File Tracking For Mobile Devices

A Thesis Submitted to the
College of Graduate and Postdoctoral Studies
in Partial Fulfillment of the Requirements
for the degree of Master of Science
in the Department of Computer Science
University of Saskatchewan
Saskatoon

By
Gao Su

Permission to Use

In presenting this thesis in partial fulfilment of the requirements for a Postgraduate degree from the University of Saskatchewan, I agree that the Libraries of this University may make it freely available for inspection. I further agree that permission for copying of this thesis in any manner, in whole or in part, for scholarly purposes may be granted by the professor or professors who supervised my thesis work or, in their absence, by the Head of the Department or the Dean of the College in which my thesis work was done. It is understood that any copying or publication or use of this thesis or parts thereof for financial gain shall not be allowed without my written permission. It is also understood that due recognition shall be given to me and to the University of Saskatchewan in any scholarly use which may be made of any material in my thesis.

Requests for permission to copy or to make other use of material in this thesis in whole or part should be addressed to:

Head of the Department of Computer Science
176 Thorvaldson Building
110 Science Place
University of Saskatchewan
Saskatoon, Saskatchewan
Canada
S7N 5C9

Abstract

Since 2010, the smart device has become an integral part of people's daily lives. The popularity of smart devices has increased dramatically. However, as the number of devices owned by an individual user increases, so does the risk of data leakage and loss. This problem has started to draw attention because the data contained on smart devices tends to be personal or sensitive in nature. Many people have so much data on their devices that they have no idea as to what they are missing when a device is lost. Although there are already some solutions for data recovery, a data backup system on a remote server, these solutions are not accessible in the non-Internet environment. Development of a data recovery system that is accessible in the non-Internet environment is essential because of the constraints of mobile devices, such as unreliable network. This research proposes an architecture that allows the data recovery in both Internet (cloud) and Non-Internet (local) network by using different connection technologies. A data tracking mechanism has also been designed to monitor data flow among multiple devices, such as the cloud server, mobile devices, and tablets. Additionally, a synchronization system has been developed to ensure the consistency of tracking information. By designing and implementing this architecture, the two problems regarding to the data: "what is where" and "who has what" are resolved.

Acknowledgements

I would first like to deliver my very great thank to my supervisor Dr. Ralph Deters. Under his supervision, I have improved my skills and abilities in different areas, not only academic, but also industrial.

I would also like to thank the rest of my thesis committees: Dr. Julita Vassileva, Dr. Eric Neufeld, and my external examiner Dr. for their encouraging words and valuable comments.

Moreover, I would like to extend my thank to all the faculties of the Department of Computer Science as well as my colleagues of MADMAC lab for their supports which helped me to overcome all the problems and confusion I had in the past two years.

Finally, I must express my very profound gratitude to my parents and to my family members for providing me with continuous encouragement throughout my study and through the process of writing this thesis. This accomplishment would not have been possible without them. Thank you.

Contents

List of Tables

List of Figures

List of Abbreviations

BLE	Bluetooth Low Energy
CM	Connection Manager
DTM	Date Transmission Manager
FSD	File Storage Database
FSM	File Storage Manager
FTS	File Track Storage
GO	Group Owner
GTM	GO Tracking Manager
IoT	Internet of Things
ISM	Information Synchronization Manager
LAN	Local Area Network
MAN	Metropolitan Area Network
NoSQL	Not only SQL
PAN	Personal Area Network
REST	Representational State Transfer
SOAP	Simple Object Access Protocol
SQL	Structured Query Language
TBUT	To Be Updated Transaction

Chapter 1

INTRODUCTION

The usage of smart phones and tablet devices by a single user is on the rise. In fact, in 2016, the Global Web Index found that, on average, most internet users aged from 16 to 64 were using multiple devices. Figure 1.1 shows that, globally, most people own and are using 3.64 different mobile devices [7].

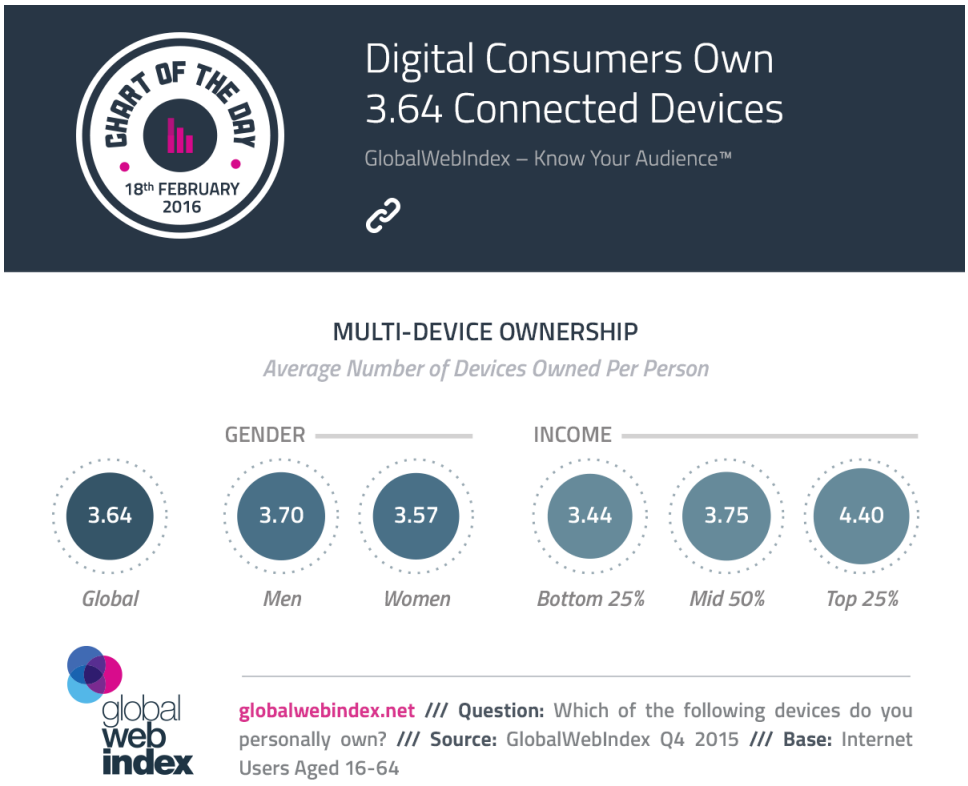

Figure 1.1: Number of devices owner by a user [7]

Many devices have the same applications installed so that the user can store and retrieve data anywhere and at anytime via different devices. However, tracking where the data is and, more importantly, who has access to these data is challenging. It is difficult for a user to recover their data if a device is lost. Moreover, complicating data transaction is the issue of intermittent connectivity of mobile devices. As soon as a user's mobile devices have lost their Internet connection, they are unable to transfer data.

The situation of data transmission is demonstrated in figure 1.2, which depicts a central server with three

different stored medical files for patients: Alex, Bob, and Charlie.

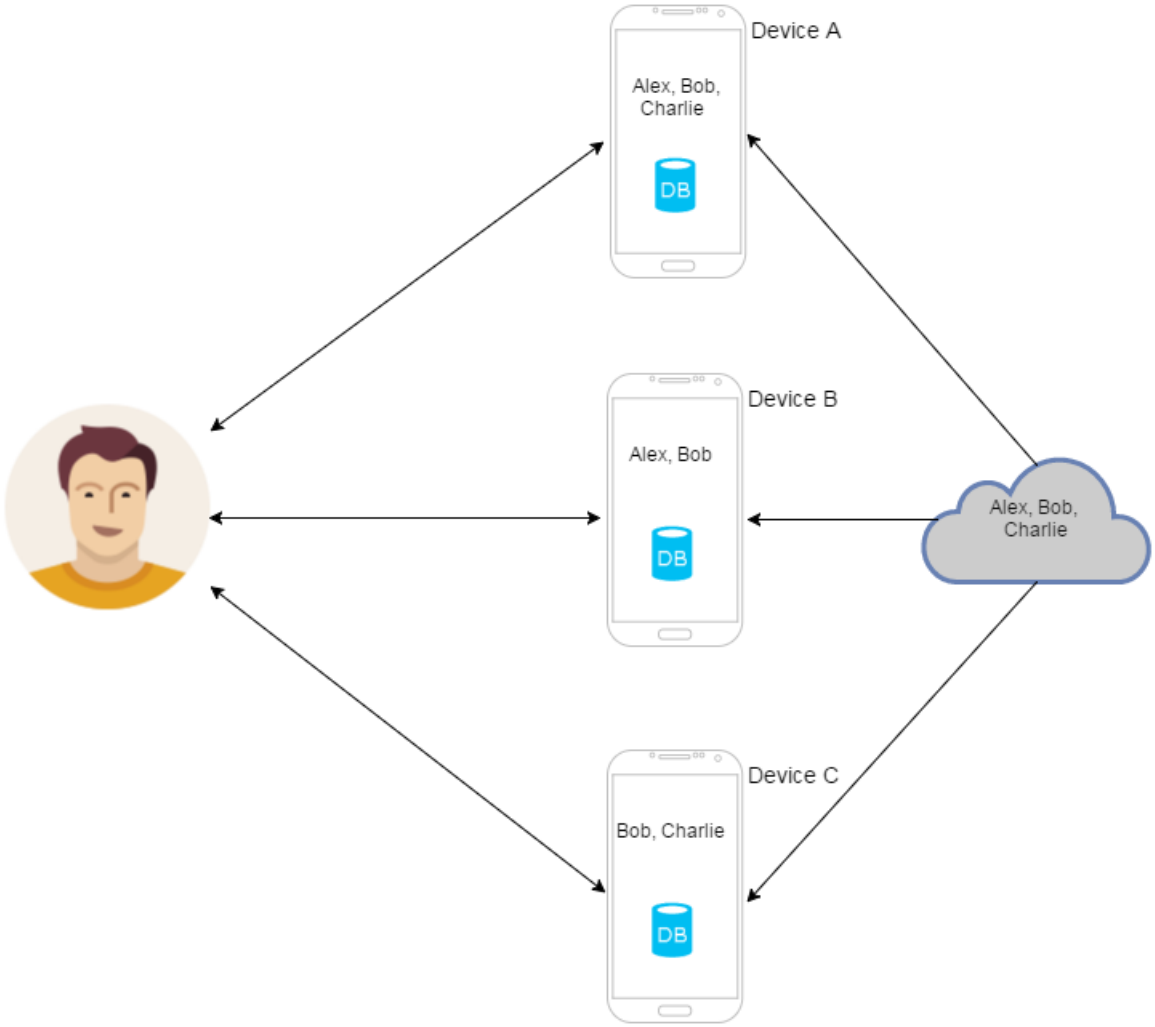

Figure 1.2: Architecture concept demonstration

In this example, a user has three devices that can retrieve the three medical files from the server. Device A requests all three files. Device B retrieves the files for Alex and Bob, while device C requests the files for Bob and Charlie. Each device stores the files locally for later access. However, the connection to the server is not permanent. If the Internet or the central server is down, it is nearly impossible for devices B and C to retrieve additional files from the server. Despite the connectivity interruptions of devices B and C to the Internet and central server, an alter way of retrieving file is to connect them directly to device A.

