Enriching Circuit Switched Mobile Phone Calls with Cooperative Web Applications

Måns Hommerberg
Abstract

**Enriching Circuit Switched Mobile Phone Calls with Cooperative Web Applications**

Måns Hommerberg

The thesis investigates the possibility to enrich standard mobile phone calls with cooperative web applications. Originating from the research field known as Computer Supported Cooperative Work (CSCW) this thesis report introduces and describes the implementation of several applications which can be used by the calling parties together during a phone call. Additionally, the report describes a proof-of-concept prototype for the Android platform, and discusses the performance of cooperative web application running on mobile devices in terms of network and CPU use.

The conclusions of the thesis describe a prototype application addressing and implementing the requirements as described by the theory of computer supported collaborated work. The performance of the running application showed to be satisfactory, both regarding to network demand and processor use.
Table of contents

1. Introduction ................................................................................................................. 4
   1.1 Background ............................................................................................................. 5
   1.2 Research Questions ............................................................................................... 8
   1.3 Delimitation ........................................................................................................... 8
   1.5 Related Work ......................................................................................................... 8

2. Theory ......................................................................................................................... 10
   2.1 Enriching a Mobile Telephone Call ................................................................. 10
   2.2 Computer-Supported Cooperative Work ......................................................... 13
      2.2.1 Communication in CSCW ....................................................................... 15
      2.2.2 Awareness in CSCW ............................................................................. 15
      2.2.3 Session Management ............................................................................. 16
      2.2.4 Concurrency Control .............................................................................. 16
   2.3 The Distributed Shared Memory Service ..................................................... 17

3 Designing a Proof-of-Concept Prototype Application ............................................. 19
   3.1 Design of a Caller Application ......................................................................... 19
   3.2 Cooperative Web Applications ........................................................................ 21
      3.2.1 Chat ........................................................................................................... 21
      3.2.2 Draw .......................................................................................................... 21
      3.3 Notes ................................................................................................................ 22
      3.3.4 Map ............................................................................................................ 22
      3.3.5 To-do list and grocery list ........................................................................ 23
      3.3.6 Directions ................................................................................................. 23
      3.3.7 Language Chat ........................................................................................ 23
      3.3.8 Balls .......................................................................................................... 23
      3.3.9 Photos ....................................................................................................... 24

4. Implementation .......................................................................................................... 25
   4.1 Android Application ......................................................................................... 25
      4.1.1 Activity 1 ..................................................................................................... 25
      4.1.2 Broadcast Receiver .................................................................................... 26
      4.1.3 Activity 2 ..................................................................................................... 28
      4.1.4 Manifest file ............................................................................................... 29
   4.2 Cooperative Web applications .......................................................................... 29
      4.2.1 Applications menu ..................................................................................... 30
      4.2.2 Chat ........................................................................................................... 31
      4.2.3 Draw .......................................................................................................... 32
      4.2.4 Map ............................................................................................................ 34
      4.2.5 To-do/grocery list ...................................................................................... 35
      4.2.6 Directions ................................................................................................. 36
      4.2.7 Balls .......................................................................................................... 38

5. Evaluation and Analysis ......................................................................................... 39

6. Conclusion ................................................................................................................. 45
   6.1 Prototype improvements .................................................................................... 45
   6.2 Further research ................................................................................................. 46

7 References ................................................................................................................ 47
1. Introduction

Today people use the mobile phone for a wide array of purposes. It can be used in order to transfer money, to stay updated with Youtube subscriptions or to track the jogging course. It is possible to do so many more things beyond the ordinary telephony functionality. The mobile phone has become a versatile, handheld device. Yet, during a phone call the mobile device still tends to be used just for talking and listening. What if some of the increased power of a mobile device could be incorporated into the regular phone call? Could that enrich the communication experience during the phone call?

The vision of the project is to research and investigate the possibility to add features to a regular phone call. This is accomplished by the exploratory development of prototype cooperative web applications for a mobile device. The applications are developed with a universal web based scope. However, the implementation of the bundling to a regular phone call is confined to the Android platform.

In order to introduce the topic of the project, some important concepts need to be declared. The introduction is split into three sections, which all contribute to introduce the topic of the thesis project and put it in the relevant context. Section 1.1 Background serves to explain what has led to the vision mentioned, as it tries to explain the contemporary context of the project topic and what prior facts motivate the work on mobile devices. The research questions, Section 1.2, present more specifically the problems to which this project seeks answers. Section 1.3 Delimitation, confines the scope of the project and declares what parts will not be further discussed. Section 1.4 Related Work presents what has been and what is currently being done within the topic of enriching communication using the ordinary Circuit Switched (CS) phone call.
1.1 Background

Since the launch of Apple’s mobile application market place, App Store, in July 2008 the availability and popularity of mobile applications have grown very fast.

![Figure 1: The rapid growth of Apple iPhone application availability and use since the launch of App Store in 2008. [18]](image)

Figure 1 illustrates the increase of mobile applications on the Apple App Store. The red line corresponds to number of application downloads and the blue line to the number of applications available on AppStore.
Another major mobile application market, Android Market was launched in October 2008, and has seen a similar pattern as depicted in Figure 2. In total, 10 billion applications have been downloaded from the Apple App Store and 3 billion from the Android Market. Figure 1 and Figure 2 also both indicate an exponential increase in the availability of smartphone applications for the respective platform.
Figures 1 and 2 indicate a rapidly increasing popularity for smartphones, such as Android or iPhone. Figure 3 indicates both that people with smartphones use their mobile device more time than people with ordinary mobile phones (60 minutes versus 40 minutes per day), and that smartphone users use their phone increasingly for purposes other than making voice calls.

The smartphone sales in the third quarter of 2010 increased 96 percent from the third quarter of the previous year [10]. The increasing use of smartphones has been called the Smartphone Revolution [8]. This thesis investigates the possibility to combine the functionality of smartphones with the core function of a telephone, making a phone call, by implementing a number of cooperative web applications and bundling them with the ordinary phone call. The cooperative web applications offer new ways to communicate. The applications provide functionalities to the users such as shared map, shared white board and shared notes. Referring to Figure 3, the incorporation of cooperative web applications would mean a sort of fusion between the voice call section and the other sections in the iPhone usage pie chart to the right.
1.2 Research Questions

There are many research issues related to enriching mobile telephony using cooperative web applications. This report will focus on three main research questions:

1. How could cooperative mobile web applications be implemented on a mobile device to enrich the communication of a phone call?
2. How could the cooperative web applications be bundled with the circuit switched telephone call of a mobile device?
3. What are the requirements on the terminal and on the network in order for the collaborative applications to work smoothly?

The approach to the first research question is an exploratory development of prototype cooperative web applications adapted to mobile devices. The approach to the second question is to develop a native application prototype that links the circuit switched phone call to the cooperative web applications. The response to the third question is a qualitative investigation of how well the shared web applications work, both with regards to the performance of the device and to the speed of the network connection used.

1.3 Delimitation

In order for the prototype applications to give a rich user experience a number external JavaScript libraries are used. They include some Google API (application programming interface) (eg. Google Maps, Google Translate), and some other API:s which improve the design and usability of mobile web applications (PhoneGap, iScroll, Jquery Mobile). The inclusion of these is important for the design of the project application, but their individual functionality is not discussed thoroughly in the scope of this report.

