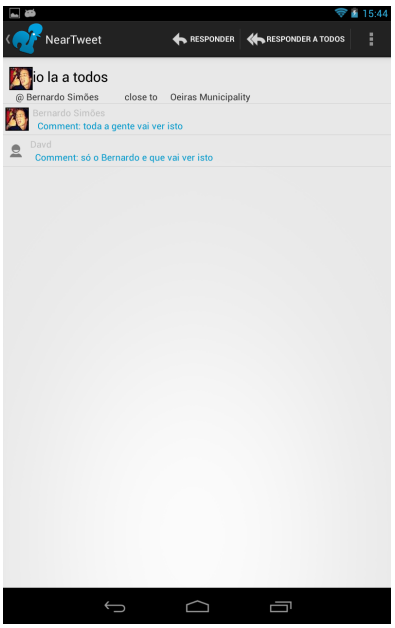




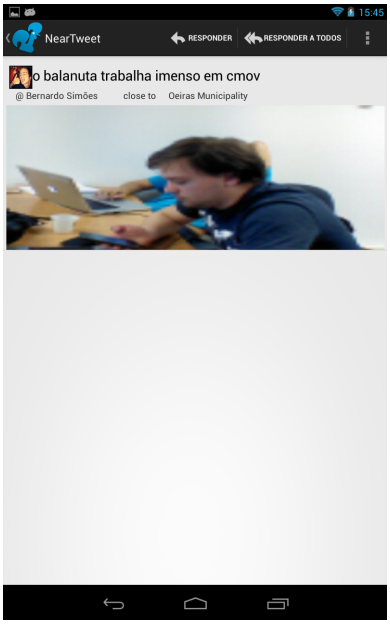
TweetDetails



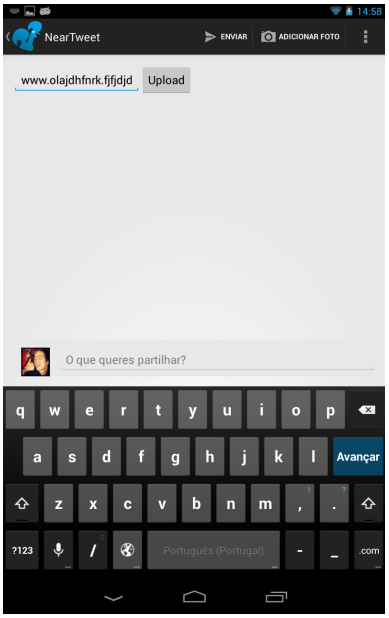
TweetDetails comments



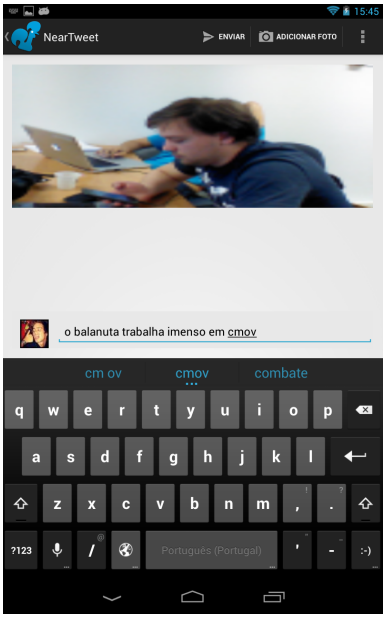
TweetDetails photo



NewTweet imagem por URL

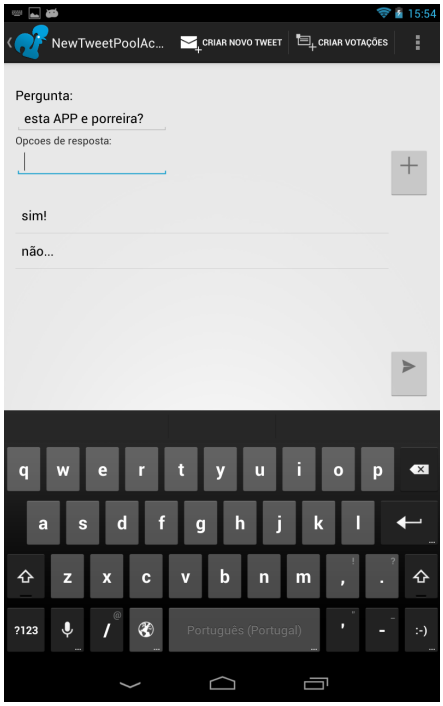


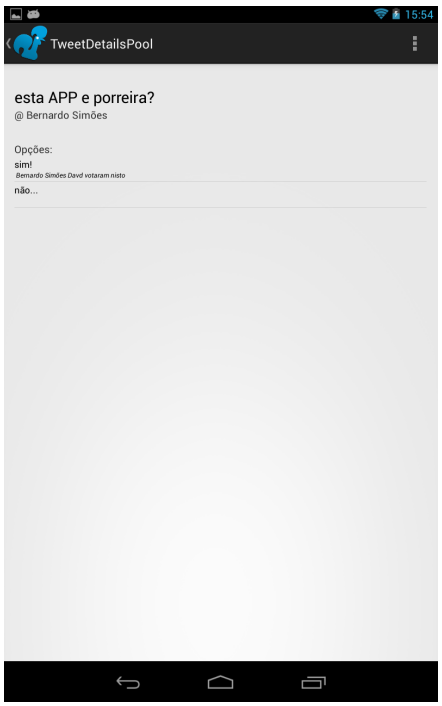
NewTweet com photo



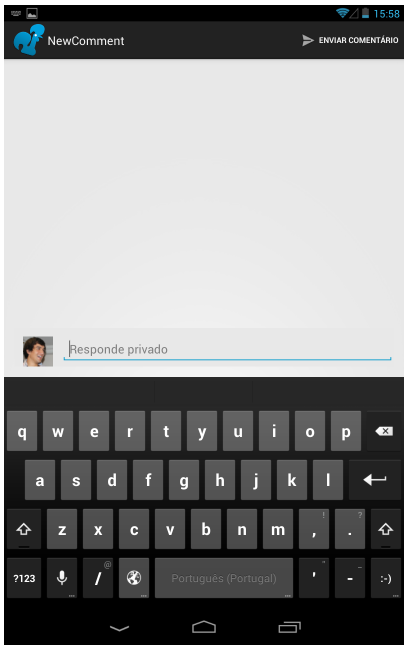
NewTweetPool

TweetPool

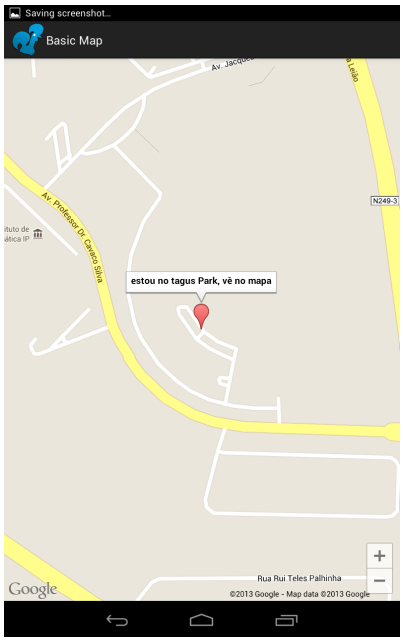




New Comment



See Tweets on Map



Blocked User