Junxi Yin

Mobile Implementation of Floating Content Service

Thesis submitted for examination for the degree of Master of Science in Technology.

Espoo, Nov 20, 2012

Thesis supervisor: Prof. Jörg Ott

Aalto University
School of Electrical Engineering

AALTO UNIVERSITY
SCHOOL OF ELECTRICAL ENGINEERING

ABSTRACT OF THE
MASTER'S THESIS

Author: Junxi Yin

Title: Implementation of Floating Content Service

Date: Nov 20, 2012 Language: English Number of Pages: 9+68

Department of Communication and Networking

Professorship: Networking Technology

Supervisor: Prof. Jörg Ott

Traditional Internet technology makes a series of assumptions which may not exist in some cases especially in the challenged environments where some or all of those requirements are hard to meet. In addition, Internet-based communications face the privacy and data security issues simply because the data is essential to be sent to and maintained in a centralized infrastructure.

DTN, by contrast, supports frequently interrupted communications by introducing a Store-and-Forward message switching mechanism: intermediate node keeps the bundle in its persistent storage and waits for a better opportunity to forward it at a later time to another custody node or directly to the final destination if reachable. Floating Content, as a DTN-based localized information sharing service exclusively dependent on mobile phones and other mobile devices, takes advantage of wireless interfaces such as Bluetooth or WLAN to communicate in ad-hoc mode.

We evaluate Floating Content service and finally implement it into mobile platforms in the form of mobile application thus benefits the users by providing an opportunistic communication mechanism even in an extreme environment, as well, decreases the security and privacy worries to a certain extent.

Keywords: Delay-Tolerant Networking (DTN), Floating Content, Simulations, Android, Location-based, Opportunistic Communication

Acknowledgements

This thesis would not have been possible without the guidance and the help of several individuals.

First and foremost, I would like to thank Professor Jörg Ott for offering me this thesis opportunity, giving me advice and teaching me about life. He is a good man, his hard working and serious attitudes toward science have been inspiring me and will inspire me for the rest of my life.

Thank Teemu Kärkkäinen from Comnet of Aalto University, for giving me advice and helping me in the process of coding. It is enjoyable to read his code and extremely helpful to form a good programming habit.

Thank all my colleagues in D301, Marcin, Mikeal, and Abu. We had lots of interesting discussions together and I sure will miss all the office time and coffee breaks.

Thank all my friends who have encouraged me to move forward. Being away from homeland is sometimes hard, but my friends helped me through and we had a really good time.

Thank Dabin Yin and Xianhui Tan for having been doing everything they could as great parents. They are always there cheering me up and stood by me through the good times and bad. Without them, I am nothing.

Finally I would like to thank all my families and all the people who have helped me for this great opportunity to study in Finland. You made my life.

Espoo, November, 2012

Junxi Yin

Abbreviations and Acronyms

AA	Application Agent
ADU	Application Data Unit
API	Application Programming Interface
BP	Bundle Protocol
BPA	Bundle Protocol Agent
BSP	Bundle Security Protocol
CLA	Convergence Layer Adapter
DNS	Domain Name System
DoS	Denial of Service
DTN	Delay-Tolerant Networking
DTNRG	Delay-Tolerant Networking Research Group
EID	Endpoint ID
FIFO	First In First Out
GPS	Global Positioning System
HCS	Helsinki City Scenario
ID	Identification
IP	Internet Protocol
IPN	Interplanetary
MRG	Minimum Reception Group
NASA	National Aeronautics and Space Administration

ONE	Opportunistic Networking Environment
OS	Operation System
PPP	Point-to-Point Protocol
RND	RaNDom
RTT	Round Trip Time
SAF	Smallest Area First
SQL	Structured Query Language
STF	Smallest Total resource consumption First
SVF	Smallest Volume First
TCP	Transmission Control Protocol
TDD	Test Driven Development
TTL	Time-To-Live
UDP	User Datagram Protocol
URI	Uniform Resource Identifier
WLAN	Wireless Local Area Network
XML	Extensible Markup Language

List of Figures

Contents

Chapter 1

Introduction

Social networks, such as Facebook, Foursquare, Google+, Flickr, Twitter, have greatly changed the manner in which people share opinions, ideas and interests with each other across geographic and time borders thus fostering relationships. By the end of 2011, 1.2 billion users worldwide – 82% of the world's Internet population over the age 15 – log on to a social networking site with roughly every one in five minutes spent on a social network, making it the most engaging online activity around the world. [1]

The tremendous development and expansion speed of social networks has brought immeasurable convenience into people's daily life. It not only makes the information much more personalized according to the users' own interests but also saves time spent on searching and waiting for certain information in the traditional social activities. Users can merely subscribe to their favorite topics or follow the fellows meaningful to them, and pick up the information sets whenever it is convenient to them simply in an asynchronous manner. This in a way helps to filter news without costing a minute as well takes full advantage of the time fragments people would have wasted (e.g. sitting on the bus, waiting for a friend in a restaurant, taking a rest during work, or any other small piece of time) by doing nothing.

Meanwhile, stimulated by the rapid development of telecommunication technologies, the mobile device industry has also been quickly growing with newer and more powerful devices emerged into our life every day. Refer to the statistics [2], over 5 billion mobile phones will be activated worldwide by 2015 and the current increasing trend with a dramatic speed is evident. The trend of using mobile phones to connect to Internet makes the term “Mobile Internet” [3] a buzzword. Growing popularity of mobile internet and applications that run on smart phones and other mobile devices makes social networking not just be limited to the Web but also carried out in an even easier way to meet the users’ needs of sharing anytime, anywhere as long as Internet is available. Among those millions of mobile applications, location-aware services such as Google Maps, FourSquare, and many other applications with location-aware features integrated are widely built around mobile phones and become quite popular for location-related inquiries as well as context sharing.

However, they are infrastructure-based, with the nature that data and user information must be stored and maintained in a centralized server-like system. This characteristic raises users’ concerns about data security, user privacy (location privacy, user information, etc.) and content validity range from the perspective of both time and space when they are making use of the online social networking. In addition to that, the connectivity prerequisite also limits the geographic range of users, for example the traveling users will have to face the expensive roaming charges or unavailable data services and network coverage. Thus, relying on infrastructure-based services may not always be the most desirable choice.

Solutions have been suggested such as using pseudonyms to communicate with servers, adopting local caching mechanism, or simply designing a completely server-less system. Delay-tolerant Networking [4] (will be abbreviated as DTN in the following paragraphs), as a mature networking technology, has provided diverse of infrastructure-less communication by making use of the store-and- forward communication mechanism. Some of them have already achieved a certain degree of success in the laboratory. Floating Content [5] [6] service is one of those candidate approaches and has been proved totally feasible in a simulation environment even with modest number of nodes supporting this service in a typical urban environment. It would be even valuable if some of those results can be applied into people's daily life.

1.1 Problem statement

DTN technology, as an approach to the heterogeneous networks that may lack continuous network connectivity, has already been widely deployed in the interplanetary communications and military communications. However, most other of the usages are limited to researches. It is really not usual to witness this mature technology applied into people's daily life. Floating Content service, as one of the DTN-based technologies, faces the same issue and somehow it has limited the wide spread of this concept.

If we consider a scenario in an extreme environment, for instance an earthquake happens and the infrastructures used to support communication services are more or less destroyed. The service providers thus cannot anymore guarantee the normal delivery of their voice services, neither the data services. Many lives could have been rescued if the SOS messages can be sent out with location information in such an emergency.

Consider another situation when you are in a flea market and want to buy a commodity. Time can be wasted for searching and comparing items scattered in every corner. Not to mention that you have to step back and forth. It could be easier to make comparisons.

Scenarios like these where the traditional internet does not work well are not uncommon and bring troubles or even worse results to us that could have been avoided. Floating Content service, as one of the candidate DTN-based solutions, has given a clear clue to fix problems happen in those environments as mentioned above.

Nevertheless, for the time being, it has achieved an initial success only in the laboratory. Applications of Floating Content service with friendly user interface are not available in the market yet, for this reason, we are proposing an API and building an application based on it, targeting at making it easier for the users to make use of Floating Content service.

1.2 Motivation

From the first time the term delay-tolerant networking and the DTN acronym was coined by Kevin Fall in 2002, it has already become a mature networking technology and has been widely deployed especially in the space and military field communications. DTN-based services such as Floating Content have also been greatly developed in the lab environment. However, it is not yet a civilian technology which can be taken full advantage of by people to benefit their daily life. This, to some extent, has limited the wide spread and further development of DTN technology.

Speaking of the reasons, firstly, platform (mainly the software) specially used as a carrier of DTN technology is not widely supported yet; Secondly, it might be costly to bring this technology from the space all the way down to the ground and set up a totally new deployment.

The open source concept has been frequently mentioned and developed in recent years. A wide variety of open-source OS and software tool kits are available to grab, for example Linux, Android OS, Web OS and so forth. These countless resources make the software development process easier and cost much lower than any time before. [7]

In addition to that, with the development of technologies and increasingly fierce competition, mobile devices are becoming cheaper and more powerful every day. Developers can even build applications by using emulators provided by the main stream mobile OS suppliers such as Google, Apple and Microsoft and publish them into the app market. [8]

Deploying opportunistic social networking into mobile devices is of great significance. It not only helps expand and ease social networking activities, but also provides opportunities for people especially who live in under-developed countries and areas to also log in to a social network.

1.3 Research Objectives and Scope

The main target of this thesis is to evaluate Floating Content, a DTN-based opportunistic information sharing service, and finally implement this infrastructure-less service into mobile platforms in the form of mobile application thus benefits the users by providing an opportunistic communication mechanism even in an extreme environment, as well, decreases the security and privacy worries to a certain extent.

Firstly, we re-run simulations to evaluate the feasibility and performances by introducing different parameters and models as [5] [6] do. Secondly, we introduce the location error issue to further evaluate the effect that location error brings to the service performance in a simulation environment. Thirdly, the API built upon DTN and used for Floating Content service realization is analyzed and studied. Based on those simulation results and the API, real implementation is then carried out. Finally, the application is deployed in a test environment and tested.

Considering the open source nature and the absolute largest market share in today’s mobile industry, we decide to adopt Android mobile operation system as our first test platform for our implementation and further studies.

1.4 Structure

This thesis is structured in five main blocks: In Chapter 1, we make an introduction to this thesis, state the problem being analyzed, explain the motivation of doing this work, finally the scope and relevant tools and methodology deployed to carry out this thesis. In Chapter 2, the Delay-tolerant Networking is elaborated, including the architecture, bundle protocol, routing, mobility models and security issues. Chapter 3 introduces the Floating Content service. It is mainly focused on the theoretical analysis about the service model, system operations, description of the Floating Content protocol. Also importantly, it states the simulation results of the service and introduces the API designed by deeply understanding of this protocol. Chapter 4 presents the application implementation upon the Floating Content API. A detailed description of the design pattern, system structure, message format, and the user interface design is provided in this section. Chapter 5 tests and evaluates the implementation in a real world scenario. It specifies the test results and an analysis. Finally, Chapter 6 concludes the thesis and discusses the possible future work.

Chapter 2

Delay Tolerant Network

DTN [4] is an overlay sitting on the top of regional networks and is designed to deal with extreme conditions caused by intermittent connectivity, long or variable delay, asymmetric data rates and high error rates in the heterogeneous networks.

Traditional networks make a series of assumptions such as an end-to-end communication path can always be found; the network topology is largely static; very short RTT; packet losses are rare etc. However, those crucial pre-conditions mentioned above may not exist in some cases especially in the challenged environments such as military battle field, rural areas and many other circumstances where some or all of those requirements are hard to meet. In addition, network-based communications face the privacy and data security issues simply because the data is essential to be sent to and maintained in a centralized infrastructure. [9]

DTN, by contrast, supports frequently interrupted communications by introducing a Store-and-Forward message switching mechanism: intermediate node keeps the bundle in its persistent storage and waits for a better opportunity to forward it at a later time to another custody node or directly to the final destination if reachable. A bundle layer shields the region-specific lower layers so that application can communicate across multiple regions. Reliable communication is achieved by the mean of custody transfer plus a variety of service classes provided for bundle transmission.