Another limitation is the choice of platform. Even though there is a huge number of smartphone platforms available today (eg. iPhone, Android, Blackberry, Windows Mobile), the prototype application is implemented only on the Android platform. The cooperative web applications, however, are practically platform independent, since they are web based. Consequently, in potential future work on the prototype, the cooperative web applications could easily be ported to other smartphone platforms.

1.5 Related Work

The idea to use the regular phone number for communication other than phone calls and ordinary text messages has been implemented before. WhatsApp is a messaging service application where users free of charge can send pictures, videos text and sound to a person in the address book who also has WhatsApp installed [11]. The users do not have to create user accounts with passwords. Instead the phone number of the mobile device is used as the personal ID. The application works for different smartphone platforms and has been a success with millions of downloads [12].

Another communication application using the phone number of the mobile device to identify the user is Viber [13]. Viber is currently only available for iPhone. With Viber a user can make voice calls to anyone who has Viber installed on their iPhone, given that the user knows the telephone number of the one to call. The data transfers over Wi-Fi or 3G, and so there is no cost for making these calls, given that the plan includes free data.
Yet another application which utilizes the regular phone call is InnoBell [14]. InnoBell is an Android application and is very similar to the prototype application discussed in this report. With InnoBell users make a regular call or start an IM-chat, and from there access what InnoBell refers to as add-ons in order to increase the communication possibilities with the other party of the phone call or IM chat. The cooperative add-ons of InnoBell include location, Youtube, transfer money and a few others. InnoBell was posted on the Android Market April 5 2011, which was right in the middle of the development of the prototype application covered in this report.

All three examples mentioned indicate how mobile applications use the ordinary telephony functionality, either the telephone number or the call itself, of the device to offer additional communication possibilities. So the development of mobile applications which in some way use circuit switched telephony is certainly an interesting topic.
2. Theory

The research questions listed in Section 1.2 contain a couple of concepts whose theoretical basis needs to be provided. Firstly, research question one mentions “enrich a mobile phone call”. In order to make this enrichment, two issues must be addressed; what does a mobile phone call actually offer today and what more could it possibly offer. Section 2.1 deals with the concepts of circuit switched mobile telephony in terms of the communication it offers to the calling parties, and plausible extensions to the communication offered.

Secondly, the research questions mention “cooperative web applications”. What are such applications? Section 2.2 addresses the concept of cooperative web applications or in broader terms computer supported cooperative work or CSCW. Within the topic of CSCW, the theoretical basis for data consistency, concurrency control and user awareness is also given.

Finally, Section 2.3 explains the theoretical functioning of the Distributed Shared Memory (DSM) Service. This is a service created by Ericsson Research which enables developers to implement their own CSCW applications. The DSM service is later used for the implementation of the prototype cooperative web applications.

2.1 Enriching a Mobile Telephone Call

A telephone call is defined as a connection over a telephone network between the calling party and the called party [15]. The telephone call allows the call parties to engage in oral communication without the need for their geographical proximity.

Given the definition of a telephone call and the increasing power and use of mobile devices mentioned in Section 1.1, it definitely seems possible to enrich the communication experience during a mobile telephone call. In order to approach the enrichment of mobile telephone calls, one needs to consider what different capacities mobile devices possess. Today’s telephones have numerous functions beyond just a microphone and a keypad. Common features include media storage and media capture with camera, high-speed internet access and several sensors such as geographical positioning, accelerometer and compass. None of these additional functions is used in a standard telephone call, so there clearly seems to be a chance to enrich the communication. Sharing media, different application functions and device sensor data with the other call party during a phone call all have the potential to enrich the communication experience.
The simple scheme in Figure 4 shows that only a fraction of the functions mobile devices possess today is used during a phone call. Consequently, one way of enriching the communication of the call would be to move the white-grey border, as indicated by the white arrow in Figure 4.

However, it does not suffice only to list functions that would be desirable to use during a telephone call. In order for an additional function use to serve any purpose, it needs to be effectively transmitted to the other call party. Just as the regular circuit switched telephone call offers reliable voice communication to the call parties, so do the additional functions need to offer reliable enriched communication. The outputs of a mobile device are the screen (visual output), speakers (sound output) and vibration (motion output). Although alternative solution might be possible, the approach chosen in this project is to link all additional inputs to the screen output.
Figure 5: The linkage of the inputs of one user to the outputs of the other user.

Figure 5 shows the desirable communication from one call party to the other party’s output; all extra input is linked to the screen output. However, the output one of the parties sees is of course not only governed by the input of the other party. It is simultaneously manipulated by the party’s own actions.
Figure 6 shows how the inputs of the call parties communicate with the outputs. The inputs of Party 1 will affect his own output and that of the other party, and the same thing goes for Party 2. So what this means is that mobile screens of the users will tend to resemble each other. This is indicated by the black circle named shared area in Figure 6.

Just as the telephone call could be compared to two people being in the same room talking to each other, would the additional functions lead to two people being in the same room. Using the additional functions however, the two people would not be talking to each other, but rather drawing together on a whiteboard, putting photos on a photo board or panning a map. So, actually, rather than having two separate and different screen outputs, the aim is to fusion these into one common screen output. This motivates the second part of the title of thesis, “cooperative web applications”, since the additional functions are cooperative in the sense that both parties are able modify the common output. Section 2.2 provides a theoretical basis for cooperative web applications, or in broader terms computer supported cooperative work.

2.2 Computer-Supported Cooperative Work

The term "computer supported cooperative work” (abbreviated CSCW) was coined in 1984 during a workshop intended to explore technology’s role in the work environment [4]. According to Carstensen & Schmidt, the theme of CSCW as a research field is “how collaborative activities and their coordination can be supported by means of computer systems” [1].

The CSCW matrix is a common way to visually conceptualize CSCW. The matrix was first presented in 1988 by Johansen [6].
Figure 7: the CSCW matrix

Figure 7 shows the CSCW matrix which indicates that CSCW can be subdivided depending on both time and space. In terms of integrating CSCW into a regular phone call, the quarter of the CSCW dealt with the most is the bottom left one; different place but same time. This cell of the CSCW matrix can be referred to as real-time distributed groupware systems. Examples of such systems include text editors, drawing systems, video conferencing, media spaces and shared whiteboards [9]. These applications represent the type of applications that are implemented in the construction of the prototype web applications for the thesis project.

According to Pichiliani and Hirata [7], CSCW applications need to meet specific requirements which are not addressed by single user applications. Firstly, CSCW must have support for communication between the users. That is, it is important for the users to know what others users want to do and their intentions. This communication link may assume shapes such as text, audio or video. Secondly, CSCW applications must provide awareness to help the users coordinate their work. In the context of CSCW, Awareness refers to an “understanding of the activity of others which provides a context for your own activity” [2]. Thirdly, Pichiliani and Hirata mention that CSCW applications must implement collaborative session management, so that users can both join an existing CSCW session or start new one. At the same time, the session management must have some security control, so that only users with the proper permission are allowed to engage in a specific CSCW session. Fourthly, CSCW applications must have some sort of concurrency control in order to manage simultaneous input from multiple users [7].

The following sections discuss with communication, awareness, session management and concurrency control for CSCW applications in more detail. Additionally, the sections explain how the prototype application addresses these four requirements for CSCW application.
development.

