University of Cambridge

Computer Science Tripos, Part IB

Group Project Team Echo

Multi-touch Conference
Final group report

Authors:
Mona Niknafs (mn407)
Yojan Patel (yp242)
Alexandru Tache (at628)
Philip Thomson (prt28)
Petar Veličković (pv273)

Client:
Catherine White, BT

27 February 2014

Contents

1 Introduction

1.1 Purpose

This main purpose of this document is to present the final version (release 1.0) of the Multi-touch Conference group project undertaken by Team Echo. This is accomplished by largely building upon the *Progress report* document that was made on 12 January 2014. The document is structured as follows:

- The **introduction** section consists of notes from the second client meeting on 13 January 2014, and a summary of the deliverables reached since then;
- The **module integration and testing notes** section consists of descriptions of various new features that have been successfully implemented since the previous report;
- The **undertaken work summary** section consists of brief summaries of each team member's contribution for completing the system (the contributions will be elaborated in more detail in each member's *personal report*);
- The **further work** section consists of proposed further work that will be implemented, should this project remain maintained in the future.

The project's source code stored on GitHub is now public and may be accessed on `https://github.com/PetarV-/CSTIB-Echo`. The project has been attributed with a CC-BY-NC-ND 3.0 license (Creative Commons Attribution-NonCommercial-NoDerivs).

1.2 Second client meeting notes

The second client meeting was held on 13 February 2014, with client Catherine White from BT. The focus of the meeting was discussion on ways in which the system's functionalities can be extended. Each module of the system was discussed, with accompanying presentation of both the Android and Touch client applications. The discussions made on each of the components are summarised below in turn.

∼ **Data Analysis** Regarding this subsystem, it was pointed out that the searching mechanism will not function as desired if different users use *synonyms*; different words to mean the same term. The system could be extended to use an alternative mechanism for keyword searching – *Marvin Minsky* approach.

∼ **Server** One possible extension to the system was identified, in particular to use a *graph database* as storage instead of a conventional relational one. There exist both advantages and disadvantages with such an implementation; of course adding new classes will be easier upon extending the server under a graph database.

∼ **Android** It was noted that *bookmarks* would be useful for users to keep track of important conversations, especially when the number of conversations in a conference becomes very large.

∼ **Android Client Viewing** *Tag clouds* were mentioned as a more interesting feature for to highlight conversation keywords and their corresponding 'weights'. The positioning of *user avatars* was discussed, with regards to the chat window.

∼ **Touchscreen** Some useful features of the current touchscreen were pointed out. Users with incompatible portable client systems can still be involved in a conference, via the main display; additionally, information relevant to a conference or conversation can be interactively shared on the display – for example *business cards*. This is similar the the *MIT Glass Infrastructure* project, extended to an interactive display environment. Potential extensions of this subsystem included: *clustering* related conversations on the screen, *facial recognition* of which Android users are around, and assigning each Touch Client its own *profile*.

$\sim$ **Touch Client Viewing** To further develop the user interface of the touch client, it was recommended to draw an *animation sequence* of user interaction events with the screen. This would give a clearer picture of what to code towards. The QR code provided on the top right corner of the chat window became hard to scan when a chat window is *rotated* by certain angles, a solution proposed was adding a function to *increase* its size for a few seconds. The *global statistics* to be provided on the touchscreen were discussed. A *dynamic map* of active users could be used to display locations of active users in a conference. The number of *users logged on* was recommended as another possible display attribute, along with number of *active users* and *posts per day*.

1.3 Deliverables reached

Prototypes $'\delta'$ and $'\lambda'$, as stated in the *Progress report*, have already been achieved.

Our final desired prototype, $'\xi'$, has been achieved through the integration of all the system modules and the development of the touchscreen client, Android application, data analysis and client to give a useful and intuitive system for the targeted user group.

In reaching the final prototype, the acceptance criteria of the system as outlined in the *Functional specification* document have been achieved and therefore the system is **completely ready** for the release.