Bundle Protocol is the bundle layer protocol. BP implements the essential functionalities and services that support the delay-tolerant characteristic. Bundle is the basic message unit with arbitrary length in DTN communications. BP takes advantages of the native internet protocols, fulfilling the capabilities such as custody-based retransmission, ability to cope with intermittent connectivity as well as scheduled, predicted, opportunistic connectivity, late binding of overlay network endpoint identifiers to constituent internet addresses. During those operations, bundles are stored in the permanent storage and forwarded to the next opportunistic node when another better chance comes. [10]

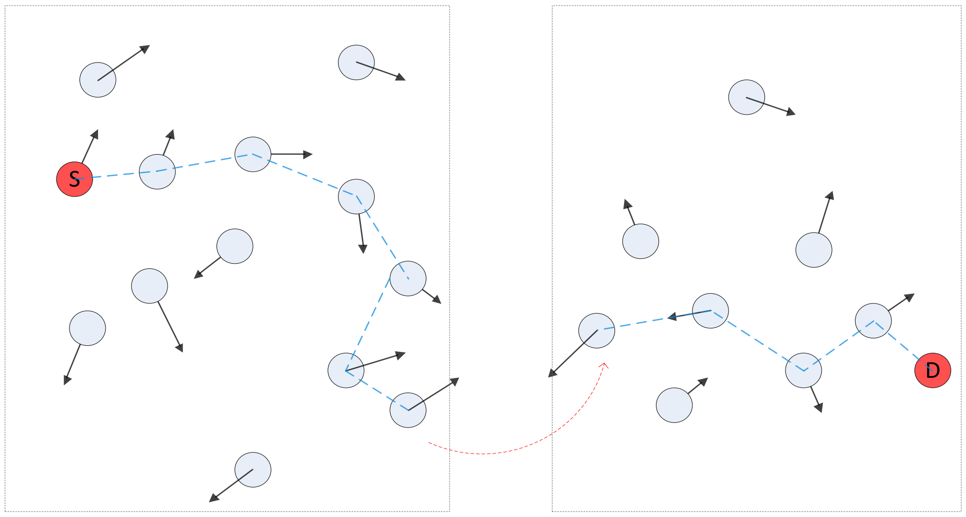


Figure 2.1 A Simple DTN Scenario

Figure 1shows a simple scenario where Delay-tolerant network is deployed. In an infrastructure-less environment, a node tries to send a message to a node which is far out of its radio range, the message will be sent to the source node's neighbor node and forward. By keeping doing this, and with the help of node mobility, there is a chance that the message will be eventually delivered to its destination.

2.1 DTN Architecture

DTN architecture was originally designed by DTNRG to provide internet-like services across interplanetary communications. Evolved from the architecture which is mainly focused on deep space communication issues, it already embraces the general concepts of networks that suffer from frequent disruptions, disconnections and high delays. It has also been proved applicable in sensor-based networks using scheduled intermittent connectivity, satellite networks with moderate delays and periodic connectivity, as well underwater acoustic networks with moderate delays and frequent interruptions due to environmental factors.

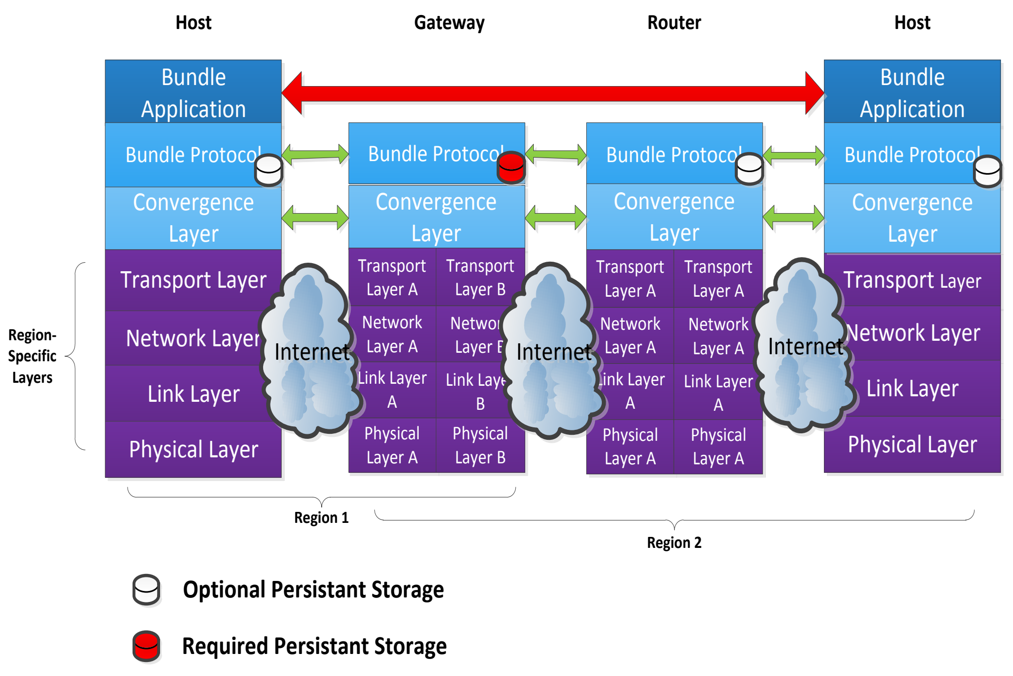


Figure 2.2 DTN Protocol Architecture

The current DTN architecture provides a common method for interconnecting heterogeneous gateways or proxies that employ store-and-forward message routing to overcome communication disruptions. Figure 2.2 depicts the DTN architecture model proposed by DTNRG.

The bundle layer, as an end-to-end message-oriented overlay, is defined above the regional layers. It stores and forwards entire or fragment bundles between nodes by using a single bundle protocol and the persistent storage, thus helping deal with frequent network interruptions that may happen in an extreme environment. [11] [12]

2.2 Bundle Protocol

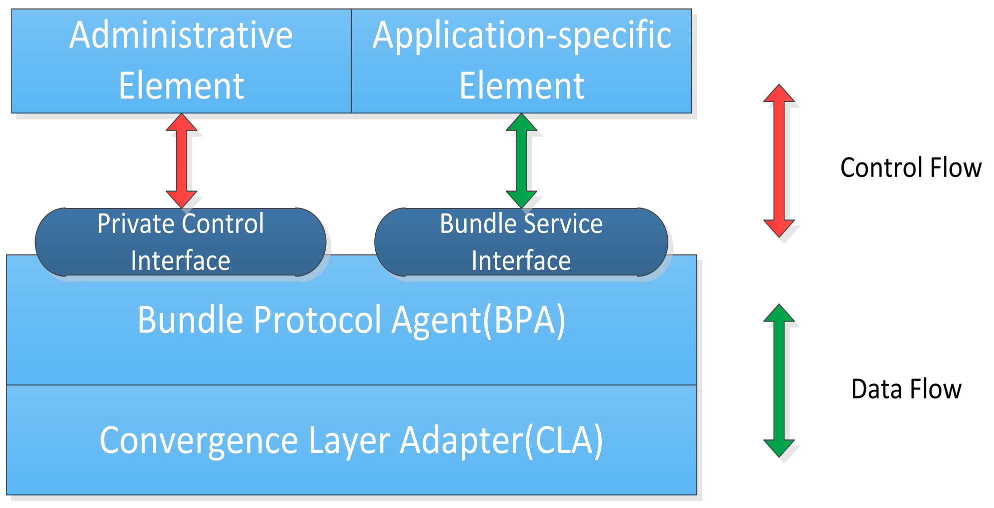
The bundle protocol (BP) is an experimental DTN protocol designed for unstable communication networks. BP groups a series of contiguous data blocks into bundles and each of them has enough semantic information to enable applications to make further progress. Bundles are generally routed in a store-and-forward manner between participating nodes over network transport technologies. These layers taking along bundles across local networks are called bundle convergence layers.

Due to the nature of store-and-forward transmission mechanism, application layer is responsible to set up the service requirements so that BP can collect the application data into bundles and send across heterogeneous network to achieve high-level service guarantees.

The capabilities of BP include: Custody-based retransmission; Late binding of overlay network endpoint identifiers to constituent internet addresses; Ability to take advantage of scheduled, predicted and opportunistic connectivity; Ability to interoperate with intermittent connectivity. [10]

2.2.1 Bundle Node

The bundle node, as shown in Figure 2.3, is an entity (host, router or gateway) that implements BP and can send or receive bundles. Each node is made up of three components: bundle protocol agent, application agent, convergence layer adapter.



Administrative Element
Application-specific Element
Control Flow
Private Control Interface
Bundle Service Interface
Bundle Protocol Agent(BPA)
Data Flow
Convergence Layer Adapter(CLA)

Figure 2.3 Bundle Node

A. Bundle Protocol Agent

The bundle protocol agent (BPA) of a node is the node component that offers the BP services and executes the procedures of the bundle protocol. The manner in which it does so is wholly an implementation matter. For example, BPA functionality might be coded into each node individually; it might be implemented as a shared library that is used in common by a number of bundle nodes on a single computer; it might be implemented as a daemon whose services are invoked via inter-process or network communication by any number of bundle nodes on one or more computers; it might be implemented in hardware.

B. Application Agent

The application agent (AA) of a node utilizes the BP services so that it can take advantage of DTN technology to communication. It is mainly made up of two parts, administrative element and application-specific element.

The application-specific element of an AA constructs, requests transmission of, accepts delivery of, and processes application-specific application data units; the only interface between the BPA and the application-specific element of the AA is the BP service interface. The administrative element of an AA constructs and requests transmission of administrative records (status reports and custody signals), and it accepts delivery of and process any custody signals that the node receives; in addition to the BP service interface, there is a (conceptual) private protocol interface between the BPA and the administrative element of the AA that enables each to direct the other to take action under specific circumstances.

C. Convergence Layer Adapter

A convergence layer adapter (CLA) sends and receives bundles on behalf of the BPA, utilizing the services of some native internet protocol that is supported in one of the internets within which the node is functionally located. The manner in which a CLA sends and receives bundles is wholly an implementation matter, exactly as described for the BPA.

2.2.2 Bundle Endpoint

As introduced in 2.2.1, a bundle node is an implementation of BP. A bundle endpoint, simply named as endpoint, is a set of (zero or more) bundle nodes. Each node is identified by a single text string named “endpoint ID”, shortly denoted as EID (details in 2.2.3). A bundle is considered as successfully delivered as long as some minimum subset of nodes in the endpoint has received the bundle without error. This subset is called Minimum Reception Group (MRG) of the endpoint.

A Singleton, as a special case of an endpoint that contains at most one node, is the most familiar sort of endpoint. Each node must be a member of at least one singleton endpoint. However, in general, the endpoint notion is meant to be broader. For instance, in the sensor network, a group of nodes might register with a common EID so that they can receive specific bundles as a single bundle endpoint. Besides, each node can register with multiple different EIDs and identify itself as a member of multiple bundle endpoints.

2.2.3 EID

EID is a name expressed using the general syntax of Uniform Resource Identifier to identify a bundle endpoint and is conveyed in the bundle block. The format is as follows,

< scheme name > : < scheme-specific part, or "SSP" >

Depending on the construction of the EID being used, there may be a provision for wildcarding some portion of an EID, which is often useful for diagnostic and routing purposes.

A node can determine the MRG of the DTN endpoint by EID. And each node is required to register at least one EID that uniquely identifies it. Applications send ADUs destined for an EID. The action that an application is willing to receive ADUs destined for a particular EID is called a “registration” and is maintained persistently by a DTN node. [13]

2.2.4 Bundle

Bundle is the basic protocol data unit used in the DTN communications. Each bundle is of arbitrary length and is consist of three main parts: a bundle header, the user data, and the control information describing the procedure of bundle process. Bundles can be fragmented, and the fragmentation will be reassembled at the final destination. [10]

A. Bundle format

Each bundle shall be a serial sequence of at least two block structures: primary bundle block and additional bundle block.

The first block in the sequence must be a primary bundle block, and only one primary bundle block is allowed in one bundle.

Additional bundle protocol blocks of other types may follow the primary block to support extensions to the bundle protocol, such as the BSP (Bundle Security Protocol). No more than one of the blocks in the sequence may be a payload block. The last block in the sequence must have the "last block" flag (in its block processing control flags) set to 1; for every other block in the bundle after the primary block, this flag must be set to zero.

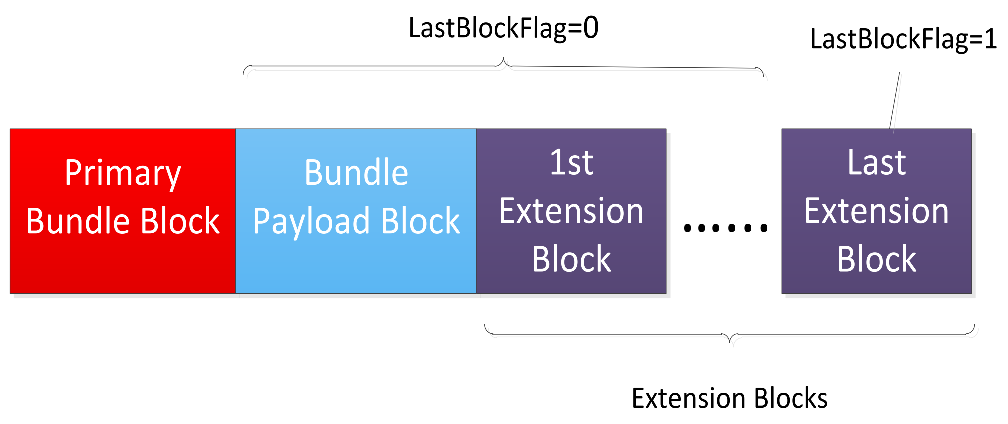


Figure 2.4 Bundle Format

B. Bundle services

A variety of services are provided by different parts of the bundle layer.