The example of data transmission and retrieval between three different devices demonstrates the need for the design of a file tracking architecture that addresses specific concerns of non-Internet networks. These concerns regarding file transaction in non-Internet networks also include addressing the challenge of a user to remember which file is stored on what device, or which devices have accessed certain files. Another concern specific to these devices is if they are lost or stolen since the chance of data recovery is extremely unlikely. To design such a high-performance system, using different technologies, protocols, and mechanisms has been

designed and applied.

The remainder sections of this thesis are structured as follow:

- Chapter 2 defines and explains the research problem,
- Chapter 3 reviews related protocols, technologies, and research,
- Chapter 4 demonstrates the architecture design in detail,
- Chapter 5 discusses implement specifications,
- Chapter 6 evaluates the performance of the designed architecture,
- Chapter 7 concludes this research and its expected contribution, and
- Chapter 8 addresses the future work that can be done.

Chapter 2

PROBLEM DEFINITION

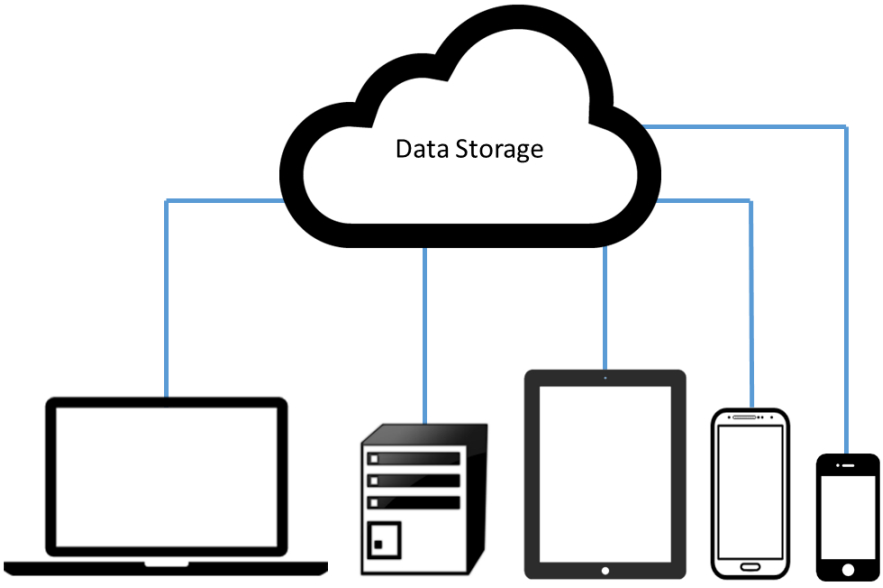

Figure 2.1: Device Connection in Cloud Based Network

Currently, there are two main models of mobile device connection: the Internet client-server based and non-Internet based (e.g. Personal Area Network, Local Area Network.) For the client-server based network, as depicted in figure 2.1, all data is stored on the central server. Each client can push as well as retrieve data to the central server. In this centralized network, each node must have an Internet connection to communicate. Conversely, non-Internet based network, figure 2.2, provides communication between devices when the Internet is not available. In non-Internet based networks, device uses specific technology to communicate directly with other devices - without the router.

2.1 Problem Description

From an individual perspective, as discussed in Chapter 1, most people personally own multiple mobile devices like smart phones, tablets, smart watches. Moreover, from an organizational perspective, businesses

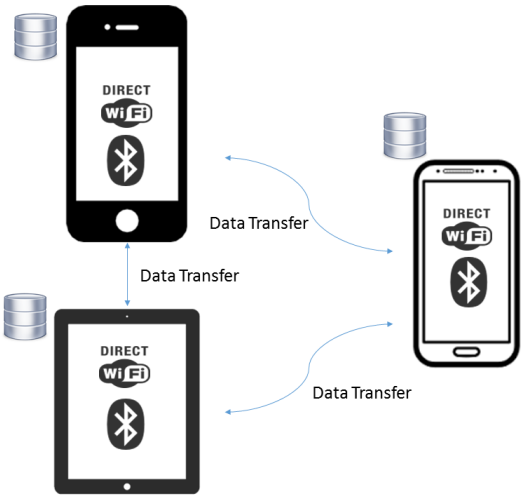

Figure 2.2: Multiple Connection in Local Network

Figure 2.3: Device Disconnected from Network

are moving away from the traditional office setup with a desktop computer to mobile side in order to give employees greater flexibility accessing data. The tradeoff for improved flexibility is that data is transferred more frequently. An increase in the frequency of data transfers in turn increases the chance of data loss. Tracking where the data is and who has access to it becomes more difficult. The issue of tracking data exposes a fundamental problem: "what is where" and "who has what". If there is something wrong with the data, or any leak of data, it is difficult for the user to find out the source of the problem. Moreover, as shown in figure 2.3, the inability of users to know what data has been lost when a device goes missing increases the risk of permanent and irreversible data loss.

For example, a doctor takes a flight from Vancouver to Toronto for a medical consultation. Before he goes to the airport, he did some research on line, and retrieved all the related medical data from the hospital's cloud server to his tablet. However, when he arrived at the hospital in Toronto, he realized that his tablet

is lost. In this case, either he is lucky enough to get his tablet back, or he will lose all the data he gathered from the hospital in Vancouver. All of the effort and time are wasted.

Moreover, due to the intermittent connectivity of the mobile network, the Internet connection among multiple mobile devices is unstable. Once the Internet service is unavailable, the file transaction cannot be performed, potentially resulting in a serious consequence during an emergency.

2.2 Research Goal

Most data management systems focus on the ideal situation where Internet access is available. But the data transactions cannot be achieved if the Internet connection is lost or unstable. Therefore, in this research, we designed and implemented a new architecture to track data among multiple mobile devices in both the Internet and non-Internet environment. We can reduce the risk of data loss and also monitor the data access by tracking the data among mobile devices. The data availability can be improved by allowing the data transaction in the non-Internet environment.

2.3 Research Challenge

There are a few challenges need to be solved in order to construct this new architecture.

- Challenge 1: Device connection in the non-Internet environment

  In [18], the connectivity of mobile devices in the network can be categorized as connected or as disconnected. Devices communicate via the Internet in most connected situations. In this architecture, the mobile devices need to communicate to others directly when the Internet service is not available. How the devices are connected in the non-Internet environment is critical as the connection method can influence the performance of the architecture.

- Challenge 2: Data consistency in the intermittent network

  The mobile device usually has unstable network connectivity and the timing of the Internet disconnection is unpredictable. The solution to the file tracking is explicit when the Internet is available. However, file transactions in the non-Internet environment needs to be considered in this research architecture. The architecture allows a device to transfer files to other devices while it is disconnected from the server. In this case, the way in which server tracks the data transactions on disconnected devices is addressed. The data integrity and consistency need to be ensured in this situation.

- Challenge 3: Bandwidth

  Low bandwith is a challenge for mobile devices and can be caused by various reasons, such as the number of devices in the network and the data transmission protocol. The architecture needs to minimize the amount of data transferred over the connection to improve the efficient of device communication.

Chapter 3

LITERATURE REVIEW

In this chapter, I will review some related research and technologies that can be included in the design of the architecture to overcome the three challenges mentioned in Chapter 2.

Challenges	Related Technologies
Device connection in the non-Internet environment	• Mobile Peer-to-Peer technology • Local Area Network
Data consistency	• Data synchronization patterns • Data replication mechanism
Network bandwidth	• Lightweight web service • Proper synchronization mechanism
Data Availability and architecture scalability	• Cloud computing • Scalable database

Table 3.1: Related technologies to overcome challenges

3.1 Mobile Cloud Computing

Tree main computing trends will be covered in this section: 1) mobile computing, 2) cloud computing, and 3) mobile cloud computing. Mobile cloud computing is a relatively new concept. It is a combination of mobile computing and cloud computing that aims to provide better availability and accessibility for mobile users.

3.1.1 Mobile Computing

"Mobile computing technology enables the mobile worker to create, access, process, store and communicate information without being constrained to a single location"[18]. The goal of mobile computing is to provide a medium between people and information without spatial and temporal constraints. There are some features of mobile computing mentioned in by Qi and Gani in [42]:

- Mobility

  Each mobile node can easily connect to others in a wired or wireless network during their movement.

- Diversity of network conditions

  The mobile device can exist in various kinds of networks, such as a wired network with high bandwidth, wireless personal area network, wireless local area network, and wireless wide area network. Some details about these networks will be covered in the later sections.

- Frequent disconnection and consistency

  Due to the limitation of battery power and network conditions, the mobile device cannot connect to others all the time. However, mobile devices can reconnect and remain consistent with the network by following a specific protocol.

- Low reliability

  A mobile computing network must address the security issues from the terminals, database, and application development because the signal is easily susceptible to interference.

According to Deepak and Dradeep[18], the author mentioned three main concepts of mobile computing:

- Hardware

  Mobile hardware contains two main categories: mobile devices or device components [53]. The mobile devices are defined as handheld computers such as smartphones, tablets, and PDAs. The device components are "defined by the size and form factor, weight, microprocessor, primary storage, secondary storage, screen size and type, means of input, means of output, battery"[18].

