Automated Content Sharing in Extended Homes through Mobile Devices

Shahzad Akhtar Awan

University of Tampere
Department of Computer Sciences
Software Development
M.Sc. thesis
Supervisor: Eleni Berki
June 2008

University of Tampere
Department of Computer Sciences
Software Development
Shahzad Awan: Automated Content Sharing in Extended Homes through Mobile Devices
M.Sc. thesis, 53 pages, 15 index and appendix pages
June 2008

Group content sharing and communication has matured from an interest to a need, but there have been few secure and cost effective solutions to safely support content exchange among user groups.

This thesis work analyzes existing content sharing solutions, specifies their limitations and presents a new solution that overcomes some of those limitations. It allows a group of users to share their mobile contents generated during a specific amount of time interval in a secure, organized, automated and cost effective way. It uses current smart phones, mobile middleware, web technologies and utilizes home infrastructure as a central repository for all shared contents rather than third party services.

A prototype of presented solution was developed which proved the feasibility of the concept and it can be used as a baseline for enhanced content sharing solutions.

Key words and terms: content sharing, mobile phones, remote access, symbian, extended home

Acknowledgements:

I would like to thank my supervisors Dr. Eleni Berki from University of Tampere and Petros Belimpasakis from Nokia Research Center, Tampere for their supervision, dedication and guidance throughout my thesis work. They guided me in every theoretical and technical aspect and also arranged an internship for me at Nokia Research Center, Tampere to carry out the implementation part of this thesis work during summer, 2007. I would also like to thank them for guiding and accompanying me in publishing two research papers related to this work.

I would like to thank Seamus Moloney, Rod Walsh and Harri Hakulinen for allowing me to be a part of their team during my internship work at Nokia Research Center and for providing valuable feedback. Also, I would like to especially thank Mihaly Borzsei for providing technical help and guidance during thesis implementation process.

I would also like to thank Professor Jyrki Nummenmaa for reviewing initial thesis draft and providing feedback.

Last but not least; I would like to thank my family and friends for their patience, prayers and moral support.

Contents

Figures:

Tables:

1. Introduction

Rapid developments in mobile technology have totally changed the way people used to perceive mobile phones. In earlier days, mobile phones were only used as devices to make phone calls. However, with the passage of time embedded features like messaging, web browsing, high quality digital and video camera, media player, third party applications support and Wireless Local Area Network (WLAN) capability have transformed mobile devices into multimedia computers. These multimedia computers have become a vital part of everyone's daily life and excessive sales of camera phones over digital still cameras is a sufficient proof of it [Camera Phone sales, 2008]. Due to these advancements, user created mobile content is on the increase, both in terms of quality and quantity [Belimpasakis & Walsh, 2006] and this raises the need of mobile content sharing. People feel the need of sharing their mobile content with friends and families especially images because it allows multiple interpretations of the content and no interpretation is exhaustive [Battarbee & Kurvinen, 2003].

Once the moment of excitement is over, people do not really bother, forget or are bored to share their content with others. Some people that have camera phones do not even transfer their captured images out of their phone. They just delete them once their phone is out of memory. Multimedia Messaging Service (MMS) is expensive and is streamlined for one to one communication. Softwares like MobShare [Sarvas, 2004] and Nokia LifeBlog [Nokia Lifeblog, 2008] mobile version provides feature to share mobile contents on semi-trusted third party web servers, but uploading contents from mobile to web is expensive, usually slow and reduces battery life. It also requires huge effort in organizing and presenting digital content and can discourage frequent use [Gossweiler & Tyler, 2004]. Using third party web services as contents repository can also lead to data misuse and generally privacy concerns [Spangler 2006].

This thesis work presents design, working and real life use case of a new solution [Awan et al., 2007]; [Belimpasakis et al., 2008b] that overcomes the deficiencies of existing content sharing solutions. The presented solution allows a group of users to participate in a common sharing session and will allow their mobile devices to create an alliance for predefined amount of time. It makes photographing social, in which participants treat photographs as common property, they feel they can use all photographs taken from the event regardless of whose device was used [Battarbee & Kurvinen, 2003]. Contents created during the session will be automatically marked for sharing and once the sharing session is over, contents with be automatically transferred to the Home PC of user who created the session using secure web connection. Contents transfer can take place either with the help of WLAN or General Packet Radio Service (GPRS) depending on user's selection. Using Home PC as contents repository, secure web connection and WLAN or GPRS for contents transfer makes presented solution reliable, secure and cost effective if desired (i.e. user selects WLAN for contents transfer).

The thesis work is divided into five chapters. Chapter two explains the key concepts such as content sharing and management, remote access to home networks and Symbian Operating System (OS). Understanding some key concepts is essential to get a clearer picture of the thesis work and its contribution to the field of content sharing. Chapter three specifies existing solutions that have been commercially developed and others as part of ongoing research related to mobile content sharing. It highlights current situation in field of content sharing and identifies key limitations of existing systems. Chapter four is the core section explaining the work carried out in this thesis project using existing systems as a baseline. It explains working of prototype system developed during this work, specifies its benefits, provides its comparison with existing systems and presents an example use case for better understanding. Chapter five is the final chapter and it summarizes the findings of thesis work. It also specifies future development aspects and research directions. The developed system's design document and code snippet are added as appendices.

2. Key Concepts

This chapter explains the key concepts related to the thesis work. Understanding of these concepts is necessary to gain better insight of research and development that has been done in the field of content sharing.

2.1 Content sharing and its paradigms:

Nowadays, digital information is usually referred as content or digital content [Wikipedia-Content Management, 2008]. A digital content item may take the form of file, image, picture, video etc. While in the past, content has been highly produced by professionals, more and more content is generated now days by non-professional users. This results in a new situation that users can be consumers as well as producers and large number of people will become potential publishers [Jans et al.,2007]. This trend has increased in popularity recently because Internet connection has become more common in compact devices and its speed is also increasing with the passage of time [Hellsten, 2006]. Digital content life cycle consists of six phases: create, update, publish, translate, archive and retrieve [Wikipedia-Content Management, 2008]. For example, content is created by one or more users, which is later on updated by other users. After some updates, content is accepted for being published. Publishing here may refer to pushing out content to others or providing digital access rights to it. Later on, content may be superseded by some other components resulting in its end of use. It is not necessary that every content passes through all the phases mentioned above. It might be possible, that a generated content is never updated or published and is simply put out of use.

So far, there are four sharing paradigms to share digital contents with other users [Belimpasakis & Walsh, 2006], which are as follows:

Showing:

Showing means displaying contents face to face, without transferring them to viewers. Major characteristic of this paradigm is that viewer does not require any special device to view the contents such as TV, mobile etc. as contents are not actually transferred to him. Showing paradigm is mostly use for sessions among family and friends.

Sending:

In sending paradigm, items to be shared are transferred from sender's device to recipient device. Common examples of sending paradigm are sending email or MMS with attachments, in which a copy of contents is sent to the recipient(s). This type of communication is most common when sender and recipients are geographically apart.

Giving:

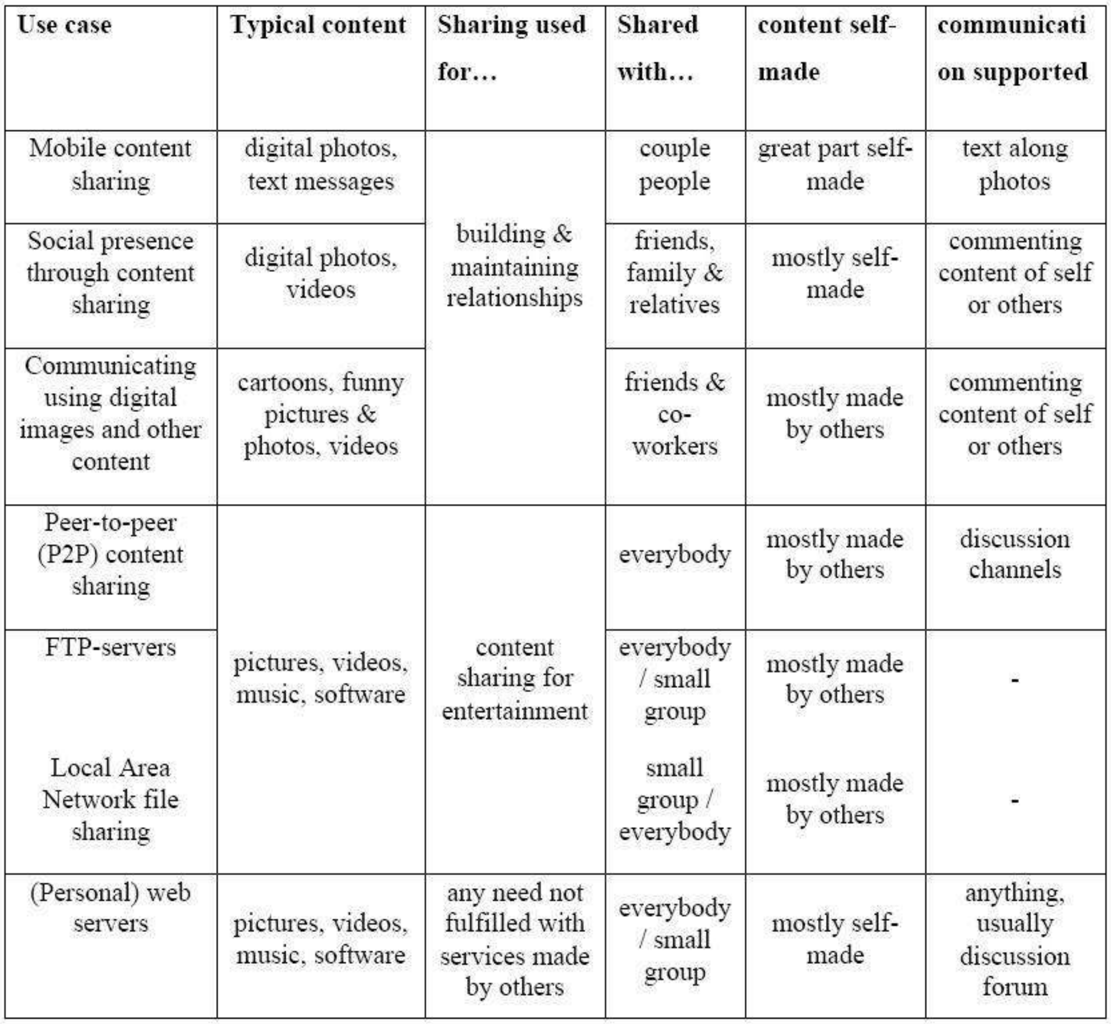
Giving paradigm refers to, face to face, handover of original/copy of contents. This paradigm usually uses short range connectivity technologies, such as WLAN or Bluetooth, and requires people to be in proximity. They most probably know each other and thus there is a relation of trust among themselves.

Offering:

Offering paradigm involves making items available face to face or remotely. Contents are not transferred to the recipient device, unless downloaded by other party and only a copy can be taken. Common examples includes offering videos and pictures on third party web servers like Flickr [Flickr, 2008], a place where other users can browse and download contents. The action of offering and downloading are asynchronous and require no personal knowledge or level of trust between two parties.

This thesis work is a hybrid combination of sending and offering paradigm, as each user's mobile device automatically sends shared contents to home PC, which can be later on offered to rest of the users in the group.

Contents that are shared among users depend on the technology they use [Hellsten, 2006]. For example, mobile content sharing usually involves sharing of text messages and images and is shared with couple of people; peer to peer content sharing usually involves sharing of images, videos and music and is shared with every body. Following table provides list of possible sharing use cases.



Use case	Typical content	Sharing used for…	Shared with…	content self-made	communication supported
Mobile content sharing	digital photos, text messages	building & maintaining relationships	couple people	great part self-made	text along photos
Social presence through content sharing	digital photos, videos		friends, family & relatives	mostly self-made	commenting content of self or others
Communicating using digital images and other content	cartoons, funny pictures & photos, videos		friends & co-workers	mostly made by others	commenting content of self or others
Peer-to-peer (P2P) content sharing	pictures, videos, music, software	content sharing for entertainment	everybody	mostly made by others	discussion channels
FTP-servers			everybody / small group	mostly made by others	-
Local Area Network file sharing			small group / everybody	mostly made by others	-
(Personal) web servers	pictures, videos, music, software	any need not fulfilled with services made by others	everybody / small group	mostly self-made	anything, usually discussion forum

Table 1: Content sharing use cases [Hellsten, 2006]

Among different content sharing types, mobile content sharing is the newest and growing fastest among them. It provides both the means of capturing and sharing contents. User creates contents using their mobile's embedded camera and shares them with friends and families either via MMS or by uploading them on third party web servers such as Flickr [Flickr, 2008]. User can also append to the pictures, metadata before sharing them with others [Hellsten, 2006].

2.2 Content Management:

As digital contents are easy to generate and share, their management is required in order to avert data misuse [Wikipedia-Content Management, 2008]. Content management is a set of processes and technologies that support the evolutionary life cycle of digital contents and may support the following processes:

- ➢ Import and creation of contents
- ➢ Identification of all key users and their roles
- ➢ Ability to track and manage multiple versions of content

Content management usually involves definition of five different user roles so that content's access rights can be managed. Description of these roles is as follows:

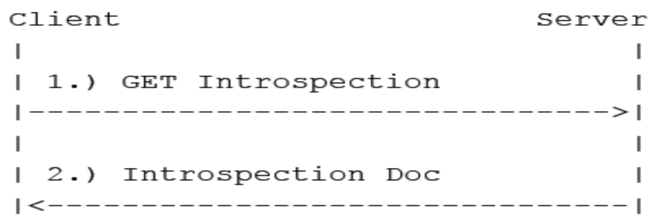
- ➢ Content: responsible for creating and editing contents
- ➢ Editor: responsible for tuning content message and may involve localization and customization
- ➢ Publisher: responsible for releasing the content
- ➢ Administrator: responsible for managing access rights for contents
- ➢ Consumer: end users, who taken in contents once they are published or shared

It is not necessary that an individual can take only one role in content management. In most cases, content creator is also the publisher and administrator.

