Presented at the DLSU Research Congress 2014
De La Salle University, Manila, Philippines
March 6-8, 2014

Unifying heterogeneous mobile messaging protocols to provide a person-centric thread of conversation

John Jefferson Chua, Marwin Terence Lao, Mohnish Singh, and Danny Cheng
College of Computer Studies
De La Salle University
jj.chua92@gmail.com, lao.marwin@gmail.com, mohnishsingh91@gmail.com, danny.cheng@delasalle.ph

Abstract: In today's scenario wherein people maintain various forms messaging services, each with its own list of contacts, no system or application exist that allows a person to communicate with another person as two people conversing without having to worry about which messaging service they are using (person-centric communication). Our system addresses this by organizing messages into a single thread of conversation regardless of messaging service used to send the message allowing for a more person-centric communication paradigm. Current protocols that are incorporated into the system include email via Gmail, XMPP thru Hangouts, and SMS thru the mobile device itself. The system is self-contained within the device and does not require server integration making it more portable. As part of the unification process, a contact disambiguation workflow is also implemented in order to support the collection of contact information of person so as the system would be able to merge together messages of the same person coming from different messaging protocols. The disambiguation process merges the different contact details of a person to serve as an index by the system to unify different messages from different protocols into a single thread. As of writing, the system is implemented using the Android platform and has already been tested and evaluated by various user demographic ranging from tech-savvy young adults to non-technical mature users and the results show that the concept was well accepted across this wide demographic regardless of the capability of the mobile device used as well. In the future, we plan to increase the number of protocols and platforms supported as well as the ability to support group-centric conversations within the system.

Key Words: unified messaging, contact disambiguation, person-centric mobile communication

1. INTRODUCTION

The number of communication technologies has increased dramatically following the advent of modern information technology, requiring users to

learn how to use ever more devices and systems and to choose the appropriate solution for every communication need. Despite these individual systems becoming useful for certain situations, these also turn out to cause inefficiency simply because there are too many kinds of these to manage (Boettner, P 2008). Ideally to solve both the complexity and inefficiency, a solution is to provide a single system, with the minimum required devices, that accommodate for all the communication systems being maintained, as much as possible including legacy ones (Lei, H. 2004).

Unified communications presents a single interface, be it hardware or software, to facilitate a multitude of communications technologies/channels available to the users. Its purpose is to gather as many channels as possible into one system, which is then deployed to as many devices as possible, in order to let the user choose the device most appropriate for any situation.

The researchers believe that there exists a way to combine unified messaging and contacts management into a single application that reinforces the concept of talking to a person regardless of the medium used. Such an application should be able to associate messages of a contact, from e-mail, IM, and SMS, into a single entry, while be able to manage contacts to ensure that the messages are organized respectively towards the perspective of people. Individually, such features are a rarity in smartphones and a complete absence when both are considered. Thus there is a need to introduce this method of "people-centric" messaging

2. METHODOLOGY

The application is a "person-centric" messaging solution, which organizes messages by the contact entry. In addition, the application attempts to fix contact entries in a semi-automated approach by employing user feedback. That is, the application can detect discrepancies in the contacts database and point to the user which entries or details in the database needs attention.

The system aims to integrate messages from e-mail, IM, and SMS; organize contacts through a people-centric view, and present a user-assisted semi-automated workflow. Specifically, it organizes messages and conversations by contact. These conversations may display IM, e-mail, and SMS messages relating to a single contact as a single conversation thread. Each of these messages should then be cached by the system. Finally, the system should be able to detect, prompt, and help address ambiguous contact details.