To further extend the system, further application layer functions would be implemented, to make the best use of the features provided by the data analysis section. (tag clouds...)

2 | Module integration and testing notes[1]

2.1 Server

2.1.1 Core server API

Integration notes

The additional work on the **server API** consisted of adding support for **avatars**, enhancing the **user model** in the database in order to store more information, and implementing **user authentication**.

∼ **Avatars** For the avatars we have decided to use *Gravatar*, a service that allows users to register, for free, an avatar using their email addresses. The avatars are not stored on our server; we only generate links to the appropriate avatars hosted on Gravatar by hashing the user's email address.

This approach has also allowed us to take advantage of Gravatar's *"identicons"* (randomly generated avatars) for users without an email or a registered avatar.

∼ **Additional user information** In order to store all the data needed by the analysis module we have added the following fields to the `User` model: first name, last name, display name, phone number, email, job title, company and gender.

We have also added a new model called **Interest** that is in a many to one relation with the user model. The new model is the equivalent of tags, but for the user, and its main purpose is to make personalised queries more relevant.

∼ **Authentication** We added the ability for the users to authenticate using their username and password. In order to achieve this we have stored the *SHA-512* hash of their password in the database.

Testing notes

The testing methods are the same as the ones described in the *Progress report*:

1. We organize and run our tests with JUnit, a general purpose Java testing framework.
2. For each test we set up a in memory database populated with predefined data, using Hibernate's support for fixtures.
3. We use the Jersey testing framework that allows us to simulate http request to the server.
4. We then inspect the output of the server and the consistency of the database after the requests.

Issues

Even though we have added authentication using hashed passwords stored in the database, we still have a security vulnerability that user should be aware of. Since our server hasn't yet been configured to run on SSL, while authenticating or setting a password, the password is sent in plain text.

2.1.2 Data analyst

Integration notes

The bulk of the additional work done on the **data analysis** module consisted of implementing additional analysis methods, most of which have become implementable only after the fully implemented `User` class had been made. The summary of the undertaken work is given in bullet point form:

[1] **Note:** This section only focuses on newly developed system features and as such does not cover the material already described in the *Progress report* document.

- **conversation searching** has been made completely *granular*; clients are now able to choose which `Conversation` parameters (names, tags, keywords) they want to query over, depending on the type of results/efficiency desired;
- **personalised queries** have been implemented; these methods represent one of the key selling points of the product, and involve recommending new conversations to a `User` upon leaving a conversation, as well as periodically notifying him/her of the interesting conversations occurring elsewhere, to emulate the concept of *overhearing* which is common for physical conferences;
- **additional methods** have been added by request of team members working on the other client implementations, and they are as follows:
  - **activity** metric for a conference, representing a number closely related to the message throughput through the conference in a given interval;
  - **M:F ratio** of users in the conference;
  - **message count** within a single conversation and within the entire conference (mostly implemented to relieve the clients of performing this computation);
  - **user count** within a conversation and within the conference (similar as previous point);
  - **contributing user count** within a single conversation (can be restricted to only currently active users).

Testing notes

The testing procedures for this module that have been outlined in the *Progress report* document have remained in use throughout this project development phase as well; the methods were tested by a combination of HTTP requests and client communication to the server hosted on Heroku. In addition, the new personalised methods resulted in creating a new static class to deal with extracting keywords from a `User` object. This has been tested locally using JUnit on users with varying levels of personal detail before being deployed. The test results were excellent with respect to both efficiency and quality.

Issues

Nothing of significance to report in terms of encountered issues.

2.2 Client

Integration notes

The additional work done on the **Client API** consisted of adding support for the new features that were added to the server: **additional fields** for the user model, user **interests**, user **authentication**, and new **statistics** from the data analysis module.

Testing notes

Testing was done as described in the *Progress report*.

