University of Waterloo
Faculty of Engineering
Department of Electrical and Computer Engineering

Real-Time Instant Messaging System

Project Specifications and Risk Assessment

Group #2015.010
Professor Werner Dietl (Consultant)
Qi Liu (20358515)
Asif Arman (20349964)
SangHoon Lee (20357600)
Danny Yan (20387735)
June 2, 2014

Table of Contents

1. High-Level Project Description

1.1 Motivation

Instant messaging systems of today are instant in name only.

Messages are sent when the send button is pressed by the sender, and no feedback to the receiver is presented during the waiting period except a wholly inadequate “user is typing” notification. This disruptive downtime between message and reply in modern instant messaging systems is becoming more apparent day by day.

In a 2007 study conducted by scientists at the Dortmund Institute for German Language and Literature, it was discovered that the average instant messaging system user discards 20% of their composed messages [1]. Evidently, messages in instant messaging systems of today are at constant risk of becoming obsolete before the send button is pressed.

Much of this awkward inefficiency could be avoided if there exists an instant messaging system that is true to its name: a system that provides instant, real-time communication between participants, without being bound by the archaic timing constraint dictated by the send button.

More recently, leaks provided by Edward Snowden on NSA’s overreach in its information collection practices has sparked a global discussion regarding government surveillance, and a surge in public demand for truly private and secure communication systems.

Some believe that the widely popular instant messaging network, Skype, is the solution to this problem as it makes use of a peer-to-peer architecture. However, this is no longer true of the network since 2012, when Microsoft replaced all of the decentralized supernodes in the Skype network (peers that had enough resources to act as relays for traffic between other peers) with servers under their control [2]. A quick look at the current Skype privacy policy verifies that Microsoft indeed reserves the right to collect “Content of instant messaging communications, Voice messages, and video messages” [3].

As of today, the demand for truly private and secure instant messaging systems has yet to be met.

1.2 Problem Statement

We aim to design an instant messaging system that mimics the free-flowing experience of natural, in-person conversations, and at the same time protects the privacy of its users as an utmost priority.

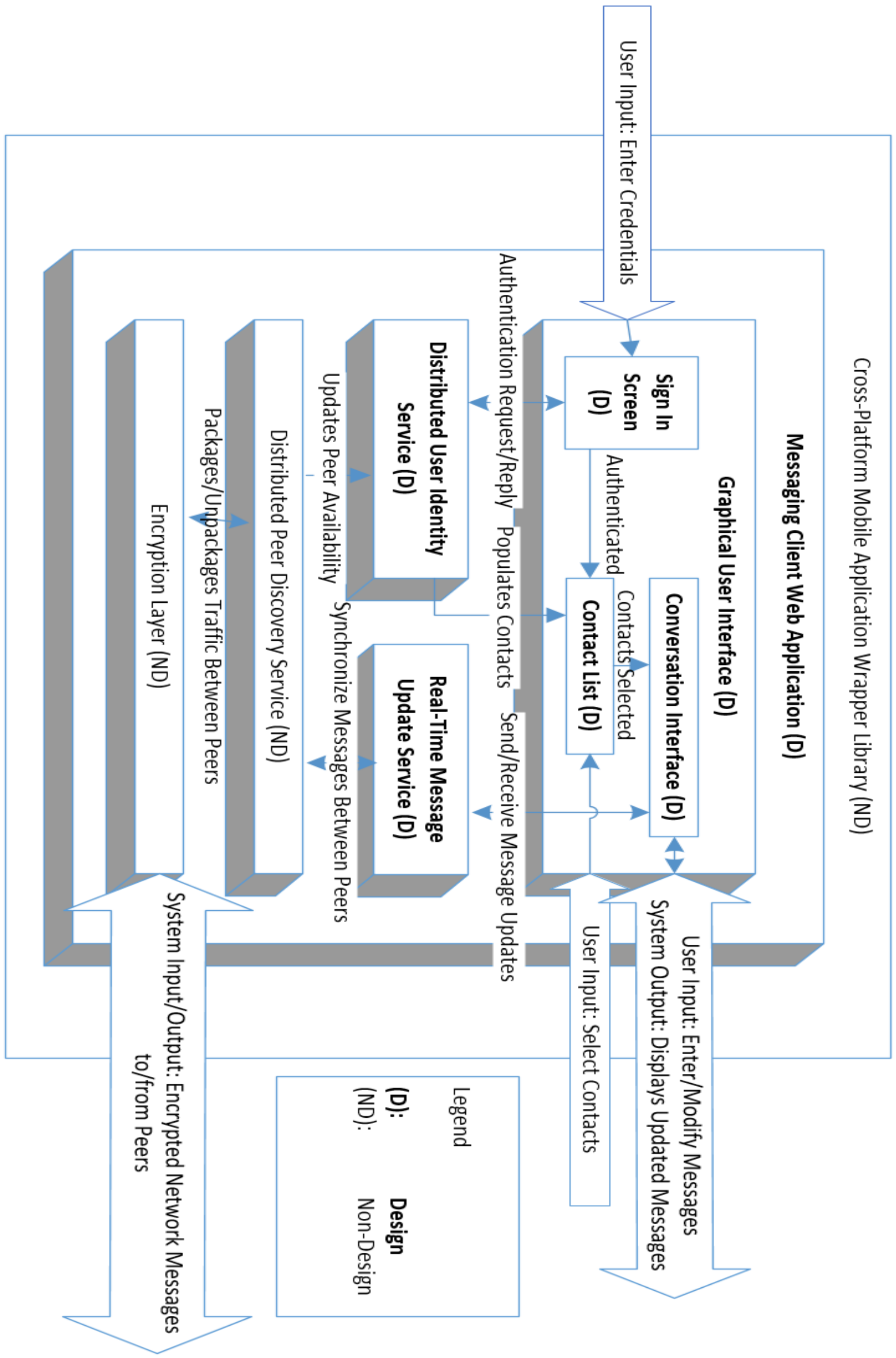
Our system will meet the former objective by updating messages on the receiver side as they are being written in real time, removing the disruptive delay between message and reply necessitated by the send button.