- Software

  Mobile software is a computer system or application designed to run on handheld computers. The common operating systems running on mobile devices includes Apple IOS, Android, Nokia Symbian, Windows. The operating system is the essential software component required to operate mobile devices. Additionally, there are varieties of mobile applications that run on mobile devices in order to provide different functionalities.Typically, there are two formats of mobile applications: native apps and web apps [34].

1. Native Apps

   A native app is programmed and designed for a specific operating system. The user downloads an app from a web store and installs it on the device. The most common stores on the market are the Apple Store and Google Play. There are two main advantages of the native app: 1) powerful user experience and 2) the ability to run offline. First, because the native app is downloaded from the app store, it can leverage the specific hardware (GPS, camera and Bluetooth) and software (picture gallery and contact list.). Second, The native app remains installed on the device and possesses the ability to run offline. There is no Internet requirement to launch the app, and the user can pause or resume the app at anytime. On the other hand, there are also some drawbacks of the native app. The biggest drawback is the incompatibility for multiple platforms. As mentioned above, beacuse there are many mobile operation systems available, each platform has its own language and tools to build an app, figure 3.1. A native app therefore requires more work and more money to meet all its needs [34].

	Apple iOS	Android	Blackberry OS	Windows Phone
Languages	Objective-C, C, C++	Java (some C, C++)	Java	C#, VB.NET and more
Tools	Xcode	Android SDK	BB Java Eclipse Plug-in	Visual Studio, Windows Phone development tools
Packaging format	.app	.apk	.cod	.xap
App stores	Apple App Store	Google Play	Blackberry App World	Windows Phone Marketplace

Figure 3.1: Multiple Mobile Platforms [31]

2. Web Apps

   A web app is a web based application that is formatted for various smartphones and tablets. The common way to access the web app is via the device's web browser. There are three main technologies used for web app development: HTML, CSS, and JavaScript. Due to the fact that web apps are developed based on a web browser, they are independent of different platforms. However, a typical web app needs to be downloaded from a web server each time it is run [34].

- Communication

"The ability of a mobile computer to communicate in some fashion with a fixed information system is a defining characteristic of mobile computing" [18]. The type and availability of communication media determines the type of the mobile applications. Deepak and Pradeep describe in [18], mobile device communication is divided into four categories:

1. Connected

   The connected status implies a continuously high-speed connection between a mobile device and an information system (e.g. server) for data transfer.

2. Weakly connected

In this situation, the mobile device and the information system communicate continuously with low speed.

3. Batch

   The mobile device is not continuously connected to the information system. The connection is established randomly to exchange data.

4. Disconnected

   Under this mode, the mobile device is incapable of communicating with the information system. The user can only access the local storage to retrieve data.

3.1.2 Cloud Computing

Mobile device hardware is not as powerful as desktops so it needs an external resource to supply additional complex computing work. This particular requirement has led to a leap in development of cloud computing. According to Rajkumar et al[8] cloud computing is:

> A type of parallel and distributed system consisting of a collection of interconnected and virtualized computers that are dynamically provisioned and presented as one or more unified computing resource(s) based on service-level agreements established through negotiation between the service provider and consumers.

The authors emphasize that the Cloud is a new generation of data centers that allocates resources to each device on demand. Therefore, cloud computing is also called on-demand computing.

Peter and Tim proposed in [37] that the cloud computing provides two main groups of models: service models and deployment models.

- Service Models

  1. Software as a Service (SaaS): is usually delivered as pay-per-use subscription, and is therefore also called "on-demand software". In this service model, a service provider develops and runs the application software in the cloud so that the consumer can access it through web portals. The consumer doesn not need to manage the fundamental cloud infrastructures like network, servers, database, which simplifies the hardware maintenance [37] [58].
  2. Platform as a Service (PaaS): in this models, the service provider offers a specific development environment for the consumers, who are the application developers. The consumers can create and deploy their applications onto the cloud infrastructure as long as they follow the standard API, libraries and languages that are supported by the provider. The developers only need to focus on the application development and do not need to worry about the hardware layers [58] [37].
  3. Infrastructure as a Service (IaaS): this model offers virtualized resources to consumers. The developers can choose the different operating systems and software running on the servers. IaaS is considered the bottom layer of cloud computing systems [58] [37].

- Deployment Models [58]

    1. Public/Internet Cloud: the cloud service is made available as a pay-as-you-go pattern to the public.
    2. Private/Enterprise Cloud: the cloud service runs within an individual or organization's own data center for internal use.
    3. Community Cloud: the cloud service is used by multiple organizations who share the same concerns or purposes.

3.1.3 Summary

To sum up, mobile computing has brought about a change in lifestyle in which mobile devices have become an essential part. Although mobile computing has many advantages as compared to the traditional PCs from the mobility and flexibility, the constraints also need to be considered when designing a mobile architecture [22]:

- Limited Resource

  Mobile devices have less computational performance, storage capacity, and battery power as compared to PCs.

- Network Bandwidth

  It is easier to cause lower or no signal conditions that result in low bandwidth because mobile devices have higher mobility.

- Intermittent Connectivity

  In mobile computing, the continuous connection is extremely difficult because of the unreliable network connection.

Cloud computing seeks to deal with data manipulation, and to provide alternative application solutions for different user levels. The combination of mobile computing and cloud computing have generated a new computing trend: mobile cloud computing. The purpose of mobile cloud computing is to improve mobile device performance by processing large and complicated computation in the cloud. In this research, the file objects and tracking information are stored on the cloud server so that the mobile devices can retrieve files when the Internet is available.

3.2 Computer and Mobile Network

Computer networks are used to allow computers to exchange information among themselves. Each device in the network is directly or indirectly connected with a specific communication technology. The most common

computer network is the Internet. In this section, some network terminologies and the different types of networks will be reviewed.

There are several various kinds of networks that are differentiated based on various functionalities. The two common classification standards are based on the size and the architecture of the network. Olivier [6] addresses two primary network architectures: the client-server and the peer-to-peer models. In Tutorialspoint[52], the network can also be distinguished based on their geographical span. There are four main categories: 1) Personal Area Network (PAN), 2) Local Area Network (LAN), 3) Metropolitan Area Network (MAN), and 4) Wide Area Network (WAN). More details will be covered in the following sections.

3.2.1 Architecture Classification

- Client-Service Model

  The client-server model is the most common network model. Each computer is either a client or a server in this kind of network. Normally, servers are powerful computers or processes that are used to manage files, database, or network traffic. Clients are the PCs or mobile devices on which users can run applications [35]. In this model, the client (service requesters) send requests to the server (resource provider), and then the server performs actions and returns the corresponding responses. The server and the client much follow the specific protocols in order to ensure they both understand the messages sent from each other [33].

- Peer-to-Peer Model

  In [1], Stephanos and Diomidis defined peer-to-peer network as the following:

  > Peer-to-peer systems are distributed systems consisting of interconnected nodes able to self-organize into network topologies with the purpose of sharing resources such as content, CPU cycles, storage and bandwidth, capable of adapting to failures and accommodating transient populations of nodes while maintaining acceptable connectivity and performance, without requiring the inter mediating or support of a global centralized server or authority.

  This definition encompasses two different peer-to-peer architectures based on the degree of centralization: a completely decentralized system or a hybrid decentralized structure. In the completely decentralized peer-to-peer architecture, which is also called pure peer-to-peer, each peer is an equal participant in the network. There is no administrative role. No central server is required to to manage and coordinate the connections. Each node has to self-organize themselves to communicate with other nodes [35].

  The hybrid decentralized peer-to-peer architecture incorporates some of the features of the client-server model. Moreover, Some of the nodes in the network are assigned a more important role and acts as a local central server for facilitating the interaction between peers [1].

  Each client in the network stores data locally that can be shared with the rest of the peers. All clients connect to a central server who contains two pieces of information [1]:

– Registered user connection information (IP address, time of connection, etc.), and

– A database table which lists the data that each client holds in the network.

A new peer connects to the central server and provides the data information the peer needed in order to join the group. The peer sends a request to the server for searching and sharing the data. The server searches through the database and returns a list of users that hold the matching information. The peer then establishes a direct connection with the specific nodes [1].

The advantage of this hybrid decentralized architecture is that it is easy to implement, and it can efficiently search and locate data. On the other hand, the main disadvantage of this architecture is scalability since there are limitations to the size of the server database and its restricted capacity. Moreover, because the whole network is connected to the local server, if the master peer abandons the system, then the services will not be available to anyone [35] [1].

3.2.2 Size Classification

- Personal Area Network (PAN)

  A Personal Are Network is the smallest network that is designed to satisfy personal needs. The PAN includes Bluetooth or Wi-Fi enabled devices such as smartphones, tablets, and wireless printers.

- Local Area Network (LAN)

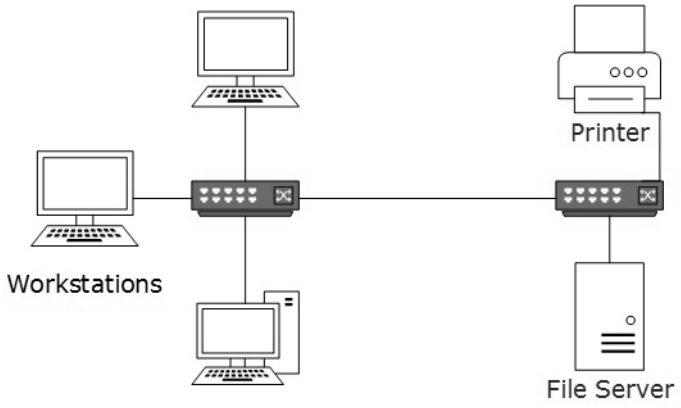

Figure 3.2: Local Area Network Coverage [52]

As defined in Tutorialspoint[52], a network that is contained in a relatively small area, like school or single building, is termed as a LAN, as shown in figure 3.2. Normally, this type of network is easy and inexpensive to install. A LAN provides a useful and efficient way of sharing the resources among the end users while it works under its local domain and does not involve any heavy routing. Some LANs may contain local servers for file storage and local applications.