The bundle protocol agent of each node is expected to provide a series of services to the node's application agent as follows: commencing a registration (registering a node in an endpoint); terminating a registration; switching a registration between Active and Passive states; transmitting a bundle to an identified bundle endpoint; canceling a transmission; polling a registration that is in the passive state; delivering a received bundle.

For the convergence layer adapter, services include sending a bundle to all bundle nodes in the minimum reception set of the endpoint identified by a specific EID that are reachable via the convergence layer protocol, and delivering to the bundle protocol agent a bundle that was sent by a remote bundle node via the convergence layer protocol.

2.2.5 Custody Transfer

BP incorporates custody transfer [14] in order to provide a reliable node-to-node transmission to the final destination. Store-and-forward mechanism is used to transmit bundles thus the responsibility of reliable transfer is passed to the next hop one by one all the way towards the final destination.

A node accepts custody transfer is called custodian. When the current custodian sends a bundle to its neighboring hop, the neighboring node shall reply with an acknowledgement or a refusal signal in a specific evaluation time interval. In case that no reply is returned before the evaluation time expires, a timer will trigger the retransmission mechanism.

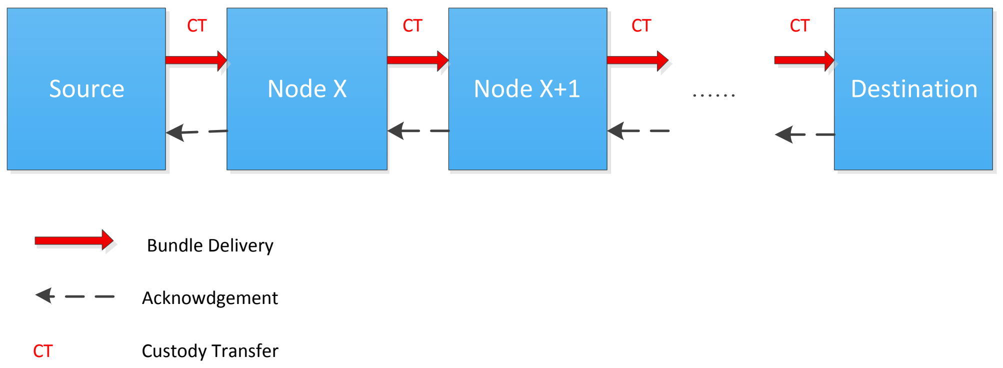


Figure 2.5 Custody Transfer

If the custody acceptance acknowledgement is sent back from the node, it becomes a custodian and is obligated to store the bundle to its local storage until another node accepts custody transfer, or the bundle TTL expires. Hence at one time, only one custodian exists in a bundle transmission process.

2.3 Routing in DTN

As all the other communication networks, DTN is able to select routing paths and transport data from a source to a destination. Differently, DTN also have to overcome the problems associated with intermittent connectivity, long or variable delay, asymmetric data rates, and high error rates by using a store-and-forward message switching mechanism. Figure 2.5 simply shows how this approach works.

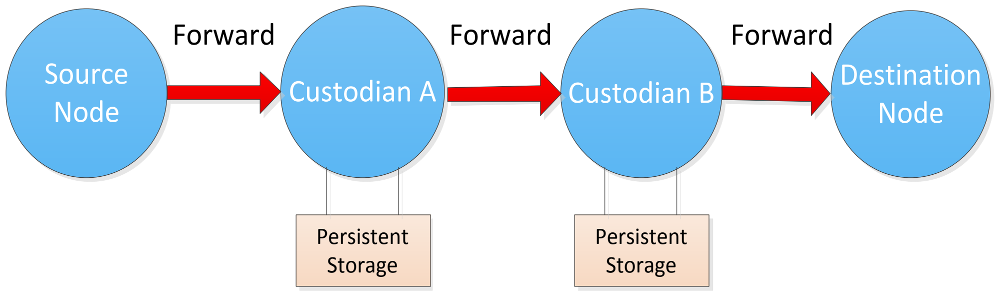


Figure 2.6 Store-and-Forward Message Switching

Store-and-Forward method is also used in today's email and voicemail systems. By adopting such a routing technique, data is stored throughout the network and forwarded when a better chance comes in hopes that it will eventually reach its destination. The local storage can keep messages indefinitely if there is enough storage space. When a communication link to the next hop may not be available for a long time, or when retransmission is needed if an error happens, adopting Store-and-Forward message switching mechanism can maximize the probability of successful transfer. [15]

2.4 Mobility Models

In a DTN-based scenario, most of the participants are mobile nodes. Very often, the research work needs to be carried out before it becomes totally successful. However, a real-world test bed for research purpose is quite hard and impractical to achieve, not only because of the difficulty of deployment, but also the unpredictable cost according to various project scales. Therefore, other approaches are needed to make this easier.

Mobility model [16] emulates the real-world movement pattern by extracting the characteristics out from the real life performance of mobile users and different objects. It plays a very important role in evaluating the protocol performances and makes some research work possible to be launched.

There are several categories of mobility models including Random Models, Models with Spatial Dependency, Models with Temporal Dependency, Models with Geographic Restriction. Choosing appropriate underlying mobility model is crucial to reflect the real life scenes in a reasonable way. Therefore, there is a real need for a deep understanding of mobility models and their impact on the protocol performance before the choice is made for simulation. Understanding some of the basic mobility models, for example Random Waypoint Model, Random Walk Model, Random Direction Model [16] etc. is quite helpful to make the comprehensive decision when conducting the simulations.

Unlike Internet, DTN has very limited resources deployed at present. It makes real-world mobility even not scalable enough for researches and studies. Relying on the mobility model in a simulation environment makes it flexible to command, also mirrors reality especially when the scale grows.

2.5 Security Considerations

Because of the high mobility and infrequent connectivity of entities in DTNs, the node density is relatively low. Thus with scarce network resource in an extreme circumstance, it is not acceptable if some unauthorized users flood the network with spam traffic easily and occupy the modest resource. That's why security service plays a crucial role and is always needed in a DTN communication. The specific security requirements and implementations vary depending on the corresponding environments and applications.

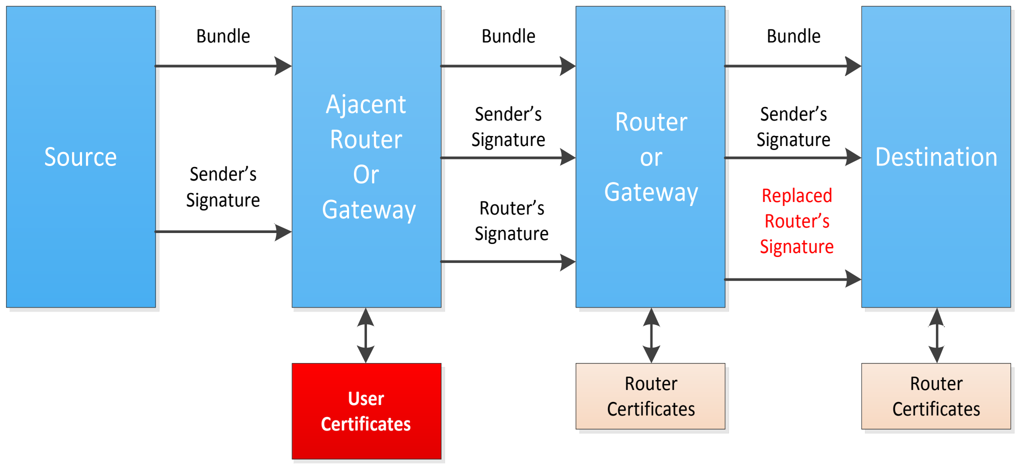


Figure 2.7 Security Steps of DTNs Using Public-key Cryptography

In most security mechanisms in the traditional networks, only user identities and the integrity of message are authenticated. In addition to that, routers and gateways are also authenticated in DTNs. Sender information is authenticated by these forwarding nodes so that network resources can be conserved by preventing the carriage of prohibited traffic at the earliest opportunity. [4]

2.6 Summary

The background and principle of DTN technology are introduced in this chapter. As an overlay designed especially to deal with communication issues in extreme environments, the DTN architecture is constructed in a very different way from the traditional network. Instead of making assumptions like rare packet losses, continuously existing communication path, interruptions and delays are tolerant by introducing Store-and-Forward mechanism. The bundle layer, as the core of DTN architecture, makes use of BP to communicate. Functionalities and working processes of different BP elements are addressed. Based on the fundamental concepts, routing mechanism is introduced with illustration of the difference compared with the traditional networks. Mobility models, required for the simulations, are also briefly presented together with the motivation why they are needed. Finally, we have discussed the security issues as it is quite different from the Internet-like networks. In next chapter, we will introduce Floating Content, the DTN-based information sharing service. The service model, system operation, protocol will be discussed. In addition to that, the simulations will be run and the API specially designed for Floating Content service will be analyzed.

Chapter 3

Floating Content Service

Floating Content is a DTN-based localized information sharing service exclusively dependent on mobile phones and other mobile devices. It takes advantage of wireless interfaces such as Bluetooth or WLAN to communicate in ad-hoc mode.

The content items can be created with customized parameters including an anchor zone, comprising geographic origin and validity radius, within which content is meaningful, and also a TTL. Other users who are interested in the content will receive a copy and further replicate within the anchor zone of this item. During the validity time, the actual lifetime and spreading of information depends on the number of interested nodes. As long as there are enough mobile nodes within the anchor zone willing to replicate and store a copy of the content, it floats. It's also possible that information disappears if there is not enough nodes around accept to take a copy or the content creator steps out of the anchor zone. In addition, once the TTL of a message expires, all the copies in other nodes will be deleted together with the original message.

The mobile devices implementing Floating Content service need to be location-aware such as adopting GPS which is built as one of the most basic functionalities in today's most mobile phones. However, because of its opportunistic nature, there are no strict limits on the error range and accuracy of the returned position data. Sufficient storage capacity is required to be available, and this is quite easy to achieve for normal mobile phones by simply setting aside a fraction of its built-in storage or expanded memory card capacity. Roughly synchronized clocks are needed to determine the status of the message, available or time-out.

It has been proved theoretically that Floating Content service can effectively solve the privacy and security related problems existing in traditional networks. Also practical simulations have confirmed the feasibility in urban environments even with modest number of mobile nodes supporting this application. [5] [6]

3.1 Service Model

All the users are assumed mobile nodes, and according to the nature of DTN, no infrastructure is provided to support this system. Mobile phones with wireless interfaces (Bluetooth or WLAN) and certain amount of storage space are used to as communication entities. Every single node may be able to get a copy of a specific message item which interests them.

Each message item has a defined anchor zone, a real world area within which the item is available. The users who create information items within a defined anchor zone and TTL must reside inside the anchor zone until they finish the message sending task. The anchor zone is a circular area defined by specifying a center point and a radius. Figure 3.1 shows an example of an anchor zone and nodes.

As the above figure shows, a node generates an information item and interested nodes keep copies floating around in the anchor zone by opportunistic communications when they meet. There is no guarantee that the items will not disappear, instead, it is explicitly allowed either when the node density is too low or when the item creator leaves the anchor zone. In addition, the information item is attached with a life time TTL and will be discarded when it expires. The mobile nodes need to be able to determine their position, e.g. by using GPS, cellular base stations, triangulation-based methods using WLAN access points, or any other approaches providing reasonable location results, thus making judges related to the anchor zone.

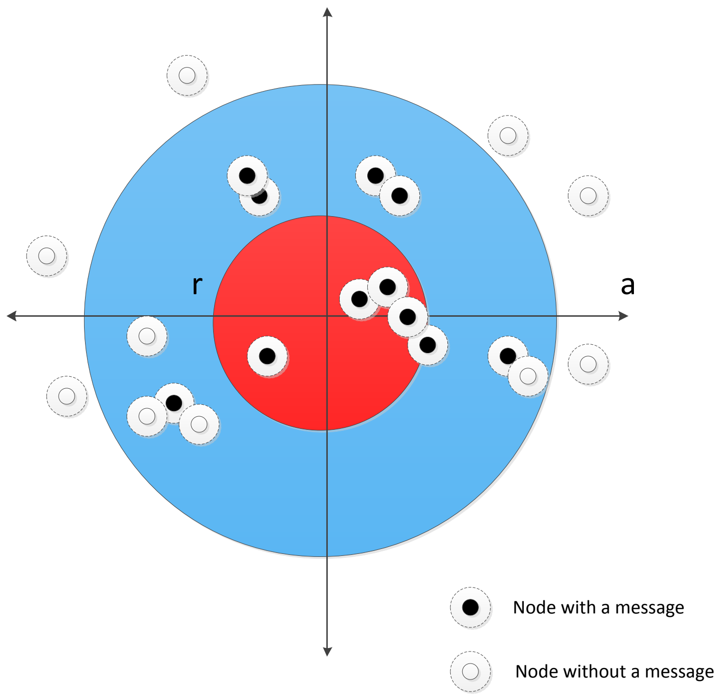


Figure 3.1 Anchor Zone of Floating Content Message

3.2 System Operation

A node creates an content item I of size S_I with a certain lifetime (TTL) and specifies an anchor zone by defining its center P and two radiuses, a and r, as shown in Figure 3.1: a represents the availability range within which the information item is kept alive with limited probability, no copies of the item can exist outside of a; r denotes the replication range within which nodes can replicate the item to others when they encounter. When two roaming nodes N_1 and N_2 meet in the anchor zone of an item I, assume N_1 has I while N_2 does not, then N_1 will replicate item I to N_2. The replication mechanism is completely based on the location of nodes, thus in the simplest situation, all the nodes in the anchor zone should have a copy of the item. Nodes are unrestrained to delete their copy of the item if they are outside of the anchor zone.

