MCKit: a Mobile App for Conferences

Lei Zhan and Dah Ming Chiu
(zl011, dmchiu)@ie.cuhk.edu.hk
Department of Information Engineering
The Chinese University of Hong Kong

This article is an editorial note submitted to CCR. It has NOT been peer reviewed. The author takes full responsibility for this article’s technical content. Comments can be posted through CCR Online.

Smart phones have become very popular. Most people attending a conference have a smartphone with them; so it is natural to think about how to build a mobile application to support a conference. In the process of organizing ACM Sigcomm 2013, we initiated a student project to build such a conference app. As a conference organizer, we had good motivation and inspiration to design functions we would like to support. In this paper, we share our experiences, in both functional design and implementation, as well as our experience in trying it out during Sigcomm 2013.

1. FUNCTIONS

The foremost question is what functions you would like to support in a conference app. We spent quite a bit of time brainstorming on it. We group the functions we designed and implemented into the following categories. A basic distinction is between online and offline functions. For the former, the app needs to contact a server to get information. If the server can be accessed via WiFi, then it is most convenient; if it can only be accessed via 3G/LTE, then it may be quite costly for people attending the conference from a different country. Offline functions do not need server access; they include functions that can rely on locally cached information, or functions that require local interactions between two smart phones. One example of such offline functions is the exchange of electronic business cards.

1.1 Publishing

The most immediately obvious function to support by a conference app is conference-related information publishing. At most conferences, each attendee gets a booklet which includes the program, schedule, and various logistics information, such as time of each activity, including coffee breaks, reception, lunch and dinner, if part of the conference. This is the minimum a conference app would support, and is what some currently used conference apps do [1, 2]. The implementation of this function is directly related to the publication of similar information on the conference web site. It can be accomplished as a special version of the conference web page adapted for the display form factor of smartphones.

There is also a calendar function that allows attendees to pick activities they want to attend, to be placed in a calendar, and get reminded in the way preferred. The calendar function is offline.

Another important kind of information is conference announcements. Even during the conference, the organizers may have important announcements to make. For example, during Sigcomm 2013 in Hong Kong, we had a relatively strong typhoon (tropical wind) passing through the vicinity of Hong Kong. The conference needed to be suspended briefly, and resumed. For such purposes, the announcement function in a mobile app has some distinct advantages (when compared to a web-based service). The announcement can be pushed to the client side, and the client side can make a sound to catch the attention of the user. The left screenshot in Figure 1 shows the announcement we sent through the mobile app to conference attendees regarding the conference rearrangement due to typhoon.

1.2 Feedback

In conjunction with information publishing is feedback collection. This is normally done at the end of a conference, by passing out a form for attendees to fill. By implementing it in a mobile app, the attendees can give feedback as the event/activity is going on and we can also collect feedback on more things.

In our design, we allow users to give scores to paper presentation and paper technical quality separately, for each paper. The right screenshot in Figure 1 gives the vote page of our design. Since each user is registered, such scoring cannot be rigged by doing it multiple times. There is also a discussion forum for users to give brief comments on each paper. Along the same line, attendees can also give feedback to other activities such as coffee break, lunch, reception and banquet.

1.3 Business card exchange

One important activity in conferences is simply to meet people, both old friends and new acquaintances. A common practice is to bring lots of business cards to exchange when making new contacts. This is something that can be easily and better achieved using a mobile app. The information contained in a business card can be encoded into a QR code. This information can be received by another person by quickly scanning the QR code of the sending person.

In our implementation, we printed such a QR code on each attendee’s badge. So exchanging business cards can be accomplished by two people allowing each other to scan each other’s badge. To make it easier to activate the scanning function, the user only needs to start the conference app and shake the smart phone a couple of times. Note, this is implemented as an offline function, as we want to allow people to do this at any time and anywhere they want to. Exchanging business cards this way allows the users to easily keep all the business cards electronically, which makes...
Figure 1: Notification Page and Vote Page

it easy to manage later on. Having all the business cards as paper cards is clumsy for retrieval, not to mention the tendency to waste paper.