- Metropolitan Area Network (MAN)

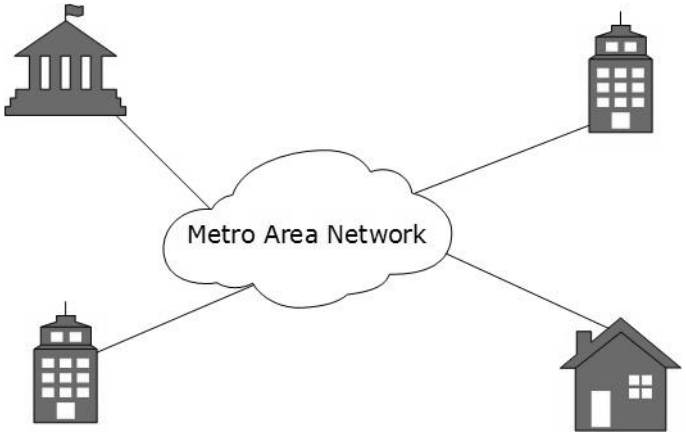

Figure 3.3: Metropolitan Area Network [52]

The MAN is more like an expansion of a LAN. The MAN generally spans throughout a metropolitan city. It can be in the form of an Ethernet, cable TV, or ATM. Much of the technologies, like fast network components as well as the connection equipment, used in LANs can also be used in MANs but with more requirements. By applying the ISPs (Internet Service Provider) [27] in the MAN, the users can expand their LAN to larger ones, as shown in figure 3.3 [52].

- Wide Area Network (WAN)

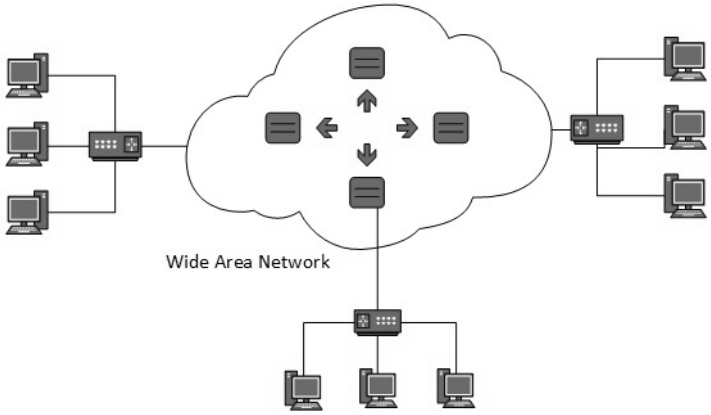

Figure 3.4: Wide Area Network [52]

When the network expands beyond the range of the MAN, a larger network is introduced and is referred to as a Wide Area Network (WAN), figure 3.4. Normally, a WAN provides connection among MANs and LANs so that everything in the network can be connected.

3.2.3 Summary

Networks can be classified according to their different component roles, client-server based and peer-to-peer based. In the client-server model, each device only has one role as either client or server. The client-server

model offers centralized control that can cause the whole system to suffer heavy delays or to completely break down when the server is offline. In the Peer-to-Peer model, two types were discussed: the completely decentralized system and the hybrid decentralized architecture. In the completely decentralized P2P system, each node is equal, in the network, and there is no master node. Conversely, in the hybrid decentralized network, some features of the client-server model are encompassed. Some peer nodes perform as the local central server in in the network.

Moreover, four categories of networks, classified based on their geography, were included: PAN, LAN, MAN, and WAN. PAN and LAN present small networks that only cover up to one floor or a building. The PAN is designed for a single person purpose of usage within 10 meters, and the LAN is more typically used by an organization or a company. Additionally, MAN and WAN refer to the larger networks that often cover a city or even an entire country. The Internet is one example of a WAN.

In this research, the goal is to design and implement an architecture that can transfer file and track file in the Internet and non-Internet environment. When the Internet is available, the client-server based network structure is applied. On the other hand, the combination of a Peer-to-Peer model and a Local Area Network will satisfy the architecture's needs in the non-Internet network.

3.3 Short-Range Mobile Peer-to-Peer Technology

Debopriyo states:"the peer-to-peer networking has shifted focus away from the static architectures of client-server model that characterised the early Internet, towards more, dynamic, mobile networks." [23] Mobile Peer-to-Peer technology is an essential part of mobile communication. Most people use mobile technologies to share information via the Internet when they are apart. The signal from the device is sent to the cell tower before a connection is made. However, this type of communication is a waste of resources if two devices are in a short distance from one another. Additionally, if one of the device is not able to connect to the Internet, then the connection between the two devices cannot be made. Therefore, the short-range mobile peer-to-peer technology is relied on to address these situations. The two main technologies that have been used widely in short-range mobile P2P networks are Bluetooth and Wi-Fi Direct.

3.3.1 Bluetooth

"Bluetooth is the connection technology which can be used as link to share data among devices within a Personal Area Network (PAN)" [50]. Bluetooth is a ubiquitous, low energy wireless link that is available in most of the mobile devices and tablet computers. The first version of Bluetooth was invented by the telecom vendor Ericsson in 1994. Overall, there have been versions released since the Bluetooth was initially developed. The two most widely used implementations are Bluetooth Basic Rate/Enhanced Data Rate (BR/EDR) as version 2.0/2.1, which is known as Classic Bluetooth and Bluetooth with Low Energy (LE) as version 4.x. Each of these versions has different usages.

Power Class	Transmission Power Level	Advertised Range
1	100 mW	100 meters
2	2.5 mW	10 meters
3	<1 mW	< 1 meters

Table 3.2: The Bluetooth Power Class [28]

- Characteristics

  The BR/EDR or Classic Bluetooth is typically used when the user needs to set up a relatively short-range and continuous connection. The key feature of classic Bluetooth is the pairing procedure. The pairing procedure requires user interaction to make a simple passkey. Conversely, the Bluetooth Low Energy was originally built for the Internet of Things (IoT) [13], which requires low energy consumption and an instant connection setup. The main advantages of classic Bluetooth are [3]:

  – Multiple-vendor interoperability.

  – Ultra-low power consumption which enables the devices to run for months with standard coin-cell batteries, and

  – Standard application development architecture which can reduce operational costs

  Moreover, in most Bluetooth Low Energy devices, there are two modes implemented: single mode and dual mode. In single mode, only the low energy protocol stack is included. In dual mode, it includes Classic Bluetooth, Bluetooth High Speed (Bluetooth 3.x), and Bluetooth Low Energy protocols so that the Bluetooth enabled devices can access any versions of Bluetooth devices. For most new generations of mobile devices, they are all embedded with Bluetooth 4.x chips.

- Cover Range

  As the Bluetooth uses radio transmissions for data exchange, the range is determined by the transmission power levels. Table 3.2 summarizes their range differences based on the various power levels.

  For most of the Bluetooth-enabled devices such as cell phones, tablets, and laptops are class 2 devices. If there is a connection setup between different classes, the range is limited by the less power level [28].

- Topology In Bluetooth network, the physical channel that connects two or more devices called piconet. In a piconet, the device which scans for other devices is called "slave", and the advertising device is called "master." Each piconet consists of one master and up to seven slaves. Any Bluetooth device can function as master or slave. While Bluetooth devices are connected, they communicate on the same physical channel by synchronizing with the common clock and hopping sequence. The common clock is identical to the master's Bluetooth clock, and the hopping sequence is generated from the Bluetooth clock and the master's Bluetooth address [4].

One Bluetooth device can participate to two or more piconets simultaneously which needs to do on a time-division multiplexing basis [2]. However, a device can never be the master of more than one piconet as the piconet is defined by the master's Bluetooth clock. For the Bluetooth device that is the member of two or more piconets is said to be connected in a scatternet as shown in figure 3.5 and figure 3.6 which are topologies of Bluetooth BR/EDR and Bluetooth Low Energy respectively [51] [4].

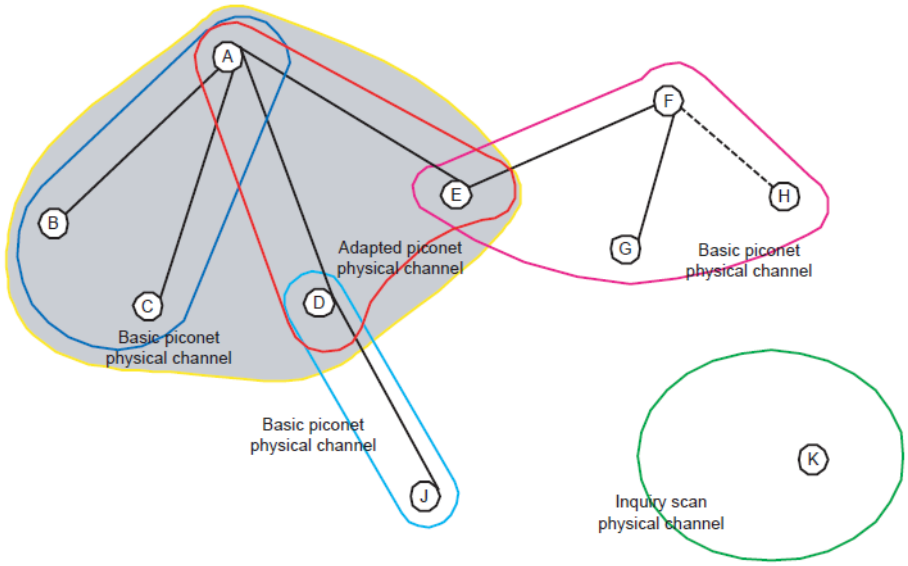

Figure 3.5: Bluetooth BR/EDR Topology [4]

In figure 3.5, it shows an example topology of Classic Bluetooth. Device A is a master device in a piconet, which is the shaded area, with device B, C, D, and E as slaves. Meanwhile, there are two more piconets: one piconet with device F as the master and device E, G, H as slaves, and the other piconet with device D as the master and device J as a slave to it. In piconet A, there are two physical channels. B and C use the basic channel to communicate with A as they are only involved in one piconet. D and E are using adapted physical channel as this kind of channel allows them to spare some radio frequencies for different piconets.

In figure 3.6, it demonstrates the topology of Bluetooth Low Energy. Device A acts as a master in a piconet which has device B and C as slaves. The difference of these two topologies is the number of physical channels needed for different slaves. In Classic Bluetooth, each slave in one piconet communicates with the master on a common physical channel. On the other hand, in Bluetooth Low Energy, each slave connect to the master on a separate channel.