In practice, the replication and deletion process takes place as follows. Assume that node N_1 having the item I, with an anchor zone defined by center point C and radius r and a. The distance between current location of node N_1 and C is denoted by d. When node N_1 meets node N_2, N_1 will replicate item I to N_2 with a replication probability p.

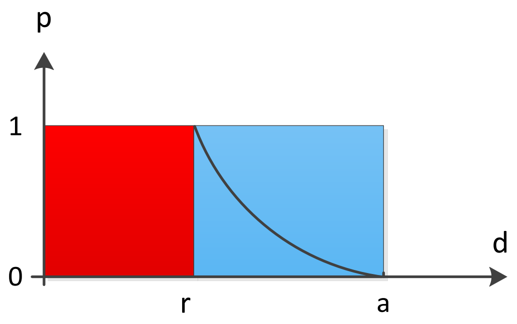


Figure 3.2 Replication Probability According to Distance from Message Center

The replication probability P_r belongs to [0, 1], and is some (decreasing) function that gives the probability of replication outside the replication range but within the availability range.

The deletion probability P_d is defined in a similar way and P_d belongs to [0, 1] as well.

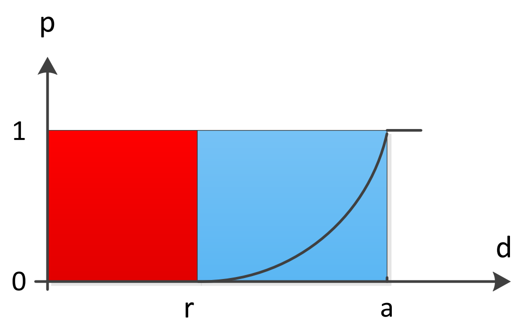


Figure 3.3 Deletion Probability

The deletion function protects buffers from filling up by using early prioritization mechanism; it is evaluated every time when a node is encountered. The oldest messages will be discarded if there is still a need to free the storage space. This definition introduces an area outside the replication range but within the availability range of the anchor zone as a buffer zone. It provides protection against items disappearing by offering transitional reduction of availability over distance especially when nodes move outside the replication range of the anchor zone for a brief moment and then return.

However, the probability is not applied for all the circumstances. One special case is when $r = a$, P_r and P_d do not make sense any longer since the "middle area" does not exist at all. [17]

Once the node steps out of the the availability area (see Figure 3.1), copies are deleted (immediately or upon encountering the next node, depending on the specific implementation).

3.3 Floating Content Protocol

Floating Content protocol as an application-layer protocol, defines the message formats and rules for exchanging Floating Content messages among the mobile nodes. A Floating Content message m is identified by a unique message id ID_m and carries its anchor zone parameters including P_m, r_m, a_m for an anchor point, its lifetime T_m in its headers, and the size S_m of content item I_m is in the message body.

Floating Content protocol specifies 4 phases to be followed when communicating as follows:

1) Nodes integrated with Floating Content service keep sending discovery beacons to their neighbors to discover peers.

2) Once a peer is found, summary message will be sent back from the discovered node. Each summary message is made up of a vector of available content items that can be replicated. For every single message item I_m in the vector, information of ID_m, S_m, (P_m, r_m, a_m) and T_m is included. The summary may be limited to what fits into one MTU size packet. If more content needs to be shared, the list of content will be spread across multiple summary messages in a round-robin fashion.

3) When the summary reaches a node, it knows what the neighbor has and can send a request for one or a subset of the content items. There is possibility that the node prioritize the replication order of messages according to certain replication policy, for example the message with the largest TTL will be replicated at first, or firstly copying the nearest message, etc. The actual replication order is decided by the specific implementations. Five policies are defined in [6]: FIFO keeps the

message order in which the messages were created/received, RND randomizes the message orders, and three further algorithms use ascending order by anchor zone, floating volume, and the product of volume and TTL. These algorithms are named as Smallest Area First (SAF), Smallest Volume First (SVF), and Smallest Total resource consumption First (STF).

With the defined replication order, messages are then exchanged as long as the connection is still on. If the nodes lose contact, the receiver will have to discard all the incomplete messages. Or if the batch is completed, the service will return to phase 2).

Message exchanges take place bidirectional and simultaneously. Protocol itself allows message exchanges with multiple nodes at one time. However, some link layer technologies may have restrictions for this. When the message exchange is happening, nodes still keep sending discovering beacons so that new reached nodes are also able to be discovered for message exchange. Nodes are allowed to append messages which they received just now to the summary list, offering to other nodes those are exchanging messages with them. The messages are deleted according to the deletion policies. With D(h) = 0, no messages will be deleted within the anchor zone α. Outside α, deletion approach depends on the mechanism adopted. For immediate deletion, messages are discarded once the nodes leave α. For upon-encounter deletion, the message deletion only happens when the next node is encountered, and nodes are allowed to wander out of α and back in a while with the message still kept. From an implementation perspective, the upon-encounter deletion policy is more sensible as it is triggered by an external event.

3.4 Simulations and Evaluations

All the simulations have been carried out by using ONE simulator [18] for feasibility in a static and idealized mobility settings as well as availability by introducing more sophisticated Helsinki City Scenario (HCS) based upon a city map (4500 m ×3400 m) of downtown Helsinki for evaluation. HCS models two types of nodes: most roam the city area following streets and walkways when moving to randomly chosen points on the map following a shortest path using pedestrian (0:5 -1:5 m/s) or car (10 - 50 km/h) speeds; some follow a set of three predefined routes as trams with their own characteristic speed (25 - 35 km/h). In addition, the location error is introduced in the sophisticated scenario for evaluate the effect it gives to the overall performances of Floating Content service in a simulation environment.

3.4.1 Feasibility Validation

By using a 2000 by 2000 m size map with certain amount of mobile nodes which are initially placed at random positions, the simulations are launched to evaluate the general feasibility of Floating Content system. Nodes move according to a random waypoint model in varied speed. Once two nodes meet, they are able to transfer Floating Content items from one node to another. Routing module is built upon the Active Router which provides message generation and logging functions.

As a crucial parameter, the node number (both static and dynamic nodes) might affect the feasibility a lot. If the required node density is too high, the condition is not likely to be encountered in a real life scenario. Simulations have shown that the node density required for 80% availability (availability is the ratio of nodes in the anchor zone with the item to the total number of nodes in the anchor zone) is around 1 node per 50 square meters, which is very easy to achieve in most typical urban environments. [5]

Mobility helps most especially in those cases when only quite limited number of nodes exist for communications. As the number of nodes increases, the fraction of mobile nodes becomes less important, even a small fraction of mobile nodes is sufficient for good availability. With certain configurations, the mixing mobile and stationary nodes can even give the better availability than having only mobile nodes.

Assuming the communication range is a perfect circle, simulation environments with the same ratio for anchor zone radius and communication range (anchor zone/communication range) behave similarly. However, this does not apply to the stationary case.

In total, the evaluation results have shown that the Floating Content service is indeed feasible and the requirements can be easily reached in a typical urban environment

3.4.2 Performance analysis

With feasibility discussed in 3.4.1, it is worth knowing more about how different parameters can affect the performances and figuring out the reasonable configuration settings which are the most preferable for real world implementation.

Most of the simulations are conducted with two different anchor zone sizes: a = r =200m or 500m. Fixed anchor locations are chosen by dividing the map with 200m by 200m squares to evaluation the Floating Content service in downtown Helsinki. Thus 352 anchor points are generated in the map shown as red crosses in Figure 3.4.

Each simulation lasts for 24 hours with the TTL set as one hour and a corresponding cooldown period of one hour as well. Two main scenarios are used for evaluation with mean values informed. The first one is across all anchor zones in the entire map, using an average weighted by the number of messages generated per anchor zone. The second one is restricted into a core area of 1.8km by 1.8km, including diverse terrains (including water front) but no boundary effects due to simulation area limits are expected.

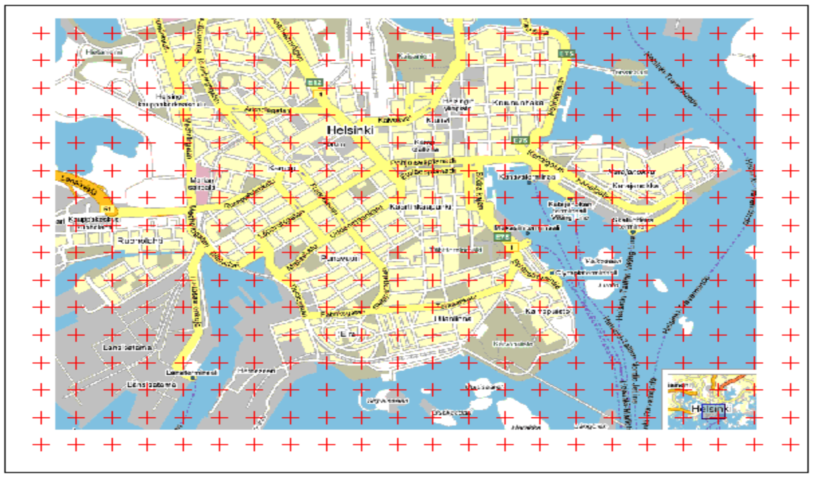


Figure 3.4 Helsinki City Scenario

The evaluation results show that even with very limited number of nodes supporting Floating Content service, it still can be feasible in the typical urban environments.

It is the users' decision what size the Anchor zone should be set. However, anchor zone are supposed to be large enough to match the node densities so that they together can make the content "floating". As a result, it is not recommended to tightly constraint the anchor zone size unless the node density is extremely high. Indicatively, it seems that anchor zones should span multiple blocks in a city for sufficient replication, but a systematic exploration is for further study.

Meanwhile, it also demonstrates that Floating Content service has limitations in the border areas such as at the waterfront where the nodes are not able to move in and out normally.

With the given location, it is possible to estimate the probability that the content will float in the simulation model. This makes it possible to know the floating expectation with the given input parameters.

Although there are limitations from synthetic mobility models (and of HCS), the macroscopic mobility and social context and interaction patterns may be of lesser significance for localized content sharing—unless selective support based upon (closed) groups or content channels come in—so the findings should also be able to apply for other scenarios. [6]

3.4.3 Location Accuracy

As a location-based service, one issue must be taken seriously is the location accuracy and location errors which may affect the performances of the whole system. Nowadays, GPS is the mostly built-in location system especially in mobile phones. Hence taking GPS as an example to study the location error has a significant meaning.

We have seen that the entire system of GPS is dependent on a network of 24 satellites orbiting the earth. While research and development work is still going on to develop more and more accurate systems, it would be a good idea to understand what the external sources of error are. Selective avalability, Satellite geometry, Satellite orbits, Reflection, Atmospheric effects, Clock inaccuracies and rounding errors, these are the main factors that affect the location accuracy.

Those random variables mentioned above that make up the error in a GPS position, including errors from multipath, ionosphere, troposphere, thermal noise and others. Assume the absolute values of different variances are independent and identically distributed, by the Central Limit Theorem [19] you might expect that the absolute value of GPS location error (for the following part, we will say only location error instead of the absolute value of location error) has approximately a Gaussian distribution [20], and indeed this is the case. With the error introduced into the ideal scenario, we carry out our simulations and make a comparison to see how different it acts in these two situations.

