Abstract

Social networking service for mobile devices is presented and evaluated. The operating principle of the service is inspired by human-like cumulative gossiping. The usage of the service is browser based and implementation utilises standard ad hoc communication between smartphones. The cumulative gossiping protocol is built on the service level and on top of the existing ad hoc communication method. The service is evaluated with six social groups. Results suggest that there is great social demand for this kind of social networking service due to similarity of human-type gossiping.

Also, a number of propositions for technical solutions to further enhance the end user experience of the service were gathered.

1. Introduction

Currently, smartphones equipped with ad hoc networking techniques and social service access are carried by people everywhere. Social web-services with central servers such as MySpace, LinkedIn and FaceBook are used to keep in touch with friends and work mates [1]. These services are also used to develop new social contacts, for example, people with same interests may find each other and communicate without having to worry about long geographical distances. In spite that smartphones provide the best way to update the profiles of social web-services in real time while on the move, smartphones provide a platform for utilizing people spatial interactions as a part of local area social service interactions. For example, many dating applications developed into smart phones utilise Bluetooth service discovery [2].

The paper focuses on a new type of mobile messaging scheme, a service called Distributed Dynamic Threads (DDT). The service enables sharing and producing content as well as cumulative interactions to other devices over Bluetooth. The service follows loosely the principles of gossip networking protocol, and it complements them by cumulative factor from individuals. A service like this could be utilized for example in areas where the network infrastructure is destroyed by a nature catastrophe. Although gossip based protocols are widely studied on networking, to our knowledge it has not been explored as a basic operating feature for spatial social services.

The paper is organised as follows. Related work is discussed in next section. Human like gossip communication is introduced in section III. Section IV presents the design of DDT followed by evaluation and conclusion sections.

2. Related work

A message thread is a chain of interrelated messages”. It consists of a start-up message and usually a title for the thread is given as well, see e.g. [3]. The thread is distributed to other users of the service and they add their replies to it. Discussion forums, internet forums or Bulletin Board Systems (BBS) are web services where people can create thread formatted discussions. They can make announcements, exchange information and naturally browse the content produced by the other members of the community [4]. Traditionally this kind of a service has been centralized; the service has been run by a central server, for example a database and a hypertext transfer protocol (HTTP) -server, and the users of the service have browsed the content using an ordinary web-browser. When the content has been modified or displayed to the user, it has always remained on the central server.

Using Bluetooth-enabled mobile devices, ad hoc networking and the concept of mobile peer-to-peer- gives an interesting aspect to thread format message exchange. Social networking is made possible in situations where people density is spatially rich. Bluetooth also provides a possibility for relatively fast data transfer facilitating social multimedia networking.

A quite similar topic to this is gossip-based MANET, which it is characterized as an infrastructure-less mobile wireless network, in which two mobile nodes communicate with each other through intermediate nodes [5]. Compared to ad hoc networks formed by non-mobile devices, MANET requires that a few things are paid special attention to. Mobile devices change their geographical location quickly and have limited bandwidth and computing resources. Some implementations of routing interfaces exist, such as Pastry [6] and ORION [7]. Both are request-response based and for both
functionalities separate routing tables are stored in each device. Nodes maintain a response-routing table for search queries by storing the routes to each node in their routing vectors. For file transfers the nodes do filtering so that if the same file is stored to multiple nodes, the routing table points to the closest node.

Haveri et al. [8] introduced a service called mCell. mCell is a portal where users are able to create conversations in different formats, for example real time or bulletin board type, within a community formed by a relatively small amount of users. In a technical level it is based on a Apache web server port on S60 platform and a custom client application. mCell contains similar elements as the service introduced in this paper but still they have fundamental differences in their operating principles. The DTT explores cumulative gossip based social networking service, which is able to differentiate and intertwine threads for enabling human-like gossip spread. According to Garbinato an ad hoc application is a self-organizing application composed of mobile and autonomous devices, interacting as peers and which relationships are made possible because of relative physical distance (collocation) [9]. DDT is not a heavily ad hoc networking-reliant service, but its operating principle for dealing with content has of autonomous flavours.