2.2.1 Communication in CSCW
As described in Section 2.1, the aim of this thesis is to enrich an already established communication link; the regular phone call. According to Pichiliani and Hirata [7], it is desirable for the communication between users in a CSCW session not to change the state of the actual cooperative application. Using a telephone call as the communication between users clearly fulfils this desirable feature. As seen in Figure 5, the telephone call communication goes from the microphone of one user to the speaker of another user, and thus parallel to the communication of the cooperative web applications which goes from one user’s input (not the microphone) to the screen of the other user.

2.2.2 Awareness in CSCW
According to Xia [9], in order for users of CSCW applications to be able to collaborate naturally and fluently it is important that they are aware of the status of the other user. Gutwin et al. [5] divide the concept of awareness into four different categories; informal awareness, social awareness, group-structural awareness and workspace awareness. Informal awareness gives the sense of who’s around and what they are up to. Social Awareness is the awareness a person gets about others in a social or conversational context. Group-structural awareness is the knowledge of people’s roles in a collaborative situation. Lastly, workspace awareness refers to the part of the awareness that the actual collaborative workspace should present. According to Gutwin et al. [5], maintaining knowledge about the workspace interaction by others is highly relevant. The way the issue of awareness is addressed in the scope of the prototype application development is that the communication established by the telephone call tries to satisfy the first three types of awareness. Although, here the telephone call has limitation compared to the awareness provided by face-to-face collaboration. A telephone call does not communicate gestures, facial expressions etc.

The last type of awareness, workspace awareness, is implemented selectively depending on the nature of the specific cooperative web application. A comparison between an instant messaging application and a shared map application can illustrate different implementation of workspace awareness. As for the instant messaging application, the only input to the application workspace is messages from either user. The users can write messages independently of one another, and as soon as a message is posted in the workspace it is clear from whom the messages came. Consequently, an instant messaging application provides explicit workspace awareness as defined by Gutwin et al. [5].

On the other hand, for the shared map, the situation looks different. Since a mobile screen offers relatively little screen estate, and since a map requires quite a lot of screen real estate, it could be argued that the exclusion of workspace awareness for a shared map application could be advantageous. Instead, in the case of a shared map application the communication and informal- and social awareness provided by the telephone call will compensate for the absence of workspace awareness.

A conceptual example might illustrate how absence of workspace awareness could be compensated by the telephone call: If a shared map application is put in the hands of two separated users without any additional means of communicating would probably result in a digital tug of war of map position, and the shared map would not serve very much purpose.
However, given an established voice communication running parallel, the shared map application becomes significantly more powerful. One user can explain directions, interesting landmarks etc, by moving, zooming and marking the map, while at the same time continuously talking to the other user.

### 2.2.3 Session Management

Session management plays a crucial role in CSCW. It determines how a cooperative web application session is started and how individuals join a specific session [9]. Furthermore, Xia [9] mentions two types of session management; explicit session management, and implicit session management. Explicit Session management means the users are required to take explicit actions to initiate the collaborative session, whereas in implicit session management the system puts users in a collaborative session for instance when they view the same page or edit the same document.

The approach for the prototype application proposed in this thesis is to use explicit session management. Actually as soon as a user dials the number (or clicks a contact's name) to start a telephone call, the user also provides the application with the address for the collaborative session. The way this is done is by using the telephone numbers of the call parties as the address for the CSCW session. Section 3.1 further explains how this is accomplished.

### 2.2.4 Concurrency Control

According to Greenberg and Marwood 1993 [3], concurrency control problems arise when the software, data and interface is distributed over several computers. If there is no concurrency control implemented for the CSCW application, users might perform conflicting operations. This could result in major user confusion since the shared content becomes inconsistent. Consequently, it is important for every CSCW system in which simultaneous operations can occur to implement concurrency control. Concurrency control can be subcategorized into two types; optimistic- and pessimistic concurrency control.

Pessimistic concurrency control techniques ensure consistency by only allowing operations and events in the right order [3]. One such technique is locking, where a user needs to acquire a lock before interacting with an object in a shared workspace. Since there is only one lock for each object in a system, only one user at a time can interact with an object, and consistency is thus maintained. This technique however can lead to slow system responsiveness since users must wait to acquire the lock before interacting [9].

Optimistic concurrency control means operations are allowed to be performed out of order, and when out of order operations are performed, the system must detect and repair possible inconsistencies [3]. One such method is Operational Transformation. This is the concurrency control method that is used in this thesis project.

Operational transformation is a concurrency control technique where local operations are executed immediately after generation. This gives a high local responsiveness. Remote operations may need to undergo transformation depending on concurrent operations before they are executed, so that the effect is the same on the local as on the remote site [9].
Figure 8: Operational transformation in a shared text editor

Figure 8 shows a simple example to illustrate the basic idea of the operational transformation technique. In the example in the figure there is a shared text editor and two users. The initial string is “abc”. User 1 wants to add an “x” at position 0 (before the “a”). At the same time, User 2 wants to remove the “c” (remove character at position 2). What operational transformation does is that it transforms the remote operations so the output remains the same for both users. In figure 8, the operation of User 2 transforms for User 1 from “removing the character at position 2” to “removing the character at position 3”, since User 1 added a character to the string. For User 2, conversely, the operation of User 1 does not need to be transformed, since removing the last character does not change the index of the first position in the string. As seen in figure 8, the resulting string is “xab” on both sites after the remote operations have been carried out. So, principally, the aim of operational transformation is to apply relevant transformation on specific operations so as to make the output converge to the intended and desired output.

2.3 The Distributed Shared Memory Service

The Distributed Shared Memory (DSM) Service is an API available for download from Ericsson Labs by Ericsson Research [16]. The purpose of the DSM service is to offer a global addressable memory space, in which every piece of the memory is assigned a unique address. Given that a user knows the unique address of the memory, it can obtain either a copy of it or a reference to it [16].

The DSM approach used in the case of this project is for the users to acquire a copy of the DSM from the server. Changes can then be made to the local copy of the DSM, whereupon the copy may be submitted to the global DSM server and the data is synchronized. The implementation supports simultaneous manipulation from different clients on their respective copy or replica. The DSM server memory then synchronizes the manipulations and makes all the replicas converge to identical states after some period of time. This is accomplished through the use of the operational transformation technique, see Section 2.2.4.
Figure 9: A scheme of the DSM service communication [16]

Figure 9 illustrates the communication between individual users and the DSM server. In the example case in the figure there are several users using computer web browsers. In the case of using the DSM service together with standard circuit switched telephony, the number of users involved are usually just two.

In the DSM service, a change on the local replica is applied immediately, without the need to request a server. After submission to the server the data synchronization is handled in the background. The background handling makes the user interface fast and responsive.

Each user can have their respective listeners for remote updates. Say one user wrote hello in the shared notes application. The remote update listener of a second user would detect this input and update the screen accordingly [16].
3 Designing a Proof-of-Concept Prototype Application

This section is split up into two main sections. First, the design of a caller application in a mobile device is explained in Section 3.1 which gives the plan for the part of the implementation that is done natively on the mobile device. This part deals with the session management of the subsequent CSCW application session.

Secondly, Section 3.2 gives the design plan for several cooperative web applications. The section discusses the idea of each cooperative web application and links it to the theory in Section 2.1 enriching mobile telephone calls and the concepts of CSCW in Section 2.2. Section 4.2 later discusses the specific implementation of the cooperative web applications.