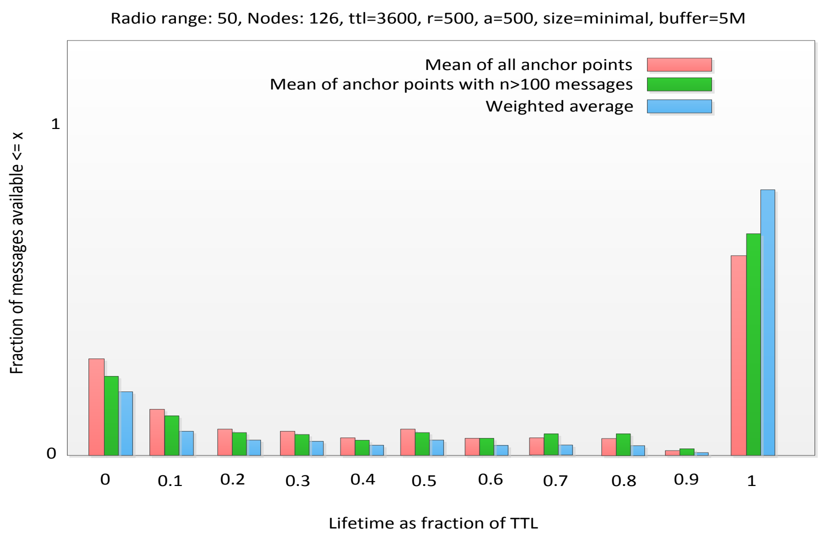


Figure 3.5 Performance of Floating Content Service without Location Error

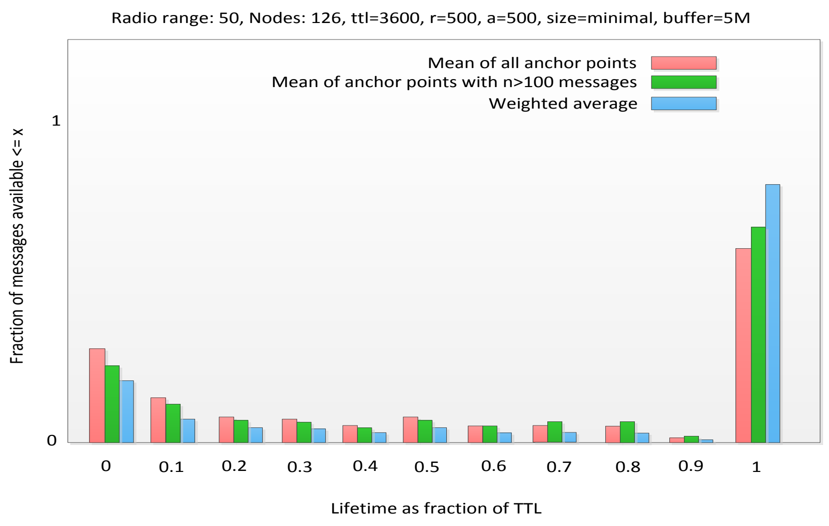


Figure 3.6 Performance of Floating Content Service with Location Error

Our result so far has shown that the location error is Gaussian distributed with assumption that all the error source variances are independent and identically distributed.

From the simulation results we can see there is barely difference with location introduced into the ideal simulation scenario.

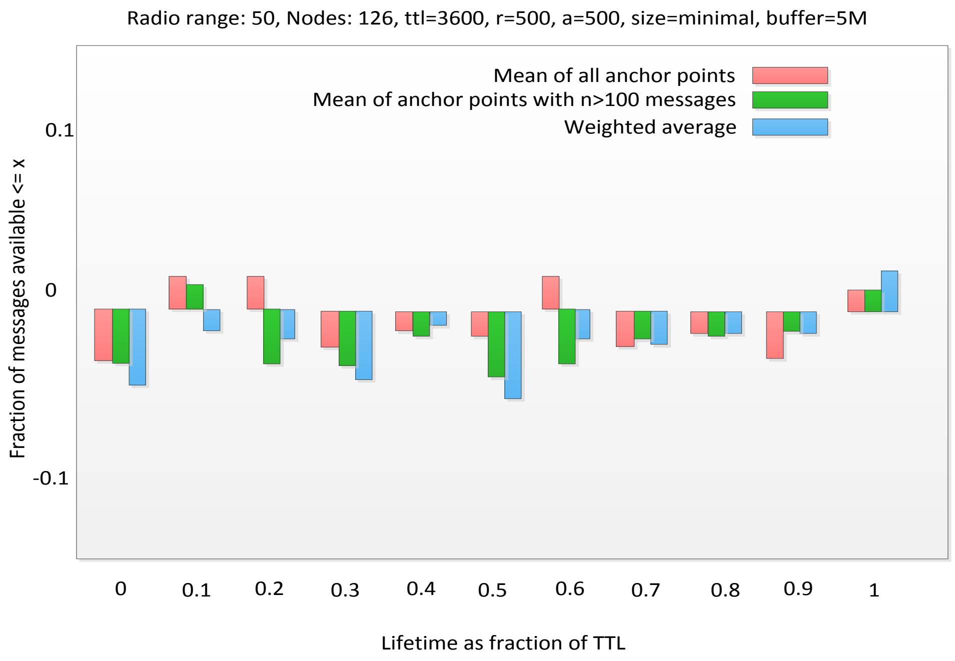


Figure 3.7 Compared Floating Content Performance as Function of Lifetime

With an even closer look at the difference between simulation results of the ideal scenario and the location error included scenario, we are more certain that the location error has very limited effect to the performance of Floating Content application. The reason it not hard to see, that when calculating the distance between two nodes, the location errors are basically canceled out. Thus no effects are created to determine the location distance when publishing a Floating Content message.

3.5 Application Programming Interface

With proved feasibility and performances according to different parameters of Floating Content service in a simulation environment, it is more meaningful to take it further into real implementation which can be realized in mobile devices such as mobile phones, thus serving for people's convenience. An API, a source code-based specification intended to be used as an interface by software components to communicate with each other, is used for this purpose so that developers and anyone who is interested in Floating Content service can build applications based on it.

The API is sitting upon a core DTN functionality layer to provide Floating Content service to the application developers. It offers access to the store-and-forward task like all the other DTN-based services. The core part of the API is an application engine that can be always running in the background once the application is started. Applications adopting this API will be able to inherit a series of functionalities including listening, queuing and sending Floating Content items that have been realized already. Also, the developers can extend these functionalities according to their needs.

As long as the application has been started, it keeps discovering available recipients and listening to the incoming messages simultaneously. Once discovered, it will try to get connected with the peer. Through a successful connection, a copy of the message can be transferred towards the recipient according to the request. If the certain message service is subscribed, the message will be delivered to the user for further use, otherwise, it will be ignored. This service filtering process is carried out in the background. How different parts of message information will be used is decided by the specific implementation.

Before a Floating Content message is sent out, the message item is firstly encapsulated in a changeable container with several predefined default values such as TTL etc. Extra information is allowed to be added into the message item according to specific implementations. The message instances are then passed to the local application engine and further encapsulated as a task for being ordered in the task stack and then wait for its time to be sent out through socket. The engine keeps searching for the pending tasks and order them according to a priority mechanism that makes a decision about which will be sent firstly, which will be discarded when the buffer is full. If there is no task needed to be handled at the moment, it will wait until a new one comes without interrupting other ongoing operations.

Considering the nature of opportunistic network, the applications may suffer from frequent disconnections. If it happens during the process of message transferring, the incomplete message will then be discarded. In another word, to ensure successful message transferring, both the recipient and the creator will have to be in the anchor zone of the message.

3.5.1 Primary API Functions

A. subscribe (*String* service)

Subscribe to a specific message service so that the application can receive the message belonging to the exact service. Each Floating Content message is attached with a service. The service set is recommended to be predefined. However, there is no strict restriction for using self-defined services from the users. User can choose which one(s) to follow. After subscribing, if the application engine receives messages from others, the messages with subscribed services will be delivered locally to the user.

B. connected (*Application* app)

As long as the application is started, the client will try to connect to the server. Once the connection has been successfully set up and the client has verified that the server supports the right API version, this method is invoked. Normally the application would like to start exchanging messages right after the connection is established. However, the particular action being taken when this method is called is totally decided by the specific implementations.

C. messageReceived (*Application* app, *String* service, *Message* msg)

Once a coming Floating Content message belonging to a certain service category is received, this method should be called. It is required that only if the client has subscribed to the message service before. The message normally will be stored somewhere (local storage, cache, etc.) in the node for later use. However, same as the "connected" method, the particular action being taken when this method is called is totally decided by the specific implementation.

D. publish (*Message* msg, *String* service)

Publish the Floating Content message to a given service. Publishing a message does not mean sending it out right away. Instead, the application engine will encapsulate this message as a PublishSenderTask and then put it into the task stack. When it becomes the task's turn in the task queue, the message will be sent out.

E. send (*OutputStream* out)

One mobile node (client) sends data to another mobile node (server) through a given OutputStream. There are several different implementations for sending different data types. All the sending tasks are queued in a task stack and each of them has exclusive ownership of the stream for the duration of the send call. Once returned, the task must not keep using it anymore.

F. hostDiscovered (*Application* app, *String* host, *int* hopcount, *long* timestamp, *double* longtitude, *double* latitude)

As long as the router finds a new peer nearby, this method should be called. By taking advantage of the input parameters, the application is able to make a decision whether or not to make a message copy to the new peer based on the predefined Floating Content anchor zone concept. As the "connected" method and "messageReceived" method, the particular actions being taken when this method is called is totally decided by the specific implementation.

G. get functions

There are a series of general methods for all the get-related operations. They are located in the message model objects. When handling the received Floating Content messages, these methods can be used to acquire the metadata encapsulated in them, for example the lifetime, file, AppTag, text message content etc. The values can be fetched both separately and as a whole package.

H. set functions

There are also some general methods locating in the Floating Content message model objects for all the set-related operations in corresponding to the getters. When create a Floating Content message, they are used to set up the necessary values for different parameters on one side. Then they can be picked up on the other side. This set and get procedure is the actual meaning of communications.

3.6 Summary

In this chapter, Floating Content service is introduced in detail from several aspects. The concept was firstly introduced, followed by the service model which illustrates how this service works in a general way. In addition to that, the detailed system operation processes and the protocol principle are described. This introduces how this protocol works in an operational level. With the fundamental knowledge, simulations are carried out to validate the feasibility, performance and how much the location error can affect it. The feasibility and performance simulations yield the similar results as in [5] and [6]. This was done mainly to calibrate my simulation system before I move on to the error estimation. The results show that the service is totally feasible even with a modest number of mobile devices installed this service. Performance is decided by several parameters, and with the ideal settings, the output is quite good. From the simulation's perspective, location error has very limited effect on the performance of Floating Content service as shown in Figure 3.5, thus the users do not need to worry much about it. At last, one API designed for Floating Content service is introduced. By taking advantage of its interfaces, developers are able to build applications easily. In next chapter, we are designing an application on the basis of this API and implementing it on Android platform.

‘

Chapter 4

Application Design and Implementation

Based on the previous simulation results and well implemented Floating Content service API, the application is ultimately designed and realized on the Android platform [21] [22] to provide an unprecedented opportunistic information sharing experience in urban areas.

The application is implemented based on Android source code version 2.1, Android OS older than 2.1 (Eclair) [23] may not be totally supported. Once the application is successfully installed in an Android phone, users may make use of this application to create and receive Floating Content message M at any location L with a predefined TTL and anchor zone by specifying its location center P and two radiuses, a and r. In our design, we try to avoid the deletion probability concept and make a and r equal value. Thus the so called P_r and P_d in Chapter 3 do not make any sense. As shown in Figure 4.1: r represents the availability range within which the information item is kept alive with limited probability, no copies of the item can exist outside of r; r also denotes the replication range within which nodes can replicate the item to others once they encounter. Together with TTL, they can affect the current status of the Floating Content message. Each message has an associated service, S, to support interest-based subscription and information filtering. M will be deleted automatically when TTL expires or the message holder steps out of the validity range. Manual deletion is also allowed by the users when they feel not interested in some messages anymore.

Nowadays in a mobile phone, especially smart phone, built-in GPS receiver is a ubiquitous component. In addition to this, quantities of free and paid positioning applications are available in the application market to assist the GPS receivers. During the whole process of Floating Content communications, GPS is responsible to provide nearly real-time location information with certain error. Specific accuracy varies according to different manufacturers and costs of the corresponding devices. However, according to the simulation results in Chapter 3, positioning error from GPS has quite limited effect to the performance of Floating Content service.

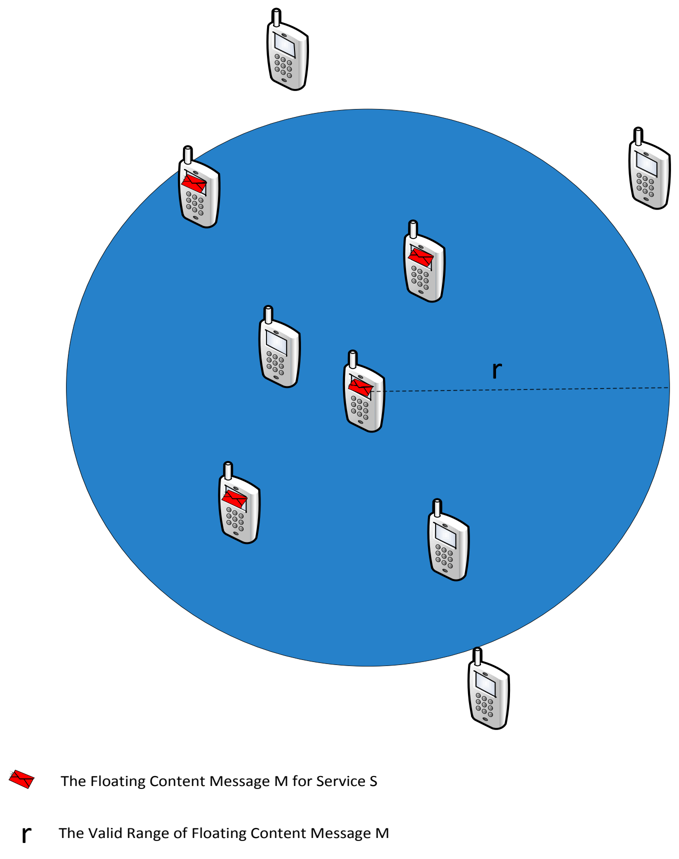

r
The Floating Content Message M for Service S
r The Valid Range of Floating Content Message M

Figure 4.1 Floating Content Application Scenario

Once a Floating Content message is created, all the mobile users within its valid range will have a chance to receive a message copy as long as they are subscribed to the exact service of the specific message. The application will not have the message delivered to the front end but ignored if the recipient is not subscribed to the service. All the copies will share the identical parameters with the original one, including AppTag, Service, Timestamp, TTL, Message Content, and Attachment.