1.4 Social functions

It is also helpful to let people find out who else are attending the conference, and to make contacts that way. In this case, we need to strike a balance between information availability and privacy. In our implementation, we let any registered user (in the conference) browse and search for other attendees, but only exposing the attendee’s name and affiliation. If one wants to get other information on a person’s business card, such as email address and phone number, the owner’s permission must be obtained first. As the owner of one’s own business card, we can either allow others to access our business card one at a time, or by doing a group permission. It is possible to make one’s own business card completely public as well. Once users have IDs, it is possible to implement all kinds of social network functions. One type of functions is one-to-one messaging, whether text or multimedia. More extended features would include grouping users and interacting with each other as a group. In theory, all functions in WhatsApp, WeChat, Twitter and Facebook can be implemented to some extent in this platform. In our case, we only implemented some simple one-to-one text messaging functions. Obviously, there are trade-offs between implementing these social network functions internally and invoking them externally relying on the existing social networking applications mentioned above.

1.5 Navigation

The conference attendees may not be familiar with the conference venue and city. Any navigational information and help should be useful, in particular maps and building floor plans.

Sigcomm 2013 was held in the Chinese University of Hong Kong (CUHK). The conference venues, including lecture halls, meeting rooms, demo and poster areas, and lunch places, spread across multiple buildings and floors. All these facilities are within walking distance to the conference hotel and public transportation. But for a visitor, some maps would be very useful. For the conference venue, we included an offline map with added labeling for the buildings we used.

Since the CUHK campus covers a large area, we also tried to provide some location identification service. This is again based on using QR code. At strategic locations people may visit, the conference helpers created small posters of the conference with a QR code. By scanning this QR code, a visitor using our conference app will see the location on the venue map, to help navigation.

For offsite navigation, visitors may use other familiar online services, such as Google Map. But since it may be expensive for visitors to access such online services, we also provided some offline maps and local information for users, such as information about public transportation.

1.6 Options selection

Finally, the conference app provides an interface for attendees to select various options the conference events offer. For Sigcomm 2013, lunch needed to be spread to two different cafeterias, and there were several lunch menu options i.e. Chicken vs Fish vs Vegetarian. The food provider wanted to get a definitive idea of the number of people opting for each choice, to help them plan. By implementing such options selection in the mobile app, attendees can also get more information about the lunch choices and locations. Since not all attendees were using the conference app, the actual lunch arrangement did not rely on this function.

2. IMPLEMENTATION

2.1 Major components

Our platform has three major components, as shown in Figure 2. They are: the Mobile App, the Management Interface and the Web Server. It is a straightforward client-server application. The Management Interface is a special client for use by conference organizers, whereas the Mobile App is the regular client for conference attendees.

Figure 2: System Framework

Mobile App

The Mobile App is the core component of our platform. We name the app as Mobile Conference Kit [3] and abbreviate it as MCKit. MCKit is for conference attendees to access conference information, send messages, share content, give feedbacks, exchange business cards and so on. Among various mobile operating systems, we picked Android and iOS for developing MCKit since they are most...
popular. After we designed the prototype, we first implemented the Android version. Then we ported the Android version to iOS.

**Management Interface**

By convention, a conference always maintains an official conference website where the organizers publish pertinent information for the conference. Since the conference app needs to publish much of the same information, a retrofitting way is to extract the information from the web site and then publish it to the mobile client. Although this is a possible method, it is quite awkward and error-prone. The initial reason for the Management Interface is for the conference organizers to create, publish and manage conference information for MCKit. Once we have the Management Interface, it is possible to support more management functions. Most notably, we can use it to make announcements in real-time. It is also through this interface we can generate various reports related to the conference, such as number of attendees, and most popular papers at the moment.

**Web Server**

Basically, the Web Server stores all the information created by the conference organizers through the Management Interface; and it is responsible for responding to requests from both MCKit and the Management Interface.

### 2.2 Data communication

#### Web server access

Both the MCKit and Management Interface need to interact with the Web Server. We implemented the server as a RESTful web service [4] using HTTP and REST principles so that it provides a uniform interface and stateless communication. The uniform interface simplifies the system architecture, allowing the server to respond to requests from different mobile operating systems, and the management interface in the same way. Being stateless helps ensure the efficiency and scalability of the server. In addition, the REST programming style simplifies the implementation process with “code on demand”.