3.3.2 Wi-Fi Direct

Wi-Fi Direct is also called Wi-Fi P2P which is defined and developed by Wi-Fi Alliance. Normally, Wi-Fi is the most common way to access the Internet. However, as people's demand increasing, the Wi-Fi technology

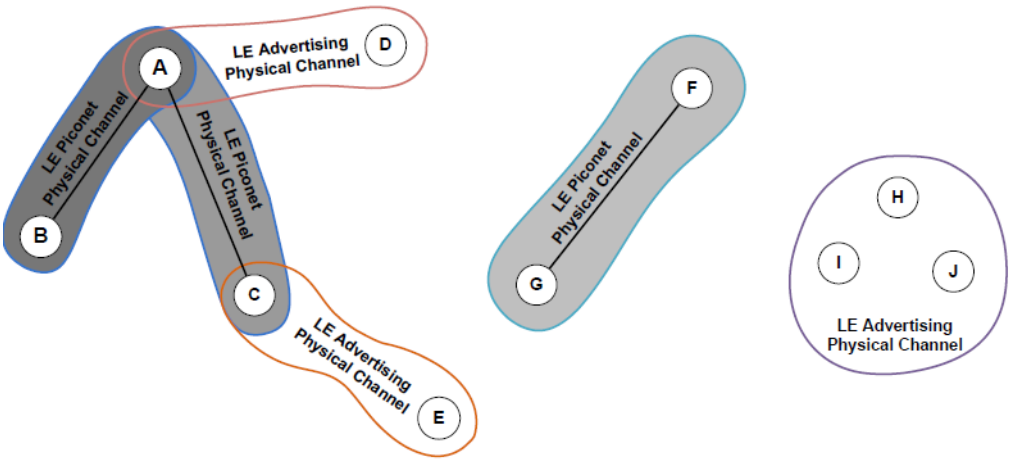

Figure 3.6: Bluetooth Low Energy Topology [4]

needs evolve into a new stage; therefore, the Wi-Fi Direct came out. It aims at enhancing device to device communications in Wi-Fi environment.

- Architecture

  As mentioned by Daniel et al in[9], the typical Wi-Fi Local Area Network is created and broadcasted by Access Point (AP). Therefore, in this network, a device is clearly defined either as an AP or a client, and each of these roles has the different set of functionalities. However, these roles in Wi-Fi Direct are defined dynamically, and a Wi-Fi Direct device has to implement the functionalities of both roles: client and AP. Meanwhile, these roles could also be executed simultaneously by the same device with different frequencies.

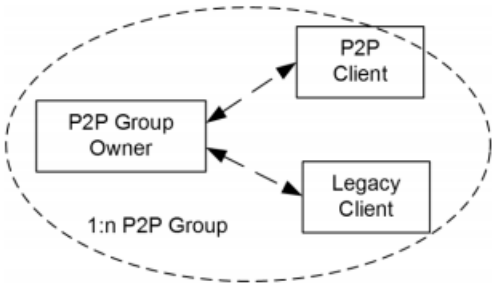

Figure 3.7: Wi-Fi Direct Topology [62]

The networks formed by Wi-Fi Direct devices are known as P2P Groups. There are two main device roles in the group: P2P Group Owner (GO) act as an AP and P2P Clients. As mentioned above, these two roles are not static in the group; therefore, when these two devices discover each other, they need to negotiate to determine which one of them to be the GO. The details of the negotiation will be covered later. Once the group is established, other devices can join the group by connecting to the GO

as connecting to the AP in the traditional Wi-Fi network as shown in figure 3.7 [62].

The default Wi-Fi Direct architecture for Android is designed for 1:N devices which means that the devices can only communicate within the same group [11]. However, the alternative architecture for inter-group communication has already be proposed by Casetti et al in [10].

In the group, the GO has to run a Dynamic Host Configuration Protocol (DHCP) server application in order to provide P2P Clients with IP addresses. Moreover, the P2P GO is the only device in the group that is allowed to cross-connect the P2P clients to an external network [9].

- Group Formation

  There are three main types of group formation in Wi-Fi Direct: standard, autonomous and persistent. Which type of formations is used depending on various situations, e.g. if the devices have been connected before, or if there are pre-shared security information available [9].

  Each of these types have different phases [29]:

  - Discovery Device

    Scan for existing P2P groups and Wi-Fi networks to establish connection

  - Determination of GO

    * devices negotiate for GO based on capabilities.
    * One device is selected as GO at formation.

  - Provisioning of the group

    * Appropriate credentials are used for group establishment.
    * Exchange credentials using Wi-Fi simple configuration.

    More details about the security will be covered in the later section.

  - Address Configuration

    The GO provides IP address to a client in the group using DHCP.

  In the standard case, all these four phases are included. At the beginning, the discovery phase is initialized. After the scan, a discovery algorithm is executed. A device selects one of the social channels, channel 1, 6 or 11, in the 2.4 GHz band as a listen channel; then the device switches its status between two states: search state and listen state. In search state, the device scans other devices by sending Probe Requests in its social channels. In the listen state, the device listens for the Probe Requests sent from other devices, and then responses to it with Probe Responses. Once the two devices found each other, the GO negotiation phase is triggered. This is a three-way handshake process: Request/Response/Confirmation. In this step, the two devices agree on which one will act as GO [29]. "The primary purpose of Group Owner Negotiation is to exchange the Group Owner Intent attribute to communicate

a measure of desire to be P2P Group Owner" [62]. The device that declares the highest Intent value becomes the GO. Besides, if one device must be the GO, the Group Owner Intent attribute has to be set to 15 as demonstrate in figure 3.8.

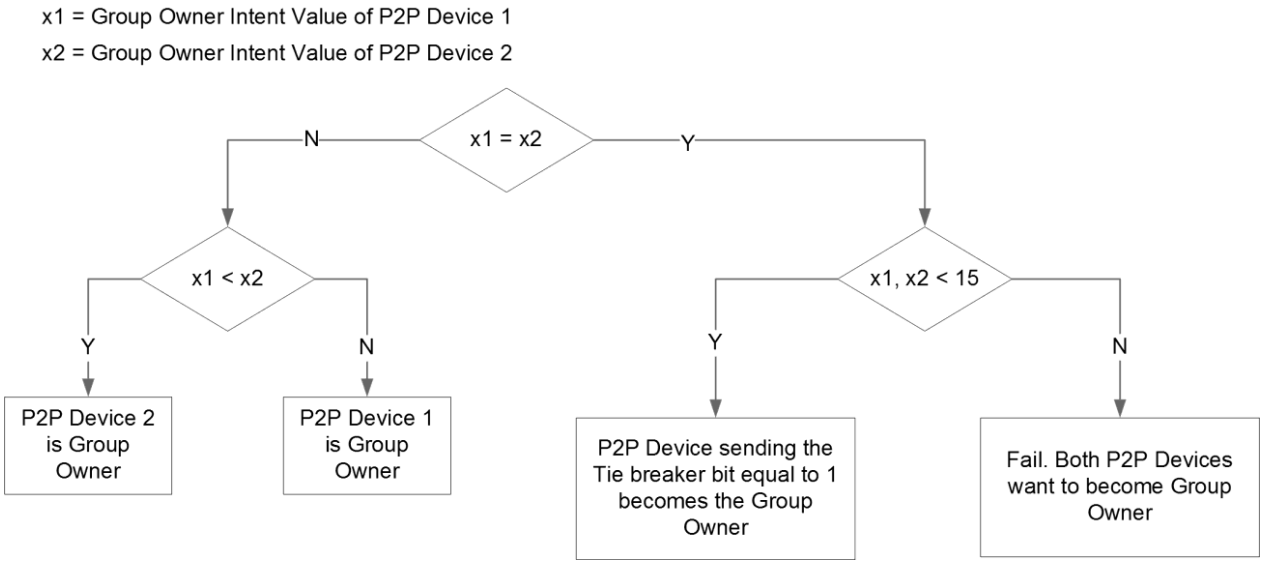

Figure 3.8: Group Owner Determination Flowchart [62]

In order to prevent the conflicts when two devices declare the same GO Intent value, a tie-breaker bit is included in the GO Negotiation Request. It is set randomly whenever the request is sent [29].

In the autonomous group formation, a P2P device may be able to create a group autonomously, and it becomes the GO. As there is only one device participate when the group is forming, the search and GO Negotiation steps are omitted. Other devices can discover the established group using traditional Wi-Fi scanning procedure, then proceed to the WPS Provisioning and Address Configuration phases [9].

The last group formation type is persistent. In this formation, P2P devices can establish a group as persistent by using a parameter in the P2P Capability attribute which contains multiple indicators [62]. In this status, the group stores network credentials which are assigned to GO and client for subsequent re-instantiations of the group. More specifically, if a device recognizes that it has formed a persistent group with the corresponding device before, then any of them can use the Invitation Procedure [62] to quickly re-connect. The standard GO Negotiation is replaced by the invitation procedure. Moreover, the provision steps are reduced as the credentials can be reused [9].

Despite which type the group formation is, each of them involves in the WPS. It is short for Wi-Fi Protected Setup [61] which is implemented to support secure connection with minimal user intervention. The WPS establishes a secure connection by generating a PIN in the client, or pressing a button in both GO and client devices. There are two parts in the operation of WPS. In the first part, the GO needs to generate and issue the network credential such as security keys to the client. In the second

part, the client disassociates and reconnects with its new authentication credentials. By doing so, if these two devices form group in the way of the persistent group, it is not necessary to go through the first phase [9].

3.3.3 Comparison

As reviewed in the previous section, Bluetooth and Wi-Fi Direct are explained in details from the aspects of characteristic, architecture, and topology. Each of them has their own advantages and drawbacks. In this section, the comparison between these two technologies will be covered.

- Network Size

  As mentioned above, the maximum number of devices for a Bluetooth piconet is eight (seven slaves and one master). It is possible to extend this number to 255 while each device does not participate in data exchange but only keep synchronization with the master's transmissions. On the other hand, according to the Wi-Fi Alliance, a Wi-Fi Direct network can be one-to-one or one-to-many. The number is expected to be smaller than the traditional access points, which is 2007 for a structured BSS (Basic Service Set [56]). Connection to multiple devices is an additional feature. The size is depending on the device manufacture [43] [60].