2.3 ATOM Protocols:

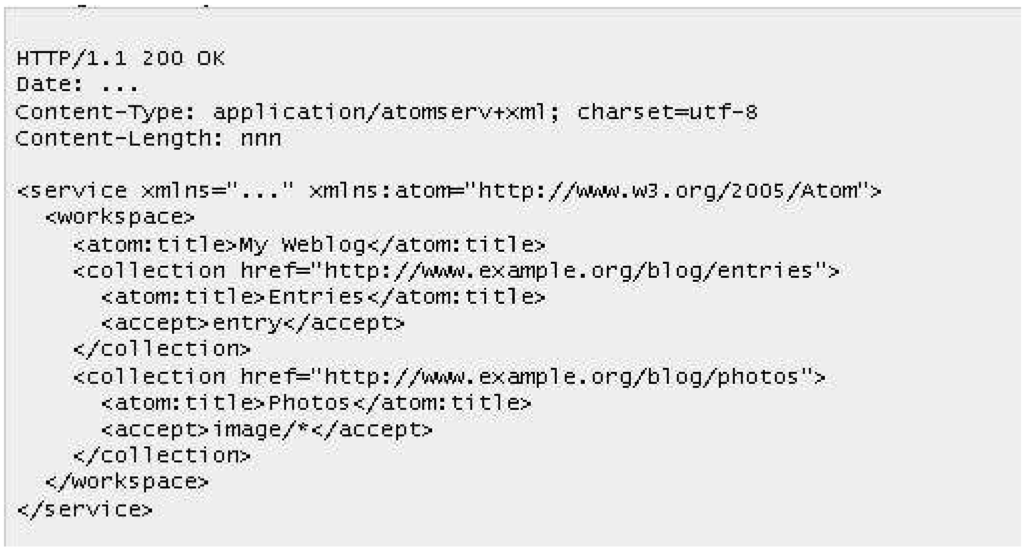
The IETF Atompub working group has created protocols for web content and metadata syndication, as well as application-level publishing and editing of web resources. The name Atom is typically used both for the "Atom Syndication Format" [Atom-Syndication Format, 2008] and "Atom Publishing Protocol (APP)" [Atom-Publishing Protocol, 2008]. Atom Syndication Format is XML-based and used for web feed description. The APP is an HTTP based protocol for creating and updating web resources. This thesis work uses Atom Publishing Protocol and Atom Syndication Format for creating and updating contents on home PC,

APP groups' resources into "collections" which are analogous to directories found in other file systems and are identified by Unique Resource Identifier (URI). The first step to using APP is to determine what collections are available and what type of resources those collections contain. This step is achieved by "Discovery" functionality provided by APP. For discovery process, Atom uses HTTP method called Get. Get is used to retrieve a representation of a known resource. In order to support collection discovery, service documents are placed on server side which represent groups of collection. During discovery process, client sends “Get Introspection request” to server and server responds by sending “Introspection Document” that enumerates the collections available to the client as shown in figures below.



```
Client                              Server
|                                        |
| 1.) GET Introspection                  |
|--------------------------------------->|
|                                        |
| 2.) Introspection Doc                  |
|<---------------------------------------|
```

Figure 1: ATOM-Requesting introspection document [Atom-Publishing Protocol, 2008]

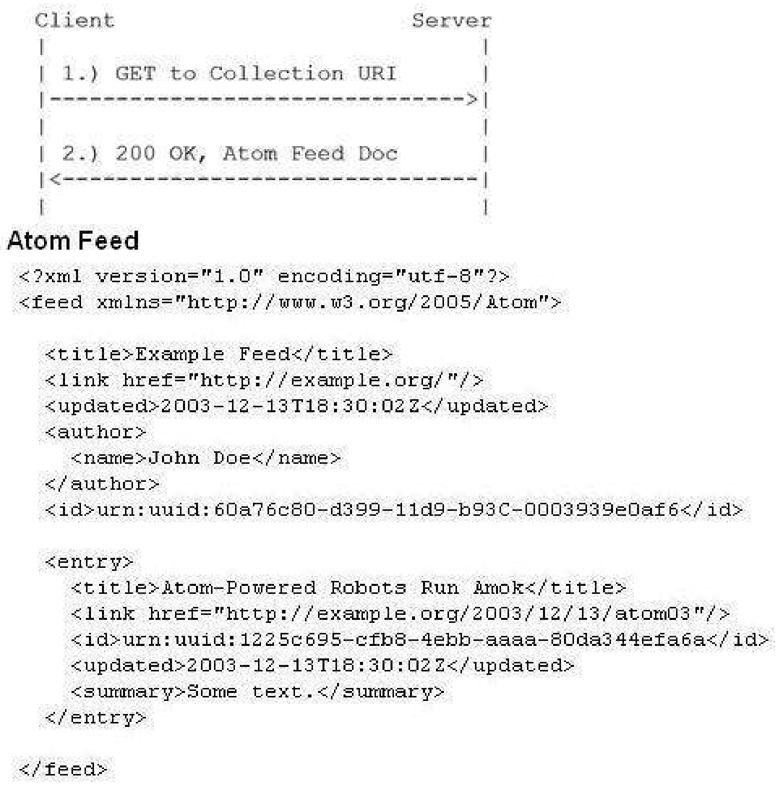


```
HTTP/1.1 200 OK
Date: ...
Content-Type: application/atomserv+xml; charset=utf-8
Content-Length: nnn

<service xmlns="..." xmlns:atom="http://www.w3.org/2005/Atom">
  <workspace>
    <atom:title>My Weblog</atom:title>
    <collection href="http://www.example.org/blog/entries">
      <atom:title>Entries</atom:title>
      <accept>entry</accept>
    </collection>
    <collection href="http://www.example.org/blog/photos">
      <atom:title>Photos</atom:title>
      <accept>image/*</accept>
    </collection>
  </workspace>
</service>
```

Figure 2: ATOM-Introspection document details [Atom enabled, 2008]

Once the client has discovered the location of a collection, it can request a listing of the collection's membership. In response to collection request, server sends Atom feed document which lists members of a collection (see figure below).



```
Client                         Server
|                                   |
| 1.) GET to Collection URI         |
|---------------------------------->|
|                                   |
| 2.) 200 OK, Atom Feed Doc         |
|<----------------------------------|
|                                   |
```

Atom Feed

```
<?xml version="1.0" encoding="utf-8"?>
<feed xmlns="http://www.w3.org/2005/Atom">

  <title>Example Feed</title>
  <link href="http://example.org/"/>
  <updated>2003-12-13T18:30:02Z</updated>
  <author>
    <name>John Doe</name>
  </author>
  <id>urn:uuid:60a76c80-d399-11d9-b93C-0003939e0af6</id>

  <entry>
    <title>Atom-Powered Robots Run Amok</title>
    <link href="http://example.org/2003/12/13/atom03"/>
    <id>urn:uuid:1225c695-cfb8-4ebb-aaaa-80da344efa6a</id>
    <updated>2003-12-13T18:30:02Z</updated>
    <summary>Some text.</summary>
  </entry>

</feed>
```

Figure 3: ATOM-Requesting collection's memberships [Atom-Syndication Format, 2008]

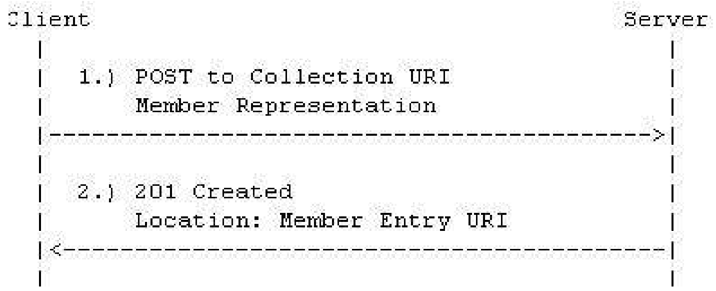
Atom feed document is composed of number of items known as entries and each entry has title, Uniform Resource Locator (URL) link, Unique ID and some other optional tags. After locating a collection and its contents, following operations can be performed on it:

- Creating a new resource
- Retrieving an existing resource
- Deleting an existing resource
- Updating a known resource

When request of above stated operations are sent to server, server responds with codes representing the status of request. Following the HTTP protocol convention, status codes of the form 2xx signal that a request was successful where as status codes of the form 4xx or 5xx signal that an error has occurred, and the request has failed.

Creating a new resource:

In order to create a new member of collection, client sends a representation of a member to the server via HTTP POST. The server responds with a response of "201 Created" and a "Location" header containing the URI of the newly-created resource as shown in figure below.

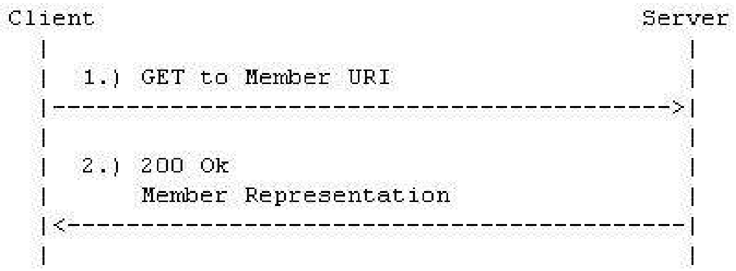


```
Client                                          Server
  |                                               |
  |  1.) POST to Collection URI                   |
  |      Member Representation                    |
  |---------------------------------------------->|
  |                                               |
  |  2.) 201 Created                              |
  |      Location: Member Entry URI               |
  |<----------------------------------------------|
  |                                               |
```

Figure 4: ATOM-Creating new resource [Atom-enabled]

Retrieving an existing resource:

Client can retrieve members of collection by sending a GET (or HEAD) request to the member's URI and server responds with an appropriate representation as shown in figure below.



```
Client                                          Server
  |                                               |
  |  1.) GET to Member URI                        |
  |---------------------------------------------->|
  |                                               |
  |  2.) 200 Ok                                   |
  |      Member Representation                    |
  |<----------------------------------------------|
  |                                               |
```

Figure 5: ATOM-Retrieving existing resources [Atom-enabled]

Deleting an existing resource:

In order to delete an existing resource, client sends a delete request along with members URI. Server responds with the status code representing the deletion process result (see figure below).

```
Client                                  Server
  |                                        |
  |  1.) DELETE to Member URI              |
  |--------------------------------------->|
  |                                        |
  |  2.) 200 OK                            |
  |<---------------------------------------|
  |                                        |
```

Figure 6: ATOM-Deleting existing resource [Atom-enabled]

Updating a known resource:

In order to update a known resource, client sends a put request along with member URI. Server responds with status code representing update process result as shown in figure below.

```
Client                                    Server
  |                                          |
  |  1.) PUT to Member URI                   |
  |      Member Representation               |
  |----------------------------------------->|
  |                                          |
  |  2.) 200 OK                              |
  |<-----------------------------------------|
```

Figure 7: ATOM-Updating known resource [Atom-enabled]

2.4 Remote access to home networks:

Due to recent trends and technologies, homes are getting equipped with various digital devices capable of creating and storing massive digital contents e.g. media server, digital frames [Belimpasakis, 2006]. Most of these devices have the capability of communicating with each other and usually remain connected to Internet 24/7. This provides the opportunity to remotely connect to home devices, over the Internet, while being away and is referred as "remote access to home networks". A simple example of remote access to home networks could be the scenario of a user remotely connecting, over his mobile device, to his home network for checking the status of the home and its devices. He could be enabled to then remotely switch devices on or off, and check the video feeds from networked security cameras.

Remote access to home networks promises very attractive scenarios but there are few challenges and problems related to *reachability, addressability and security* which holds it back from being widely available.

Reachability:

Usually Internet Service Providers [ISP] provide only one public Internet Protocol (IP) address to home customers, which turns out to be insufficient in the case when multiple devices within home are network enabled. Thus, users are forced to use Network Address Translation (NAT) on their home gateway, which then provides private IP addresses to home devices. As private IP addresses can not be routed from outside the home, all external requests need to be addresses to the home gateway, and from there forwarded to the appropriate in-home device. This requires some special administration skills from the end users, as the so-called "port forwarding" needs to be enabled and configured on their home gateway.

Addressability:

Usually, public IP addresses provided by ISP's are dynamic and might change at random time. Therefore, it is possible that while accessing home's resources remotely, public IP address of gateway changes, making it inaccessible and resulting in possible data loss. However, this problem has been addressed by the dynamic Domain Name System (DNS) solution [Vixie et al. 1997] , which makes sure that host name is always resolved to the latest known IP address of the home gateway, so that remote clients are not affected (see detailed figure below).

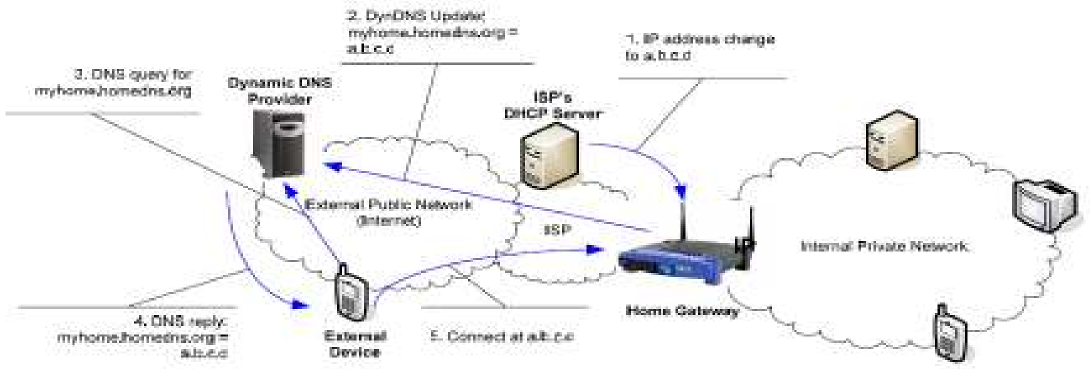

2. DynDNS Update: myhome.homedns.org = a.b.c.c
1. IP address change to a.b.c.d
3. DNS query for myhome.homedns.org
Dynamic DNS Provider
ISP's DHCP Server
External Public Network (Internet)
ISP
Internal Private Network
Home Gateway
4. DNS reply: myhome.homedns.org = a.b.c.c
External Device
5. Connect at a.b.c.c

Figure 8: DNS environment overview [Belimpasakis, 2006]

Security:

Making home network accessible remotely using public IP addresses raises security concerns. If unauthorized users could gain access to home network, they could potentially access all digital contents and could violate privacy of residents. Therefore, security issues must be dealt properly before making home network remotely accessible. Following techniques can minimize security threats:

a. **Authentication:** Users accessing home network remotely should be properly authenticated using some credentials before granting them access to home's devices.
b. **Encryption:** All data transfer between remote users and home network overlays over the public Internet. Therefore, it is important to use some encryption algorithms to assure secure data communication, such those provided by the Secure HTTP (HTTPS).
c. **Authorization:** Access to each and every home network devices should be handled individually to make sure that remote user gets access to only required devices.

2.5 Extended homes:

The term extended homes refers to family members accessing communication and media devices located at home while being away form it [Nokia-White paper, 2007].Extended environment is becoming significant with the passage of time because people today tend to be more mobile in their daily activities, both when working and in free time. This trend raises the need to have access to digital home contents 24/7, so that people can share their experiences with family members regardless of time and space. Following figure provides a clear distinction between physical and extended home.

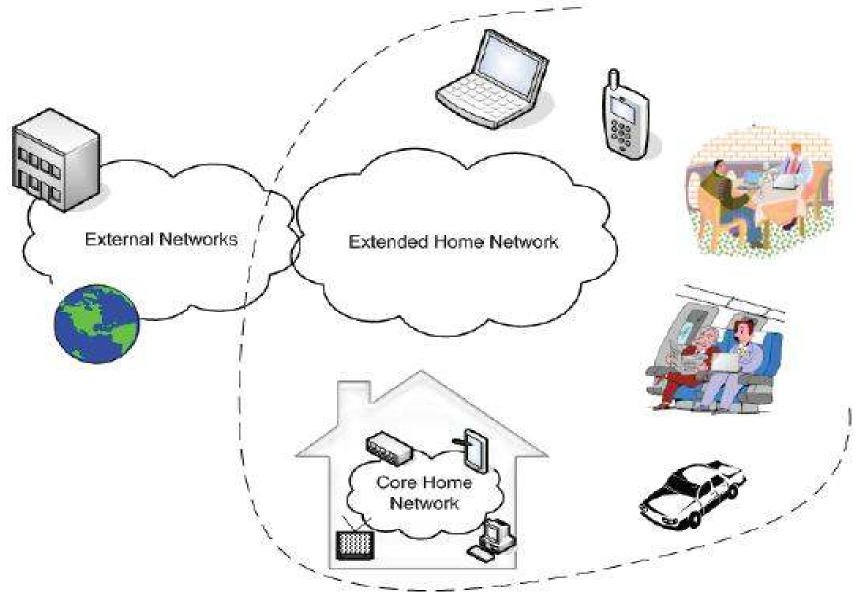


Figure 9: Physical VS extended home [Achilleopoulos et al., 2007]

A home domain can be extended either spatially or functionally. Extending home spatially means controlling and accessing home digital contents remotely while being away from it [Nokia-Extended Home, 2008]. For example, a user can connect to his/her home's media server and stream some TV episodes while being away. Extending home spatially means introducing home interface to external services.

Among other devices capable of accessing home's digital contents while being away, mobile devices can be considered as a suitable option. They have become personal devices that are always carried by users and have required capabilities to securely and remotely connect to home's digital contents.