When the users are reading a message, they can save it to their local storage for later use if they find it very useful or feel like further forwarding it to the people around. When they find it useless, manual deletion can be executed. None of these actions from recipients will do any changes to the original message and other message copies. They are discarded only when TTL expires or the message holders are out of the valid anchor zone of the corresponding message item. Once a message is deleted by its creator, it will not be further copied anymore. The existing copies will be dropped when TTL expires or the message holders delete them manually. As time goes by and as the nodes move around, the “floating” message will “sink” in the end.

Each Floating Content message is identified by a unique message ID. So once the message has been received, it won’t be replicated anymore. This more or less helps protect users from spams. However, there is no guarantee that someone won’t send the same message for multiple times by recreating the messages with a different message ID. Instead, we consider the subscription mechanism and location-aware feature has already decreased this potential crisis.

Note: All the applications share the same port number 7744 for setting up connections and message transfer, this number is used only for prototyping. The specific port number must be decided during the process of development and make sure it is not a frequent used port number on your mobile phone by other applications.

4.1 Application System Architecture

A service engine keeps running in the background ever since the start of the application. The key functionality includes: discovering new peers and updating the data information of existing peers; listening to and receiving the incoming messages; sending the messages encapsulated and queued in the task stack; automatically updating the local database and the user interface periodically; accepting subscriptions from the user. [24]

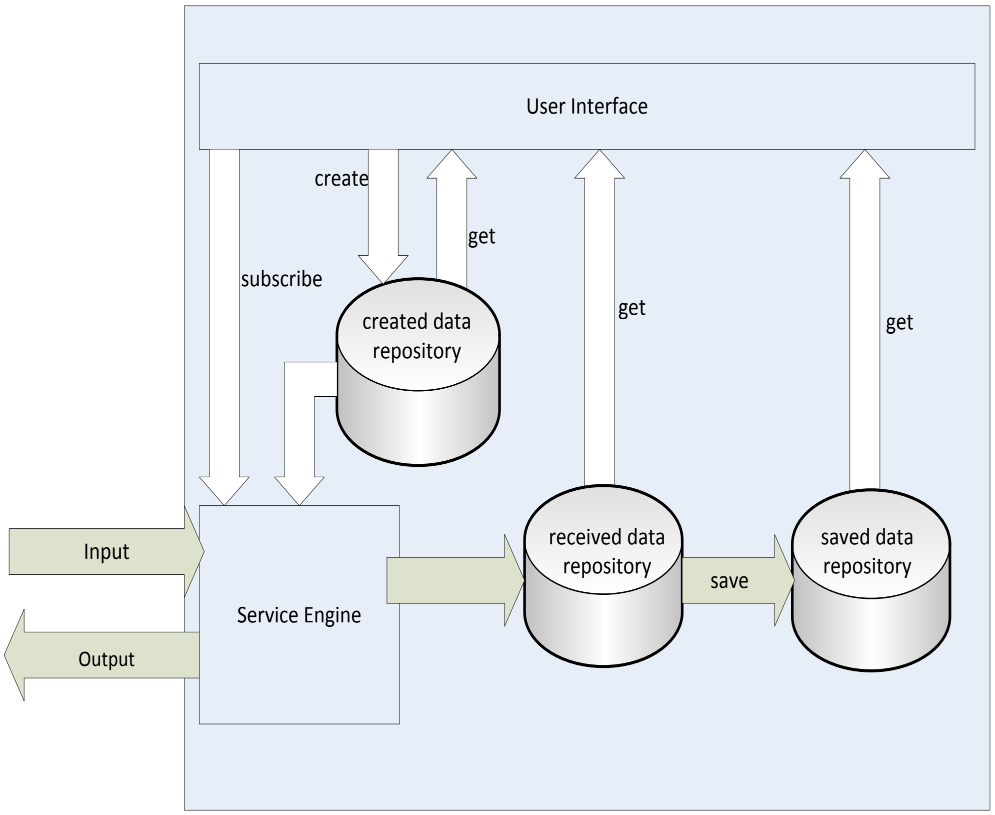


Figure 4.2 System Structure of Floating Content Application

Figure 2 illustrates an overall structure of a mobile node. Such a system offers reliable communication mechanism: sending message to any mobile phones around, and the message content is possible to reach to other users if the receiver forwards it within its lifetime.

The local data center is made up of three individual database tables and storage space, separately used to describe and keep created messages, received messages and saved messages. Separate data centers work as a whole but make the message operations more flexible. Android devices normally provide both database and storage support thus making the data operations quite convenient and greatly reliable.

Creating a Floating Content message makes an insertion to the created data center. Also the file included in this message will be copied to the created storage space. All the created messages share the identical parameters with their recipients. Content can be deleted, and it will not affect the messages copies in their defined lifetime. However, if a user deletes an own-created message, the message will not be replicated anymore. Once the TTL of the created message expires, all the copies including the original message will be removed from their data centers automatically.

A Floating Content message acquired from other mobile node will show up in the specific part of the data center for received messages if the recipient has subscribed to the exact service attached to the received message. They can be read by the user from the user interface. If the service is not subscribed, the message will neither be inserted into the received data center nor be delivered to the front end but simply be discarded. This process is carried out in the background and user will not even notice it.

In addition to that, the received messages can be saved for further use when the recipients are reading them. Once saved, TTL will not be able to affect the status of the messages in the saved data center anymore. However, this will not affect the operations in the received data center. Any message in the received data center will only be available when the left valid time TTL is a positive value. Expiration will lead a deletion of the certain received message copies, removals from both database record and the actual attachment file from the local storage. Manual deletion is also allowed for received copies. However, the action does not affect anyone else except for the copy holder itself.

Users of the application are able to subscribe to certain predefined services attached to Floating Content messages. Once subscribed, the application will be able to deliver certain messages to the front end. If no service is subscribed, the messages attached with default service will be delivered to the users.

4.2 Message Format

The basic format of a message used in the Floating Content Application is shown in Figure 3 below.

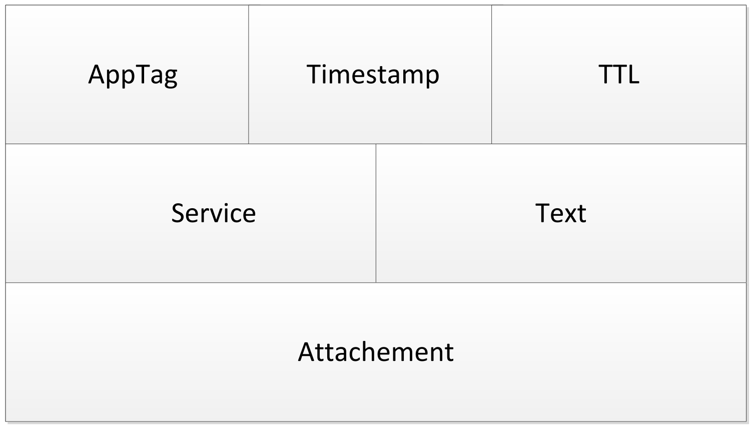


Figure 4.3 Message Format of Floating Content Application

Each part of the message is a variable of a certain data type. There is no limit to the length of any message part. The octets are shown here for convenience in representation.

AppTag is the unique identifier to distinguish Floating Content messages. It can be used to refer to the specific messages in the data centers. Each message the application receives will have the AppTag checked to avoid duplicated messages.

Timestamp is the system time when the message is created and sent. It can be used to uniquely identify the message and as an index for the database operations. Timestamp is a “double” type value.

TTL (Time-To-Live) is the time period length after which the corresponding message will become invalid. The bubble color will change along with the decrease of TTL. It is an “int” type variable without limit. However, for the Floating Content service, the larger TTL will lower the receiving priority of messages.

Service is an interest-based parameter used to subscribe only for interested categories of messages, thereby cut down the unnecessary and unwanted information. Plus, different bubbles associated with different services will show in the window according to the message categories. Service is a variable value of String type.

Attachment and Attachment Name together defines the file-specific part. User can attach the files from local storage and send it out together with the text message. Currently only image attachments are supported.

Message is the text message content as typical messengers use. There is no limitation to the text length. However, it is not encouraged to make the content too lengthy.

4.3 Application methods

Based on the Floating Content API, we make a further development and create essential functions to be compatible with the Android environment.

A. send(*Message* msg)

   Create a Floating Content message with an expiry time limit and anchor zone. Insert the message data it to the *created* data center. The parameters include a unique message ID, data payload comprised with text message content, attachment file and information related to the file, the lifetime length TTL, and a message service. Right after this method is invoked, the application will make a copy to anyone else it meets within the anchor zone of this message.

B. delete(*String* appTag)

   It is a local operation used to remove messages which are not any longer useful to the users from a specific local data center. The deleted messages will not be available to view or for further use. Deletion of received messages will not affect other message copies as long as the messages are in their defined lifetime. However, if a user deletes an own-created message, the message can not be replicated anymore.

C. save(*Message* msg)

   Keep a local copy of any available received message to the saved data center for later use. Saved messages will not be removed from the saved data center due to TTL expiration or because the creator is stepped out of the valid message range but only be deleted manually by the user. However, this action does not affect the operations in the received data center or any other ongoing operations.

D. subscribe(*String* service)

Register the given application services in the system for filtering the incoming Floating Content messages that only match the corresponding subscribed services. There are several predefined services such as sports, movie, music, etc. that the users can choose to follow. Self-defined category is not supported at present. If no service is specified, the default service is assigned to the message.

E. get functions

When receiving messages and doing other local operations, getters can be used to acquire and display messages as well as message-related information when the user feels like looking it over during its valid time. Normally this will happen automatically as long as the application is running and Floating Content messages come.

F. set functions

When creating a message, configure the message parameters when creating a Floating Content message. Users can decide what content they are going to send and which service a message belongs to by adopting some setters. When receiving a message, there are setters mainly used for the user interface, value assignment and attachment display.

G. update

Refresh the user interface by inquiring the local data center. There are two ways of doing update. Automatically update every other certain time period. Manual update is also supported by press update button in the main user interfaces.

4.4 Action Flow

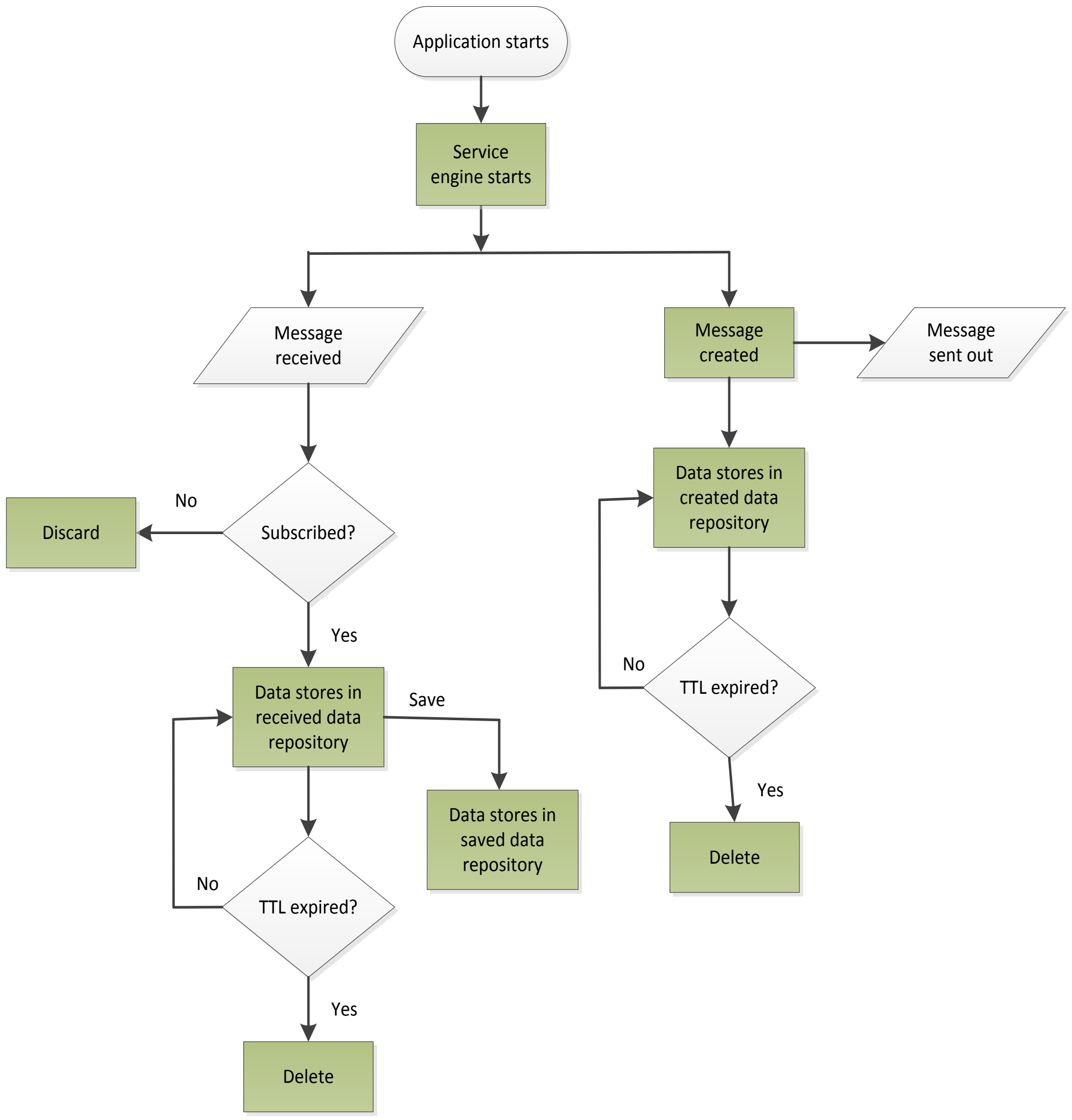


Figure 4.4 Action Flow of Floating Content Application

The figure (Figure 4.4) above simply describes the action flow since the start of the application.