- Speed

  Based on the information from [5], the data rate of Bluetooth Basic Rate is 1 Megabit per second (Mb/s) and the Enhanced Data Rate is 2 or 3 Mb/s. For Bluetooth 3.0 which is designed for high-speed transfer, the speed can be up to 25 Mb/s. However, since the Bluetooth Low Energy is designed for use cases that require low data rate and has lower duty cycles, the speed for it is 1 Mb/s. On the other hand, the Wi-Fi Alliance mentioned on their website: Wi-Fi Direct supports typical Wi-Fi speed which can be as high as 250 Mb/s, but this speed depends on which protocols the devices follow e.g. 802.11 a, g, or n as well as the physical environment of devices [60].

- Compatibility

  As stated in 3.3.1, the Bluetooth Low Energy has single and dual modes. For single mode device, it is not backward compatible with Classic Bluetooth protocol. For dual mode implemented device, it is allowed to connect with both Classic Bluetooth and new Low Energy devices. For the Wi-Fi Direct device, it is not only compatible with the same certificated Wi-Fi Direct, it can also communicate with legacy Wi-Fi devices which are not implemented with Wi-Fi Direct.

3.3.4 Conclusion

These two technologies have been widely used by electronic devices. In this research, we are focusing on designing an architecture that can track mobile application data in the Local Area Network. Our architecture

requires high scalability whenever a new device joins the group. Moreover, the transmission rate is also critical for this research as it requires a high frequency of data transaction. Therefore, after considering these factors, Wi-Fi Direct is applied to our architecture as the communication technology in the non-Internet environment.

3.4 Web Service

Web Service (WS) is a collection of protocols, and that can be used to exchange data between applications over a network despite which programming languages the applications written in or the platforms the systems are running on [55].

There are four steps in the Web Service process [59]:

- The service provider and requester become known to each other. By known, it means these two components have the awareness of each other's existence.
- These two entities agree on the service description and semantics that will manage the interaction between them. "The semantics of a web service is the shared expectation about the behavior of the service, in particular in response to messages that are sent to it." The web service description is a specification of the web service interface. It defines the message formats, datatypes, etc.
- The service description and semantics are input to the requester and the provider.
- These two entities exchange messages and data.

There are two main Web Services that are used broadly: the Representational State Transfer (RESTful) and the Simple Object Access Protocol (SOAP). In the following sections, these two services will be explained and compared from different aspects.

3.4.1 RESTful

RESTful stands for Representational State Transfer. It was first introduced by Roy Fielding in [20] in 2000. It is normally, but not always, communicate over the HTTP protocol which is a stateless, client-server protocol. As stated by Elkstein[19], "despite being simple, REST is fully-featured; there's basically nothing you can do in Web Services that can't be done with a RESTful architecture." The way RESTful works is demonstrated in figure 3.9. The consumer sends request to the provider which is usually the web server; then the server sends back the response in XML.

In one paper of Cesare [40], he mentioned that REST architecture is based on four principles:

- Resource identification through URI

  Each resource can be referred by a unique universal resource identifier (URI) which provide a global addressing space for resources.

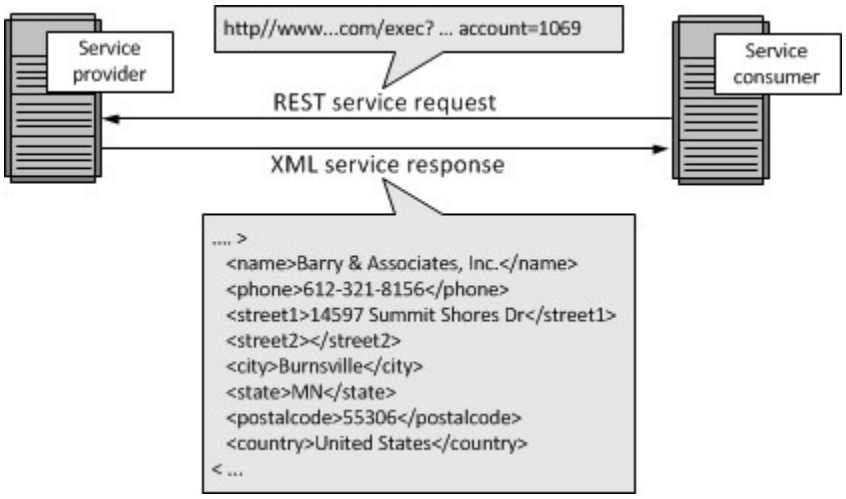

Figure 3.9: REST Message Exchange [44]

- Uniform interface

  Resources use four main standard HTTP operations: PUT (create), GET (read), POST (update), DELETE (delete). Each of these operations has different functionalities. PUT creates a new resource which can be removed by DELETE or transferred to a new state by POST, and GET is for retrieving current stat of a resource.

- Self-descriptive message

  In order to access the resource with more efficient way, REST decouples them from their representation to a variety of formats (HTML, XML, plain text, etc.).

- Stateful interaction through hyperlinks

  Each interaction between the client and the resources is stateless, i.e., request messages send from the client are self-contained which means that no client context is stored on the server between requests. All the information needed to service the request is included in each request from the client.

The main strengthes of REST described in [40]:

- It has already been well-known by W3C/IETF and the infrastructure has become pervasive. Clients and servers based on HTTP protocol are widely available for all primary programming languages and operation systems. Additionally, the HTTP port 80 is commonly left usable in most firewall configurations.

- It is inexpensive to acquire because of its lightweight. The requirement needed to build an interaction using RESTful is slight as developers only need a traditional web browser to do tests instead of a well-developed software.

- Efficiency in processing messages. As the REST choices lightweight message formats such as JavaScript Object Notation (JSON) [16], or plain text (for very simple data types), which can optimize the performance of a web service.

3.4.2 SOAP

SOAP is the abbreviation for Simple Object Access Protocol, which is a protocol specification for exchanging information in the implementation of Web Service between the service consumer and provider over the network. It uses XML as its message format. Normally, it is used over HTTP the same as the RESTful, but other protocols may also be used, i.e. Simple Mail Transfer Protocol (SMTP) [14].

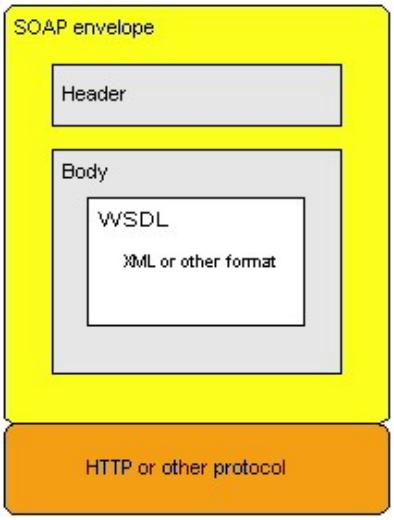

Figure 3.10: SOAP Message Elements [45]

The SOAP message is consisted of four XML elements as shown in figure 3.10. The top level element is called envelope which contains a header and a body. The envelope normally identifies the document as a SOAP message and also defines the message structure. The header in the envelope is an optional element to provide additional message-layer infrastructure information such as authentication, encoding data and Quality of Service (QoS) configuration. The body contains the payload of the message which can be defined using the Web Service Description Language (WSDL) [54] specification. The last element is the fault which carries a client or server error information. [40][45]

Comparing to the REST, the SOAP also has some advantages points [49]:

- Protocol Transparency and independence. While using SOAP, the same message can be transported across different middleware systems via different protocols not only HTTP.

- Interoperability. SOAP can work between different platforms and operating systems. Therefore, the data can be retrieved from the legacy systems via SOAP as long as the data is encoded in XML or other readable formats.

3.4.3 Summary

As stated above, both SOAP and RESTful have their own strengths that allow them to meet the different needs of applications. Nevertheless, in this research, RESTful is more suitable for use as the communication protocol between the service consumer and provider because it requires less bandwidth in data transactions. There are two main factors that are considered:

- Packet Size

  As this research aims to design an architecture based on the mobile environment, which may have limited network bandwidth and CPU performance than that of PCs, the size of the packet becomes an essential element. According to Brian [49], a SOAP message has a large amount of overhead because of the XML and the protocol design. Generating and parsing an XML message is a time-consuming task. In fact, the message generating and parsing can consume a lot of bandwidth, and the data transmission may lead to network latency. However, REST can decouple the resources for lightweight message formats such as JSON, which is easier to parse.

- Interoperability

  Since there are large number of different versions of SOAP that were developed. It is often that one vendor will use a version of SOAP that is slightly different from another vendor's. This mismatch in versions causes a problem when the cross-platform application is developed. On the other hand, REST has the advantage over SOAP because all developers must use it as an HTTP stack, which can be hosted on most devices today. Therefore, REST has better interoperability than SOAP. [21]

3.5 Data Synchronization

In a well-developed network, which can contains hundreds of devices, some identical pieces of data may be stored on each of these devices so that more stable data processing can be achieved. However, many data processing tasks are been done while the devices are disconnected from the network. Therefore, some data inconsistences can occur in the network. Due to the possibility of data inconsistency, the data synchronization and replication absolutely required in order to ensure data integrity, reliability and availability [32]. In order to keep the data consistent, as well as to overcome the limited mobile bandwidth, there are two categories of synchronization patterns that have been reviewed: data storage and availability patterns, and data transfer patterns [63].

3.5.1 Data Storage Patterns

In this pattern, the questions: "how much data should be stored?" and "how much data should be available without further transfer of data?" are addressed. Usually, mobile devices have limited resources (local storage

capacity, power, and processing capabilities) and some network constraints, which is why these two questions become the main concerns. There are two forms of storage included in this section: Partial Storage and Complete Storage [36].

- Partial Storage

  In this pattern, the device only synchronizes and stores data as needed to enhance the network performance and storage capacity. It is used in two main scenarios: the data set is either too large to store or is not needed in the application. The major advantage of this is the reduction of locally storage space as only needed data is stored. However, the weakness is also critical. If the network is lost, the data loading can fail and result in the termination of application [36].