2.6 Symbian OS:

Mobile phones have special requirements for operating system and require their operating system to run reliably for long hours [Wang, 2007]. Mobile phone users do not reboot their mobiles frequently and important information like email messages, contacts, calendar entries, call logs etc must be preserved at all costs. Fundamental features like telephony and messaging should be working properly at all times, battery consumption should be efficient, boot up time should not be longer and device needs to be responsive in all situations. Meeting these contradictory requirements can only be done if whole operating system is designed for efficiency [Symbian, 2008].

Symbian OS was one of the operating systems that were developed keeping above stated requirements in mind. The five key points – small mobile devices, mass market, intermittent wireless connectivity, diversity of products and an open platform for independent software developers – are the premises on which Symbian OS was designed and developed. This makes it distinct from any desktop, workstation or server operating system. This also makes Symbian OS different from embedded operating systems, or any of its competitors, which were not designed with all these key points in mind.

First symbian OS was developed based on Psion Software's EPOC Release 5 and did not support third party applications [Wang, 2007]. From version 6 onwards, EPOC OS was named Symbian and it supported third party applications. So far, nine versions of Symbian OS have been released and version 9.2 is the most recent version in use.

Currently, Symbian is the market leading operating system for advanced data-enabled mobile phones licensed by the world's leading mobile phone manufacturers. In 2005, Symbian had slight over half market share, followed by Linux and Microsoft Windows mobile and currently over 110 Million symbian smartphones have been shipped [Symbian, 2008].

2.7 S60 Platform

Currently there are three main platforms that run on top of Symbian OS, which are UIQ, S80 and S60. Among these platforms, S60 is the world's leading platform offering a feature-rich software base for phones with advanced data capabilities [S60 platform, 2006]. Some important services provided by S60 platform are as follows:

- Personal Information Management (PIM) application services
- Messaging application services
- Browser application services
- S60 Java application services
- UI framework services
- Location services
- Web based services

- Multimedia services
- Communication services
- S60 OS extensions
- Flash

High level architecture of S60 platform can be divided into following four categories:

- S60 platform services
- S60 application services
- S60 Java
- S60 OS extensions

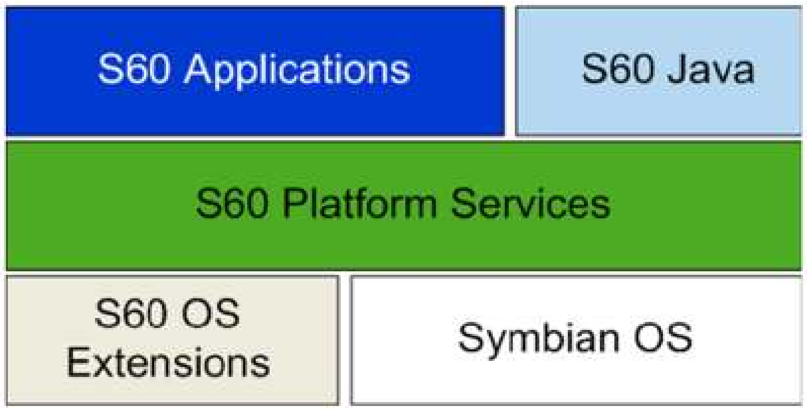


Figure 10: S60 High level architecture [Symbian, 2008]

S60 platform services:

S60 application services are those services which are in direct use of end users. Such applications include PIM, messaging, web browser etc. PIM service includes features such as phonebook, calendar, notes etc. Messaging services includes features such as Short Messaging Service (SMS), MMS etc.

S60 Java services:

S60 platform supports Java platform Micro Edition (Java ME) and complies with Java Technology for the wireless industry specification for Java technology enabled mobile phones.

S60 platform services:

Platform services include general services such as locations services, web based services, multimedia services etc. It mainly consists of engines and servers which are running at back end and assist other components of platform to run fluently.

Symbian OS extensions:

S60 extensions connect applications with device hardware functions like vibration, lights and battery charge status.

S60 Releases:

In order to introduce new features and functionalities in S60, new releases are made continuously [S60, 2008] as shown in figures below. Edition level raises contain major improvements and architectural changes. Feature Packs, on the other hand are used to introduce new features on top of existing architecture. Each feature pack also includes all the functionalities of previous feature packs. Among S60 released editions, S60 3rd Edition is the newest edition and it added new level of flexibility and security into platform. It consists of Feature Pack 1 and 2 (see figure 11). Feature Pack 1 introduced Firmware upgrade over the Air feature and new open source browser. Whereas, Feature Pack 2 extended symbian development environment with Open C and provided better multimedia performance (see figure 12).

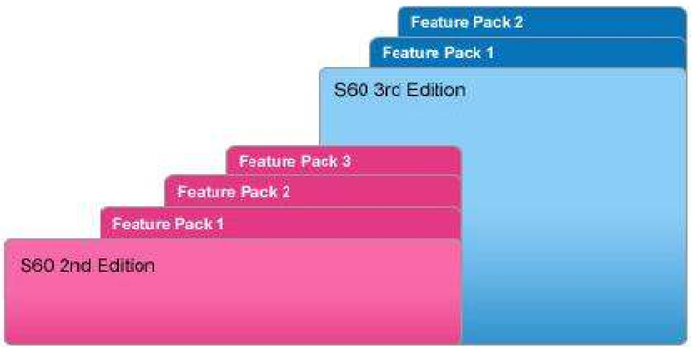


Figure 11:S60 Editions and feature packs [S60, 2008]

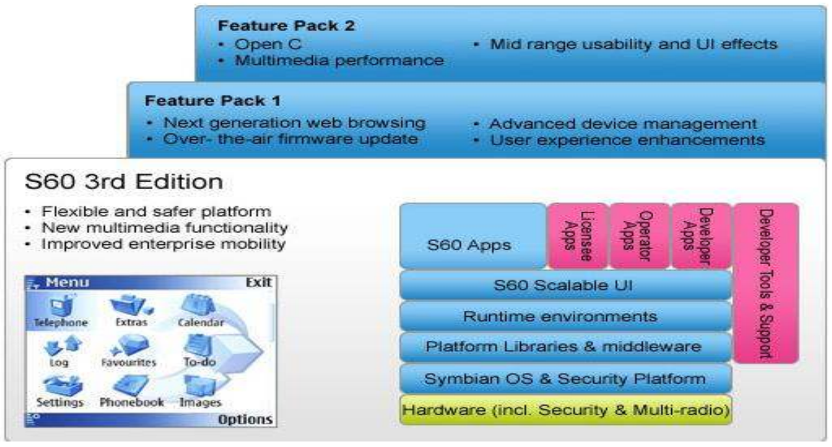

Feature Pack 2
• Open C
• Multimedia performance
• Mid range usability and UI effects
Feature Pack 1
• Next generation web browsing
• Over- the-air firmware update
• Advanced device management
• User experience enhancements
S60 3rd Edition
• Flexible and safer platform
• New multimedia functionality
• Improved enterprise mobility
Menu
Exit
Telephone
Extras
Calendar
Log
Favourites
To-do
Settings
Phonebook
Images
Options
S60 Apps
Licensee Apps
Operator Apps
Developer Apps
Developer Tools & Support
S60 Scalable UI
Runtime environments
Platform Libraries & middleware
Symbian OS & Security Platform
Hardware (incl. Security & Multi-radio)

Figure 12: S60 3rd edition features [S60, 2008]

2.8 Mobile phones as multimedia computers:

A multimedia computer is a computer with optimized high performance multimedia capabilities, allowing rich multimedia experience [Multimedia Computers Wikipedia, 2008]. True multimedia computers have high processing power, huge memory, good quality graphics cards and TV tuners. However, today high performance devices have become compact and all necessary features of multimedia computers can be found in new mobile devices like Nokia N-series. Today's multimedia computers (referring to mobile devices from here onwards) offer all functionalities of a PC and many portable single purpose devices in a connected mobile device that is always carried by user and always connected [Symbian, 2008]. As multimedia computers have programmable OS, people can create and install third applications and customize it. Some of the common features of today's multimedia computers are high quality embedded digital camera, DVD quality video recording, music and video player, full browser capabilities, blogging , internal and external (i.e. memory card) support, email access, wide range connectivity options (i.e. WLAN, Bluetooth, Infra red, UPnP), internet calls support and Global Positioning System (GPS). Earlier, consumers had to buy multiple products to acquire previously stated functionalities but today all these features are embedded in a single compact multimedia computer.

3. Existing content sharing solutions

In this chapter, existing content sharing solutions that have been developed on commercial and research level are reviewed. It explains the working of existing systems followed by personal opinion identifying their key benefits and limitations.

3.1 Flipper:

Flipper is a client-server application that provides automated content sharing on group rather than individual level and minimizes the usage barriers to sharing [Counts & Fellheimer, 2004]. Its client software runs on mobile device where as server side runs on a desktop PC. In order to share images with other people, user can create/edit a buddy list and send participation invitation to other users. After invitation is accepted, participating users are displayed as buddy tiles around the perimeter of the screen as shown in figure 13. User can navigate through buddy's photo simply by selecting his tile and his shared photos are displayed in the centre of the screen.

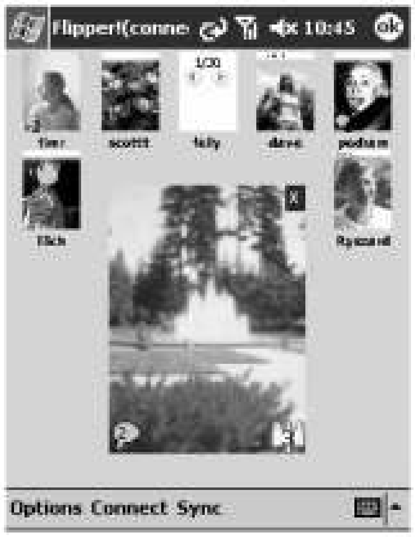


Figure 13: Flipper main screen, expanded image in centre, buddy tiles around parameter [Counts & Fellheimer, 2004]

Users can continuously update their shared folder by synchronizing it with mobile camera. After synchronization is successful, application periodically checks the files in the folder against images that have already been shared by the user. New images will be uploaded automatically in the background and will be visible to all people in the buddy list. Users can also delete images they shared removing the image from all buddies' in the list.

In order to develop the system, Structured Query Language (SQL) database and Active Server Pages were used. Active Server Pages acts as a middleware module that communicates with Flipper client and SQL database. In order to keep sharing consistent, each client device polls server after every eight minutes by sending HTTP POST request to it. The server in turn returns any new content in XML format with information specifying content structure (which contents go with which buddies etc) to the client.

In my point of view, flipper is quite useful system as it allows sharing of contents on group rather than individual level. It uses GPRS for sharing information which is useful as users do not have to be physically located together in order to share images using Bluetooth, WLAN etc like FunkyShare system [Doubell et al., 2005]. However, usage of GPRS for retrieving and viewing high resolution pictures can result in high bandwidth and battery consumption. Secondly, images of each user are located in their own album and there is no available option to create an integrated album containing all users' images. End user has to go through each and every album to view all the contents.

3.2 FunkyShare:

FunkyShare is an application software that enhances the co-located synchronous photo sharing experience as well as involved social interactions around this experience [Doubell et al., 2005]. It allows users to use PDA and mobile phones to share photographs in a co-located setting group. It consists of two parts, GUI and the backend networking functions of the application.

GUI of FunkyShare was developed using an Application Procedure Interface (API) called GAPI Draw. GAPI Draw was designed specifically for graphical applications of mobile devices. Networking backend was developed using OpenTrek, an API that plugs into GAPI Draw and adds wireless networking capabilities to the application. FunkyShare allows user to create and manage sharing session, based on Wi-Fi adhoc network. All communication packets are broadcasted to all devices that are part of sharing session and any unacknowledged packets are sent again assuring communication reliability.

GUI of FunkyShare is divided into public and private parts. Private part shows only those images that are private to device user, where as public part of GUI shows those images that are visible to all users in the session as shown in figure 14 below. In order to share a private image, user can simply drag it from private part of GUI to public part. The photograph is then shown in public space of all other devices in the session.

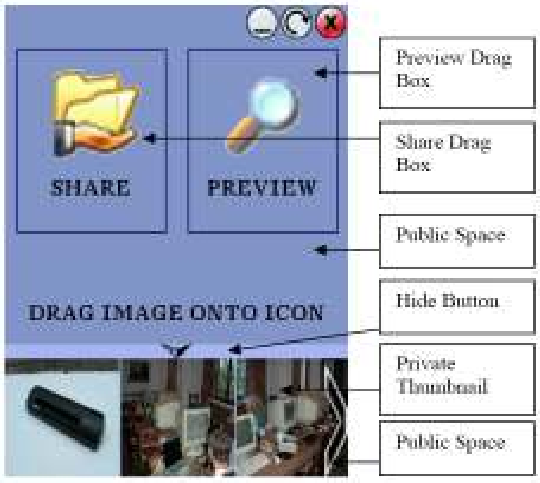


Figure 14: FunkyShare Photograph sharing screen [Doubell et al., 2005]

Application GUI also provides the feature to zoom rotate and move images, as shown in figure 15 and 16. If a user in session performs any of these stated operations on a public image, the operation is sent to all other devices that are part of session.

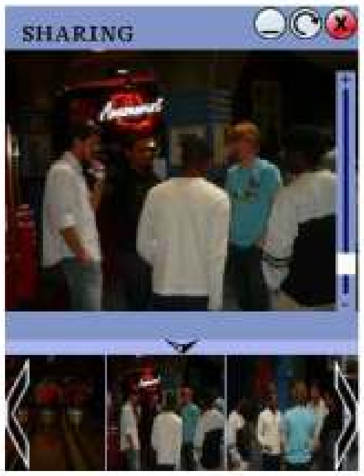


Figure 15: FunkyShare-User viewing shared images [Doubell et al., 2005]

Figure 16: FunkyShare-User zooming shared image [Doubell et al., 2005]

In my opinion, GUI of FunkyShare is user friendly and system is cost effective as it uses Wi-Fi for establishing and maintaining sharing group. However, using Wi-Fi as the only means for communication makes it unsuitable for usage in open spaces like parks, picnic spots etc. Even for indoors, users have to be always within range of Wi-Fi to be part of sharing group. Another limitation of this system is dragging an image to public UI section of image for sharing, which can become cumbersome as number of images to be shared increase.

3.3 mGroup

mGroup is a group level sharing system which tries to overcome the limitation of MMS for some phones i.e. each MMS recipient can not see the list of other recipients [Jacucci et al.,2005]. This is an important aspect in maintaining a group communication and by not having visible information about other recipients; effective group communication is not possible. mGroup tries to solve this dilemma by providing the facility of creating and sharing of "multimedia experiences" by groups at large scale events. It is based on the following principles:

- *Story based communication spaces*: Users can create media stories and can invite specific members. In the media stories, members can share collections of media items, by creating messages that open discussions, or by replying to existing messages. By using media stories, users can create different media spaces to support different discourses like in chat rooms.

- *Automatic album creation for the post-event reliving of experience*: Every media story has an independent Hyper Text Markup Language (HTML) page protected by password. Every time a message is sent to a story, sever adds the new message to the HTML page. In this way, an annotated album of group's joint experience is created and can be viewed later on.

- *Support for communication presence:* Each message that is sent on media story is delivered immediately to other story participants, who can reply with similar one to many messages. In addition, online/offline status of each member is also shown in mGroup, as shown in figure below.