Issues

We have encountered a multitude of issues while making the client library integrate fully with Android; unfortunately, our library depends on features of Java SE 6 that are only partially implemented by the Android development library. Hence, we had to include jars with the missing Java library classes in order for everything to work as intended. This successfully solved the problem, and the Android client now functions as any other client implementation.

2.2.1 Touchscreen client

Integration notes

All of the concurrency issues mentioned in the *Progress report* document have been resolved using **locks**, which have been made fine grained to improve the efficiency of the program. The **touchscreen client** has had the **global statistics panel** and the individual **conversation stats** added. These additions have brought lots of new difficulties to the server connection, residing with the need for the client to poll

the server. Among other questions this brings up, is clearly: How often should the client poll the server for statistics?

The design of the global stats panel has also been updated and changed to display different **views**. This maximises the size of each of the statistics/charts. Also the panel has had the shape change from an ellipse to a **circle**; this change makes it easier for the global statistics panel to be viewed from all sides as rotation causes less disturbance to other panels.

2.2.2 Android client

Integration notes

The additional work carried out on the **Android client** was a major refactoring of the original prototype. Initially, a new connection to the server was made on every request (every message send, receive, load view etc.) in order to get an idea of how the Android client will interact with the other components of the system. The refactoring carried out has been to use an **Android Service** that outlives the actual application with the single instance of the connection. This has sped up network calls greatly and the app runs more smoothly.

User authentication has been implemented on the server allowing the dummy login screen to be replaced with the actual user authentication with personalised data along with **avatars** to be displayed alongside each conversation. An activity where the user can modify their **profile** (display name, email, phone number, interests etc.) has also been added. The addition of the Service has also enabled **system notifications** to notify the user of any activity when not in the application. The ability for the users to register interests and other information has also resulted in the system including an original feature called ***"Overhearing"*** which periodically notifies the user of other conversations the user may want to join in the future to encourage the user to always participate in the conversation most relevant to their interests.

Additionally, the conversations have an additional view where one can see all active users and view their profiles and **call** if the other user has entered a phone number.

Testing notes

The app was given to various people both of a technical background and without to offer feedback on the user interface, functionality and the system as a whole. The general opinions were very favourable and any subtle hints I felt were justified have been implemented.

Technical testing also took place in various forms by overloading the app with several dummy conversations (testing the list views), many messages at once (testing the notifications) and using the messaging system to communicate with other members of the group on a few occasions to discuss various topics.

Issues

The main issue has been the introduction of many useful features being introduced in a very recent Android API (nested fragments) which has constrained the compatible devices more than expected. However, given more time this can easily be rectified by using the Support Android library. Additionally, dedicated landscape layouts were not created as it essentially doubles the amount of development time which would have limited the the functionality given the timing constraints.

2.2.3 Computer client

Additional work on the text-based **computer client** revolved largely around implementing support for the newly-implemented `User` class (particularly handling secure authentication), so that this could then be easily replicated in the other client implementations.

It remained a command-line tool throughout this phase of the project, and it likely will maintain this form in possible future iterations.

3 Summary of undertaken work

3.1 M. Niknafs (mn407)

During the project I took on a more managerial role which involved some **administration** work, such as taking minutes at the regular project meetings and client meetings. I worked in the **design process** of the **touchscreen client** user interface through modelling animation sequences of user interaction and implementing this on the screen. I am currently working towards implementing **tag clouds** to be displayed in the local and global panels on the Touch client. I have learnt that *constant group communication* is key to a successful project and system; thankfully, in our group this was present from the outset.

3.2 Y. Patel (yp242)

In the system, my main role was to design and develop the **Android application** that acts as the main point of user interaction. As such, a lot of time was spent making the UI as intuitive as possible. To achieve this, I have created various Android **activities**, **fragments** and **services** to carry out the front-end handling of input as well as **adapters** and the use of the Client API to interact with the server. I also developed all the **layout** files (*.xml) as well as the non-standard **graphics** used in the app. The app also provides various external functionality such as QR code scanning to join a conversation as well as dialling a phone number. My contribution to the system as a whole was consistent input along with the rest of the team on the direction the product would take, functionality it should and should not support as well as various design ideas on the touchscreen client. I was also working closely with the API developers to ensure they understood the methods that had to be implemented for the Android app to work as intended.