3.1 Design of a Caller Application

There are many ways in which a caller application initializing a CSCW session with cooperative web applications could be designed. Principally, the application has two important tasks to complete. Firstly, the users should be able to access the cooperative web applications when they are in a phone call. Secondly, the application shall provide a unique shared session, so that only the two participants share the application and no external users can enter.

One plausible approach to implement a caller application could be to use the internal telephony API of a mobile phone. Internal API is API designated to be used by the operating system itself rather than by third party applications. It is however possible, although generally not recommended, to incorporate internal API in third party application. Using internal telephony API has some advantages and some disadvantages. One advantage is that it is possible to override the built in phone screen, and the developer can let the application control the call (show call duration, get the number directly from the call, activate speakerphone, end the call). This enables the development of an application with total control of an ongoing phone call. However, using internal telephony API in this way also means a couple of complications. Firstly, it is not always the user wants to reach the enhanced features. Overriding the built in caller application means the application has to control the call functionality. This means a new interface for the user for something that is not a new functionality. Secondly, and most importantly, when using the internal API the developer takes a risk. These internal API are not generally meant to be used by third party application developers. The API might change without a notice, rendering applications built on the internal API malfunctioning. Thirdly, using the internal telephony API means a major permission has to be granted from the user. This might scare potential users, since always on install they have to grant all permissions the application needs in order to function.

Because of the potential risks, the caller application design chosen in this project is, therefore, such that the application first shows up discretely as a notification in the top bar when a phone call is started. The users see their ordinary call screen, but get a notification with a link to the shared applications menu. Then upon entering the menu the application strives to be very cooperative, and the screens of the users follow each other. What this means is that few tasks will have to be carried out on both devices. Instead input from one user will propagate to the screen of the other user. Another approach could be to, whenever an action is performed, such as a selection in the shared menu, send an alert to the other user asking if the other user wants to engage in the same activity.

The other important task for the application is to provide a unique session. Getting the unique
address could possibly be accomplished in several ways. The approach chosen for the prototype application is using the phone numbers of the call parties. Using phone numbers is a suitable approach since every phone number is unique and both parties in a call have access to this information; the caller knows his own number and which number is called, while the called party knows his own number and the number of the incoming call. Then, additionally, for the address to become the same on both devices, the parties can use the two numbers and perform an operation such as multiplication or string concatenation of the numbers with smallest or largest number first, that guarantees identical addresses.

Another method to set up the session management for the cooperative application could be to use another protocol such as SIP (session initiation protocol) in order to negotiate a common and unique address. This alternative method however, would require more signaling and increase the complexity of the application.

![Figure 10: The sequential procedure of the caller application.](image)

The block diagram in Figure 10 shows the idea for the sequential procedure of the caller application. Then figure illustrates the key steps of the application for both users/call parties. The first screen of the upper user, A1, is where the phone call is requested. At this initial stage the number of the called party is saved. The first screen of the lower user, B1, corresponds to this. Screen B1 is the standard screen for an incoming phone call, but in the background the number of the calling party is collected by the application.

Screens A2 and B2 are almost identical on the two users. The difference is that the notification on screen A1 appears before the call has started. This is because the phone state for the calling party counts as off hook while waiting for the called party to pick up. Screen B2, on the other hand illustrates an ongoing call. Because of this there is a chance that the calling party enters the menu, screen AB, before the called party. What might happen then is that the calling party starts a shared app in the belief that the menu screen of the called party would synchronize and follow along. If the called party did not reach the menu before the calling party started the app, this
would not be the case. However, since they are on a phone call at the same time, issues like that could easily be solved with voice communication.

One way to tackle this problem, and also a way to make the users slightly more dependent on each other would be not to allow users to start a cooperative web application on their own. Making a selection in the menu would return in highlighting the choice, on both screens. Subsequently, the other user has to choose the highlighted item in order for both to start the shared application chosen. While this approach gives more individual control to each user, it is also one step extra. Say, for example, the calling party wants to see where in Paris his friend is living. He calls his friend, and both enter the menu. Then they would have to agree on both entering the map application, instead of the called party just starts the map, and the calling party is taken there automatically. Consequently, letting any user start an application for both users might lead to conflicts in choice, but generally, it is likely to make the user experience faster. And since the users always have the possibility to communicate with voice, conflicting choices can always be resolved verbally.

3.2 Cooperative Web Applications

In order to investigate the concept of CSCW applications, a number of mobile adapted prototype cooperative web applications were developed. These are; chat, draw, notes, map, to-do, directions, language chat, balls, groceries and photos. The following sections analyze how each cooperative web application developed enriches the communication as described in Section 2.1 and how the different web applications implement workspace awareness (see Section 2.2.2).

3.2.1 Chat

Obviously an instant message chat needs two participants to function. It might be argued that a chat application does not enrich the communication of a phone call since you can say anything you can express in text. However, there might be situations where the ability to speak is suddenly compromised, such as in noisy places. Also, sometimes you might want to share things more privately without saying out loud. Thirdly, sending a clickable link or e-mail address is easier than saying it and having to repeat it several times. Hence, the Chat application presents a natural cooperation opportunity and provides a way of communication, absent in the standard CS phone call.

3.2.2 Draw

Drawing as an activity does generally not require more than one person. However, often the intention of the drawing is to show it to someone. Moreover, drawing is clearly something that cannot be done simply by talking. And in many cases it can add a useful communication channel to the standard voice interaction. Say for instance that the calling party should explain to the called party what an isosceles triangle is. Having the ability to draw the triangle on a shared drawing canvas while simultaneously using the phone call to explain, makes explaining the concept much easier. Furthermore, when integrating the photo capture functionality into the call, the communication can be even more enriched. Continuing the above example, say the called party faces a math problem where he needs to detect which one of a set of triangles is isosceles. He can then take a photo of the set of triangles, and share it with the calling party, who in turn can annotate the photo and explain the problem at hand.
The draw can be divided into three layers; one for the background, and two initially transparent layers for the calling parties. This enables for simultaneous input on separate layers, and the sharing of a common background. Figure 11 shows an illustration of this.

![Layer Design Illustration](image)

**Figure 11: the layer design of the draw application**

The design chosen, as illustrated in Figure 11, allows some customization of the degree of user awareness. For instance, by changing the color of the lines in the remote layer, the user can see who drew what. And if this is not desired, complete abstraction of user awareness can be achieved by setting the same line colors for the two layers. Another possible design for the draw application could be to have a vector based drawing. However, the technical implementation of a vector based drawing web application for mobile devices would be more laborious than the pixel based design chosen.

### 3.3.3 Notes

Notes is an instance of a collaborative application which completely lacks user awareness. It lets the users collaborate naturally, as they would on a common sheet of paper. It is just like a text editor, but with two simultaneous input points. The usage fields can include cooperatively composing an email, a message or log. Compared to the chat application which is more for communication between the users, the notes application serves as a collaborative composition tool.

### 3.3.4 Map

The map is another collaborative application that does not distinguish between the two different users. In combination with the established voice call, the map is a powerful collaborative application which definitely enriches communication possibilities during the telephone call. While other solutions could be either to send the location address to the other party so that he could look it up, or to take a screen shot of the map to send it as a picture, the shared map application sends the message across much faster and easier. Every operation made on the map is shared between the users. So users can show each other exactly what they want from a Google Map; ordinary map, satellite view or Google StreetView on any location. Achieving the same communication using just voice communication would clearly not be possible, so a shared map application evidently seems to enrich the communication of a standard phone call.
3.3.5 To-do list and grocery list
The To-do list and grocery list are two simple collaborative applications. Just as the map application, the to-do list and grocery list give low user awareness; it is not possible to see from whom the posts came. They work just as a regular to-do list or grocery list; items can be added, and items can be added. But both these tasks can be completed by either user.