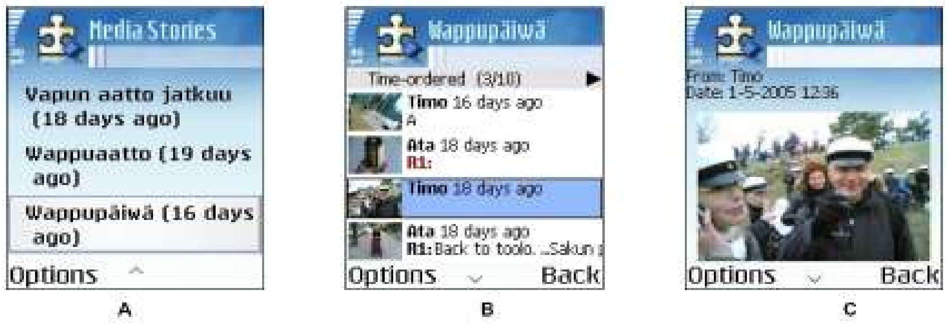


Figure 17: mGroup media story snapshots [Jacucci et al., 2005]

Figure 17 shows pieces of content on a phone screen from a small scale user trial. Figure 17A shows media story selection screen along with information about when last message was post on the story group. Choosing specific story takes user to interface 17B, which lists all messages posted in the story, ordered according to date in ascending order. By selecting specific message from the list in interface 17B, user can see the full message as shown in figure 17C. Option menu in figure 17B and 17C allows user to create his/her own messages. A message can be a reply to selected story or can instigate a new discussion story. Message is uploaded to the server in the background, from which it is distributed to other media story members.

In my opinion, mGroup can be a suitable system for group level communication as it allows group of users to participate in group discussion and also keeps track of it. However, using GPRS to post messages during outdoor activities can result in high bandwidth, processing and memory consumption, especially when message includes high resolution images.

3.4 Mobile Web Server

In latest S60 devices it is possible to install a Mobile Web Server (MWS), which provides a global URL and HTTP access to it [Nokia MWS, 2007]. MWS allows a user to create his/her own mobile web site referred as mob site, share his/her mobile contents, calendar and web applications. MWS uses application software that allows owner to manage web server's user accounts, server settings and shared folders access rights as shown in figure 18.

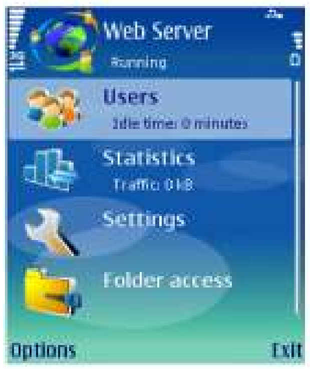


Figure 18: View of MWS application software main menu[Nokia MWS, 2007]

Owner of MWS can share the following applications with other users:

- Camera
- Blog
- Guestbook
- Send SMS
- Messaging
- Phone log
- Contacts
- Gallery

- Calendar

In order to grant access to other people to use web applications on mobile website, owner has to first create user accounts with MWS application. After accounts are created, users are granted access rights separately to each web application and some web applications also allow granting access rights to a more detailed level. Owner can also create groups of users and can grant access rights to particular groups.

Among other web applications, gallery web application allows users to view mobile albums (i.e. images, video and audio clips) of MWS owner. Gallery web application categorizes albums as private, phone and memory card. Private album is only visible to MWS owner, where as visibility of phone and memory card album to users depends on their access rights.

In my opinion, MWS is an easy to manage software as contents sharing and its administration are done on same device. However, there are certain limitations in this solution. First, once traffic limit exceeds, MWS stops working which makes its services unavailable to other users. Second, owner is charged for all users accessing its web server which results in high cost. Third, running web server on mobile device requires processing power all the time, resulting in high battery consumption [Rahmati et al., 2007]. Fourth, user can get out of memory since all contents are stored on device.

3.5 Mobilog

Mobilog is a framework that tries to resolve the dilemma of manually entering basic information in mobile blogs by partially automating multimedia blog’s data entry with context relevant annotation [Cemerlang et al., 2006]. It provides an option to automatically add metadata as part of blog such as time, place and weather at which blog’s image was created. It also provides the feature to automatically add user’s personal information such as name, birth date etc to enrich blog entry. Information such as time, place and weather can be retrieved using GPS and weather services. Personal information can be retrieved from user’s profile stored on mobile device. Following figure gives an overview of Mobilog framework.

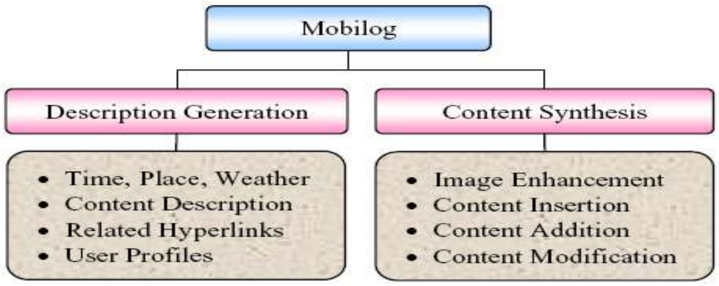
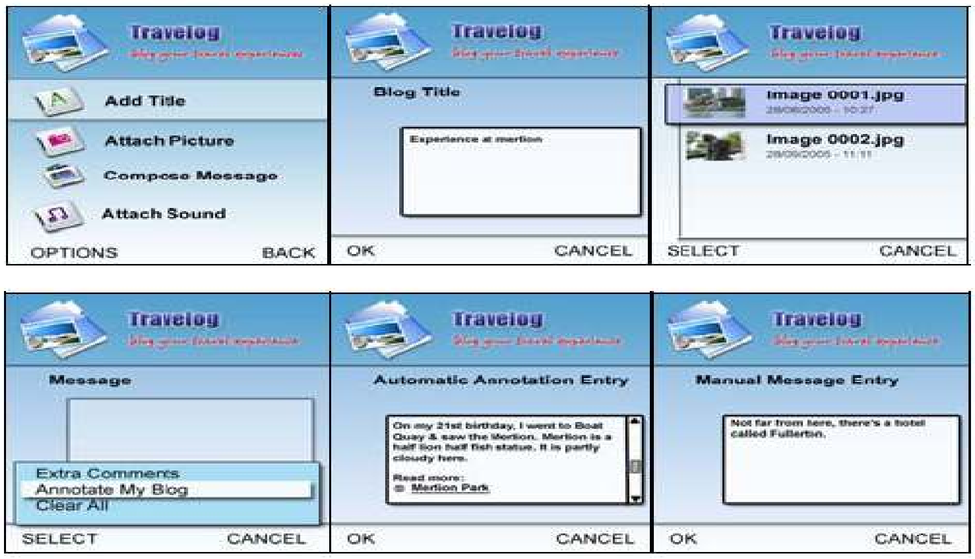


Figure 19: Mobilog framework [Cemerlang et al., 2006]

In order to utilize Mobilog framework, an application called Travelog was developed on top of it. Main objective of Travelog was to improve the blogging experience of tourists. It allows users to create blogs based on pictures, content relevant annotation and self generated text (i.e. blog title and body). Making use of SnapToTell server [Lim et al., 2004], Travelog system extracts scenery information based on captured photo and contextual information. It then extracts keywords from scenery information returned by SnapToTell server, searches for them on Google Search Engine and returns hyperlinks to related websites. The returned results and user comments are then used to create and upload a blog on user selected website such as Flickr [Flickr, 2008]. Following figures give an overview of Travelog system.



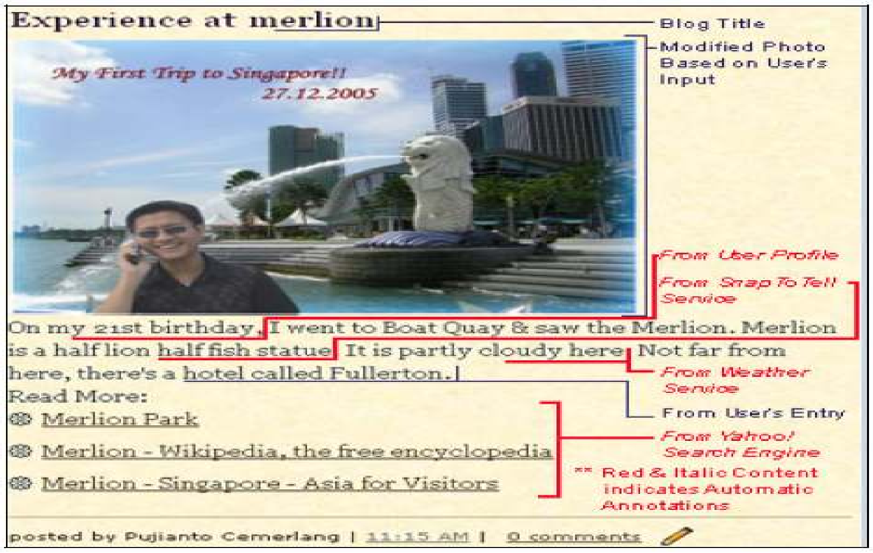
Experience at merlion

On my 21st birthday, I went to Boat Quay & saw the Merlion. Merlion is a half lion half fish statue. It is partly cloudy here.

Read more:
Merlion Park

Not far from here, there's a hotel called Fullerton.

Figure 20: Blog composition on mobile phone using Travelog [Cemerlang et al., 2006]



Experience at merlion

My First Trip to Singapore!! 27.12.2005

On my 21st birthday, I went to Boat Quay & saw the Merlion. Merlion is a half lion half fish statue. It is partly cloudy here. Not far from here, there's a hotel called Fullerton.

Read More:

- Merlion Park
- Merlion - Wikipedia, the free encyclopedia
- Merlion - Singapore - Asia for Visitors

posted by Pujianto Cemerlang | 11:15 AM | 0 comments

Blog Title
Modified Photo Based on User's Input
From User Profile
From Snap To Tell Service
From Weather Service
From User's Entry
From Yahoo! Search Engine
** Red & Italic Content indicates Automatic Annotations

Figure 21: A sample blog entry in the Travelog system [Cemerlang et al., 2006]

In my opinion, Mobilog is an intelligent system as it automatically attains required metadata and adds it to blogging text. However, there are also two limitations in this system. First, as blogs are uploaded on third party web servers, there is a threat of data misuse. Secondly, it uses GPRS to upload blog which can include high resolution images, resulting in high bandwidth, processing and memory consumption.

3.6 One Push

"One Push" is a mobile camera application which allows owners to share their mobile contents with group of users via email [Look et al., 2004]. It provides "one push sharing" feature, which sends mobile generated images to a specific group with single input. It is similar in functionality to standard camera application, but the only limitation is that email is the only means for transferring pictures of the phone. It allows user to specify three people with whom s/he want to share pictures. Names of selected people appear on the top level menu, allowing the user to select his most common recipients rather than going through recipients list each time after taking a picture (see figure below). Once picture and recipients are selected, application sends this data to server using GPRS, which in turn sends the picture as an email attachment to the selected recipient.

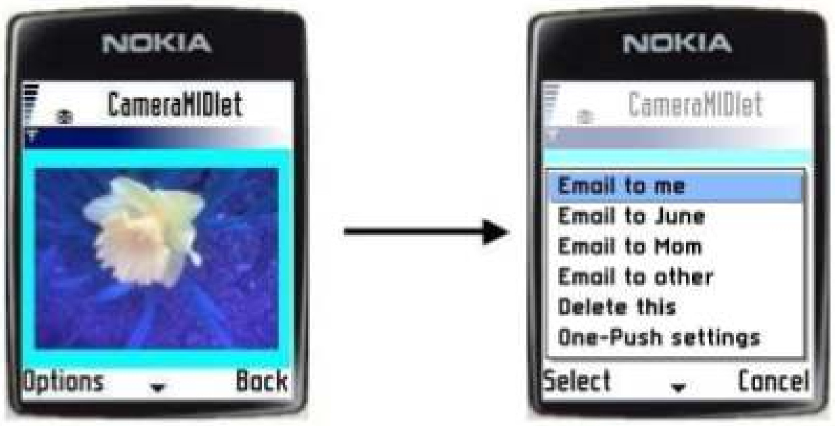


Figure 22: One Push system snapshots [Look et al., 2004]

In my opinion, there are four limitations which might prevent it from being the first choice for sharing contents. First, it uses GPRS to send image as an email attachment to selected recipients, which results in higher bandwidth and memory consumption [Pagonis, 2003]. Second, only three users can be selected directly from the menu to send email with single input, which is insufficient for people with large social circle. Third, it does not provide a feature to select multiple recipients or group of recipients for sending email at the same time, which does not make it suitable for group sharing. In case user has to share an image with N number of user, N emails have to be sent one after another requiring user intervention after sending every email. Fourth, since one image can be sent at a time, sending N number of images will require sending N number of e-mails to recipients, making it difficult to accumulate received items.

3.7 Orb

Orb software allows users to access their home computer's contents remotely on other internet connection based devices such as mobile phones [Orb, 2008]. It turns home PC into personal broadcasting machine, allowing users to create, share and enjoy their media contents at any time and at any place. Users can access home PC contents from any internet connected device such as PDAs, mobile phones, internet tablets, Play Station 3 etc. by visiting mycast.orb.com site and by providing required credentials.

Access to home PC is controlled by unique user name and password. All contents stay on Home PC and are sent using 128-bit and 256-bit encryption. Additionally, personal data is not stored but routed.

Among other sharing facilities, images can be shared with friends and families either by sending them SMS link of desired image or by sending them album invitations via email containing shared album URL. Following features related to images sharing are provided by Orb:

- No upload and download delays for sharing photos, as they are streamed directly from home PC
- It automatically checks the bandwidth, screen size and resolution of viewing device and optimizes the digital contents and speed accordingly.
- Users can easily search for photos by name
- It also provides the feature of sending photos from mobile phone to home PC.

In my opinion Orb is a useful solution as it allows users to access home PC contents while being away from it, supporting the idea of extended homes [Nokia-White paper, 2007]. However, continuous data streaming of data on mobile device results in higher data and memory consumption. Images selection and uploading to home PC has to be done manually by user and becomes cumbersome as number of contents increase.

3.8 Nokia Lifeblog:

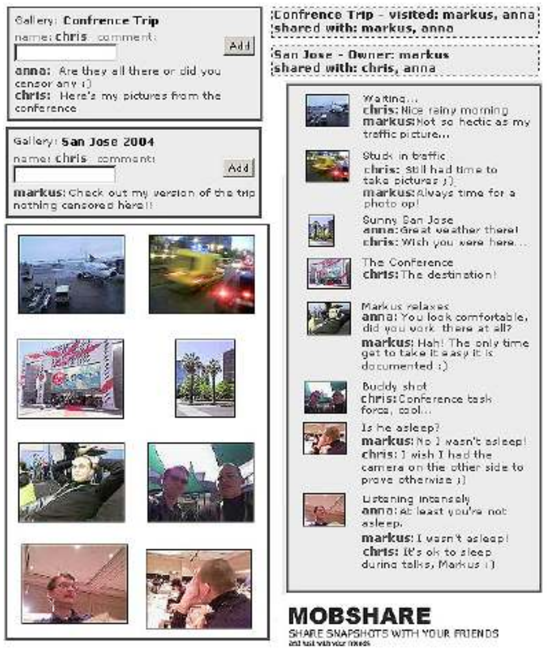
Nokia Lifeblog is a digital photo album tool designed with mobile phone photographers and bogglers in mind [Nokia Lifeblog, 2008]. Using Nokia Lifeblog, user can either create a blog on mobile and upload it directly on web using GPRS connection or can connect to his Home PC using USB or Bluetooth (see figure below) and can transfer mobile contents on it. In latter case, Lifeblog phone and PC software automatically organizes digital media between mobile phone and PC so that users can view, search, edit, and share images and messages. Sharing is provided by either sending specific contents through e-mail or by uploading them on internet so that they can be used in public.

Figure 23: Mobile connectivity with Lifeblog software[Nokia Lifeblog, 2008]

In my opinion, Nokia Lifeblog is an easy to use and useful solution. It can either be real time if contents are directly uploaded form mobile or cost efficient if contents are first transferred to home PC and then shared with friends and families. However, in first case it uses semi-trusted third party web servers such as Flick [Flickr, 2008] as contents repository and can result in data misuse. Secondly, contents to be shared are searched and uploaded manually each time, making it a laborious job as the number of contents to be shared increase. Thirdly, contents are uploaded one at a time resulting in waiting time for end user.