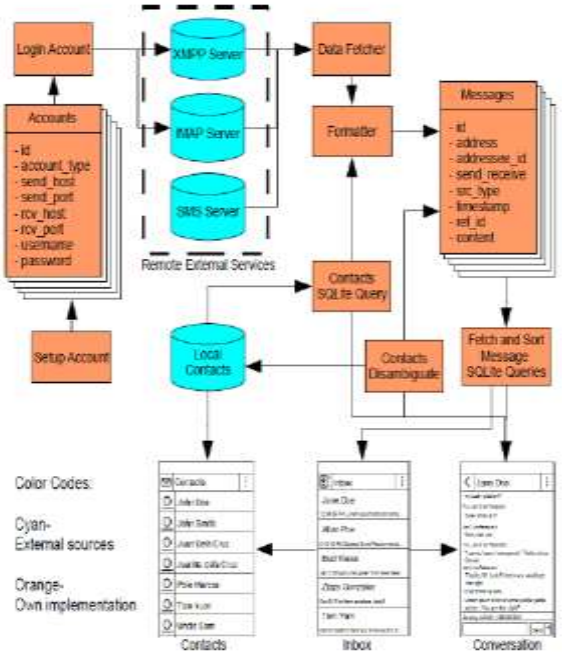

Fig. 1. System Architecture

Getting the contacts as well as sending and receiving SMS messages will require the use of the development platform's API's for contacts and messaging. Implementing support for Gmail will be to access its mail server through the Internet Message Access Protocol (IMAP) and Simple Mail Transfer Protocol (SMTP).

For this, some external libraries exist for supporting the said protocols such as JavaMail or Apache Commons Net. Lastly, for the Google Talk support, the application will make use of the Smack XMPP API, which is popular with many application

developers primarily for its familiarity. It must be noted, however, that other implementations of Smack do exist, particularly a Smack, which is essentially a version of Smack optimized for the Android platform.

Android applications, such as this project, avail of access to a database for its use through a provided SQLite API bundled with the SDK. The project, being primarily a messaging application, employs this API to store and display messages as well as to store and access accounts data. Meanwhile, access to the contacts database will be provided by another API designed for that purpose. To sum up, messages and accounts will be independently stored and accessed within the application; while the contacts database will be that provided by the SDK, which equates to the contacts stored on the device which can be used by

The accounts module consists of the accounts view and accounts controls, which is the interface to the database. In the system architecture, both "Setup Account" and "Login Account" involve accessing the interface. The latter is responsible for fetching the account details saved in the database for login into separate utilities for sending and receiving messages. More details about utilities will be discussed in another section. The former, meanwhile, has a corresponding view which the user may use to add new accounts for sending and receiving messages from it.

When the application starts, the data fetcher is activated. The data fetcher is a data structure of independent components which enable the system to send and receive messages from different services. These components are called the utilities. Utilities are defined in this context as "sub-programs" which allow individual accounts to send and receive messages. These contain the account instance itself, an identifier to differentiate from other services, and abstract methods for connecting, sending, receiving, and disconnecting. Additionally, implementing a utility requires an interface on which it could callback to the service for further processing, especially whenever a message is received or sent by the utility.

The inbox view is basically a summary of all conversations transacted through the application. Where in other implementations, conversations are sorted by date and contact detail, the system agglomerates messages belonging to the contact entity, which may have one or more details. The resulting list displays the list of conversations by date and contact, where the latest message of the contact corresponds to the last message sent or received using one of among the details registered to that contact.

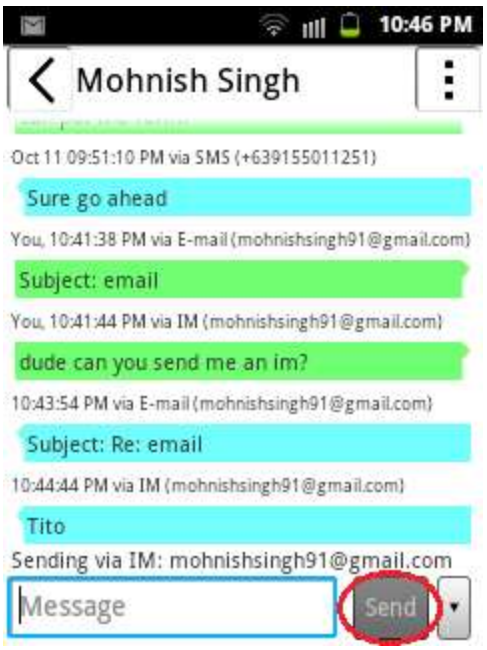

Fig. 2. Conversation Thread Display

The conversation view displays the thread of the selected contact. This view is accessible from the inbox view, when an existing conversation is selected. It is also accessible in the details view as soon as a detail is selected, after when a contact is selected in the contacts view. The latter path is the navigation step to send a new message from a contact

not listed in the inbox view. As such, the contacts view doubles as the process to send a new message, thereby eliminating the need to implement an action to directly create a new message, since a message is intended to contain a recipient. In other words, the application reverses the process of sending the message by making the user select the contact first before composing the message. Further description on navigating through sending a new message outside of the inbox view will be detailed in the next section.