The server response for each MCKit request is user-specific, so it is necessary for the server to authenticate the user. Authentication is based on HTTPS to protect user privacy.

#### Keepalive

Most of the data communication is initiated by the mobile client, such as to fetch or refresh static conference information, vote for conference talks, send instant messages to other users and so on. In some scenarios, however, the server wants to “push” some data to the mobile client. These include, for example, announcements made by the organizers and messages from friends or chat forums.

There are “push” services available on different mobile platforms, such as the Google Cloud Messaging for Android [5] and the Apple Push Notification Service for iOS [6]. Adopting these mechanisms, however, complicates the server implementation. For this reason, we developed our own “push” service. The mobile client sends keepalive requests periodically to fetch any available message or notification.

The period between sending keepalive request should be configured properly to balance between “push” delay and the energy consumption and loading at the mobile device.

In our implementation, we set a small period (5 seconds) when MCKit is running in the foreground, but a large one (30 seconds) when it is running in the background.

**Offline communication**

For non-server accesses, we rely on “offline” communication. Good examples are business card exchange and offline map geo-location. The basic mechanism we use for offline communication is by scanning QR code [7]. The functions for generating and scanning QR code are embedded into MCKit.

QR code has become very popular, especially in mobile applications. Compared to conventional bar code, QR code has higher information capacity. It can handle all types of data and up to 7,089 characters can be encoded in one QR code symbol. As we pack more information into a QR code, however, it is more difficult (requiring a more steady hand, or taking more time) to scan and decode the QR code. In our implementation, we try to filter out unnecessary information and pack only the essential information into the QR code to facilitate its readability, hence ensure good user experience.

### 2.3 Offline map

An offline map is of great value when WiFi is not available and 3G/LTE is too expensive to use. A natural approach is to find a high resolution campus map image, and use the image directly for display, drag, zoom in and zoom out. The problem is that this can easily overload the RAM of a mobile device.

Our solution is to use multiple images (layers) of the same map, each with a different resolution. In addition, we cut each image into small tiles of equal size. As a user zooms in and out the map, we use the layer of suitable resolution and load the tiles to fill the mobile screen on-demand. The idea is to use higher resolution only when users want to view more details, and keep the RAM usage at a reasonable level. Figure 3 shows the resolution and corresponding number of tiles for each layer in our implementation.

Originally, labels (e.g. building names) are embedded in a map image. In order to fit the text size of these labels to the screen and support the functions of searching and marking different buildings, we extracted out the labels from the original map image. Each time, after the map is drawn on the screen, the labels (with fixed font size) are put on top of the map according to their relative positions.

An important function that comes with a map is geo-location. With online maps, geo-location is often supported

![Figure 3: Layered and Tile-based Offline Map](image-url)
either using WiFi-based positioning [8], or using a satellite navigation system (GPS) [9]. For offline maps, however, providing geo-location is an interesting challenge. Again, we resort to using QR code.

We picked some important buildings and landmarks, and created QR codes encoding their information. Then we printed those QR codes on small sign cards and posted them at the corresponding locations. As illustrated in the right part of Figure 4, a user can discover her location by scanning the QR code on the sign card, and MCKit will automatically indicate the location of the corresponding building (or landmark) on the map.

3. EXPERIENCES

We knew it would not be easy to get users to pick up a new mobile app and learn to use it just for the purpose of a 3-day conference. Our strategies were to (a) do promotion, (b) give incentives, and (c) make it as painless as possible. For promotion, we made an announcement to all those who registered for the conference (including workshops) just a few days before the conference. This was quite successful, as we observed quite a lot of installations immediately after the announcement. For incentives, we tried to persuade a local mobile service operator to consider giving each attendee a SIM card valid for the week of the conference, in exchange for some publicity value. In Hong Kong, the mobile service market is very competitive, with 4-5 operators. The week-long SIM cards do not cost that much; so we were hopeful. Unfortunately, no operator took our offer. In retrospect, we should have planned some other incentives. For making MCKit easy to use, we made the following preparation before the conference.