3.9 MobShare

MobShare is a mobile picture sharing software that enables immediate, controlled, and organized sharing of pictures along with the ability to discuss the shared pictures [Sarvas et al., 2004]. It's a client-server software, in which client side runs of mobile device where as server side software runs on a web server. Users are able to upload their captured images to an organized web album and can also arrange their order. Basic metadata (i.e. time, date and place) is automatically attained by client software and is posted along with the image on server. After image uploading is complete, user can select list of users from his contact book who can access his/album. After list of users is selected, SMS is sent to them containing invitation along with required credentials to visit the album on server. All users visiting the album of owner can provide comments on a specific image or on complete folder as shown in figure below.



Gallery: **Confrence Trip**
name: **chris** comment:
anna: Are they all there or did you censor any ;)
chris: Here's my pictures from the conference

Gallery: **San Jose 2004**
name: **chris** comment:
markus: Check out my version of the trip nothing censored here!!

Confrence Trip - visited: markus, anna
shared with: markus, anna

San Jose - Owner: markus
shared with: chris, anna

Waiting...
chris: Nice rainy morning
markus: Not so hectic as my traffic picture...

Stuck in traffic
chris: Still had time to take pictures ;)
markus: Always time for a photo op!

Sunny San Jose
anna: Great weather there!
chris: Wish you were here...

The Conference
chris: The destination!

Markus relaxes
anna: You look comfortable, did you work there at all?
markus: Hah! The only time get to take it easy it is documented :)

Buddy shot
chris: Conference task force, cool...

Is he asleep?
markus: No I wasn't asleep!
chris: I wish I had the camera on the other side to prove otherwise ;)

Listening intensely
anna: At least you're not asleep.
markus: I wasn't asleep!
chris: It's ok to sleep during talks, Markus ;)

Figure 24: MobShare snapshots [Sarvas et al., 2004]

In my opinion, MobShare is an easy to use and useful software and has acted as a baseline for recent systems related to content sharing. It provides features such as content sharing and discussion without sacrificing user's privacy. However, as images are transferred directly from mobile device, high bandwidth is consumed as it usually takes 19-30 seconds to upload single 640 X 480 pixels image. 19-30 seconds of waiting time can thwart end users as they have to wait for image uploading to proceed ahead.

3.10 Flickr:

Flickr is a popular sharing web site based on third party web servers [Flickr, 2008]. It allows users to categorize their pictures as private and public so that they can be shared with others accordingly. If a picture is marked as public, it can be viewed by every visitor. If a picture is marked as private, user gets some further more options to either make it visible to just himself, or also to his friends. It also allows users to categorize their pictures with tags, which makes them ease to search. Tags could be name, place, some event etc.

Users of new Nokia N-series mobile devices such as N95 etc can directly upload their mobile images on their Flickr accounts. This feature allows them to share their feelings with friends and families while on the move. Users simply have to configure Flickr credentials on their mobile once and later can use those credentials to post contents. However, uploading contents directly from mobile device results in higher bandwidth consumption and this software is most suitable for people having flat rate mobile connections.

3.11 Conclusion

In this chapter, we reviewed ten systems which provide content sharing for individual or group of users. All systems suffer either from high bandwidth consumption or data privacy threat or user intervention during whole process. None of these systems provide combined features such as privacy through storage in a trusted machine, easy “one click” invitation and participation in the group, automatic content upload in background when WLAN connection is available and automatic content offering to all participants in the session. Next section presents a solution that provides group content sharing in secure and cost effective way without user intervention during the whole process

4. Presented Solution

4.1 Overview:

Automated Content Sharing in Extended Homes (ASCEH) system is a client-server solution, in which client software runs on mobile device whereas server software runs on home PC [Awan et al., 2007] ;[Belimpasakis et al., 2008b]. Main objective of this solution is to provide cost effective, automated and secure content sharing among group of users. This solution principally automates mobile content sharing by reusing users' smart home resources, such as storage and Internet connection, without other parties' interventions or risks that might lead to contents misuse. Thus, by reusing currently available Information and Communication Technology (ICT), it creates an enhanced facility and subsequent service that will empower user groups to act on their needs at no extra cost for additional equipment.

It allows group of mobile devices to create an alliance for the sake of content sharing. One of the users (*referred as primary user from here onwards*) among the group can create a common sharing session on his mobile device for pre-defined amount of time. After session creation, its participation invitation can be sent to other users (*referred as secondary user from here onwards*) using SMS. Sharing session starts on primary user's mobile device after invitations are sent. The invitation for participation also includes credentials and description on how to remotely access primary user's home PC (see figure below).

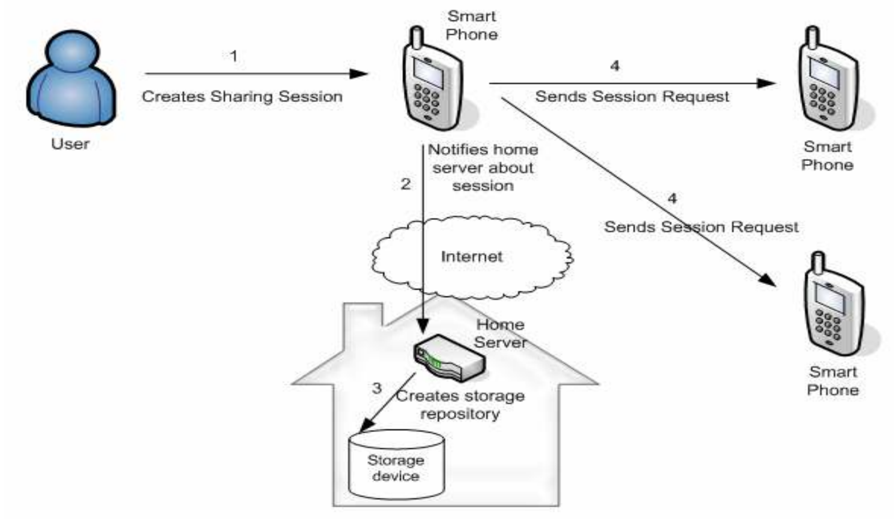


Figure 25: ACSEH-Initiating session creation and sending invitations for participation in group content sharing [Awan et al., 2007]

Sharing session starts on secondary user's mobile device after invitation is accepted and is independent of invitation acceptance/rejection by other users in the group. All contents created during sharing session are automatically marked for sharing on primary and secondary user's mobile device. After sharing session is over, mobile devices of all users in the group connect to Home PC of primary user either via GPRS or WLAN, depending on specified configurations for session. After connection is established, all contents created during sharing session are automatically transferred to primary user's home PC, as shown in figure below:

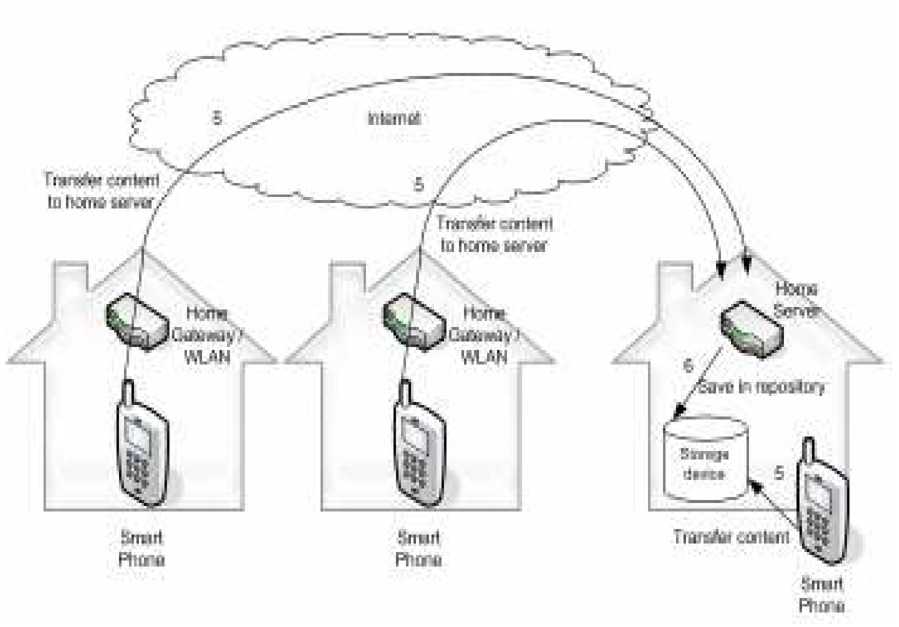


Figure 26: ACSEH-Remotely connecting to primary user's home and transferring contents [Awan et al., 2007]

4.2 Detailed description

This subsection provides detailed theoretical and technical description of ASCEH system. It specifies pre-requisites necessary for ACSEH system's working and description of six steps carried out during content sharing (figure 25 & 26).

Pre-requisites for ASCEH system:

Following pre-conditions should be met to assure successful group content sharing:

I. Home PC is up and running
II. Server software is running properly on home PC
III. Sufficient space exists on home PC to store shared contents
IV. Content sharing account is already created on primary user's mobile device

First three conditions are simple and do not require further description, so only fourth condition will be discussed in details. ACSEH requires primary user to have a valid sharing account on his mobile device before starting group sharing session. In order to create a sharing account, primary user uses server software to create a unique 8-digit pin code. After successful generation of pin code, primary user inputs it to client software running on mobile device (see figure below). After reception of pin code, client software connects to home PC using GPRS or WLAN and creates an account on it.

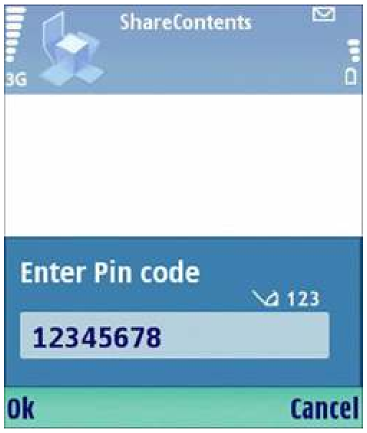


Figure 27: ACSEH-creating sharing account on home PC using 8-digit pin code [Belimpasakis et al., 2008b]

Sharing account remains valid for lifetime and can be used any time to connect to home PC. It should be noted that only primary user's account creation requires direct contact with home PC. Accounts for secondary users are created by primary user's mobile device using its sharing account and remote connection with home PC.

Step 1: Creating sharing session on primary user's mobile device

Content sharing on group level in ASCEH system starts with creation of common sharing session on primary user's mobile device. Primary user specifies sharing session name, duration, means of contents transfer (i.e. GPRS or WLAN), access point and list of session invitation recipients as shown in figures below. Duration of sharing session can range from one minute to 99 hours. Recipients are selected from contact book of primary user's mobile device.

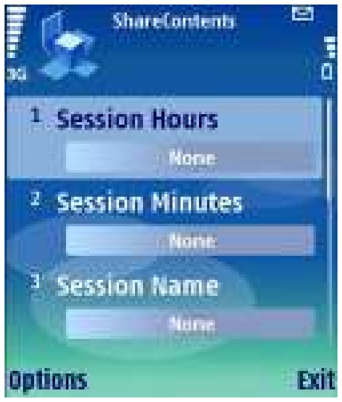


Figure 28: ACSEH-Providing sharing session details [Belimpasakis et al., 2008b]

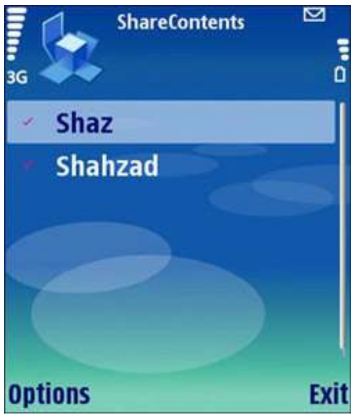


Figure 29: ACSEH- and selecting list of recipients from contacts list [Belimpasakis et al., 2008b]

Step 2: Notifying home PC about sharing session

After primary user provides new session details on his mobile device, it remotely connects to home PC using sharing account. It provides sharing session information to home PC and requests it to create accounts for recipients. After sharing accounts are created, primary user's mobile device retrieves them and appends them to sharing session invitation along with other information.

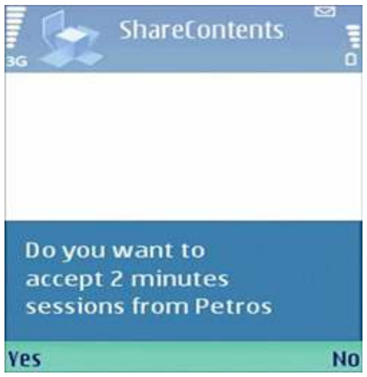
Step 3: Creation of storage repository on home PC

When primary user’s mobile device remotely connects to home PC during step 2, it provides sharing session name along with other details. Home PC creates a directory with this name and automatically redirects all sharing session contents to this directory. Storing session contents in one directory provides ease of content management.

Step 4: Sending session request to recipients

After successful completion of step 2 and 3, primary user’s mobile device sends session invitation to recipients using low level messaging APIs in background. Session invitation is encrypted using ceaser cipher and is in the format “*$$$_Auto_$$$Share_#SessionTime # SessionName#8-digit pin code”*. Using ceaser cipher technique was just an indication that session invitation can be encrypted and stronger encryption technique like triple DES should be used in real life. After session invitations are sent, sharing session starts on primary user’s mobile device for specified time. All images created during sharing session are automatically marked for sharing and are supposed to be transferred to home PC after session expires.

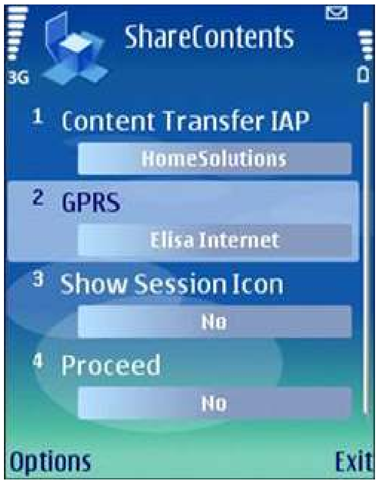
When sharing session invitation is received at recipients end, ACSEH client software detects it and requests secondary user to either accept or reject sharing session, as shown in figure below:



Do you want to accept 2 minutes sessions from Petros

Figure 30: ACSEH-Sharing session notification at recipients end [Belimpasakis et al., 2008b]

Selecting ‘yes’ means secondary user provides his consent to be a part of sharing session for specified time. After accepting session invitation, client software requests secondary user to select Internet Access Protocol (IAP) and GPRS connection from list of configured access points as shown in figure below. GPRS connection is used to remotely connect to primary user's home PC to create a sharing account using 8-digit pin code, which was part of session invitation. Selected IAP can vary from GPRS to WLAN and is used for actual transfer of contents. Client software also provides an option to show/hide sharing session icon used to indicate that sharing session is still active, as shown in figure below.


ShareContents

1 Content Transfer IAP
HomeSolutions

2 GPRS
Elisa Internet

3 Show Session Icon
No

4 Proceed
No

Figure 31: ACSEH- IAP and GPRS selection at recipients end [Belimpasakis et al., 2008b]

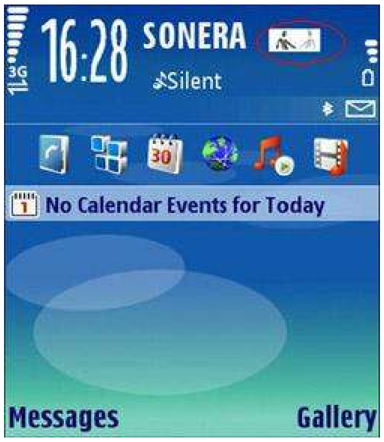


Figure 32: ACSEH-Displaying sharing session icon to indicate that sharing session is still active [Belimpasakis et al., 2008b]

After providing required details, sharing session starts on secondary user's mobile device and all contents created during session are automatically marked for sharing.