Ideally, we will meet the latter objective by implementing our system as a truly decentralized peer-to-peer architecture with no centrally controlled servers of any kind. Messages in this system will thus travel directly from the sender to the recipient, through a completely encrypted channel, giving no opportunity for any third-party to access message contents.

Alternatively, if a pure peer-to-peer system proves to be impractical for any reason, we will attempt to design a hybrid system that makes use of central servers for only peer discovery and authentication.

1.3 Block Diagram

Figure 1: Block diagram of project components and inputs/outputs


Cross-Platform Mobile Application Wrapper Library (ND)
Messaging Client Web Application (D)
Graphical User Interface (D)
Sign In Screen (D)
Contact List (D)
Conversation Interface (D)
Distributed User Identity Service (D)
Real-Time Message Update Service (D)
Distributed Peer Discovery Service (ND)
Encryption Layer (ND)
User Input: Enter Credentials
Authentication Request/Reply
Authenticated
Contacts Selected
Populates Contacts
Send/Receive Message Updates
Updates Peer Availability
Synchronize Messages Between Peers
Packages/Unpackages Traffic Between Peers
User Input: Select Contacts
User Input: Enter/Modify Messages
System Output: Displays Updated Messages
System Input/Output: Encrypted Network Messages to/from Peers
Legend
(D): Design
(ND): Non-Design

2. Project Specifications

2.1 Functional Specifications

Table 1: List of functional specifications

Functional Specification #	Essential / Non-Essential	Description
1	Essential	Messages should be displayed in real time: the receiver should see the message that the sender is typing as it is being typed. The average update time should be less than 2 seconds assuming both sender and receiver are under an ideal and stable network.
2	Essential	Every message sent over the network will be encrypted. Messages should use at least 128 bit encryption scheme.
3	Essential	Users are not limited to one conversation. They will be able to participate in multiple conversations concurrently.
4	Essential	Users should have access to the same contact list on every device that they sign in to.
5	Essential	Each member of the user’s contact list will display an availability indicator showing they are available to communicate.
6	Essential	Users must successfully authenticate themselves through a sign-in screen to obtain access to their contact list as well as send and receive messages.
7	Essential	Communication between users is routed through a private P2P channel using Distributed Hash Tables (DHT) for peer discovery. A client-server architecture will be used for peer discovery as a fall back in case suitable DHT libraries are not found.
8	Non-Essential	The frequency of real time message updates should be user adjustable. Both sender and receiver should be able to modify how often messages are to be updated within 0.1 second level of granularity. If they wish, users should be able to turn off the real time component completely so that the application behaves like traditional messaging systems that send messages only on demand.
9	Non-Essential	Message queuing will be supported. If the receiver of the message is offline (unavailable), messages can be queued, and the receiver will receive the message when he/she next becomes available.
10	Non-Essential	Group chat should be supported. At least 4 users should be able to participate in the same real-time conversation with no noticeable performance degradations.
11	Non-Essential	Group conversation messages should remain synchronized. Every user should have the same consistent view of the messages in terms of ordering and content.

12	Non-Essential	The user can choose to save conversation history. When a user reconnects to their account (on a different device) they will be able to view their previous conversations between the other users and groups.
13	Non-Essential	A user should be able to use the application on more than 1 device concurrently. State between each instance of the application should be synchronized to within 10 seconds of delay under ideal network conditions.
14	Non-Essential	Real time widgets will be supported that allow users to perform more than just sending messages. Examples may include a drawing widget, voice communication widget, video widget, etc.
15	Non-Essential	Users should be able to search through their contact list and messages. Searching will be done in real time and filter through new messages as they arrive.