Various different kinds of technologies have been introduced in this chapter; discussion forums, P2P, MANET and Bluetooth. When all of these are combined, interesting ideas can be developed. For example, Zhao [10] described a technique called message ferrying. A message is created using a mobile device and the message is stored in the device’s memory. When the mobile device meets another device, the message is transferred to its memory as well. This device or MANET node in this case, carries the message further to another device. Distributed dynamic threads use a similar kind of message ferrying, but with a difference that the message content is formed cumulatively in each node.

3. Human like gossip communication

This section discusses characteristics of human-like gossip communication with cumulative feature. A model for communication is developed and studied.

Figure 1 illustrates the principle of cumulative gossip based communication. Two pieces of information, with topics $x$ and $y$, are created in social context A becoming to the knowledge of the members of this social context, including persons $n$ and $m$. The person $n$ changes their location into different social context. The participants in the new social context B have opportunity to share topics $x$, $y$ and provide additional related input $x_1$, $y_1$ to them. Similarly, participants in the new social context C have opportunity to share topics $x$, $y$ and provide additional related input $x_2$, $y_2$ to them. Participants in social contexts 1 and 3 have cumulative content $x+x_1$, $y+y_1$, $x+x_2$, $y+y_2$. Now, these cumulative threads may further evolve over several different social contexts. Eventually topics $x+x_1$, $y+y_1$, $x+x_2$, $y+y_2$ may collide when totally different threads with the same background are combined.

This kind of scenario leads to interesting situations; the thread can divide into multiple branches if the users storing the same thread move to different environments This also resembles a more “human” kind of messaging than, for example, short message service (SMS) or IM. In real life people exchange gossip; one person tells a piece of gossip to a second one and the second one tells it to a third person. Persons gossips are flavoured by their own ideas, background and opinions. The idea of DDT is very similar; threads are passed from one terminal to another and new information is added on the way.

![Figure 1: Human like cumulative gossip communication behaviour.](image1)

This gives an interesting opportunity for social services to gain novelty to content when independent contributions are intertwined for both parties as combined new content. The key element then is a procedure for indexing the content universally and combining the content as meaningful thread structure.

The illustration of the principle in Fig. 1 is visualised as an example of mobile social service with text content in Fig. 2. The initial content $x$ in the gossip, or in a messaging thread, is created. Person moves to different social contexts and threads get different follow-ups which are added at the end of threads.

![Figure 2: Human like cumulative gossip communication behaviour.](image2)
4. Design and implementation

In our initial implementation for text based content the DDT realises message chains similar to discussion boards. First a title and content for the start-up message are created and other users can create replies to the chain. The fundamental difference to traditional discussion forums is that message chains are distributed in a decentralized manner.

All DDT enabled terminals can create threads. This means that they can write a start-up message, the first message, contributions to a thread and give the thread a title. Other terminals can fetch the thread from the terminal and create replies to the thread. The whole thread content and the newly created reply are stored in the memory of the terminal that fetched the thread, and this terminal starts to distribute the thread, including the reply as well. The originator of the thread does not hold the content of the reply unless it fetches the new version of the thread from the reply creator.

4.1. Architecture

The software implementation was done on smartphone S60 Symbian platform. The implementation consists of three main components; a client, a server and a content creating application. In addition to this, two file type handlers are implemented to transfer data from the web browser to the DDT client.

The DDT client connects to the DDT server of the other terminal using a protocol that is described later in this paper. All the communication is done over Bluetooth RFCOMM protocol. The DDT client sends a request to the DDT server and the server answers with a response. All the threads are stored in XHTML files in the memories of the devices.

The user interface is realized using the web browser of the device. An XHTML page is shown via the browser and it lists the actions that the user can execute as hyperlinks. When the user selects a link the recognizer passes the information to the DDT client. The content creation application is started in a similar manner; the user selects a hyperlink and the content creating application is started in new-message or reply-mode. The content creating application has a feature that uses Bluetooth communication as well; it sends a request to the DDT server’s terminal to indicate about new content in a thread with a vibration signal. Next section describes this in more detail.