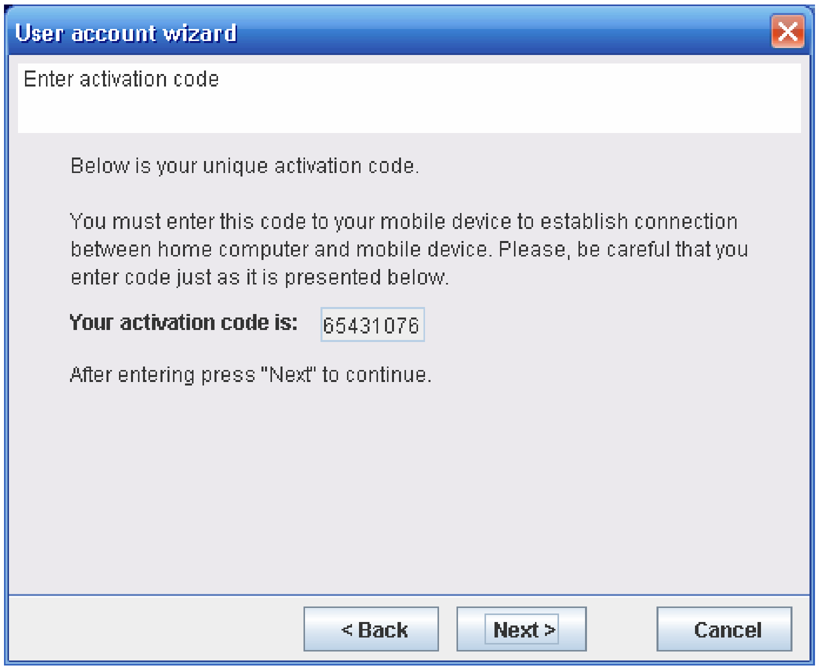
Step 5&6: Transferring contents to home server and saving them in repository

After sharing session is over on primary and secondary user's mobile device, it identifies images that were created during session and creates a list of it. It user has selected GPRS from list of IAP's as mean of content transfer during session creation or reception, contents are transferred immediately after session expiration to primary users home PC. Immediate transfer of contents makes system real time. However, if user has selected specific WLAN as means of content transfer, then mobile device continuously polls for it in background. Once WLAN is found, it automatically connects to it and transfers shared contents to home PC of primary user. Using WLAN as an IAP makes system cost effective but reduces battery life due to continuous polling for selected WLAN.

4.3 Client and server software description:

Server software:

Server side software for ACSEH system was developed using Java language. It provides secure remote access to home PC contents using ATOM protocol. It provides feature of creating/managing sharing session accounts as shown in figure below:



User account wizard

Enter activation code

Below is your unique activation code.

You must enter this code to your mobile device to establish connection between home computer and mobile device. Please, be careful that you enter code just as it is presented below.

Your activation code is: 65431076

After entering press "Next" to continue.

Figure 33: ACSEH-Creating 8-digit pin code for client software

During account creation process, it creates a unique 8-digit pin code which must be used by client software to create an account on it. For security reasons, generated pin code expires after two hours time period and must be used before that time.

After account is created successfully, shared folders can be assigned to it (see figure below). Client software can remotely connect to server and can add/remove contents from these shared folders.

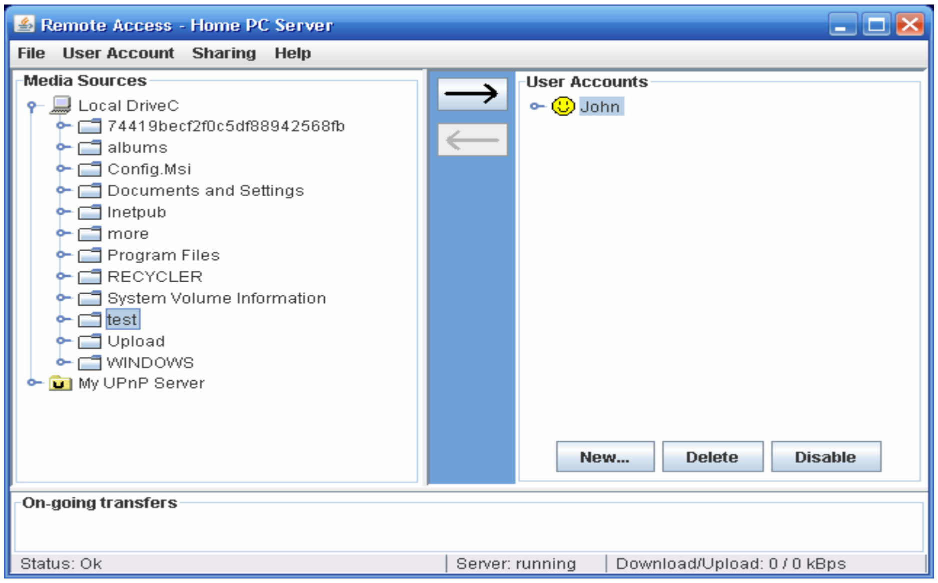


Figure 34: ACSEH-Assigning shared folders to users

Server side software was already developed at Nokia Research Center and required minor modifications to make it compatible with client software. Main implementation part of thesis work was to develop the client software for content sharing system (the S60 client), as the server side software was provided to me ready for usage.

Client software:

Client side software for ACSEH system was developed on Symbian OS, S60 3.1 platform using Codewarrior as an IDE. It was developed on top of sharing middleware, that allows 3rd party application developers to create applications with sharing capabilities, while being agnostic of lower level sharing technologies and protocols [Belimpasakis et al., 2008a]. It introduced a new layer, between applications and content sharing protocol libraries, which provides a generic API to applications, for using any kind of content transfer protocol, as shown in figure below:

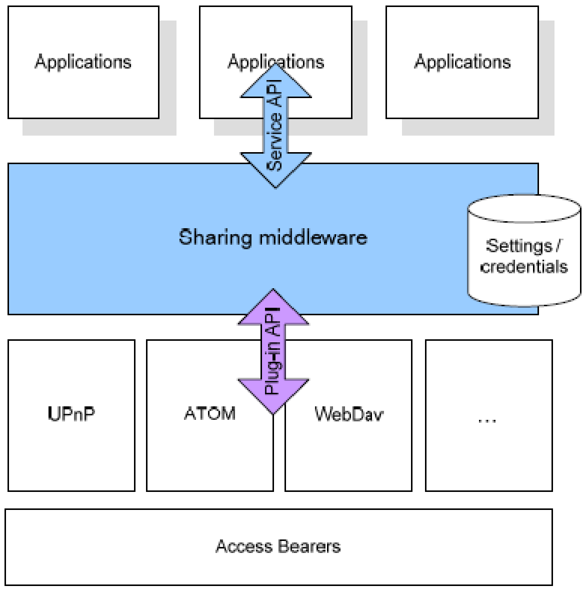


Figure 35: ACSEH-Sharing Middleware simplified design [Belimpasakis et al., 2008a]

Middleware provides list of sharing accounts to application and user can decide which sharing account can be used to transfer contents. The Sharing Middleware handles the lower level content transfers, and on top of it S60 sharing client was developed. Some small parts of sample software code were reused, but most of the application code was developed from scratch. The development followed an iterative approach. It started with the development of a group content sharing solution using Flickr [Flickr] as contents repository, instead of the home PC that was our final target. This system used GPRS to transfer contents to user's Flickr account. After testing of system's functionalities, it was further extended to introduce WLAN for contents transfer. WLAN usage made the system cost effective as system used to wait for WLAN availability in its environment to transfer contents. After system's testing, it was used as a baseline to develop ASCEH system using home PC as contents repository and WLAN or GPRS as means of contents transfer. Middleware's remote access plug-in was used for remotely connecting to home PC and accessing user's folders.

4.4 Real life use case:

Four friends go out for skiing and intend to share their captured moments. They all use their camera phones to capture images for five hours and then go back home. By the time they get back home, moment of excitement is over and they are bored to share their contents. May be, one of them send images via e-mail, one of them uploads on his Flickr account. Due to usage of different content repositories, all captured contents will be scattered and friends will not be able to view all contents.

A good alternative to this dilemma can be usage of ACSEH system. Moving back to the moment when all friends meet for skiing, they can all use ACSEH client to automatically share their captured contents. Assume that one of them creates a common sharing session for 5 hours, names it Skiing_07 and selects home WLAN as an IAP for contents transfer. Client software notifies home PC about sharing session. Home PC allocates resources for this session, creates a directory called Skiing_07 and generates accounts for session invitation recipients. Then an invitation to participate is easily sent to rest of the friends using SMS as shown in figure below.

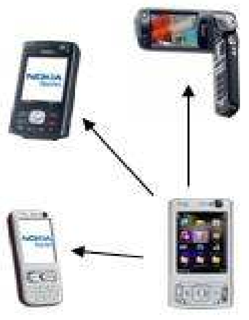


Figure 36: ACSEH-Sending session invitation to friends

All friends accept the invitation and select their respective home WLAN as mean of content transfer. After invitation reception, all contents created during next 5 hours will be automatically marked for sharing. At the end of the day, everybody goes back to their homes. When mobile devices detect home WLAN, they automatically establish a connection to the home server of primary user, using the details included in the original invitation. After connection is established, all shared contents are transferred to a common location using free home connection as shown in figure below.

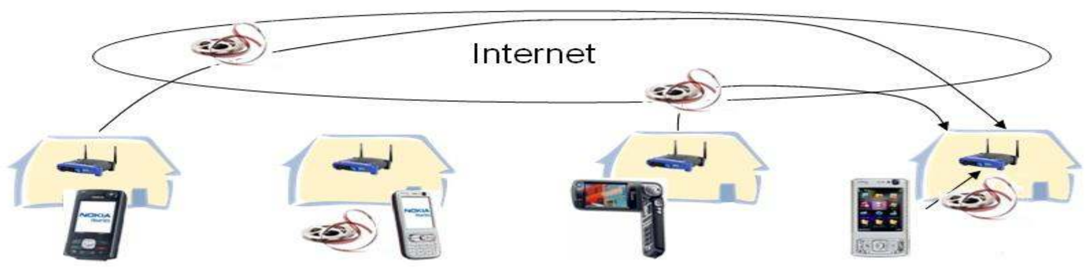


Figure 37: ACSEH-Connecting to primary user's home PC to send contents generated during sharing session

4.5 Comparison with existing systems:

Following table compares ACSEH system with existing content sharing solutions Comparison is done on the basis of cost effectiveness, data privacy, metadata usage, ease of use and content sharing level (i.e. single/group). It highlights the pros and cons of content sharing solutions discussed in details in section 3 & 4.

	Uses third party web server for storing contents	Cost effective	Data privacy	Requires contents to be moved on PC for sharing	Uses Metadata	Shared data of multiple users can be automatically accumulated at single point	Easiness of usage
MobShare	Yes	No	Yes	No	Yes	No	Yes
Nokia LifeBlog used with PC album	No	Yes	Yes	Yes	Yes	No	Yes
Nokia LifeBlog used with online blogging	Yes	No	No	No	Yes	No	No
Flipper	Yes	Yes	Yes	No	No	No	Yes
mGroup	Yes	No	No	No	Yes	Yes	Yes
Mobile Web Server	No	Yes	Yes	No	No	No	No
One Push	No	No	Yes	No	No	No	Yes
Orb	No	No	Yes	Yes	No	No	Yes
ACSEH System	No	Yes	Yes	Yes (transparently to the user)	No	Yes	Yes
Flickr	Yes	N/A	No	No	Yes	Yes	N/A

Table 2: ACSEH system comparison with existing content sharing systems

5. Future work and conclusion:

5.1 Future work:

ACSEH system is a prototype and should be considered as a baseline for future content sharing solutions, rather than a final solution. It provides sufficient grounds for creating improved content sharing solutions and there are many directions towards which future work can be directed. Some of possible extensions to existing work are as follows:

Allowing multiple sharing sessions:

Currently, ACSEH system provides only one active sharing session option. If a user accepts a new sharing session invitation while previous sharing session is active, previous session will be terminated and a new session will be started. In order to maintain multiple sharing sessions, modifications and feature extensions will be required at client side only, as server side already supports multiple sharing sessions.

Providing multiple options for sending/receiving session invitation:

In ACSEH system, session invitations are sent/receive using SMS. Usage of SMS is valuable if primary and secondary users are not geographically located together. However, its usage results in increased cost if users are located in near distance. Therefore, it is important to provide multiple means of sending/receiving sharing session invitation. Using Bluetooth can be a better option if users are present in 10 meters distance range.

Using multiple contents repositories:

Currently, ACSEH system uses only one home PC as a contents repository. Using only one machine as contents repository can result in data loss if it shuts down or gets out of space. Therefore, there is a need to extend and modify client software, so that it can use multiple machines as contents repository.

Sharing multiple content types:

During special occasions, people not only capture images using their mobile phone's embedded cameras but also generate audio and video files and want to share them as well with friends and families. ACSEH system provides the feature of sharing images only generated during sharing session. However, this functionality can be easily extended to share audio and video files as well. Minor modifications will be required at client side as server side already supports multiple files format.

Appending metadata to contents:

Currently, ACSEH system does not append any metadata to the contents. Location, time and name of participants can be acquired using GPS and Bluetooth technology and can be appended with the content.

5.2 Conclusion:

This thesis work specifies the importance of content sharing in day to day life. It highlights the features and limitations of existing sharing solutions with respect to current need of users.

One of the main outcomes of thesis work is an automated, secure and cost effective content sharing solution using home Infrastructure for content storage. It removes hurdle of manual sharing and creates a seamless network of users which can be friends, family members etc. Using existing home infrastructure, users can be in total control of their content even when sharing it without relying on semi-trusted third party web servers. Taking into consideration the social aspects of sharing and the privacy concerns of user, the presented system is a solution to real world problem that people face daily. It significantly improves user experience, as it minimizes the required actions for sharing content to a simple acceptance of an invitation. This allows the users to enjoy the moment while the system takes care of shared contents in an ambient way.

Using presented system as baseline, further research and development can be done to answer some open questions such as implications of having multiple sharing sessions active at the same time. From technical point of view, running home PC all the time to act like a server is not realistic. However, it should be considered as an intermediate step to a more robust and ambient architecture.

References:

[Awan et al., 2007] Awan A, Belimpasakis P, Berki E, Walsh A, Automated Content Sharing in Extended Home through Mobile Devices - A quality solution for group communication. *Proceedings of Berki et al. (EDs) BCS Software Quality Management XV. Software Quality in the Knowledge Society. pp. 381-38.* Tampere, Finland, August 2007,

[Achilleopoulos et al., 2007] N. Achilleopoulos, F. Alvarez, P. Belimpasakis, P. Daras, C. Guerrero Lopez, I. Laso, M. Pelt, and J.C. Point (Eds.), "User Centric Media - Future and Challenges in European Research", Coordinated by Networked Media Unit of the DG Information Society and Media of the European Commission, November 2007. ISBN 978-92-79-06865-2.

[Atom-Syndication Format, 2008] http://www.ietf.org/rfc/rfc4287.txt

[Atom-Publishing Protocol, 2008] http://www.rfc-editor.org/rfc/rfc5023.txt

[Atom enabled, 2008] http://www.atomenabled.org/developers/protocol/#whatIsAtom

[Belimpasakis et al., 2008a] P. Belimpasakis, J-P. Luoma and M. Börzsei, "Content Sharing Middleware for Mobile Devices". *Proceedings of the First International Conference on MOBILe Wireless MiddleWARE, Operating Systems, and Applications*, Innsbruck, Austria ,February 2008.

[Belimpasakis et al., 2008b] P. Belimpasakis, S. A. Awan, E. Berki, "Mobile Content Sharing Utilizing the Home Infrastructure"-pending. *Proceedings of the 2nd IEEE Conference and Exhibition on Next Generation Mobile Applications, Services and Technologies*, Wales, UK, September 2008.