3.3.6 Directions
The directions application provides geographical directions between the calling parties. The application adds to the regular phone call a feature that would be near impossible using just the voice communication. The application is minimal as far as sharing data is concerned. The only thing that is sent is the position of the clients, and possibly waypoints. Everything else in the map application (panning map, zoom, satellite/map view, StreetView) works, but is not shared between the clients.
The usages of it could be many. For instance, say two persons are in a city neither of them knows very well, and they split up. When they want to meet, they can call each other and start the directions application. Then they immediately see their positions, and the closest connecting route. Also, if one of them decides they should meet at, say the library, he clicks the library, and the routes are recalculated with regards to the waypoint inserted.
This is an instance of a collaboration web application which implements some user awareness. The screens of the users are not identical but rather mirror images. The directions shown always go from the location of the user, to the location of the remote user. It does not depend on what location was first sent to the DSM service.

3.3.7 Language Chat
The language chat basically works the same way as the ordinary chat. But it translates the messages to the language of the client. Say the calling party writes messages in Swedish, while the called party writes messages in French, the incoming French messages on the calling party device will translate to Swedish and vice versa.
The usage field of this can be to bridge communication barriers; one party might handle spoken language poorly, and might need help from his mother tongue language to communicate effectively. The language chat application can also be used for educational purposes. The calling parties can all the time speak in whichever languages they prefer, but say one of them wants to practice French, whereas the other one wants to practice German. Simply type messages in the desired language and the incoming messages will translate to this.
The language chat is of course limited to the performance of Google Translate. Some languages are not covered very well, and some languages are not translated at all. But for simple purposes the performance is overall adequate.

3.3.8 Balls
The balls application is simply just an application to show what can be done. It is not a finished application that delivers a rich user experience. However, it shows well how the sensors of a smartphone can be used to communicate. The way in which one user tilts his telephone is reflected on the other user. While there can be a couple of useful implementations of the device accelerometer sharing, the balls application shows how sharing accelerometer data can be turned into a game.
3.3.9 Photos
The photos application is designed to look the same on both users’ screens, providing little user awareness. It tries to mimic a physical photo board. Both users can add photos independently of one another to the shared photo board. One advantage of a photo application working this way compared to other ways of sending photos such as mms is that there is no need to choose recipient, since the Photos application is designed just as an in-call photo share application. Also it does not store the photos on the phone, but only presents them in the application web view. So the application should not be considered as a photo sending application, but rather as a shared photo board where users can post their photos during the call.
4. Implementation

The project has resulted in the development of an Android prototype application providing access to several collaborative web applications through an in-app WebView to be used during a phone call. The implementation of the prototype application has been divided into two parts. Firstly, the native application implementation on the Android device is treated in Section 4.1. This section thus explains how the phone is able to go from the call to the cooperative web applications. Secondly, the implementation of the cooperative web applications is addressed in Section 4.2.

4.1 Android Application

Android applications can be built up from four types of application components; activities, content providers, broadcast receivers and services (see appendix for Android basics). The prototype application uses three of these; activity, content provider and broadcast receiver. All of these communicate with one another to a varying extent. The following subsections explain the implementation of the application components in the order they are executed.

4.1.1 Activity 1

In order for the creation of the unique address to be successful, the application needs to know the number of the telephone. Consequently, the first time the user starts the application a dialog window asks for the user’s telephone number. The implementation of this uses the application’s shared preferences. In the shared preferences any application can save data specific to the application. When first starting the application, the activity class looks for a data post in the shared preferences called myNumber. When the application runs for the first time, the shared preferences don’t hold this data, and consequently the application prompts the user for the telephone number.

```java
String mynumber = pref.getString("myNumber", "no number");
if (mynumber.equals("no number")){
    AlertDialog.Builder alert = new AlertDialog.Builder(this);
    ... 
```

The first line in the code snippet above goes into the preferences and looks for a string variable with name myNumber. If it does not exist, the string “no number” is returned. The second line checks if “no number” was returned. If so, the dialog prompting the user number is built.
Figure 12: The first screen of the Android application

Figure 12 shows what the phone number dialog window looks like in the application. Upon giving the number, the user is then displayed his or her list of contacts. The collection of contacts is an instance of a content provider in the application. The contacts database is a global database on the telephone which can be accessed by any application given that the proper permissions are granted.

The next step in the application sequence is clicking a contact to make a phone call to that person. This action fires intent to use the telephone in the android device. The following snippet defines how (the variable number holds the number to be called):

```java
Intent i = new Intent(Intent.ACTION_CALL);
i.setData(Uri.parse("tel:"+ number));
startActivity(i);
```

Doing this, however, makes the telephone leave the application, and a call screen is shown. So just before the call activity is started the number called is stored in the shared preferences, just like the user’s own number, but with the reference name frNumber.

When the call screen is shown the first activity class has finished its job. The first activity is responsible for the acquisition of telephone numbers; both the number of the phone, and the number of the person being called. The activity explained in this section corresponds to the first screen of the caller, A1 in Figure 10 in Section 3.1.

4.1.2 Broadcast Receiver

The next step in the application is to monitor phone calls. This task is handled by a broadcast receiver. The broadcast receiver class listens for changes of the phone’s call state and then performs different tasks depending on whether there is an incoming call or an outgoing call. The Android telephony API has defined three call states; idle, ringing and off hook. When a phone
number is called the state changes to off hook. The broadcast receiver first detects just a change and then checks for the new state with:

```java
telephonyManager.getCallState() == TelephonyManager.CALL_STATE_OFFHOOK
```

If the above statement returns true, then the broadcast receiver class creates a notification of the possibility to enhance the call and puts the notification on the Android notification bar. The notification in turn links to the next activity. Figure 13 shows the screen when making a call. In the top, the notification is circled. The second screen shows the notification on the pulled down notification list.

![Figure 13: The notification when starting an outgoing call, and the Android notification bar pulled down](image)

The other usage case of the broadcast receiver is when the phone receives a call. In this case the broadcast receiver does not do the same thing as it does when a call is made. When receiving a call the application does not know the number of the calling party. But bundled with the intent that starts the broadcast receiver, there is an intent variable called `extra_incoming_number`. This one is null for all call states except the ringing state. Consequently, it needs to be picked up in this state. So the only thing that the broadcast receiver does in the case of call state = ringing is that it saves the incoming number to the application’s shared preferences with the reference name `frNumber`, just as the first activity class did in the case of the outgoing call.
Bundle bundle = intent.getExtras();
if (telephonyManager.getCallState()==TelephonyManager.CALL_STATE_RINGING)
{
    String number =
        bundle.getString(TelephonyManager.EXTRA_INCOMING_NUMBER);
...

The code above shows how the incoming number is collected in the broadcast receiver. The first line extracts the information in the intent. The second line controls if the phone is ringing. If the phone is ringing, the number of the incoming call is saved.

When the called party answers the incoming call, the broadcast receiver runs again. This time the call state is off hook and so the broadcast receiver creates a notification in the Android notification bar, just as it did in the case of an outgoing call.

Figure 14: Notification for incoming call.