4.2. Communication protocol

Every DDT enabled terminal has the server and the client applications installed, and this makes both receiving and sending requests possible for each terminal. Data transfer between the DDT client and the server is based on a simple request-response protocol; the DDT client sends a request and the DDT server sends a reply to the request. These kinds of HTTP-like protocols have been used earlier for P2P traffic in applications like Gnutella [11]. In DDT the request format is GET <thread_id> <request_type><line break>. The thread id is a 32 bit unique identifier for a thread. In some cases the thread id is not needed for the request, for example when requesting a list of threads available, but on the server side it is expected to be included in every GET-request. This is handled with a dummy thread id, which is a 32 bit random integer.

Figure 3 illustrates the messaging sequence that is performed when a specific thread is transferred to another terminal. Three different requests have to be made if a specific thread is fetched from the remote terminal. First, a list of threads stored in the remote device must be obtained. Secondly, the list of files belonging to this specific thread is obtained, and finally the thread content itself is obtained. This is done by executing a device discovery to search for DDT enabled devices in the Bluetooth range. The user selects a device that (s)he wants to view. The DDT client sends a “GET THREADLIST” message to the DDT server of the other device. The response from the DDT server contains a list of threads stored in the device. When the list of threads is received, the DDT client sends a “GET FILELIST” request separately for each thread. DDT server sends a list of files belonging to the specified thread as a response. The client receives the response and generates a NEW-file from the file listing received.

![Figure 3: A messaging sequence in a scenario where the client selects and fetches a specific thread from the server.](image)

An XHTML-page is generated from the thread list and from the thread titles. Every hyperlink in this XHTML-page points to a NEW-file that was generated. When the user selects a link, the NEW-file is opened and a “GET <filename>” request per each line in the file is sent. The response from the server contains the file data and the data is written to the terminal memory. A folder for the thread is created in the terminal memory as well at this point. After all the files have been received, the terminal browser is redirected to “index.html” that is received with every thread and located in the thread folder created earlier.

The user who starts the message chain is informed when new content is added to the thread, if (s)he is on the communication range. Earlier studies made by Abras [12]
claim that an indication to a content creator about other users using the content is important.

A reply is created with the message editor and after the user has created the reply (s)he submits it from the message editor UI. The reply is written to the terminal memory (to file “index.html”) and a “GET VIBRA” request is sent to the DDT server of a remote terminal (the thread starter’s terminal). This will run the vibra motor on the DDT server terminal for a brief period of time. No reply is sent back by the DDT server for the “GET VIBRA” request.

4.3. Thread distribution and branching

There are several scenarios how threads may distribute, and there are also a number of possibilities of how branching and combination of different branches may occur. To assure end user understanding of the operating principle and a reasonable user experience special attention must be paid on managing thread branching and presentation of content on a UI. The combining of branches is handled by the DDT client. When a thread is fetched, the fetching device first checks whether a thread with the same UID is already stored in the device. If this is the case the client device checks the thread lists, goes through thread contents and combines the two versions using time stamps in the replies. A uniform thread is formed so that the replies are in order according to the creation time. If UID on the particular topic does not exist, a new branch is added. In our prototype implementation for text based content only UIDs and time are used for managing the branching.

4.4. User interface

This section describes the user interface implementation. Three different applications are shown to the user: DDT client, message editor and DDT server. The first two of these are launched through the UI. The client is controlled with the web browser of the device. The user interface is implemented with an XHTML-page. By selecting the appropriate links in a page, the user can execute desired functions. The user can create a new thread, discover other DDT enabled devices and browse their threads or browse the threads stored in the local device.

The page is illustrated in the left hand side of Figure 4. This page contains four options that the user can select, depending on the action (s)he wants to execute. The first one of the links “New thread” is a hyperlink to a SUB-file that contains keyword “NEW” as content. When the user selects the hyperlink, the .SUB recognizer code is run. It will start the message editor in a new thread creation mode. The rest of the hyperlinks in the page are used to control the DDT client and they point to .NEW-files.