Create accounts and QR codes

To simplify the life of users, we prepared MCKit accounts for all conference attendees in advance using their registration information. Besides, we generated the QR code for each attendee containing his/her account information and attached it to the back of his/her badge. This allows the attendee to login to MCKit quickly by simply scanning the QR code on the badge. In addition, we also created and printed a QR code on the front of each badge, for the purpose of exchanging business cards.

Create initial contacts

MCKit allows attendees to quickly build up a social network to share content and communicate with each other. We define the user relationship of MCKit as “follow”, which is similar to that of Twitter. If user A “follows” user B, A can view and browse all content posted by B and A will also appear in the “follower” list of B.

To help initialize the social network, we extracted all co-author pairs from papers accepted by the main conference and workshops of Sigcomm 2013. Among the accounts we had prepared, we assigned users of each co-author pair to “follow” each other so that users will see an initial social network when they use MCKit for the first time.

Create an account for organizer

To provide more help and support for conference attendees, we set up a special account “Sigcomm 2013” in the social network of MCKit. We set the special account to “follow” all attendees and vice versa. In this way, “Sigcomm 2013” can receive queries from any attendee and make announcements to all of them. The left screen shot in Figure 4 shows the announcements made by “Sigcomm 2013” for special arrangement of the conference due to typhoon in the social network of MCKit.

Figure 4: Announcements and Offline Geo-location

3.1 Installation statistics

Android versus iOS

We captured the installation report of both the Android version and the iOS version from Google Play Market and Apple App Store. Figure 5 presents the daily installation numbers and total installation numbers on Android and iOS.

There were a total of 305 valid installations and around two thirds of them were on iOS devices, the rest were on Android devices. This gives us an indication of the choice of mobile platforms (Android vs iOS) in the Sigcomm community. There was a peak value of daily installation for both Android and iOS on Aug 10, which was due to the promotion email sent to conference attendees on that day.

Country distribution

We summarized the installation distribution based on which country an installation is from. Figure 6 shows the top five countries ordering by the installation numbers. Besides, it also shows the distribution of Android and iOS for these...
countries. It is interesting to note some differences, reflecting different market shares of Android and iOS in different countries.

Figure 6: Installation # in Countries

Android brands

Figure 7 shows the brands that appeared among the Android installations. There are a total of 104 installations of MCKit on Android devices and half of them are from Samsung, while both Google and HTC take around 16% of the total installations. Although this is a small sample, it gives an indication of the possible market shares of Android brands in Sigcomm community.

Figure 7: Installation # of Android Brands

3.2 User statistics

Sigcomm 2013 had over 770 attendees. We created accounts for most of them (except those who registered after the day we created accounts). Out of all the users with accounts, 327 distinct users have logged in and used MCKit at least once. Table 1 summarized the number of active users seen each day. 320 out of the total 327 active users performed “join the conference”, which means they browsed content related to the conference.

After Sigcomm 2013, we analyzed the user activities based on the requests from MCKit recorded on the Web Server. Table 2 shows the total number of each request (top 10) during the whole conference. As shown in Table 2, users showed great interests on profiles of conference attendees and conference relevant information. However, they did not participate much in providing feedback for talks and communicating with each other. As for talk feedback, we only received 33 votes for 18 talks from 12 users and 4 comments for 2 talks from 2 users, while there were only a few users sharing content and sending messages on the social platform.

4. DISCUSSION

In this section, we discuss what we learned from this experience, not only making observations based on the data we collected, but also reflecting on what we wished we had done, or what could have been done differently.

Mobile platforms

From the installation statistics, we feel it was the right decision to implement MCKit on both Android and iOS. For our case, we implemented our design on Android first, and then ported it to iOS. The porting part was done in a hurry because we did not find someone familiar enough with iOS until late in the development cycle. For this reason, some of the functions appeared awkward in the iOS version. In retrospect, we need more time to consider how to design each function on iOS as well based on the constraints on iOS, and possibly revisit the original design if necessary.

An alternative approach is to develop a browser-based client. The advantage would be a more uniform user-interface and less implementation work across mobile platforms. But some functions, especially device dependent functions, may be harder to implement and being browser-based does not allow offline access from users. Whether to develop the client side as an app or as browser-based always has many trade-offs.