Figure 14 shows the notification popping up in the case of an incoming call. The only difference between the call screen in Figure 14 and the call screen shown in Figure 13 is that the call has started in the case of Figure 14, whereas in Figure 13, the caller is waiting for the calling party to answer the phone.

The broadcast receiver class explained in this section corresponds to the second screens of both the caller and the called party as shown in the sequential procedure of the application in Figure 10, Section 3.1. So, as seen in Figure 13, the broadcast receiver gives to the user a link to the cooperative web applications, as proposed by the model in Figure 10.

4.1.3 Activity 2

The second activity class is the last and shortest part of the Android application. It corresponds to the right-most screen in Figure 10, Section 3.1, where both the called party and the caller see the same screen. The actual task of the second activity is to link the phone call session to the shared web application menu. In order to enhance the possibilities of the shared web applications this
class is written as a PhoneGap class. Basically this just means that the activity consists of a WebView that covers the entire screen. The PhoneGap library then contains multiple JavaScript interface functions, with which JavaScript code of the web applications can use Android native features such as the camera and accelerometer [17].

The second activity is started when the user clicks the notification in the notification bar created by the broadcast receiver. The first thing the class does is collecting the telephone numbers and combine them to create a unique address:

```java
if(frNumber.compareTo(myNumber)>0){
    address = frNumber + myNumber;
} else {
    address = myNumber + frNumber;
}
```

The first line of the code above checks whether frNumber is lexicographically greater than myNumber. If the comparison returns true, the address becomes frNumber concatenated with myNumber, and vice versa if the comparison returns false.

The other task completed by the second activity is to load the web view with the correct URL:

```java
loadUrl("file:///android_asset/www/menu.html?" + address);
```

The line above adds the unique address created from the phone numbers to the URL of the cooperative web applications menu, and then loads the file in the WebView.

When the web page has loaded, the Android implementation part is finished. It has taken the call parties from the regular call to the menu of shared application. Note that the URL links to a local html-file rather than a file on the internet. In order to reduce loading time, all html-, style-, JavaScript- and image files are put in a local resource folder in the application. Putting the files locally is also the recommended use by the PhoneGap docs [21].

4.1.4 Manifest file
The Android manifest file is a required file in every Android application. It lists all classes used in the application project as well as all permissions the user has to grant in order for the application to function. In this project the manifest file is considerably extensive since the application should be capable of a wide array of different tasks. As for permissions, the application framework needs to be granted by the user to use READ_CONTACTS, CALL_PHONE, INTERNET and READ_PHONE_STATE. In addition, with regards to the cooperative web applications two further permissions need be added in the manifest file; CAMERA, ACCELEROMETER, LOCATION.

4.2 Cooperative Web applications
As soon as the users click the notification, they enter into the cooperative web applications menu, which is presented as a WebView in the application. The following sections describe how the implementation of the different cooperative web applications discussed in Section 3.2 was made. The implementation texts focus on the implementation of the cooperative session rather than the aesthetic design of the applications.
4.2.1 Applications menu

The applications menu serves as the start page which shows the list of shared applications. The page itself is quite simple. The only task it needs to handle specifically is that both users enter a cooperative application independently of who clicked the application icon. It fulfills this task by sending the chosen item to the DSM every time a choice is made.

```javascript
set("selected", item);
```

At the same time it listens for changes made on the server:

```javascript
hash.onRemoteUpdate = function(op){
  if(op.key=="selected"){
    friendSel(op.value);
  }
}
```

The code above shows the listener for remote updates. If there is a remote update with an operation key “selected”, the menu executes the function `friendSel` with the value of the remote application. The function `friendSel` simply changes the URL of the visible page to the URL of the chosen application.

Figure 15: the applications menu. The user can go between the screens by swiping a finger.

Figure 15 shows a screenshot of the applications menu running on a mobile device, with a number of different cooperative web applications to choose from.
4.2.2 Chat
Creating a chat application using the DSM service does not require many lines of code. Every
time the user writes anything it prints it on the screen as an outgoing chat message, and sends it to
the DSM-server. Every time the DSM server is remotely updated the chat client prints the remote
update as an incoming chat message.

Figure 16: A screenshot from the chat application

Figure 16 shows what the chat application looks like. One of the messages in Figure 16 contains
a clickable link
4.2.3 Draw

The fundament of the draw application is the HTML 5 canvas element. Or rather three canvas elements; one for background, one for the remote user, and one for the user himself.

As the finger is moved across the screen a listener detects the coordinates, and draws a line on the user canvas. At the same time it sends the screen relative coordinates to the DSM. On a remote update the application transforms the relative coordinates to actual coordinates on the screen and draws the line. This adjusts for the display error that different screen sizes could cause.

![Diagram of screen sizes and coordinates](image1.png)

**Figure 17: The treatment of different screen sizes in the draw application**

Figure 17 shows how the difference in screen sizes is adjusted for. Another feature of the draw application is the possibility to choose background. With the help of PhoneGap API, the camera of the mobile device is incorporated into the application (given the device has a camera). The picture, either directly from the camera or from the user’s stored photos, is translated into a Base64 encoded string, which is drawn on the background canvas and then sent to the DSM server. Upon a remote update, the user detects an image represented by a Base64 encoded string, and then draws the image on its local background canvas relative to the window dimensions. It is important, if the camera has high resolution, to strongly reduce the size of the image for it to be able to be sent as a base64 encoded string. The output of the image will still be good since the draw application does not allow zooming.
Figure 18 shows what the draw application can look like when it runs. In Figure 18, there are two lines drawn; one is drawn by the user and one by the remote user.
4.2.4 Map
The map application puts a Google Map in the WebView. The map can be panned, zoomed and so on, just as a Google Map. But all the states of the map are shared with the other user. The states of the map are position, zoom level, map type, overlay markers and Street View point of view. Included in the Google Maps API are listeners to identify a change in either of these states. When a change is detected by either of the state change listeners the new value is sent to the DSM server. And whenever a remote change is detected the map view of the user updates accordingly. When performing the update of the map it is important to clearly state that it is a remote change, so that the Google Maps listeners don’t repeat the procedure, and create an infinite loop.

![Map application](image)

Figure 19: The Map application. Lower left corner shows the zoom meter, upper left shows the Street View Pegman and upper right corner shows the switch to toggle between map modes. In addition, two markers have been placed.

Figure 19 shows the map applications running on two Android devices. As seen in Figure 19, the position, zoom and markers of the map are the same on both devices.
4.2.5 To-do/grocery list

The To-do list and grocery list are simple applications which share their state across the two users using the DSM service. Each time either client adds a grocery item or a to-do task the item appears on the other client’s list too. And when completing a task the completion is synchronized across the screen.

![Figure 20: The to-do application](image1)

![Figure 21: the grocery list application](image2)

Figure 20 and Figure 21 show the To-do list and the grocery list respectively. The implementation of the to-do list uses jQuery Mobile. jQuery Mobile includes a JavaScript file and a CSS file targeted at making the web interface mobile friendly. Tokens of this seen in Figure 21 include full-width text input fields, the design of the bullet list and the rounded buttons. Both the
text input fields, the bullet list and the buttons have been “auto-styled” by the jQuery Mobile framework. The grocery list, on the other hand, does not include jQuery Mobile. Instead a mobile-targeted styling has been implemented manually.