When the application firstly starts, it will initialize the app engine and try to connect the server. Once they are connected, the app engine starts to receive messages, according to the subscription list stored in the local database, the engine decides whether or not to put the received message into the local buffer. The server keeps checking if the message in the buffer is expired and executes deletion. If the received message is saved, it means it is persisted in the local data repository and will not be deleted because of TTL expiration.

When a message is created, while sending it out, it makes a local copy for that message and stored in the local data repository. It will be deleted according the TTL mechanism as well. However, this is extremely convenient for resending a message.

4.5 User Interface

The importance of creating a good user experience in order to achieve better services has grown in prominence over the past years. Essentially, users need to find what they want, when they want it, with minimal complexity and to a satisfactory quality. Therefore, to achieve a good user experience, one of the most important standards is to build a user friendly interface. Apple, as a typical example, has been doing a very good job in providing smooth user experience thus achieved crucial success in the initial mobile market.

For DTNs, most research results are limited in a lab environment. Some tries were kicked off into the application market and got people’s attention through research conferences and workshops, such as GRID [25], a DTN-based local social application.

The UI design differs a lot depending on the service platform the application is running on. For mobile phones, the design can be even harder than any other platform like PC or tablet because of its quite small screen size and restricted means of expressing.

The target devices are mobile phones with different screen sizes. On small devices, a list of messages make it hard for the users the find what they want when they need. For those who have an eye-sight issue, this becomes even worse. In addition, considering the characteristics of Floating Content service, especially the status changes according to TTL and distance from the originating point, the list-pattern UI can affect the expression of Floating Content service in a bad way.

Based on those above, changing the traditional messenger UI into more visual emphasized might bring a better and easier user experience for both searching and reading Floating Content messages.

4.5.1. Design

The main idea and motivation of our user interface design is to make this application easy to use, thus introducing DTN technology into people's daily life and making convenience to them. Besides, a creative design theme can make the application look good.

Considering the service name "Floating Content" and convenient way to display messages, the whole design is based on "floating bubble" theme. Each bubble stands for a Floating Content message. Bubbles differ because of their services and message status – the distance between the current location and the originating position, the left lifetime of the message. Bubble shows and disappears in the main windows so that users can easily access to the actual message content by one or two clicks.

A. Programming Language

The programming language used for the design is Java and XML. Java is used for the functionalities while XML is used for the interface. Over two thousand lines of code are implemented. The mostly used libraries include both J2SE library (e.g. List, Map, String, File) and Android library (e.g. Activity, Service, SQLiteOpenHelper, AnimationDrawable). The specific structure and descriptions for each class of the source code are listed in Appendix A. The list and descriptions for the xml files are listed in Appendix B.

B. Different Bubbles

The bubbles are the main elements in the main tabs. Each bubble is made up with a transparent bubble and a colored action picture in the middle of the bubble. The purpose is to make the bubble easy to recognize even when the users take a quick glance at the interface. The action in the middle is decided by the message service. The color of the action picture can be green, red, or blue, according to the message status. This is also designed for the user to distinguish the messages in a visual manner.

C. Bubble Color and Message Status

Each bubble displaying in the user interface represents a Floating Content message. The message status keeps changing according to the ratio T_r between the left valid time ttl and the total life time TTL of the message.

$$T_r = ttl/TTL$$

The bubble color changes according to message status when the next update is triggered, automatically referring to the update period or manually by the user. The initial bubble color is green, and then changes into blue after 33% of the time consumed, then red when there is only

33% life time left. Once Tr reaches 0 (which means ttl reaches 0), the message is deleted and the corresponding bubble also vanishes from the user interface.

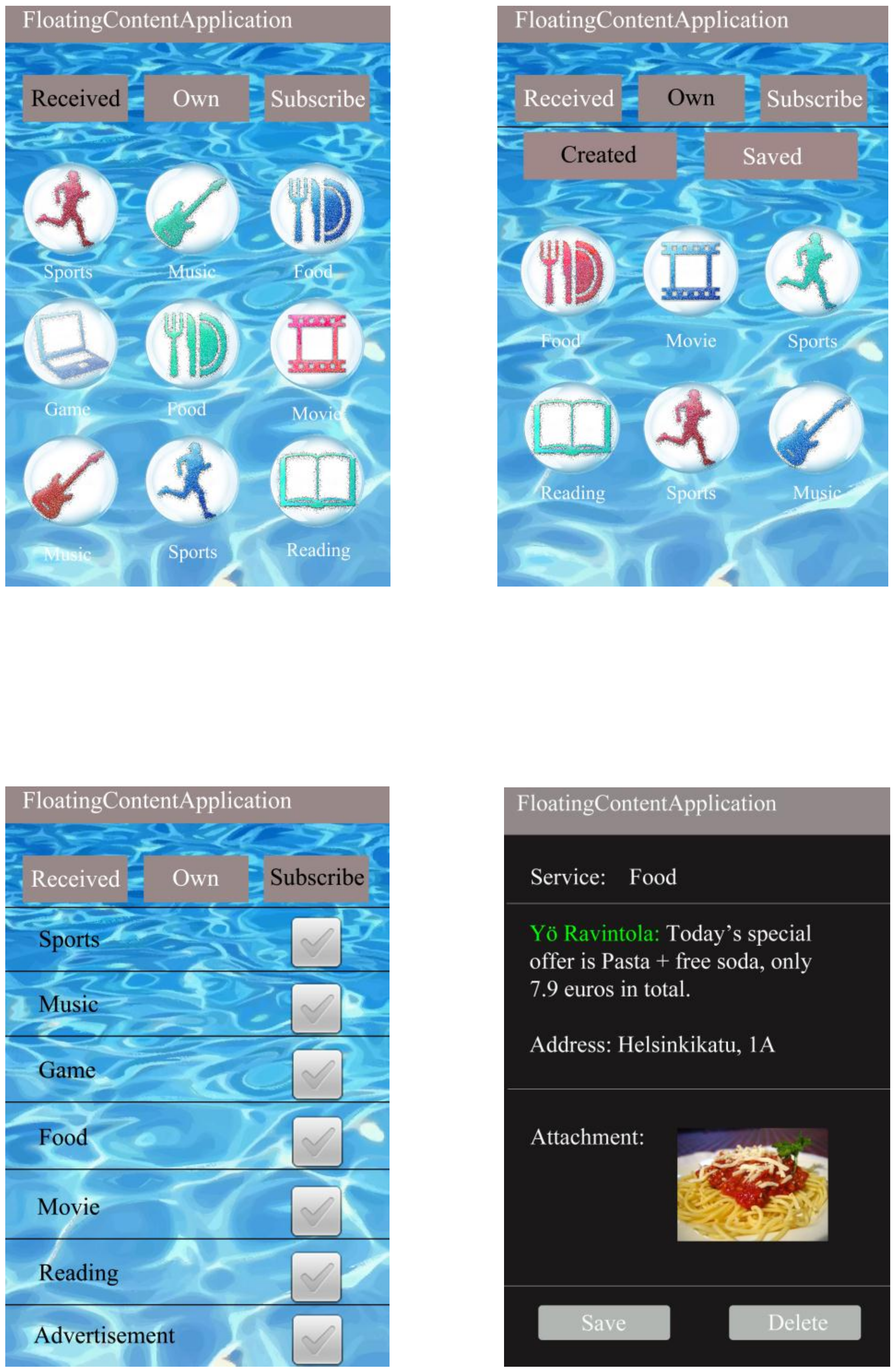
D. Usage

Creating and editing a Floating Content message is basically similar like the typical mobile text messengers originally integrated in the mobile phones people normally use. All the technical issues are hidden behind the view. Even the users have no idea what DTN is neither what Floating Content service is, they can still make use of floating bubble application just like using a normal text messengers, without worrying about the fact how the messages are transferred. By simply clicking the bubble displayed in the main interfaces, the user can get messages they have, which is more interesting than the traditional plain text interface. As a consequence, users do not need to worry the operation changes at all. Instead, it’s easier and more fun to use this application.

Due to the necessity of location information, before any Floating Content messages are received or sent, the user is required to make sure that the GPS service on the mobile phone is activated and it must keep running as long as the application is running.

4.5.2 Screenshots

Figure 4.5 shows four screenshots of the application. The first and second are the main tabs of the application, displaying messages as bubbles with animations in the background. The third is the subscription page where user can subscribe or unsubscribe to the specific service(s). The last one is the message viewer where the message content is displayed.



Service: Food

Yö Ravintola: Today's special offer is Pasta + free soda, only 7.9 euros in total.

Address: Helsinkikatu, 1A

Attachment:

Figure 4.5 Floating Content Application Screenshots

4.6 Summary

In this Chapter, we have presented our application design and implementation for Floating Content service. The system architecture is introduced in detail. The message is made up with certain message format, and the functionality of each part is studied. Some application methods can be used to operate the low layer protocols thus making it Floating Content application. We also presented the action flow of the application, showing how the application operated in the program and the relationships among different methods and operations. In addition to that, we have stated the motivation and final design of the user interface. We introduced the application specific issues and usage of it. Finally, we presented the screenshots of the main user interface. In next chapter, we will carry out the test and try to have a deep understanding about Floating Content service through a real-life scenario. Also, we are going to find out the issues we still need to improve and care about.

Chapter 5

Test and Evaluation

Software test is an essential step to provide the information about the quality of the product or service both during and after the development stage. The process includes executing a program or application with the intent of finding software bugs. Then the developers can fix them before the product is released. Nowadays, there is even a third way to do it. Some tests those define code requirements are written before the development starts. Passing the tests later on confirms correct behavior as developers evolve and refactor the code. The most famous one is TDD [26]. No matter when the tests are carried out, they are playing a very crucial role in the whole process of software development.

Before the test started, there should be a set of requirements or expected performance ready in mind or in document. Then, the developers are supposed to decide the test environment, for example in which scenario, on which platform, which tools to use, etc. Once the test environment is set up, the real tests can be started. With the test results, developers can evaluate the performance through a series of evaluation processes and based on certain standards.

In our work, the main target is to evaluate the feasibility and performance of Floating Content application in a real life environment. By activating the application in two mobile phones, we can see how the application is running and how friendly is the user experience. However, because of limited resources, the tests are carried out in a limited range of area instead of the whole city as it is in the simulation environments.

5.1 Test Environment

We adopt two HTC Nexus One [27] phones with Android 2.3 Gingerbread [28] operation system installed in as our test bed. In addition, each phone support and has originally installed SQLite [29] relational database.

5.1.1 Mobile Platform

Android is a powerful software stack for mobile devices that includes an operation system, middleware and key applications. As an open source platform, it empowers developers to contribute to its development and build applications on the basis of its API to the Android Market as well choose any kind of revenue model (freeware, shareware, ad-driven, paid applications, etc.) they want. For the manufacturers, no licensing or royalty fee is charged by Google, thereby dramatically decreases the R&D and maintenance cost.

With benefits to the users, developers, manufacturers, Google itself and the whole mobile industry, Android has been growing during the past few years and become the largest mobile platform all over the world. [30]

5.1.2 Database

SQLite is a public-domain software library that implements a lightweight relational database management system. It does not have a client/server architecture which is built in most large-scale database systems. The entire database engine is integrated into the application that needs to access a database and the database itself is packaged in a single file containing the layout as well as the actual data in different tables. Besides, from an end-user standpoint, SQLite requires nothing to install and configure.

All these simplicities make SQLite lie deep within nearly every mobile OS to store user-defined data records and to process data queries and delete. Android, as one of the mainstream mobile OSs, assuredly has a built-in SQLite database. [25]

5.2 Test Result and Evaluation

In a Wi-Fi area, two mobile phones with Floating Content service installed are used to communicate. After all, this topic is not to deploy a systematic testing environment. Therefore, only two phones are used to test the basic performance of Floating Content service in a real world environment and the user interface operations as well.