2.2 Non-Functional Specifications

Table 2: List of non-functional specifications

Non-functional specification	Essential / Non-Essential	Description
Efficiency	Essential	Efficiency is one of the most important non-functional specifications that our system requires. The system needs to be as efficient as possible in both local and network resource usage. In terms of network usage, on average it should not surpass 1MB per 10000 characters sent/received. In terms of local resource usage, it should never use more than 50MB of memory under normal operation as a mobile or web application.
Security	Essential	Our system needs to safeguard the privacy of the user as an utmost priority. All messaging traffic needs to be encrypted and resist tampering and eavesdropping. Only the intended recipient should be able to decrypt the messages efficiently with his private key, and be able to validate that it came from the expected sender.
Dependability	Essential	Our system needs to be very dependable. Instant messaging systems are required to be very reliable and robust. It should perform within our design limits without failure over time and should be able to respond adequately to unanticipated runtime conditions. Therefore, all messages should be delivered in the correct order and without any errors under ideal network conditions. Also, all failures in delivery due to poor networking conditions needs to be notified to the user.
Usability	Essential	Our system should be highly usable, with an intuitive and responsive real-time interface. Our system needs to have minimal input latency

		under all supported platforms. There should not be more than 0.5 second of lag between user input and interface response (not counting network latency) as long as the system has sufficient local resources for normal operation.
Maintainability	Non-Essential	Our system needs to be easily maintainable so we can update the client and add features without needing to perform architecture overhauls. Our component topology needs to be laid out in a layered architecture with high degrees of separation, with automated tests covering all important features to detect regression issues when changes are made.
Portability	Non-Essential	As our system may need to run on multiple platforms, portability becomes important aspect of our system. It should be able to run on android, iOS mobile platforms in addition to the original web platform, retaining all of functional and non-functional specifications. Once we have completed the core specifications for the web application, this can be accomplished using open source libraries that wrap the web application with platform specific interfaces.

3. Risk Assessment

Table 3: List of possible risks, their probabilities and potential impact

	Nature of Situation	Potential impact	Probability of the Situation
Quality, stability, interoperability and maturity of open-source libraries needed for certain project components.	Certain functionality for the project are dependent on open-source libraries for operation (for instance, the Encryption layer and the DHT library used for peer discovery). Some of the open-source libraries we plan to use are under heavy development and may or may not contain incomplete features or show-stopping bugs. In addition, compatibility is another concern we have regarding use of open-source libraries, as some of them may conflict with aspects our code or other libraries we're using.	This situation can cause a devastating impact to our project success. If any of the depended open source libraries creates conflict during the development of the project, it can greatly delay our project completion time and decrease the chances of the project successfulness.	The probability of this situation is considered to be high. Due to the usage of open-source libraries, we have no guaranteed support in the case where a library malfunctions or does not perform as we expect. Finding possible alternatives to each library we plan to use can greatly reduce the impact of this situation occurring. Having a list of alternative libraries that can substitute

			dysfunctional or incompatible libraries means we can switch between them to avoid failure or halting our project. We also have the option of contributing to the open-source project that is causing issues for us or fork the project for our own use. However this option requires a great amount of domain specific technical expertise, and a huge maintenance overhead. Thus it is best avoided if alternative libraries exist. No matter what we do, it is still possible for the worst case to occur. Therefore, to mediate the damage caused by the situation, we may end up needing to reduce the scope or remove certain features from the project, or fall back to alternative implementations.
Failure to meet time constraints for certain deliverables.	Time management is an important factor in determining the completion of project before the deadline because we need to invest	Failure to complete the project under the deadline can result to academic penalties. The produced delays	The probability of this situation is medium because group members have a flexible schedule

	enough time to complete each component of the project. Our project treads on many uncharted territories in the field of instant messaging and requires knowledge of several advanced networking topics, so time required for completing each component can be difficult to estimate.	by poor time management can also increase or create more potential delays as the project progresses. Time to market will also be effected by delays in the project. If we take too long, we risk losing our competitive edge if/when another developer releases a similar instant messaging system.	throughout the duration of project to reduce the chances of this situation occurring. If timing does become a significant constraint, we can consider reducing the scope of the project by focusing only on our core specifications.
Team member leaving or disbandment of the team.	In every project involving two or more people, some degree of conflict will be inevitable. Members may decide to leave the team in the worst case. Also, there is a chance that certain members may need to leave the program in the second term due to failing out or other special circumstances.	In the situation where a team member were to leave the group, it will greatly affect the project and the remaining team members. Addition time required to cover for the remaining work of the missing member, and the morale of the team will be heavily affected. In the worst case, the entire team can disband and abandon the project.	The probability of this situation to occur is low. The team members have worked on previous projects involving cooperation and teamwork. To mitigate the risk of any single member leaving, we will try to distribute work so that at least two people in the group are domain experts on any single component or technology used in our project.

References

[1] T. Simonite, "New Scientist Blogs," 07 December 2007. [Online]. Available: http://www.newscientist.com/blog/technology/2007/12/instant-message-irrelevance.html. [Accessed 01 June 2014].

[2] D. Goodin, "Skype replaces P2P supernodes with Linux boxes hosted by Microsoft (updated)," 01 May 2012. [Online]. Available: http://arstechnica.com/business/2012/05/skype-replaces-p2p-supernodes-with-linux-boxes-hosted-by-microsoft/. [Accessed 01 June 2014].

[3] Microsoft, "Skype Privacy Policy," 2014. [Online]. Available: http://www.skype.com/en/legal/privacy/. [Accessed 01 June 2014].