Contacts management encompasses the contacts view as well as the contacts disambiguation function. The application further differentiates itself by introducing a semi-automated disambiguation process. Implementing a contacts management subsystem is justifiable because the system relies on the accuracy of contacts organization to display the conversation view.

The application's contacts view displays contacts from the platform and does not cache any information into the local database. As such, all modifications made from the application reflect within and outside of the application.

When a contact is selected from the list, the user is taken to the details view, which displays the contact details. The application presents a unique feature in which contacts disambiguation is integrated within its workflow. Specifically, the function is called in the conversation view, right after displaying the messages. The system checks whether the contact is registered in the device's database. Beforehand, conversations with a detail not yet registered appear with that detail in the header. The same holds true when that conversation is selected and the user is taken into the conversation view. The system is able to recognize that the detail is not registered, and a non-obtrusive notification appears on top of the thread, suggesting the user to associate the detail to a contact to help organize the conversation. The user may then choose to ignore the suggestion or act upon it.

3. RESULTS AND DISCUSSION

The mobile application was tested in Android mobile phones with platforms 2.0 and up. The group conducted both internal testing and external testing to test the mobile application. Internal testing involves testing the application within the group's supervision. On the other hand, external testing is done by testing the application to users online and providing them an application website (www.beavermessenger.tk). The website contains the testing process of the mobile application. Prior to testing, the phone must be setup in the following conditions:

- The user needs to enable the installation of non-market Android applications in his/her Android phone.
- The user should add his contacts to his Google Talk/Hangouts buddy list in order for him to properly send and receive instant messages in the mobile application. Unknown contacts may be identified within the application through a non-obtrusive prompt.

To perform the benchmarking tests, the group has acquired five devices with varying specifications. For all other tests, a different device which meets the minimum specifications of the application is used.

After the mobile application is tested within the group's supervision, the group has gathered a total of 32 users to conduct the external testing of the mobile application. The external testing revolves around the users and the mobile application website. The website consists of the following pages: signing-up, installing the application, testing page, and getting user evaluation and feedback. It contains the procedures on how to download, install, test, and evaluate the mobile application.

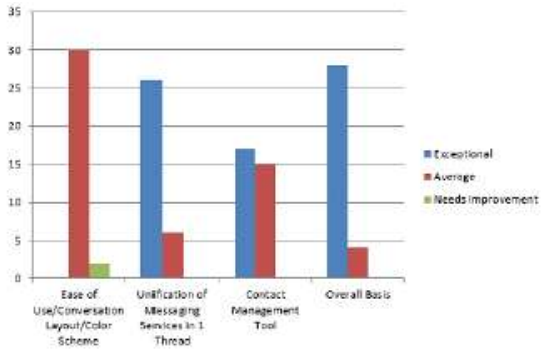

Fig. 3. User Survey Results

Overall, the users find the mobile application to be novel and generally useful; the former of which in the sense that there is no existing application that can unify different messaging platforms into one conversation thread, and the latter that they do not need to switch applications for each messaging platform. Some users, however, suggested improving on the mobile application's user interface in order to enhance user experience and productivity. What really makes them want to use the application is also the disambiguation feature, a tool they have not really come across and happens to be inside the application.

4. CONCLUSIONS

Overall, the thesis proves that developing a people-centric messaging application is not only achievable, but that it also generates interest among smartphone users who would like a more convenient messaging solution that not only facilitates messaging from different services, but also presents these messages in a more logical way, that is by contact.