Integration with web publishing

Publishing conference information to MCKit, in our case, was not synchronized with publication to the conference web page. This means double amount of work, and risk of having different versions of the conference information. It is desirable and important to integrate the mechanisms so that conference information can be published and updated to both places at the same time. This can be done, for example, by having a common content management system that push content to both places.

Social functions

During our design phase, we considered integrating other social networks into our app, such as Facebook, Twitter, Sina Weibo and so on. We finally decided to build our own social network for the academic community for a variety of
reasons. Our attendees may not be connected to the same external social networks; besides, it appeared simpler for us to implement our own.

The experience was that the social functions did not get much use, even though we had initialized it with the co-author network. Perhaps this should not be too surprising, since even with the popular external social networks, the operator needs to constantly prompt new users to get them to start participating. We will reconsider integrating popular external social networks into MCKit.

On the other hand, according to usage statistics, users appear to be quite interested in browsing profiles of other attendees. In this aspect, we could have done a better job in showing more information (optionally).

Security and privacy
In our current design, all users are allowed to browse and view all available conference content, even those who are not conference attendees. For Sigcomm 2013, the information published to MCKit is manually derived from the conference website, so there is no privacy issue. But for a full-function version, we need to allow the conference organizers to control what content is available to conference attendees, which can be different than the completely public content. Besides the information published to the web site, the conference app contains a lot more information dynamically generated by users, such as talk votes and comments, e-tickets reservation, which are more sensitive. Such dynamic information should have access control that may restrict access from only attendees. Since MCKit already uses HTTPS to authenticate users, we only need to add the access control part.

In addition, we have included some controls on the visibility of user profile information. That is, user A can view complete profile information of user B only if B has “followed” A. Otherwise, part of the information, such as email and phone number, would not be visible. For future versions, we will add ways for users to configure such access control policies.

On-site registration
In our pilot use, we prepared user accounts, QR codes for business card, QR codes for quick login for the attendees who registered the conference online at least two days before the conference opening. For those who did on-site registration, we set a help desk to create user account, generate and print QR codes for them. Everything was processed manually and took a long time, which made the help desk very crowded. It became even worse when the attendees were catching a talk. A more efficient way to handle the on-site registration is necessary. We can do this by further automating and optimizing the configuration steps in MCKit for registering a new attendee.

Multiple conference support
From the very beginning, our goal is always to design a conference app as an extensible platform, used to support different conferences and events. For this reason, we do not organize everything based on a single conference. Instead, MCKit handles conferences and attendees separately.

Implementation-wise, conferences are managed by package, i.e. users can browse available conferences and download conference content from our web server. The conference information and feedback are viewed and processed inside the app package. The built-in tools of MCKit, such as calendar, favorite talks and maps, are designed to support all conferences.

Attendees (users) are handled as a first class entity of MCKit, which is independent from Conferences. A user can register and login to MCKit, browse and view available conferences and view conference content. MCKit identifies user by email, which makes it possible to connect a user with a conference if the email is also used by the user to register that conference. It is possible that different emails are used by a user to register for different conferences. To handle such cases, secondary or additional emails can be introduced to merge the registered conferences for the user; this function will be added in later versions of MCKit.

5. CONCLUDING REMARKS
MCKit was a student project, carried out in the span of a few months. Motivated by the needs of a particular conference, we tried to consider the general requirements for supporting a conference with mobile devices in our design. The implementation was rushed, short-handed and without adequate testing, resulting relatively low usage level. Nonetheless, it is valuable to document our design thinking and experiences gained in this effort, which is the purpose of this article. Some conference organizers have contacted us about using MCKit for their conferences. We are working on a way to sustain this project. If you are interested in using MCKit, please contact us.

Acknowledgement
For the implementation of the Android version, we acknowledge the help from Liang Lin and Si Yaxiong, who are from Nanjing University of Post and Telecommunications. For the iOS version, we acknowledge the work from Chen Dan, who is from Huazhong University of Science and Technology. We want to also acknowledge and thank the continuous help and support from fellow student Tom Z. J. Fu, who contributed ideas during the design, prototyping, implementation, as well as the final trial. Finally, we also thank Samuel, KY Leung from the Sigcomm 2013 organization team, and Prof Lawrence Yeung who gave us good suggestions.

6. REFERENCES