Through the test, we learn that the text messages are fully supported. The messages float in the time length of TTL defined in the message. The bubble color changes according to the validity time left of the corresponding Floating Content message. Only messages of subscribed services can be received while the unsubscribed are filtered.

Several files are attached in the messages (each message is allowed to carry one file) for testing as well. Figure 5.1 shows the data we collected from the tests, roughly illustrating the time consumption for transferring files of different sizes.

From the data, we can see that between two devices, the transmission speed using Floating Content service is similar with the actual Wi-Fi speed, and in this case, it is roughly 1MB/s. (The time recorded is in second-level, so the file smaller than 1MB takes still 1 second)

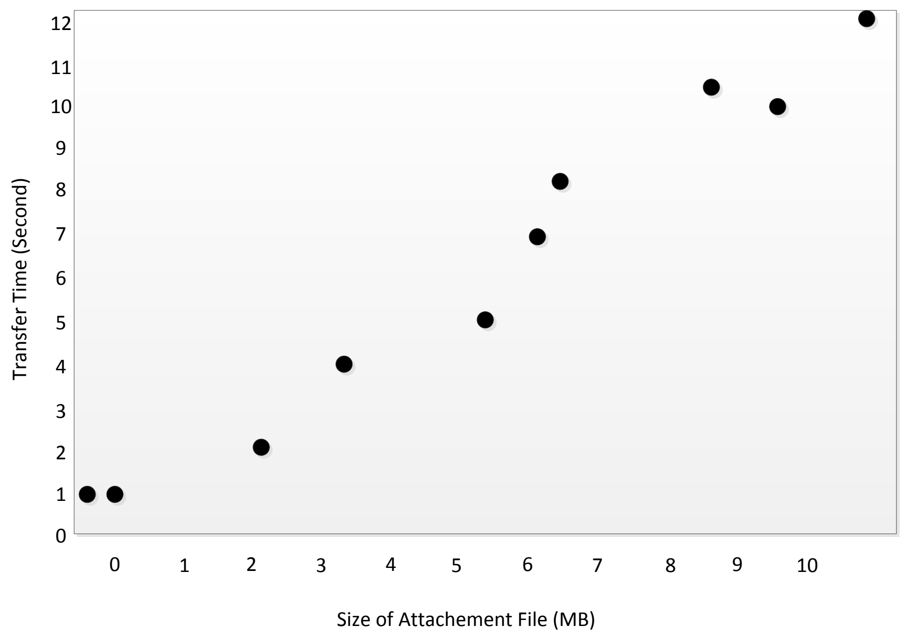


Figure 5.1 Transfer Speed of Floating Content Service between Two Devices

5.4 Summary

In this chapter, our main work is carrying out the tests to prove the feasibility of Floating Content service and learn about the user experience through practice. Since the target of this thesis is not to build a systematic testing environment. Hence we adopted only two mobile phones to evaluate the feasibility of Floating Content service in a real world environment, meanwhile, testing the operations of the user interface. The test results show that the text messages are fully supported and able to float for a certain time according to the TTL definition. The user interface is quite easy to use and convey the concept of Floating Content. The transfer speed is more or less same as the actual Wi-Fi speed while communications only happen between two devices.

Chapter 6

Conclusion and future work

6.1 Conclusion

In this thesis, we have mainly analyzed Floating Content, a DTN-based localized information sharing service exclusively dependent on mobile phones and other mobile devices. A user friendly mobile application with Floating Content service integrated is the main contribution that can benefits the users by providing an opportunistic communication mechanism even in an extreme environment, as well, decreases the security and privacy worries to the lowest. Based on the test results from various real life scenarios, the proposed floating bubble application based on Android platform appears to be user friendly and useful for information sharing especially in urban areas.

Before the application is implemented and tested, preliminary work is carried out to provide a theoretical support and fundamental conditions. Initial theoretical demonstrations help us have a certain understanding of DTN technology, an overlay designed to deal with frequent disconnections. With different assumptions from traditional Internet, DTN has a different architecture that makes the working mechanism quite different. BP, the core protocol used in DTN communications, is introduced along with different routing mechanism and security considerations.

Floating Content service, as the key concept of our application, is built on the basis of DTN technology. Based on the service model, we analyzed the system operations and studied Floating Content protocol, which defines the message format and rules for exchanging Floating Content messages among mobile nodes.

After the theoretical studies, we start evaluation and implementation. It takes us 4 major steps in total to make sure the application is feasible and user friendly.

Firstly, we run several simulations with ONE simulator [18] to evaluate the feasibility and performances by introducing different parameters and models. In addition, the location error is introduced to evaluate the effect it brings to the performance. The results show that Floating Content service is totally feasible with the easily reachable requirements in typical urban environment and the location error has very limited influence to the overall performance of Floating Content application.

Secondly, the API built upon DTN and used for Floating Content service realization is studied. With the API, we are able to further develop it to make it more suitable for our application.

Thirdly, the implementation is carried out based on the simulation results and API. The application is designed to make it easy to use and user friendly. It is implemented on Android platform because of its open source nature and absolute largest market sharing which makes it possible to popularize this application and Floating Content service.

Finally, the application is deployed in a test environment and tested. From the testing results, we can see that the Floating Content service is feasible in a simple real world environment. It is able to make use of wireless network to achieve a basic communication. The transfer speed between two nodes is similar with the Wi-Fi speed. The user interface is easy to use and able to convey the concept of Floating Content.

6.2 Future Work

This application is focused only on the implementation of Floating Content, an infrastructure-less DTN service. Infrastructure-based network is totally ignored in our case. However, taking advantage of existing Internet service would also be needed in some scenarios. Integration Floating Content service together with traditional Internet messaging functionalities can make this application even more useful. Thus users can switch between these two modes easily according to their needs in certain scenarios. Hence this would be a direction to look at in the future.

The test is carried out only between two devices in this thesis to illustrate the feasibility of Floating Content service in a real world scenario. However, this is not enough to find out the performance details. In the future work, deploying a systematic testing environment in the urban area is valuable to find out more about Floating Content service and make improvements.

In addition, this application is developed on Android OS, it would be nice if it can be implemented also on other platforms, for instance, Apple iOS, Windows Mobile OS, etc. As long as Floating Content service is implemented in different mobile platforms, even more complicated direction would be making it cross-platform. Then users are able to communicate with Floating Content service regardless of the platform issues.

Bibliography

[1] John Lovett, “Beyond Surface-Level Social Media, Using Analytical Assets for Generation-Next Marketing, March 2012.

[2] Alcatel-Lecent, “Alcatel-Lucent Eco-Substainable Wireless Solutions, Leveraging the value of green”, 2009.

[3] Abbas Jamalipour, “An Introduction to Wireless Mobile Internet”, in The Wireless Mobile Internet: Architectures, Protocols, and Services, 2003.

[4] Forrest Warthman, “Delay-Tolerant Networks (DTNs), A Tutorial”, May 2003.

[5] J. Kangasharju, J. Ott, and O. Karkilahti, “Floating Content: Information Availability in Urban Environments,” in Proc. of IEEE Percom 2010, Work in Progress session, March 2010.

[6] J. Ott, E. Hyytiä, P. Lassila, T. Vaegs, and J. Kangasharju, “Floating Content: Information Sharing in Urban Areas,” in IEEE Percom 2011.

[7] Chris DiBona, Sam Ockman, “Open Source, Voice from the Open Source Revolution”, O’Reilly Media, January 1999.

[8] Adrian Holzer, Jan Ondrus, “Trends in Mobile Application Development”, Mobile Wireless Middleware, Operating Systems, and Applications - Workshop, Springer Berlin Heidelberg, 2009.

[9] Martin W. Murhammer, Orcun Atakan, Stefan Bretz, Larry R. Pugh, Kazunari Suzuki, David H. Wood, “TCP/IP Tutorial and Technical Overview”, October 1998.

[10] K. Scott, S. Burleigh, “Bundle Protocol Specification”, 2007.

[11] Internet Research Task Force, “Delay-Tolerant Networking Architecture”, April 2007.

[12] Kevin Fall, “A Delay-Tolerant Network Architecture for Challenged Internets”, Proc. SIGCOMM 2003, August 2003.

[13] Loren Clare, Scott Burleigh, Keith Scott, “Endpoint Naming for Space Delay / Disruption Tolerant Networking”, in Aerospace Conference, 2010 IEEE, March 2010.

[14] Kevin Fall, Wei Hong, Samuel Madden, “Custody Transfer for Reliable Delivery in Delay Tolerant Networks”, Technical Report IRB-TR-03-030, Intel Research Berkeley, 2003.

[15] Jian Shen, Sangman Moh, Ilyong Chung, “Routing Protocols in Delay Tolerant Networks: A Comparative Survey”, in The 23rd International Technical Conference on Circuits/Systems, Computers and Communications, July 2008.

[16] Tracy Camp, Jeff Boleng, Vanessa Davies, “A Survey of Mobility Models for Ad Hoc Network Research”, Wireless Communication & Mobile Computing (WCMC): Special issue on Mobile Ad Hoc Networking: Research, Trends and Applications, vol. 2, no. 5, pp. 483-502, 2002.

[17] E. Hyytiä, J. Virtamo, P. Lassila, J. Kangasharju, and J. Ott, “When does content float? characterizing availability of anchored information in opportunistic content sharing”, in IEEE INFOCOM, Shanghai, China, April 2011.

[18] A. Keränen, J. Ott, and T. Kärkkäinen, “The ONE Simulator for DTN Protocol Evaluation”, in SIMUTools ’09:Proceedings of the 2nd International Conference on Simulation Tools and Techniques, NewYork, NY, USA: ICST, 2009.

[19] Sheldon Ross, “Probability and Statistics for Engineers and Scientists Fourth Edition”, USA, 2009, pp.206-223.

[20] K. Krishnamoorthy, “Handbook of Statistical Distributions with Applications”, Chapman & Hall/CRC, Boca Raton, 2006.

[21] Ed Burnette, “Hello Android, Introducing Google’s Mobile Development Platform”, May 2008.

[22] J.F. DiMarzio, “Android, A Programmer’s Guide”, USA, 2008.

[23]Boost Mobile, “Android OS 2.1 Éclair”, September 2011.

[24] J. Ott and J. Kangasharju, Opportunistic Content Sharing Applications, in ACM MobiHoc NOM workshop, June 2012.

[25] Karthik Budigere, Binoy Chemmagate, Junxi Yin, Eero Martela, Markus Nurminen, Ankit Kumar, Richa Khera, Nutan Sawant, “GRID: A DTN based Localized Social Networking and Messaging Application”, In Proceedings of the 6th ACM workshop on Challenged networks (CHANTS '11) at Las Vegas, USA, 73-74.

[26] David Janzen, Hossein Saiedian, "Test-Driven Development: Concepts, Taxonomy, and Future Direction", IEEE Computer Society, September 2005.

[27]Google, "Nexus One, User's Guide", March 2010.

[28]Motorola Mobilty, User Guide-Android 2.3(Gingerbread), 2011.

[29] Jay A. Kreibich, "Using SQLite", USA, 2010.

[30]Adrian Holzer, Jan Ondrus, "Mobile Application Market: A Developer's Perspective", Telematics & Informatics, Feb 2011.

Appendix A

Source Code structure and Descriptions

fi.tkk.netlab.dtn.floatingcontent.db

-*Database.java*

Handling the data storage for subscription list

fi.tkk.netlab.dtn.floatingcontent.util

-*BubbletUtils.java*

Handling bubbles, for example bubble color changes

-*FileUtils.java*

Local file operation tool kit

fi.tkk.netlab.dtn.floatingcontent.ui

-*FloatingContentApplicationMain.java*

The main class to run when the application starts, it is used to do some initializations

-*BackgroundService.java*

The main service class implements the Floating Content features

-*MessageViwer.java*

Abstract class used to be inherited for different message views

-*ReceivedTab.java*

The tab displays the received message bubbles

-*ReceivedMessageViewer.java*

The sub class of MessageView.java and shows the received message content

-*CreatedTab.java*

The tab displays the created message bubbles

-*CreatedMessageViewer.java*

The sub class of MessageView.java and shows the created message content

-*SavedTab.java*

The tab displays the saved message bubbles

-*SavedMessageViewer.java*

The sub class of MessageView.java and shows the saved message content

-*SubcriptionTab.java*

The tab displays the available subscription services with checkboxes

Appendix B

XML File List and Descriptions

-*main.xml*

The main ui container which contains the three tabs

-*received_tab.xml*

The tab for displaying the received message bubbles

-*own_tab.xml*

The tab is made up of 2 sub tabs, created_tab and saved_tab

-*created_tab.xml*

The tab for displaying the created message bubbles

-*saved_tab.xml*

The tab for displaying the saved message bubbles

-*subscription_tab.xml*

The tab ui is made up of a list of checkboxes

-*checkbox.xml*

The stylesheet for each checkbox

-*single_bubble.xml*

The stylesheet for each bubble

-*message.xml*

The stylesheet for the message content