Using a mobile designated library can have both advantages and disadvantages. Firstly, it makes it lots easier to format pages to look good on a mobile device. Secondly, transitions between pages can be done to mimic native applications. So, if the purpose is simply to present content, using an external mobile framework is a good option. However, for pages which have a more dynamic content, the external libraries are not as dependable, since they require the content to conform exactly to their predefined rules. They also reduce the degree of freedom for development, since you have to choose from a limited selection of themes and colours. It is possible to edit the libraries, but that requires a lot of extra work, and essentially you are then creating your own system anyway.

4.2.6 Directions

The directions application is another instance of Google Maps, but a bit different from the Map application mentioned in Section 4.2.4. For the Directions application there is no synchronization of the map states. Both users can control their position and zoom independently, so there is no need for map state change listeners in the case of the Directions application. Instead, what is shared between the users are their respective positions. Using the new feature of HTML5, Geolocation, the application is able to locate the user’s location. When the application has acquired the position of both itself and the remote user, it sends a directions request to Google Maps. If the request is successfully treated, the directions application puts the directions overlay onto the map, and adds directions instructions to a show/hide full screen HTML div element. Geolocation, directions request and the DSM service listener for remote updates all use callback functions. So the timing and order of execution code can vary a bit. Therefore it is important to make sure both users get both locations.
Figure 22: The system used in order to guarantee the acquisition of both clients’ position

Figure 22 illustrates how both users get to know both locations in the directions applications. In the execution example shown in Figure 22, the Geolocation of User 1 turns out to be fast. It sends the location to the DSM server. But since the Geolocation of User 2 is not finished by the time it detects the remote update, nothing happens. Eventually the Geolocation function resolves the location of User 2, and the position is sent to the DSM. User 1 in turn receives the position, and immediately after sends his position to the DSM a second time. This time User 2 is ready to accept the remote position, and directions can be drawn onto both users’ maps.

A final feature of the directions app is the possibility to add waypoints. When the directions have been drawn, clicking the map sends the click location to the server and then redraws the directions from the client’s position via the way point and on to the position of the other client. On a remote addition of a waypoint the same thing happens.
Figure 23: The directions application for two users. The position of the user is marked with an A and the position of the remote user is marked with a B.

Figure 22 shows what the Directions application can look like. For the test execution shown in Figure 22, the devices of the users were not vastly separated, and so directions path is very short.

4.2.7 Balls

The balls application is an experimental application, which uses the accelerometer feature of the Android handset. Each user sees her own ball as red and the other user’s ball as blue, to implement workspace awareness for the cooperative application as described in Section 2.2.2. Both users control their own ball by tilting the device. They can interact by bumping into each other. The application uses PhoneGap in order to listen to the device’s accelerometer information. The accelerometer provides acceleration on all axes. In order to make the balls move according to the tilt of the device, the acceleration data needs to be integrated to position.

\[
\begin{align*}
\text{vx} &= \text{vx} - \text{accelerationfactor} \cdot \text{acceleration}.x \cdot \Delta t; \\
\text{vy} &= \text{vy} + \text{accelerationfactor} \cdot \text{acceleration}.y \cdot \Delta t; \\
\text{x} &= \text{x} + \text{vx} \cdot \Delta t; \\
\text{y} &= \text{y} + \text{vy} \cdot \Delta t;
\end{align*}
\]

\(\Delta t\) in the above code is the time between accelerometer readings. The readings from the accelerometer are given in \(\text{m/s}^2\), and the output position needs to be given in pixels on the screen. The variable \text{accelerationfactor} \text{ therefore serves to make the conversion from meters to pixels. Rather than calculating a relationship, an appropriate value of \text{accelerationfactor} was derived through empirical testing.}
5. Evaluation and Analysis

One way to test the performance and response of the applications and the DSM-model would be to check how long time it takes for the updates to be made. This can be implemented with JavaScript. One simple performance test file can be created that simply sends the exact time a button is clicked or an event is triggered, to the DSM memory. The other client listens for these updates, and on receipt, it records the exact time, and compare it to the value send over the DSM. Variation in the evaluation tests can be established both by varying the frequency of DSM updates, and the size of the data sent to the DSM.

Figure 24 shows times taken for sending mouse move events between two users, but on the same computer (two web browser windows). The frequency of mouse move events is high, and resembles the situation when the position of the map is moved, the point of view of the Street View is changed or when a line is drawn in the Draw application.

![Data transmission chart](image)

**Figure 24: delay until remote updates are propagated**

Figure 24 shows the times taken for every single update to the DSM server. In total around 250 DSM updates were made. The test was performed on a MacBook Pro with fast Ethernet cable internet. Figure 24 indicates that the time of transmission is generally very fast, around 25 milliseconds. For the human eye, this is a very short delay. Consequently, the performance of high frequency data transmission applications can be said to work well. The user experience as far as time delay is concerned of applications such as draw and map will therefore be good. No tests were performed on the audio delay of the standard phone call. It could have been interesting to see which one would be the fastest, the DSM update or the audio through the phone call.

The performance test shown Figure 24 is dependent on synchronized clocks on the two users. Since two users were simulated on the same computer, they use the exact same computer time. A mobile device also has a machine clock built in, but it cannot test with two web browser windows simultaneously as the computer can. And the machine clocks on different handsets (or
computers) are not precisely synchronized. Consequently, just comparing the time of reception on the second device with the time of sending from the first device would not give a reliable result.

Instead, to make a performance test on mobile devices, a time is recorded on the first device just before a message is sent to the DSM server. The second device detects the remote update, and sends a confirmation to the DSM server. In turn, the first device notes this remote update and checks the time again. This final time can now be compared with the first time. The result is the time taken for the transmission of two DSM updates. Figure 25 illustrates the idea for the performance test for mobile devices.

![Diagram showing the test method for mobile devices.](image)

**Figure 25: Implementation scheme for experimental testing of the delay until DSM updates are propagated**

The test method displayed in Figure 25 was applied both for small data sizes and high update frequency (which is the case in applications Draw, Map and Balls) and big data sizes and low update frequency (photo application).

Figure 25 shows the outcome of a test with small data size and high update frequency.
Figure 26: Times taken for 100 updates to the DSM memory.

Figure 26 shows the times taken for updating the DSM memory a little more than 100 times. Actually, every update is in fact two updates, since the second device, detecting the first update, makes a new update, which the first device then compares to the start time. The test was made with SonyEricsson XPERIA Arc mobile phones, and Wi-Fi internet connection. The speed of the Wi-Fi connection was measured to be 9.5 Mbit/s downlink and 6.2 Mbit/s uplink (average values from three measures with Bredbandskollen Android application). The average time observed in figure 24 seems to be around 50 milliseconds. Since the test for the mobile devices made two DSM updates, this value needs to be split, which makes the average transmission time 25 ms. 25 ms was indeed the value observed for the computer test. Consequently, the DSM does not seem to be strongly dependent on the platform.

The same test program was also run without Wi-Fi connection but with 3G connection instead. Also this test was with SonyEricsson XPERIA Arc. The outcome of the test is shown in Figure 27.
Figure 27: Time taken for circa 100 updates to the DSM memory

Figure 27 shows the time taken for updating the DSM memory using 3G instead of Wi-Fi connection. The speed of the 3G was measured to be 1.0 Mbit/s downlink and 1.37 Mbit/s uplink. Figure 27 indicates an average propagation time for the DSM updates of around 150 ms. Undeniably, this is about three times as slow as the average time when the phones had access to Wi-Fi, shown in Figure 26. Although time delay of 150 ms can be claimed to be within the acceptable time delay, the test shown in Figure 27 indicates that the performance of the cooperative web is relatively dependent on internet connection speed and that they might not perform satisfactorily in places with bad connection to a 3G network.