3.3 A. Tache (at628)

I have primarily been responsible for implementing the **server** (excluding the data analysis module) and the base **client** library that communicates with it. My main contributions to the project can be summarised as follows:

- Creating a REST based **HTTP Server** which provides data as JSON;
- Storing and retrieving the data from an SQL **database**;
- Providing efficient message updates using **HTTP Streaming**;
- Creating a **client library** to communicate to the server that works both on a desktop computer and on the Android OS.

3.4 P. R. Thomson (prt28)

I have primarily been responsible for the implementation and testing of the **touchscreen client**, which has had many successes, mainly the fact that several components of the system are fully functional, including:

- touch movements;
- touch gestures;
- server connection;
- message display;
- statistics display.

The touchscreen client has one primary failure in that you cannot dynamically add or remove conversation panes, which has taught me a lesson that the conversation panes should have been made as separate components, which would have allowed for greater flexibility in the design.

3.5 P. Veličković (pv273)

At the start of the project I have decided to take up implementation and testing of the **data analysis** module of the server. My responsibility for this module remained until the end of the project, however my final contribution list spans a much wider range of areas. A summary of that list is given below:

- several contributing **ideas** during meetings, out of which the most important one is the initial proposition for what ended up being the final **multi-touch screen user interface layout**;
- complete implementation and testing of the **data analysis** server module - this involved devising the **natural language processing algorithms** described in the *Progress report* document;
- several **bug fixes** as well as a few **additional classes** in the core server API;
- **typesetting** all of the formal report documents and presentation slides in LaTeX.

4 | Further work

Most of the further work that we propose for the project, should it continue to be maintained past the code deadline, revolve around what was already proposed in the functional specification document; several ideas given to us by the client have also made it on this list due to time constraints. A brief summary of them, for future reference, is as follows:

- Further improving the **metrics** and **quality** of the data analysis module; this would require, among other things, implementing support for handling **synonyms** with respect to keyword extraction, considering the tradeoffs with using a **lemmatizer** instead of a stemmer to reduce a word to its base form (perhaps allowing the client to choose), etc.
- Consider the different available **database models** for the server as an alternative to the relational model, and their relative merits.
- Expand the **abilities** of the **multi-touch screen**, allowing users to be able to use one of them to integrate with a conference, in the absence of an Android application for input; with virtual/remote keyboards for instance. This would likely require a much larger screen.
- As an addition to the previous point, users could be allowed to have a **personalised section** of the screen, which they can lock in their absence and use to access conversation/stats/files relevant to them. This concept was heavily proposed in the planning stage but ended up being discarded for a more global approach; however with a larger screen and less time constraints this would definitely be a useful extension.
- Integrating **voice messaging** (resembling phone conversations) where each attendee would be wearing a headset associated to the conversation they are currently partaking in, and augmenting the data analysis module with sound processing.
  - We got a partial implementation of voice communication already by enabling users to call each other from the Android app (provided the callee has supplied his phone number), and further development would go along this direction of allowing "sound messages" to be broadcast within a conversation.
- Integrating **remote sharing** and **annotation** of various files such as spreadsheets, PowerPoint presentations, source code, PDFs etc. is something that would be extremely valuable to have in an office conference environment, and as such is highly ranked on this list.
- Expanding our platform base, in particular, by implementing an **iOS client application** similar to the Android one. This is something that should be left for a time when we're confident that we'd like to ship this product to a wider user base, as successfully developing for iOS and being able to publish to the iTunes Store requires an Apple Developer account, which is not free of charge.