Overall, the thesis proves that developing a people-centric messaging application is not only achievable, but that it also generates interest among smartphone users who would like a more convenient messaging solution that not only facilitates messaging from different services, but also presents these messages in a more logical way, that is by contact.

- Messages are sent and received through SMS, IM, email, given that the user has enabled these services.
- Conversations are organized by contact instead of its individual details.
- Unknown contacts may be identified within the application through a non-obtrusive prompt.

To achieve these, however, other problems pertaining to its development had to be resolved, such as designing a singular message format for all services and formulating an architecture to manage accounts and handle sending and receiving of messages. Eventually, the result is an application that behaves like a unified messaging solution, wherein messages from different messaging systems are gathered into a single interface.

Internal testing certifies that all the major features of the application have been accounted for and are suitable for further testing, specifically in benchmarking and profiling. It is also revealed that the application is able to run on entry-level smartphones and performs even better on devices with higher specifications, satisfying the implicit goal of the application to be compatible with as many devices on the market as possible.

Further, while comparison, analysis, and profiling reveals that the application performs its tasks more slowly than other related messaging applications, the cause is justified to be the added number of processes involved in achieving the differentiated capabilities of the messaging solution. This, however, should not significantly impact performance on real-world usage.

Feedback from limited public testing suggests that the respondents are positive towards the people-centered messaging solution; thereby allowing the group to deduce that there is an importance for conversations to stay coherent, regardless of the medium the parties of it use. Providing an interface that combines messages belonging to a contact from different details becomes the logical solution to this need.

With regards to contact management, reception towards the feature has been lukewarm. While the respondents agree that the feature complements the overall experience, some prefer to have this function automated. Because of this, the group believes that while contacts disambiguation complements a people-centric messaging solution, it should rather be

considered as a secondary feature.

Overall, the thesis provides evidence that developing a people-centric messaging application is not only achievable, but that it also generates interest among smartphone users who would like a more convenient messaging solution that not only facilitates messaging from different services, but also presents these messages in a more logical way, that is by contact.

5. REFERENCES

Bhawani, A. 2012. How to set SMS storage limit & auto delete old SMS on Android. Retrieved January 11, 2013 from Android Advices http://androidadvices.com/set-sms-storage-limit-auto-delete-sms-android/

Boettner, P., Gupta, M., Wu, Y., & Andrew, A. 2008. Towards policy driven self-configuration of user-centric communication. In Proceedings of the ACM Southeast Conference '09 (Clemson, SC, USA, 2009). ACM.

Cardwell, L. 2008. Unified communications: Cutting through the hype. Retrieved October 12, 2012 from http://viewer.media.bitpipe.com/1206484657_637/1206511483_362/SearchUC-v5.pdf

Grundy, J. n.d. Preventing an iPhone from deleting texts. Retrieved January 11, 2013 from http://smallbusiness.chron.com/preventing-iphone-deleting-texts-48521.html

Lei, H. and Ranganathan, A. 2004. Context-aware unified communications. In Proceedings of the IEEE International Conference on Mobile Data Management. (Berkeley, CA, USA). IEEE.

Rosenberg, A. 2008. Don't miss the "unified messaging" boat in UC. Retrieved October 12, 2012 from http://www.ucstrategies.com/detail.aspx?id=2492&terms=mobile+collaboration

Seligstein, J. 2011. See the messages that matter. Retrieved November 21, 2012 from The Facebook Blog https://www.facebook.com/blog/blog.php?post=452288242130

SEVEN Networks. 2012. Ping FAQ's. Retrieved October 15, 2012 from http://www.seven.com/FAQ.php

Telecomindiaonline.com. 2011. Synchronica introduces pre-RCS unified messaging with Mobile Gateway 6. Retrieved October 14, 2012 from http://www.telecomindiaonline.com/synchronica-introduces-pre-rcs-unified-messaging-with-mobile-gateway-6.html

Yeung, K. 2012. AOL unveils Alto, its new cloud-based application designed to bring calm to your email inbox. Retrieved October 24, 2012 from The Next Web http://thenextweb.com/apps/2012/10/18/aol-alto-email-application