In order to provide a wider performance evaluation, tests for greater amounts of data need to be carried out. Figure 28 depicts the outcome of a test with large data size and low update frequency. The data sent is a photo in the form of a Base64-encoded string.
Figure 28 shows the time taken from a photo being taken, to the sender of the photo detecting a DSM update made by the receiver of the photo declaring successful photo reception. The photo size is calculated from the length of the base64 encoded string. Each character in this string is built from 6 bits, whereas a byte holds 8 bits. Hence, multiplying the string length with 6/8 or 3/4 results in the size expressed in bytes.

Figure 28 indicates a linear relation between the file size and the time taken to send and update. It also illustrates the importance of reducing file size. The test was made with a Sony Ericsson Xperia Arc which has a camera resolution of 8 megapixels, with picture file sizes around 1000 kB if no quality reduction is made. Assuming a linear correlation, sending a full resolution photo would take approximately one and a half minute. This time is too long, and in many cases longer than the duration of the actual call. Another technique to implement a photo sharing application would be to upload photos to a server and use the DSM service just to send the address to the photo. Perhaps, that would result in better transmission times. However, due to lack of time and lack of server, this method was neither implemented nor tested.

Another way in which the performance of the shared applications can be tested is to use the Google Chrome task manager on a computer. It provides data on CPU and network usage for individual Google Chrome tabs and windows. So in order to use it to test the DSM performance and demands, two windows were opened, to serve as two users. Updates were made in one browser window, while observing the task manager.
<table>
<thead>
<tr>
<th>Application</th>
<th>CPU (%)</th>
<th>Network (Bytes/s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>menu</td>
<td>1</td>
<td>70</td>
</tr>
<tr>
<td>Chat</td>
<td>1</td>
<td>286</td>
</tr>
<tr>
<td>Draw</td>
<td>12</td>
<td>35000</td>
</tr>
<tr>
<td>Notes</td>
<td>2</td>
<td>2630</td>
</tr>
<tr>
<td>Map</td>
<td>13</td>
<td>1 142 000</td>
</tr>
<tr>
<td>To-do</td>
<td>7</td>
<td>350</td>
</tr>
<tr>
<td>Direction</td>
<td>10</td>
<td>502 200</td>
</tr>
<tr>
<td>Translated chat</td>
<td>1</td>
<td>749</td>
</tr>
<tr>
<td>Groceries</td>
<td>5</td>
<td>290</td>
</tr>
<tr>
<td>Photo (only reception)</td>
<td>15</td>
<td>511 100</td>
</tr>
<tr>
<td>Balls (only reception)</td>
<td>1</td>
<td>11 400</td>
</tr>
</tbody>
</table>

Table 1: CPU and Network data for the shared applications. Some applications use phone specific input, in which case the computer can only receive updates, not post them itself. These applications are marked with “(only reception)”

Table 1 shows how much CPU and network that was maximally used by the computer when the shared applications ran. The CPU is shown in percent of the CPU capacity. The test was run on a 2.00 GHz 32-bit Windows Vista computer.

The data displayed in table 1 show that the applications do not consume an excessive amount of the computer’s resources. The max value for network is 1 MB/s in the map application. What would happen in case that speed could not be provided is simply that the map would load a little slower. As for CPU, all of the applications use just a little fraction of the total resources. The maximum observation was 15% in the Photo application. Additionally, this was just a momentary observation. It was not constantly using 15% of the CPU, but peaked when loading the photo to the screen.

However, what needs to be considered as far as this topic is concerned is that a telephone is less powerful than a computer. The newest Android phones on the market have 1 GHz-processors and 512 MB RAM.[22] Given there is no major background activity soaking the power, it seems that all the shared applications developed will run without problem on mobile devices. This was also the user experience observed from very small scale testing.
6. Conclusion

The thesis has investigated the concept of CSCW in combination with circuit switched telephony, or more specifically how cooperative web applications can be implemented and bundled with mobile telephone call. The project has resulted in a prototype mobile application allowing users first to make a standard phone call and then engage in a CSCW session using either of ten cooperative web applications implemented.

In order to be able to correctly address the issue of enriching communication with cooperative mobile application, the research field CSCW has been thoroughly examined. In accordance with CSCW theory the development of the cooperative applications was governed by the four requirements for an effective CSCW session; communication, awareness, session management and concurrency control. By tying the functionality of the cooperative applications to those four concepts, the applications successfully serve to enrich the communication of a circuit switched telephone call between two mobile devices.

The second prime part of the project was to find a plausible way to actually bundle the cooperative applications developed to the phone call of a mobile device. This task was addressed by the development of a proof-of-concept application developed for the Android platform, and demonstrated on a SonyEricsson XPERIA Arc unit. Using the Android API for telephony, the application could listen for different phone states of the telephone. In case of an ongoing call, the application will awake, grab the phone numbers of the active call, and link the user to the interface of cooperative application which has been set up with the phone numbers as the unique identifier.

The final part of the project assessed the functionality of the applications running on an actual mobile device. By implementing test applications which mimicked the functionality of the actual applications, and taking times for update propagation, the cooperative web applications were tested, and the results were analyzed. The results of the tests showed that the cooperative web applications generally did work satisfactorily for a mobile device. However, the speed of message propagation tended to dip when the applications were run from a 3G network connection.

To sum, the project originated with a theoretical basis within the computer science field commonly known as computer supported collaborative work. Based on this theory, a proof-of-concept application was developed for a mobile platform. The performance of the prototype application was finally tested.

6.1 Prototype improvements

One thing that would have been desirable to remove is the initial contacts list. In all Android devices there already is a native contacts list. So there is no need to create an extra. However, since getting the numbers from the call is impossible without using the internal API, this has to be the solution if the internal API is excluded.

Furthermore, the solution presented still has several potential enhancements. One major improvement would be to add a server database with all numbers having signed up with the application. This would allow the notification to appear selectively depending on whether the person calling actually made his call from within the application. Otherwise clicking the menu would still result in the menu appearing, but with no other user.
6.2 Further research

For the prototype application to be tested in a broader perspective, it would be interesting to put the Android prototype application on the Android Market, or in other ways let the prototype application be tested by a large number of users. This could give the thesis project additional feedback on what cooperative web applications work well from a user experience perspective. It could also be a way to find out new possible cooperative web applications, and thus new ways to enrich the communication experience during a standard circuit switch telephone call. Another further work for the project would be to implement the application on more platforms. Currently, a prototype application exists only for the Android platform. Widening the scope of the application to include iPhone, Blackberry and Windows mobile could provide further information on how to enrich the communication between different smartphone devices. Yet another possible improvement for the prototype would be to turn the cooperative web applications into native cooperative applications. A disadvantage with deploying the CSCW application as web application is that the user interface can seem to look a bit “webbish”. It is likely that native cooperative functionality would give a smoother user experience.
7 References


[17] [http://www.phonegap.com/features](http://www.phonegap.com/features), extracted 2011-06-02


[21] [http://www.phonegap.com/start#android](http://www.phonegap.com/start#android), extracted 2011-06-07