- Complete Storage

  This storage pattern stores data on each device before it is needed. This storage pattern is desired while the network connection is not available or not stable. Nevertheless, while synchronizing the whole dataset to each device, the device storage usage is increased as well as the bandwidth [36].

3.5.2 Data Transfer Patterns

This pattern addresses the problem of data transfer quantity: "how can we synchronize between sets of data to minimize the data flow?" As discussed already, the network bandwidth is one of the main concerns of the mobile application. Thus, the data flow between mobile devices needs to be minimized to optimize the mobile network performance. There are three data transfer categories are covered in this section: Full Transfer, Timestamp Transfer, and Mathematical Transfer [36].

- Full Transfer

  In this transfer category, the whole dataset is transferred once the synchronization event is triggered. It is designed to satisfy the demand of the application which has a small dataset. The benefit of this pattern is the data consistency. Either the client or the server device sends the entire data to the other one, whose dataset is then replaced. In this way, the data on both devices can be identical. On the other hand, the data redundancy is the disadvantage. The whole data is sent no matter how much data is modified. Sending unchanged data can waste the network bandwidth and device storage.

- Timestamp Transfer

  The goal of this transfer pattern is to reduce the data redundancy and improve the network performance. During data synchronization, only those changed data since the last synchronization are sent according to the last-changed timestamp. Though this pattern can boost the network performance, the database design and synchronization implementation can be more complicated than Full Transfer. Moreover, the data deletion is not handled well in this model.

- Mathematical Transfer

  This pattern also only synchronizes parts of the data that are changed since the last synchronization, but with a mathematical method. This form of transfer is more efficiently than the timestamps, which should be used in the application that requires for absolute minimum bandwidth. However, it is a highly context-dependent method, different codes may be used for various types of data, and the process of reconciliation can be much more complex than Timestamp Transfer.

3.5.3 Mobile Data Replication

In many cases, a mobile device often only accesses to its local data, and most of these data are retrieved from other devices. Due to the constraints of the mobile device such as weak or intermittent connectivity and security vulnerabilities, data replication is essential for mobile data management to increase availability and cut communication costs [46].

The primary question in mobile replicated data system is how to handle data updates and synchronization in multiple replicas in different devices. All the updates to different replicas can be rejected if the connectivity is weak or not exist. Therefore, an appropriate data replication strategy should be adopted in this research.

In [46] and [25], the data replication can be divided into different groups based on the update modes (synchronous replication, asynchronous replication, and two-tier replication) and sources of updates (single-master and a group of masters).

3.5.3.1 Synchronous Data Replication

In synchronous data replication (SDR), data updates are applied to all replicas at all nodes as part of one atomic transaction, which means when one data copy is updated, all other replicas need to be synchronized concurrently. This replication can avoid data conflict and ensure serializable execution. However, it comes with several costs. Since the additional messages are included in the transaction, the replication process can result in a bad performance as well as more transaction response time. Moreover, if one replica is not available during the replication procedure, it may cause failure of a commit. Though, a technique called voting [57] can be used to solve this problem, it is very slow and can cause deadlocks. Due to the mobile device has limited bandwidth and unstable network connectivity, this replication is not applicable for mobile environment [46][25].

3.5.3.2 Asynchronous Data Replication

Under asynchronous data replication algorithm, the data updates are done on one single node in the network. After the transaction is committed, the replica update is propagated to other nodes. In this way, the update transaction requires less networking resources than the synchronous replication. Nevertheless, it also has some shortcomings. Since the data updates are committed by various transactions, the data consistency can be a challenging, and a conflict might occur [46][25].

Each of these two replication models can be used in different update control mechanisms as described following. There is no difference for synchronous replication in these two mechanisms as it does not matter on which device the data is modified. On the other hand, there are critical differences for asynchronous replication.

3.5.3.3 Group Update (Non-Master)

In the group update mechanism, which is also called multi-master model, the data can be modified by any nodes in the network. For the asynchronous replication in this model, the update transaction is sent to every node after the transaction committed on the update origination device. However, there is a possibility that two nodes may modify the same data object locally. In this case, a reconcile protocol should be applied. Normally, timestamps are the most common approach to sequence the updates. Each update request carries new data and the timestamp of the old object. A transaction is committed when the timestamp in the request is the same as the local timestamp of replicas. On the other hand, if these two values are not the same, which means the object might be modified by more than one node, then the update transaction is rejected, and update reconciliation is submitted. This can cause waits in the whole network. The waiting time is a variable which depends on the reconciliation frequency [25].

3.5.3.4 Single-Master Update

As mentioned above, in [25], Jim et al brought up two main updates originations: single-master and group. In the single-master model, each node in the network has a master node who is the only device that has the authorization to update the primary copy of the data object. All other replicas can only read the data. If any of the replicas wants to modify the object, it needs to send a request to the master node. Since this single-master property, there are no conflicts in the system which can ensure the data consistency. For the asynchronous replication in this model, the master node broadcasts the replica updates to all the slave nodes after the master transaction commits. After the slave nodes receive the updates, they use the timestamp, which is the same as in the multi-master, to ensure that all the replicas are in the same state. If the slave's local timestamp is older than the incoming one, the transaction will proceed. Otherwise, it will ignore the updates. To sum up, the asynchronous replication in single-master has better response time comparing to the asynchronous replication in a multi-master network as the previous replication pattern sends out fewer messages. However, single-master mechanism requires the slave to connect the master node and participate in an atomic data transaction as long as there is an update. Therefore, it is not a suitable for mobile applications [25].

3.5.3.5 Two-Tier Data Replication

In [48] and [41], both of these paper brought up a protocol called Two-tier Replication to ensure the data consistency in mobile networks. This protocol was first introduced by Gray et al in [25]. The main idea is

simple; each data object is mastered by a node in the system, whenever there is a modification made on the mobile nodes, they can make a tentative update, and then check if the update is acceptable by the base nodes.

In the two-tier replication protocol, there are two varieties of nodes (mobile and base), two data versions on mobile nodes (master and tentative), and two types of data transactions (base and tentative) [25]:

- Mobile Nodes

  The set of nodes that are disconnected from the network often. They have a replica of the database stored locally which can be modified while they are offline.

- Base Nodes

  The set of nodes that are always connected to the system. They also have a replica of the database stored but with the most updated version.

- Master Version

  The latest data value retrieved from the object master while it was connected. The mobile nodes may have different versions as the object master has.

- Tentative Version

  The local data object that is updated by the tentative transactions while the node is disconnected.

- Base Transactions

  The collection of transactions that only work on master data, and they generate a new version of master data. They only involve the connected nodes.

- Tentative Transactions

  The collection of transactions that work on the local tentative version of data and produce new tentative versions. They also generate base transactions to be committed on the base nodes while they are reconnected.

This replication model can handle both device connected and disconnected cases. In the connected scenario, the two-tier system will run as an asynchronous replication in single-master condition. However, the procedure in the disconnected case is more complex. In the beginning, a mobile node has a replica of the master data from a base node. After it loses the connection to the base node, it can make tentative transactions on that master data locally. These transactions generate a tentative data version locates on the mobile node. After the mobile node reconnects to the base node, there are several processes for mobile node to operate [25]:

- The tentative data versions are omitted as they will be refreshed after the connection.

- All the local replica updates and tentative transactions are sent to the base node for synchronization.

- Receive all the replica updates from the base node and the feedback of each tentative transaction sent before.

On the other hand, the base node also has some operations to run corresponding to the mobile nodes [25]:

- Send all the updated data objects to the reconnected mobile nodes.
- Receive the replica update and tentative transactions from the mobile nodes.
- Rerun the tentative transactions in the order they were committed on the mobile node, and then it will send the results to the mobile node.
- After the base transactions are committed without conflicts, all the updates will be propagated to all replica nodes.

3.5.4 Summary

Normally, the solution of data storage and transfer for a large system on PCs would be to increase the network performance and extend the storage space. However, extending the storage space is impractical on mobile devices. Therefore, how the data is stored and transferred in a mobile network becomes a critical question. Regarding, the storage pattern in data synchronization, the exact type to use in the mobile device can be ambiguous. The storage pattern is therefore determined by the size of datasets and the device storage capacity. The timestamp works the best for a mobile application's transfer pattern. The full transfer can affect the network's bandwidth and occupy extra mobile storage. Although, the mathematical pattern can enhance the network performance, but it is not reusable for different data types.

Moreover, in the previous analysis, there different data replication architecturse were compared: synchronous replication, asynchronous replication in both single-master and multi-master, and two-tier replication. In both synchronous and asynchronous replication, nodes are required to be connected in order to update the replicas. Furthermore, the synchronous replication and asynchronous replication in multi-master condition can cause a long-time waits and deadlock; although, the asynchronous replication in the single-master model may have better performance than the synchronous replication. For two-tier replication, it supports data updates while mobile devices are disconnected, and it combines the advantages of both synchronous and asynchronous replication [25].

3.6 Database

Generally, databases refer to the organized collections of data so that it can easily be accessed and managed. The database can be classified based on their organizational approach. The most common style is the relational database which is opposing to the non-relational database. In this section, the two main prevalent databases: SQL and NoSQL (Not-only SQL) will be presented and compared.

3.6.1 Relational Database / SQL

SQL stands for Structured Query Language. It is an interface language that is used for relational database applications. In a relational database, a table is used to organize the different set of data. In a table, each row represents a record and each column represents a field of that record.

Each record in the table can be retrieved by a primary key which uniquely identifies each record in a table; the primary key can avoid the data redundant in a table. Additionally, there are some relations among different tables in a database schema. The linkage is connected by the foreign key. "A foreign key value represents a reference to the tuple containing the matching candidate key value (The referenced tuple)" [17].

In [26], Christoforos states that the normalization of the schema is an important design aspect of a relation database. There are three steps are described in [15] [17]:

- First Normal Form (1NF): Create a new table for each set of related data to avoid repeating data so that all tuples in a table are unique.
- Second Normal Form (2NF): If a set of values are the same for various records in a table, then transfer them to a new table and link them via a foreign key.
- Third Normal Form (3NF): Those columns that do not depend on the primary key should be removed.

In a relational database, the ACID transaction properties are used to ensure the reliability: Atomicity, Consistency, Isolation, and Durability.