The “Discovery” selection is used to search for other DDT enabled devices. A pop-up window with a list of surrounding devices is displayed. The user can select the one from where (s)he wants to fetch the list of threads stored in the device. Behind the link keyword “DISCOVERY” is used in a NEW-file.

The last option on the bottom of the page is “My threads”, which is used to generate an index of the threads stored in the local device. An XHTML-page is generated from the index and displayed to the user. The keyword used in the NEW-file is “MYTHREADS”. The “Devices” link shows nearby devices with names and UIDs.

Figure 4: The XHTML-page acting as a main user interface for the DDT client.

An application is needed to create the user replies because for technical reasons they can not be created with the terminal browser. Message editor is a simple application which contains a menu, text input field and a submit button. Message editor is launched automatically when the user selects “New thread” from the client UI or “Add reply” in an existing thread. On the right hand side of Figure 4 is an example of a thread with the “Add reply” option.

An application is needed to create the user replies because for technical reasons they cannot be created with the terminal browser. Message editor is a simple application which contains a menu, text input field and a submit button. Message editor is launched automatically when the user selects “New thread” from the client UI or “Add reply” in an existing thread. On the right hand side of Figure 4 is an example of a thread with the “Add reply” option.

Figure 5 illustrates the message editor. The first picture on the left hand side is the text input field. The user can type in text as in any other applications on the terminal using numbers, capital letters etc. The picture on the right shows the function when the submit button (in the bottom right corner) is pressed in case a new thread is created. The application asks the user for the thread title and the user presses “OK” when done. If the user created a reply to an existing thread, no title is asked for. After this, the message is saved and contributed to the thread. The message editor was developed on the base of the AknExEditor example software provided with the S60 SDK.

Figure 5: The message editor.

Figure 5 illustrates the message editor. The first picture on the left hand side is the text input field. The user can type in text as in any other applications on the terminal using numbers, capital letters etc. The picture on the right shows the function when the submit button (in the bottom right corner) is pressed in case a new thread is created. The application asks the user for the thread title and the user presses “OK” when done. If the user created a reply to an existing thread, no title is asked for. After this, the message is saved and contributed to the thread. The message editor was developed on the base of the AknExEditor example software provided with the S60 SDK.

5. End user evaluation

An end user evaluation was arranged to recognize indications on the development needs and to evaluate
end user understanding of local area social networking with different operating principle.

5.1. Test setup

The evaluation consisted of three different phases. First, the users were given a short introduction about DDT as a messaging method followed by instructions on how to use the service with mobile terminals.

The users were then given a scenario during which they used the service in multiple ways. During the scenario a group of 2-3 users were simultaneously using the service for messaging. The test group was allowed to use the service as long as they wanted, but still it was ensured that all the functions in the service were used. After they had used the service a survey was conducted to collect feedback from the test group. The survey was categorized into two groups of questions. The first questions studied DDT as a messaging method in general and the rest handled UI related factors. All the questions were open ended and people were encouraged to answer very freely. In addition to the survey, the users were observed during the user test session.

The total number of test users was 12. The users were located in the same room and observed during the test session. If they encountered problems and did not know how to perform the desired action, they were guided.

5.2. Results with discussion

The results of the survey and user observation were categorized into two main categories. A summary of the questions asked in the survey and the results got is presented in Table 1. Since users provided actively a number of comments to actions during the session, we present the discussion here together with the results.

In general DDT as a messaging method was experienced very positive (55 positive, 14 negative opinions). However, according to some users the suitability depends on the context; how many DDT users there are near by and how actively they create new threads. The reasoning for this was linked to the polling of threads; first a DDT device discovery must be executed to fetch a list of threads stored in the device, and then the thread itself must be fetched to check whether or not it contains new content. When there are multiple DDT enabled devices in the Bluetooth range, multiple devices must be checked for new content. In contrast, some users felt that the large density of DDT users in the environment could be a positive factor. It would increase the efficiency and speed of information distribution. In announcement type of information sharing this would be a remarkable fact.