[Belimpasakis & Walsh, 2006] Belimpasakis P, Walsh R, User Created Content in the Extended Home. *Proceedings of the 15th IST Mobile & Wireless Communication Summit.* Myconos, Greece, June 2006,

[Battarbee & Kurvinien, 2003] Battarbee,K., Kurvinien,E. “Supporting creativity – co-experience in MMS”. *Proceedings of COST269*, Helsinki, Finland, 2003.

[Belimpasakis, 2006] Petros Belimpasakis, Remote Access to Home Services Utilizing Dynamic DNS and Web Technologies. Tampere University of Technology, MSc Thesis, November 2006. Also available as http://research.nokia.com/files/homedns_thesis.pdf

[Counts & Fellheimer, 2004] Scott Counts, Eric Fellheimer, Supporting Social Presence through Lightweight Photo Sharing On and Off the Desktop.

Proceedings of the ACM CHI 2004 Conference, Vienna, Austria, April 24-29 2004

[Cemerlang et al., 2006] Pujianto Cemerlang, Joo-Hwee Lim, Yilun You, Jun Zhang1 and Jean-Pierre Chevallet, "Towards Automatic Mobile Blogging". *Proceedings of Multimedia and Expo IEEE conference.* Toronto, July 2006

[Camera Phone sales, 2008] http://www.3g.co.uk/PR/April2005/1336.htm

[Doubell et al., 2005] Dane Doubell, Philip Arkcoll, Gary marsden and Dynal Patel, "Co-Located Photo Sharing on Mobile Devices", October 2005.

[Flickr, 2008] www.flickr.com

[Gossweiler & Joshua Tyler, 2004] Rich Gossweiler & Joshua Tyler, "PLOG: Easily Create Digital Pictures Stories through Cell Phone Cameras". *Proceedings of International Workshop on Ubiquitous Computing.* Porto Portugal April 13-14, 2004.

[Hellstén, 2006] Tuomas Hellstén, Seminar on Multimedia Content Sharing, Helsinki University of Technology, 2006.

[Jacucci et al., 2005] Giulio Jacucci, Antti Oulasvirta, Antti Salovaara, Risto Sarvas, "Supporting the Shared Experience of Spectators through Mobile Group Media", *Proceeding of Group 2005 Sanibel Island, Florida, USA, ACM Press November 6*

[Jans et.al, 2007] Greet Jans, Jeroen Vanattenhoven & David Geerts,"Social Requirements for Shairng Information and Experiences". *Proceedings of Computer Human Interaction CHI-2007.* California, 2007.

[Lim et al., 2004] J.H. Lim, J.-P Chevallet, & S.N. Merah, "SnapToTell: ubiquitous information access from camera. A picture-driven tourist information directory service". P*roceedings of Mobile Human Computer Interaction -HCI 2004*, pp. 21-27, 2004

[Look et al., 2004] Gary Look, Robert Laddaga, and Howard Shrobe, "One-Push Sharing: Facilitating Picture Sharing From Camera Phones". *Proceedings of Mobile human-computer interaction- Mobile HCI.* Glasgow, September 2004.

[Multimedia Computers Wikipedia, 2008] en.wikipedia.org/wiki/Multimedia_computer

[Nokia-White paper, 2007] NOKIA, Bringing mobility to homes short paper

[Nokia MWS, 2007] Mobile Web Server: Technology White Paper

[Nokia-Extended Home, 2008] http://research.nokia.com/extendedhome/index.html

[Nokia Lifeblog, 2008] http://europe.nokia.com/photos

[Orb, 2008] http://www.orb.com/orb/

[Pagonis, 2003] John Pagonis, "While paper: GPRS considerations for mobile email", 2003

[Rahmati et al., 2007] Ahmad Rahmati, Angela Qian, and Lin Zhong, "Understanding Human-Battery Interaction on Mobile Phones". *Proceedings of Mobile HCI'07,* 2007.

[Sarvas, 2004] Sarvas R, Media Content Metadata and Mobile Picture Sharing. *Proceedings of the 11th Finnish Artificial Intelligence Conference STeP 2004*. Vantaa, Finland, September 1-3.

[Sarvas et al., 2004] Risto Sarvas, Mikko Viikari, Juha Pesonen, and Hanno Nevanlinna, MobShare: Controlled and Immediate Sharing of Mobile Images. *Proceedings of MM'04*, New York, USA, October 10–16, 2004.

[Spangler 2006] W. Spangler, S. Hartzel, G. Mordechai, "Exploring the privacy implications of addressable advertising and viewer profiling", Communications of the ACM, Volume 49, Issue 5, May 2006.

[S60 Platform, 2006] White paper: S60 Platform: Basics, Nokia, October 16th, 2006

[Symbian, 2008] www.symbian.com

[S60, 2008] www.s60.com

[Vixie et al. 1997] Vixie P., Thomson S., Rekhter Y., Bound J.,"Dynamic Updates in the Domain Name System (DNS Update)", RFC 2136, IETF, April 1997

[Wang, 2007] Kui Wang, Post-mortem Debug and Software Failure Analysis on Symbian OS, University of Tampere, Department of Computer Science, MSc Thesis, January 2007. Also available as http://tutkielmat.uta.fi/pdf/gradu01561.pdf

[Wikipedia-Content Management, 2008] en.wikipedia.org/wiki/Content_management

Appendix 1: List of Acronyms and abbreviations

ACSEH	Automated Content Sharing in Extended Homes
APP	Atom Publishing Protocol
HTTP	Hyper Text Transfer Protocol
UPnP	Universal Plug and Play
GPS	Global Positioning System
IDE	Integrated Development Environment
URI	Unified Resource Identifier
URL	Uniform Resource Locator
ISP	Internet Service Provider
DNS	Domain Named System
MMS	Multimedia Messaging Service
SMS	Short messaging service
GPRS	General Packet Radio Service
IAP	Internet Access Point
PIM	Personal Information Management
API	Application Procedure Interface
HTML	Hyper Text Markup Language
SQL	Structured Query Language
WLAN	Wireless Local Area Network
XML	Extensible Markup Language
OS	Operating System
PC	Personal Computer
IP	Internet Protocol
S60	Series 60

Appendix 2: System Design Document

Class Diagram:

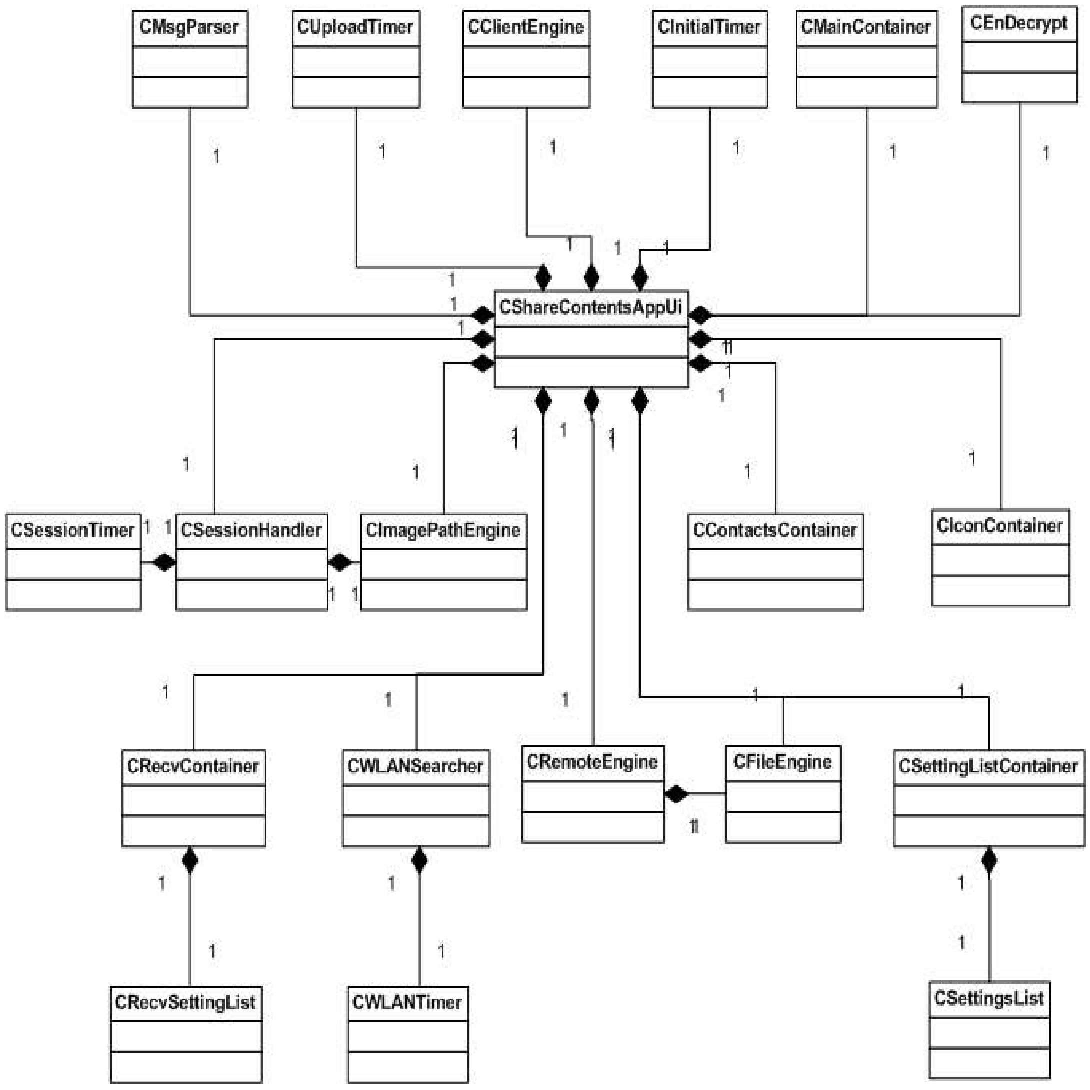


Figure 38: ACSEH System Class Diagram

Classes' description:

CContactsContainer:

Purpose: This class connects to contacts database and extracts list of phone contacts stored in it. It creates a multi-selection list based on those contacts which is used by primary user to select multiple recipients for sharing session.

CEnDecrypt:

Purpose: This class is used to encrypt and decrypt SMS message that contains session information (i.e. session name and time) and required credentials to connect to Home PC remotely. It uses reversing and ceaser cipher as an encryption and decryption technique.

FileEngine:

Purpose: This class is used to perform all filing related operations for ACSEH system.

CClientEngine:

Purpose: This class is used to create a HTTP client, which connects to primary user's home PC to download generated pins. These pins are later on send using SMS to recipients along with session details. These pins are used by recipients' mobiles to connect to primary user's home PC to create an account during initial stage and to upload contents during later stage.

CIconContainer:

Purpose: This class is used to create and remove sharing session icon on mobile screen. Sharing session icon is displayed on mobile screen to indicate that sharing session is still active and all contents created during that time are automatically being marked for sharing.

CImagePathEngine:

Purpose: This class is used to extract list of images created during sharing session. Mobile images are filtered based on date and time to extract only those images that were created during sharing session. It extracts images name, path, date and time of creation that are used by CRemoteEngine class during image uploading.

CMsgParser:

Purpose: This class is used to parse message at the receiving end. It parses the message to extract session name, session time and pin code used to connect to Home PC of primary user.

CRecvContainer:

Purpose: This class is used at recipients end and constructs a setting list which requests GPRS name and WLAN name from recipients

CRemoteEngine:

Purpose: This class is responsible for handling all required remote actions with the home PC of primary user. It is used to create account, connect and upload contents remotely on primary user's home PC.

CSessionHandler:

Purpose: This class is used to create and handle sharing session. It gets control from CShareContentAppUi class and passes control back and forth to it. It creates sharing session, searches for user specified WLAN after sharing session is over and notifies CShareContentAppUi about session expiration and WLAN detection.

CSessionActive:

Purpose: This class is acts as a simple timer. It is used by CSessionHanlder to set time for session.

CSettingListContainer:

Purpose: This class is used to create setting list used by primary user to specify WLAN and GPRS name

CShareContentsAppui:

Purpose: This class act as a controller for both sending and receiving end and controls the behavior and execution of whole application. It is responsible for displaying and removing sharing session icon, for triggering sharing session, for sending application in background and for bringing it in foreground.

Sequence Diagrams:

ConnectInitially:

Diagram:

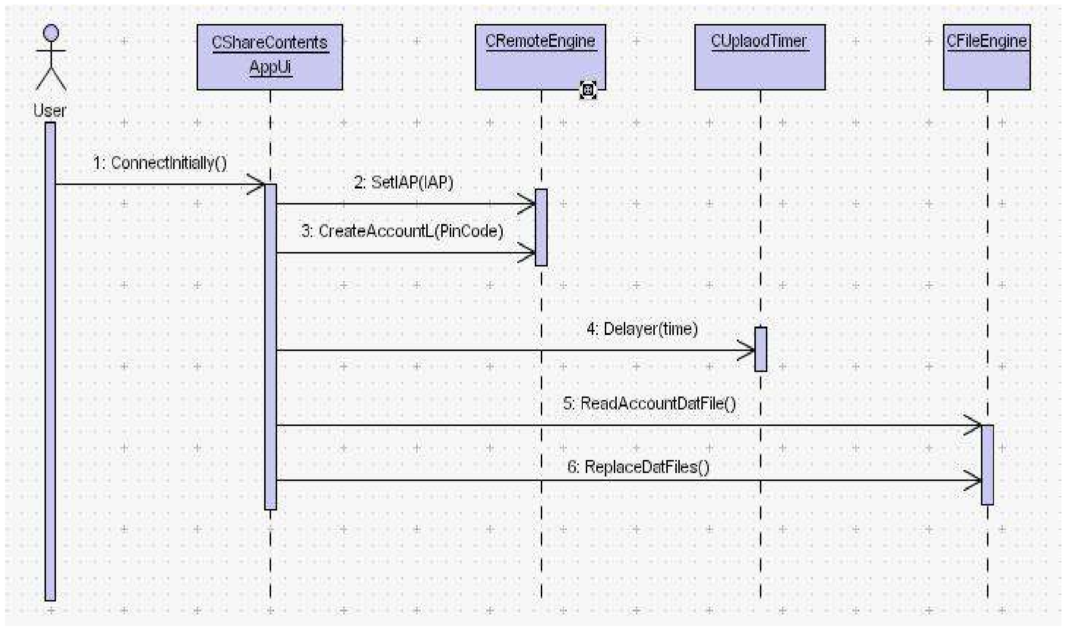


Figure 39: ACSEH System Sequence diagram of ConnectInitially use case

Description: This module creates an account at Home PC of primary user. After account is created, it disconnects from Home PC and extracts dynamic IP from a newly generated file. Dynamic IP address is used in later stage to connect to Home PC of primary user to download required credentials for session recipients.