Two test users commented that when the content is collected cumulatively, meaning that all the messages belonging to the thread are transferred all the time with the thread, the service can be considered quite fail-safe. If one terminal runs out of battery or is not able to run the DDT service for some reason, other terminals can still exchange data and all the data produced by the lost terminal is still available in the other terminals’ memories. It was also commented that the messaging efficiency is not heavily dependent on the activity of a single user, and this was justified with the same reason as the previous fact; i.e. that all the content that is produced by a user is stored in other terminals as well.

Table 1. The questions of the user test. The answers are categorized into positive (+), neutral (O) and negative (-) columns.

<table>
<thead>
<tr>
<th>Questions – DDT as a messaging method</th>
<th>+</th>
<th>O</th>
<th>-</th>
</tr>
</thead>
<tbody>
<tr>
<td>Do you find vibra alert suitable for DDT?</td>
<td>55</td>
<td>3</td>
<td>14</td>
</tr>
<tr>
<td>Do you think there’s a situation where DDT could be handier than SMS?</td>
<td>10</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Total User experience</td>
<td>76</td>
<td>13</td>
<td>31</td>
</tr>
</tbody>
</table>

Users suggested that the content production should be integrated to the existing mobile phone functions. If a user writes an SMS or takes a photo with the mobile phone’s camera, the user should be able to use this newly created information with DDT as well with minimal effort.

The test group was asked to share their opinions about concepts where DDT could be applied. The most common comment was, made by a quarter of the test users that the service is suitable for searching for people and friends. It was proposed to be used in a big crowd, maybe in a mass event, to find someone. Feedback was also made that the service could be used for arranging and sending invitations to meetings or different kinds of events. This kind of information exchanging does not require communication on a rapid speed since the information that is distributed can be considered to have rather low priority.

The users were asked about a user effort needed to use the service. Eight out of eleven users felt that the effort
was within bearable limits, but this requires that some initial conditions exist; the users should be familiar with the application. This makes it easier to adopt the idea of DDT because there is no need to learn trivial things related to messaging, such as typing etc.

Current user task for device discovery was found not to be suitable. One quarter of the test users said that the device discovery should be automatic, and users should be able to subscribe to certain threads. Then an indicator would be sent to inform the user about new content in a thread. It was also commented that the terminal should alert the user when another DDT enabled terminal comes into the range of user’s device.

Privacy was identified as an important topic by one group, two propositions for solving the problem was given. Firstly, user should be able to limit the group of terminals to whom the own terminal is shown in device discovery. The second idea was to limit the group to whom some specific threads are shown.

The survey also studied the question as to how users should be informed about new content in a thread. The current solution, sending a signal that causes the thread initiator’s terminal to vibrate, was found suitable by one third of the test group and the group of people to whom the vibration is sent should be considered. Some users felt that all users who have contributed content to the thread should receive the signal always when the thread is updated. Some users suggested that the user should be able to turn off the vibration and replace it with an audio alert or some visual alert, for example with a flashing screen or an icon.

6. Conclusions

Social networking service based on spatial cumulative gossiping for mobile devices is presented and evaluated. The service is evaluated with six social groups. Results suggest that due to similarity of human-type gossiping there is great social demand for the social networking service. End user evaluation was positive. Several improvements for technical solutions to further enhance the end user experience of the service were suggested. However, the major drawback of DDT was the pull type of service, meaning that the users have to fetch the content themselves from other terminals. The messaging method itself, mainly because of its cumulative manner of distributing content, was considered to be quite efficient and users easily adopted the idea.

DDT is suitable for local area social networking applications. These include distributing announcement type of information, but when the need is for more real-time communication other, more conventional, messaging methods were considered to be more useful.

DDT could be valuable aid also in crisis areas. It can be used in situations where the cellular infrastructure is out of order, such as earthquake areas and still there may be high density of people in a small area. Emergency messages from mobile phone users could be collected cumulatively and transmitted to an area where cellular network infrastructure is available. Then all the collected messages could be transferred to an authority providing first aid in the area.

References