StartSessionSending:

Diagram:

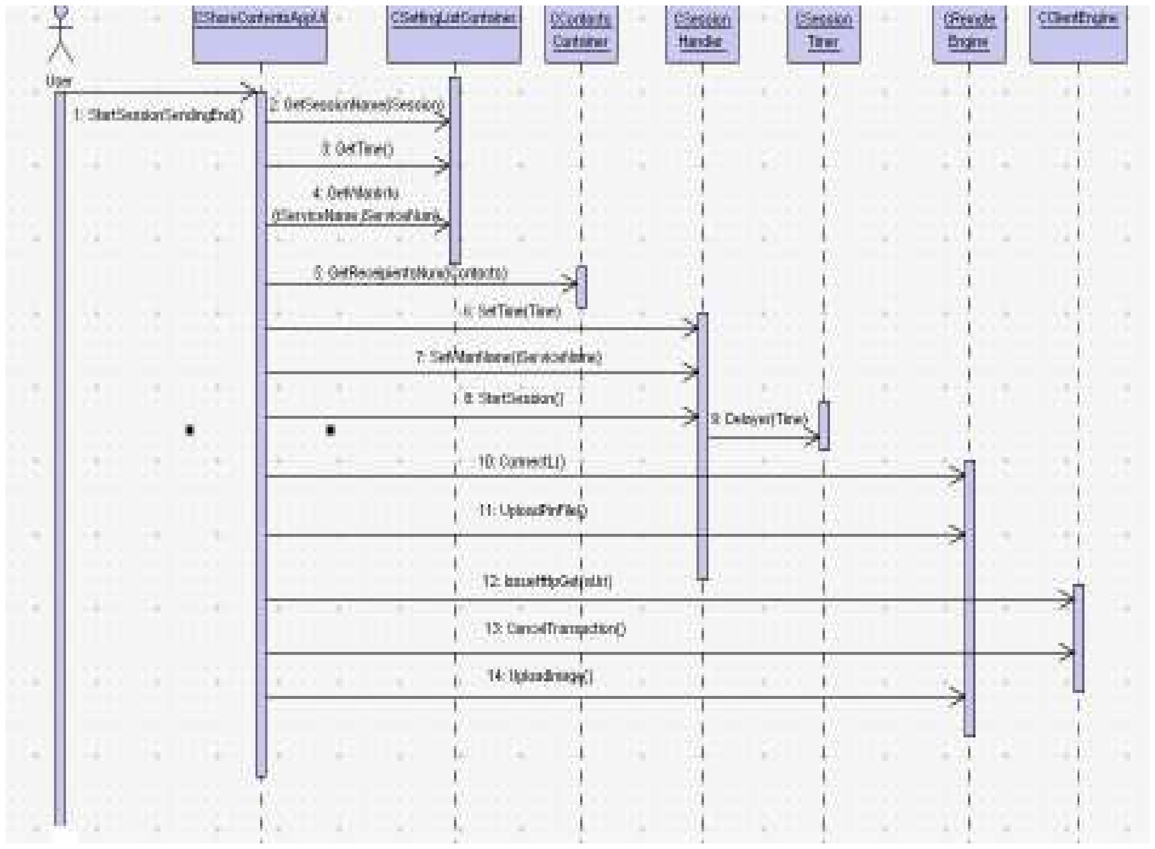


Figure 40: ACSEH System Sequence diagram of StartSessionSending case

Description: It gets WLAN information, session name and time from CSettingListContainer and contacts information from CContactsContainer. It then writes this information in “pin_request.txt” file. It then sets IAP for remote engine and connects to Home PC to upload “pin_request.txt “file and to download generated pins using HTTP client engine. It then disconnects from Home PC and sends session invitation including pin code downloaded from Home PC to recipients. It then calls CSessionHandler to start the session after passing it required information. CSessionHandler calls CSessionTimer to complete the session time and after session time is complete, it calls CImagePathEngine to get list of images taken during session. CSessionHandler then looks for user specified WLAN and once it is found, it calls CShareContentsAppUi to upload the images. CShareContentsAppUi writes images information into file, connects to Home PC and uploads the pictures using user specified WLAN.

StartSessionReceivingEnd:

Diagram:

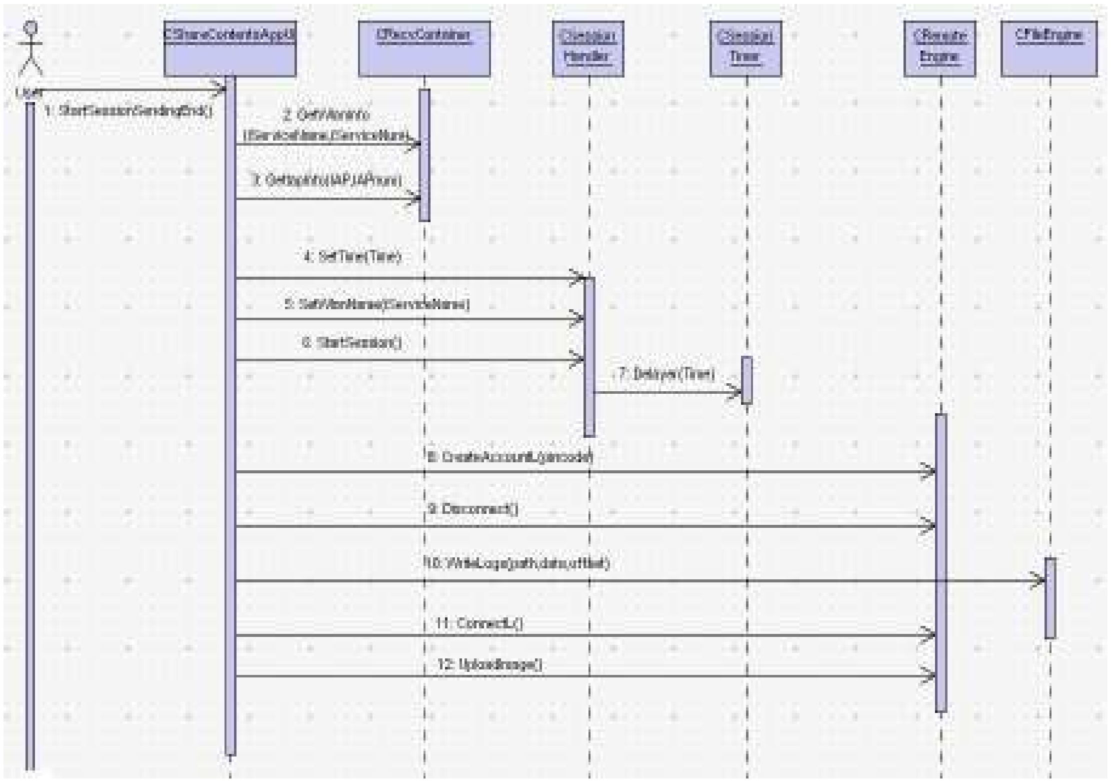


Figure 41: ACSEH System Sequence diagram of StartSessionReceivingEnd case

Description: It parses invitation messages and extracts session name, time and pin code from it. It gets WLAN and GPRS connection information from CRecvContainer. It sets IAP and connects to Home PC of primary user to create an account using Pin code. After account is created, it disconnects from Home PC. It then calls CSessionHanlder to start the session after passing it session time and WLAN name to look for, after session is over. CSessionHandler calls CSessionTimer to complete the session time and after session time is complete, it calls CImagePathEngine to get list of images taken during session. CSessionHandler then looks for user specified WLAN and once it is found, it calls CShareContentsAppUi to upload the images. CShareContentsAppUi writes images information into file, connects to Home PC and uploads the pictures using user specified WLAN.

Appendix 3: Thesis implementation code snippet

```
void CSessionHandler::StartSession()
{
      iImagePathEngine->SetTimeFormat24();
      iImagePathEngine->SetDateFormatJapanese();
      iStartTime.HomeTime();
      iEndTime.HomeTime();
      TTimeIntervalMinutes timeMinutes(iTime);
      iEndTime=iEndTime.operator +(timeMinutes);
      TDateTime tempDate;
      tempDate=iStartTime.DateTime();
      iStartDays=(tempDate.Year()*365);
      iStartDays+=(tempDate.Month()*30);
      iStartDays+=30;
      iStartDays+=tempDate.Day();
      iStartDays+=1;
      tempDate=iEndTime.DateTime();
      iEndDays=(tempDate.Year()*365);
      iEndDays+=(tempDate.Month()*30);
      iEndDays+=30;
      iEndDays+=tempDate.Day();
      iEndDays+=1;
      tempDate=iStartTime.DateTime();
      iStime=tempDate.Hour();
      iStime=iStime*60; //converting hours into mins
      iStime+=tempDate.Minute();
      tempDate=iEndTime.DateTime();
      iEtime=tempDate.Hour();
      iEtime=iEtime*60; //converting hours into mins
      iEtime+=tempDate.Minute();
      iNumTimerCalled=0;
      iRepeater=iTime/35; // symbian timers upper limit is 35 minutes
      iRemainder=iTime%35; //
      if(iRepeater==0) // no need to call timer more than once
      {
            iRequireRepeat=EFalse;
      }
      else  // timer has to be called more than once
      {
            iRequireRepeat=ETrue;
      }
      StartTimer();
}

void CSessionHandler::GetImageMetaData()
{
      iImagePathEngine->SetTimeFormat24();
      iImagePathEngine->SetDateFormatJapanese();
      iImagePathEngine->StartFileList();
      iImagePathEngine->GetFileListItemsL(iPicPathArray);
      iImagePathEngine->CloseSesison(); // close file session
      iImagePathEngine-
      >FilterByDate(iPicPathArray,iFiltered,iStartDays,iEndDays);
      iImagePathEngine-
      >FilterByTime(iFiltered,iPicPathArray,iStime,iEtime);
      iAppUi->RemoveIcon();
```

```
   iSessionOver=ETrue;
   iSessionTimer->Cancel();
   iTime=10000000;
   iSessionTimer->Delayer(iTime);
}
```

TBool CSessionHandler::SearchWLAN()

```
{
      TInt index=-1;
      iWLAN.Reset();
      TBuf<32> netName;
      RConnectionMonitor monitor;
      TPckgBuf<TConnMonNetworkNames> pkgNetworks;
      TRequestStatus status;
      monitor.ConnectL();
      CleanupClosePushL(monitor);
      monitor.GetPckgAttribute(EBearerIdWLAN, 0, KNetworkNames);
      User::WaitForRequest( status ) ;
      User::LeaveIfError(status.Int());
      for(TUint i=0; i<pkgNetworks().iCount; i++)
      {
        netName.Copy(pkgNetworks().iNetwork[i].iName);
        iWLAN.AppendL(netName);
      }
    // close server session
    monitor.Close();
    CleanupStack::PopAndDestroy(&monitor);
    index=GetWLANIndex();
    if(index>=0) // WLAN found
    {

      return ETrue;
    }

    else // call timer again to continue searching
    {
            return EFalse;
    }

}
```

void CRemoteEngine::DoCreateAccountL()

```
{
      iRADEngine->SetDisclaimerAcceptedL();
      iRADEngine->CreateAccountL( iCode );

}
```

void CRemoteEngine::UploadPinFile()

```
{
      if(!iConnected)
      {
            iRADEngine->ConnectL( iRADEngine->CurrentAccount() );
      }
      TBuf<250> FilePath(KPinFilePath);
```

```
	iRADEngine->UploadFileL(FilePath);

}
```

void CRemoteEngine::UploadImage()

```
{
	RenameImages();
	if(!iConnected)
	{
		iRADEngine->ConnectL( iRADEngine->CurrentAccount() );
	}
	TBuf<100> Logs(KNewFilePath);
	TBuf<300> Path;
	RFs fsSession;
	User::LeaveIfError(fsSession.Connect());
	RFileReadStream readStream;
	readStream.PushL();
	User::LeaveIfError(readStream.Open(fsSession,Logs,EFileRead));
	CLineReader *iReader=CLineReader::NewL(readStream);
	TInt aPos=0;
	TInt aErr=0;
	while(aErr!=KErrEof)
	{
		Path.Zero();
			iReader->ReadLineL(aPos,aErr);
		Path.Copy(iReader->iBufPtr);
		iRADEngine->UploadFileL(Path);
		}
	CleanupStack::PopAndDestroy();
	delete iReader;
	iReader=NULL;
	fsSession.Close();
}
```

void CImagePathEngine::ParseImageData(TDes& aImgData,TDes& aPath,TDes& aDate,TDes& aTime)

```
{
	aPath.Zero();
	aDate.Zero();
	aTime.Zero();
	TInt HashIndex=aImgData.Find(KHash);
	iDateComplete=EFalse;
	for(TInt i=0;i<HashIndex;i++)  // filling up aPath
	{
		aPath.Append((TChar)aImgData.operator [](i));
	}
	HashIndex++;
	for(TInt i=HashIndex;i<aImgData.Length();i++)
	{
		if(aImgData.operator [](i)==35)
			{
				iDateComplete=ETrue;
			}
```

```
                  else
                  {
                  if(!iDateComplete)
                  {

                        aDate.Append((TChar)aImgData.operator [](i));
                  }

                  else // fill up aTime
                  {

                        aTime.Append((TChar)aImgData.operator [](i));
                  }

            }
      }

}
```

void CImagePathEngine::GetImgFolder(TDes& aImageData,TDes& aFolder,TDes& aImageName)

```
{
      TBuf<100> Path; // gets path after parsing
      TBuf<50> Date;    // gets date after parsing
      TBuf<50> Time;  // gets time after parsing
      TBuf<50> tempFolder;
      TBuf<50> tempImage;
      tempFolder.Zero();
      tempImage.Zero();
      aImageName.Zero();
      aFolder.Zero();
      TChar temp;
      TInt tempNum;
      TBool ImageDone=EFalse;
      TInt length;
      ParseImageData(aImageData,Path,Date,Time);
      length=Path.Length();
      length--;
      for(TInt i=length;i>=0;i--)
      {
            tempNum=Path.operator [](i);
            if(tempNum==92)// ascii for \ is 92
            {
                  ImageDone=ETrue;
            }

            temp=(TChar)tempNum;
            if(!ImageDone)
            {
                  tempImage.Append(temp);
            }

            else
            {
                  tempFolder.Append(temp);
            }
      }

      length=tempFolder.Length();
      length--;

      for(TInt i=length;i>=0;i--)
      {
            tempNum=tempFolder.operator [](i);
            aFolder.Append((TChar)tempNum);
      }

      length=tempImage.Length();
      length--;

      for(TInt i=length;i>=0;i--)
      {
            tempNum=tempImage.operator [](i);
            aImageName.Append((TChar)tempNum);
      }
}
```

void CClientEngine::IssueHTTPGetL(const TDesC8& aUri)

```
{

        SetupConnectionL();
        TUriParser8 uri;
        uri.Parse(aUri);
        RStringF
        method=iSession.StringPool().StringF(HTTP::EGET,RHTTPSession::Get
        Table());
        iTransaction=iSession.OpenTransactionL(uri,*this,method);
        RHTTPHeaders hdr=iTransaction.Request().GetHeaderCollection();
        SetHeaderL(hdr, HTTP::EUserAgent, KUserAgent);
        SetHeaderL(hdr, HTTP::EAccept, KAccept);
        iTransaction.SubmitL();
        iRunning = ETrue;
}
```

TBool CMsgParser::Parser2(TDes& aMsgData,TDes& aTime,TDes& aBasketName,TDes& iPinCode)

```
{
        TInt HashCounter;// counts number of hashes in the code
        TInt HashIndex;
        TInt i; // used to traverse and access each item of the
        TChar TempChar;
        TInt TempNum;
        TInt MsgLength=aMsgData.Length();
        TInt IndetifierIndex=aMsgData.Find(KIdentifier);
        if(IndetifierIndex==KErrNotFound)
        {
            return EFalse;
        }
```

```
HashIndex=aMsgData.Find(KHash);
if(HashIndex==KErrNotFound)
{
      return EFalse;
}

i=HashIndex+1; // start picking up data from first hash onwards
HashCounter=1; // as one # is found

for(i;i<MsgLength;i++)
{
            TempNum=aMsgData.operator [](i);
            if(TempNum==35) // ascii value for # =35
            {
                  HashCounter++;
            }

            // its soem data, typecase num into character
            else
            {
                  TempChar=(TChar) TempNum;

                        switch(HashCounter)
                        {
                              case 1:
                              {
                                    aTime.Append(TempChar);

                              }
                              break;

                              case 2:  // retrieving basket name
                              {
                                    aBasketName.Append(TempChar);
                              }
                              break;

                              case 3:
                              {
                                    iPinCode.Append(TempChar);
                              }
                              break;

                        }     // switch ends here

            }// else ends here

      } // for loop ends here

if(aTime.Length()==0 ||aBasketName.Length()==0 ||
iPinCode.Length()==0)
{
      return EFalse
}
```