- Atomicity

  All or none of the transactions are finished. If one of the transaction operations fails, the entire transaction fails.

- Consistency

  The database needs to remain consistent before and after the transaction commits.

- Isolation

  Each transaction needs to be completed serially so that the associated data does not affect others.

- Durability

  After a transaction is committed, the data will remain in the state.

Despite the high usage of the relational database system, it has shortcomings. The biggest problem is horizontal scalability. "Traditional database products have comparatively little or no ability to scale horizontally on these applications." [12]. As explained by Rick [12], the term "horizontal scalability" refers to the capability to distribute both the data and data operations over a large number of servers without RAM or disk sharing. It differs from the "vertical scalability" which is the ability to utilize multiple cores

that share RAM and disks. This problem gets more critical in a Peer-to-Peer system as the data is spread on different nodes. Once a tuple of the table is changed on one node, then the update is needed on each node on which the data is located. If the node is offline, then the data inconsistency may occur [47].

3.6.2 Non-Relational Database / NoSQL

As the constraints of relational database mentioned above, NoSQL, which is also known as "not only SQL" or non-relational database, has started exploding during the late 90s to early 2000s. The goal of developing NoSQL database is to provide high availability, flexibility, and performance for large-scale Internet applications. Normally, NoSQL is the term used to include all the non-relational databases. As the data types and user cases can be very wildly in these databases, the NoSQL databases are induced into four types [30]:

- Key-Value stores

  In these databases, the data values are paired by keys, like a hash table. A good example to explain this is a file system where the path acts as the key and the file acts as the data contents. If there is a change been made, the entire value other than the key needs to be updated.

- Graph stores

  This type of database specializes in handling interconnected data. The nodes and edges are the primary concepts in the graph database. The nodes correspond to the data record, and the edges present the relations between the nodes. "The strength of a graph database is in traversing the connections between nodes" [30].

- Column Stores

  In a relational database, all the data are stored in a specific table's row. In a column database, it normally serializes all the values of the same column together so that the retrieval of an aggregated data on a particular attribute fast[30].

- Document Stores

  In this database, the data record is stored as a document on the disk. The document can be thought as collections of key-value pairs where keys are normally strings and values can be different data types (strings, numeric, Booleans, arrays, and other key-value pairs, etc.). The popular syntaxes for document database are XML and JSON.

Rick [12] states that the NoSQL databases normally have six features as following:

- The ability to scale "simple operation" (key lookups, read and writes of one data record or a small amount of records) horizontally over many servers.

- The ability to replicate and distribute data through large number of servers.

- Simple call level interface compare to the SQL.
- A weaker transaction properties than the ACID.
- Distributed index and RAM are efficiently used.
- Strong ability to dynamically add new attributes to existing data records.

Apparently, the most important feature of NoSQL is the horizontal scaling. This allows the database to support a large amount of data read/write operations. Meanwhile, as mentioned in the fourth feature above, NoSQL databases generally do not support the ACID properties; instead, the BASE is brought up [47] [39]:

- Basically Available

  Use replication technique to minimize the possibility of data unavailability, and partitioning the data into smaller parts on multiple storage servers to reduce the failure risk.

- Soft State

  In ACID, the database requires all data to remain consistent. For the NoSQL database, it allows the data to be inconsistent and turns over the data consistent designing to developers.

- Eventually Consistent

  In the NoSQL database, the data consistency is achieved eventually at some future point in contrast to ACID, which enforces the consistency as soon as the transaction is committed.

By promoting BASE instead of ACID, better performance, availability, and scalability can be achieved. However, the instantaneous consistency is traded for availability in CAP [22]. In the CAP theorem [24], it states that a system can only achieve two out of three of the following properties: consistency, availability, and partition-tolerance. According to Fu[22], due to the requirement of different applications, the CAP trade-off is different. In the mobile environment, as the network connection is unpredictable, the partition-tolerance is required to prevent message loss.

3.6.3 Summary

In conclusion, the relational database has been the most dominant database system for decades. However, as the number of web and mobile applications dramatically increases, more and more data is generated. Therefore, a more flexible and stable database solution is required. For this reason, the NoSQL database has been brought up. Compared to an SQL database, the NoSQL database system has the following advantages [30]:

- Flexibility

  In a relational database, the data values are referenced by an attribute that is pre-defined in a schema, which can be a constraint for a developer when a new attribute needs to be added. For NoSQL,

"schema flexibility and intuitive data structures are key features" [30]. Schema updates are generally not required in NoSQL database, and complex database schema changes are eliminated by codebase modifications.

- Scalability

  As stated in section 3.6, database scale refers to both vertical scale and horizontal scale. NoSQL databases are more specialized to horizontal scale, which can spread the data across a cluster of nodes more easily than the relational databases.

- Availability

  The downtime of a database can result in both customer and data loss. In some NoSQL databases, different data replication architectures are offered, which provide high availability for data reads and writes. In NoSQL, even if one or more nodes goes down, the other nodes in the network can continue data operations without influence.

3.7 Conclusion

In this chapter, numerous approaches were described in order to solve the key challenges that were mentioned in Chapter 2, and these are:

- Device Connection in the non-Internet network

  Mostly, device connections can be made via the Internet. When Internet service is available, mobile devices can send requests to the cloud server. Various technologies have been brought up for the non-Internet environment, but Wi-Fi Direct and Bluetooth are the most dominant technologies. Through comparison of these two technologies from different aspects, Wi-Fi Direct was found to be more suitable for this research due to its network capacity, compatibility, and transmission speed.

- Tracking information consistency in the intermittent network

  The file tracking information is stored on the cloud server database when the Internet is available. However, as the architecture permits mobile devices to transfer files in a non-Internet network, the file also needs to be tracked when the Internet connection is lost. Therefore, a synchronization mechanism needs to be designed to ensure the integrity and consistency of the tracking information. In section 3.5, several data replications were reviewed. Two-tier replication was determined to be the most applicable synchronization model for the mobile device. It can ensure the data consistency while the devices are disconnected from the server.

- Bandwidth

  Based on the reviews in section 3.1, limited resources, network bandwidth, and intermittent connectivity are the three main constraints that mobile computing has. The constraints of limited resources can be

solved by good database model and design. Unstable connectivity causes data inconsistency, which can be resolved by a proper synchronization pattern. The architecture can achieve high network bandwidth from two aspects:

  - Restful lightweight data transmission technique, and
  - Proper synchronization mechanism to minimize the transferred updates

Besides the reviews of technologies that can be used to solve this research's challenges, some other research was analyzed for the purpose of improving the scalability and availability of the architecture:

- Mobile cloud computing

  Mobile cloud computing combines the advantages of mobile computing and cloud computing. The native application of mobile cloud computing can achieve high computation capacity as well as optimized local storage space. In this research, the file objects and file tracking information are stored on the cloud server for mobile devices to access.

- Scalable database

  Our architecture needs to have the ability to scale well because devices can, at any time, join or leave the network. NoSQL database is broadly used due to its advantages in availability and horizontal scalability compared to the relational database. Therefore, it is desired for the local data storage to store tracking information.

Chapter 4

DESIGN AND ARCHITECTURE

Overall, there are five essential technologies in the design of the file tracking architecture:

- Mobile Peer-to-Peer (P2P),
- Mobile cloud computing,
- Web service,
- Different synchronization mechanisms, and
- Scalable database.

These five technologies were reviewed and compared in Chapter 3. The review showed that, when comparing them from various aspects, such as the maximum number of devices in the network, compatibility, and data transmission speed, Wi-Fi Direct is more advantageous than Bluetooth for mobile P2P technology in the non-Internet environment. With respect to synchronization mechanisms, a proper synchronization mechanism is needed in order to keep the tracking information consistency on the mobile device. The synchronization mechanism that was used included two-tier replication pattern and time stamp pattern because they allow better consistency and lower bandwidth usage. RESTful, a lightweight protocol that can achieve lower bandwidth usage than SOAP, was utilized for data transmission protocol between multiple devices. Finally, scalability is a valuable property in a distributed system. In our file tracking architecture, a NoSQL database was used because it is more specialized to horizontal scalability than the traditional relation database. In this chapter, an architecture that integrates these four technologies to track data in mobile networks will be designed and proposed.

4.1 Overview

As stated in chapter 3, there are two main types of networks that have been widely used: Local Area Network (LAN) and Wide Area Network (WAN). The LAN is usually constructed on the P2P network architecture for connection setup. On the other hand, the Internet is the representative of WAN. The data tracking mechanism in these two networks was implemented in our architecture.

4.1.1 Data Tracking in WAN

In the WAN network, there are two primary roles: server and client. The server and client have different jobs and responsibilities as shown in figure 4.1.

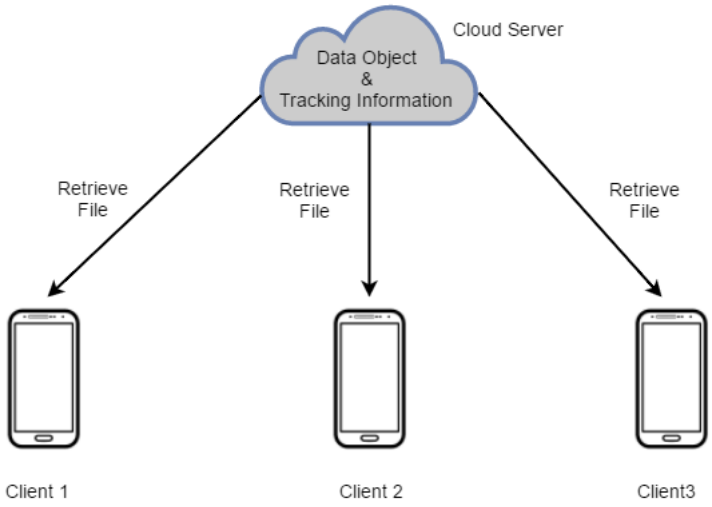

Figure 4.1: Basic Cloud Server and Clients Architecture

- Client

  The clients in this network are usually the mobile devices. The clients do not need to do the cumbersome computation process as the cloud server can handle the majority of the data manipulations. The primary responsibility of the client device is to send requests to the server and retrieve data from the server. It acts as the medium between the user and the cloud.

- Cloud Server

  The main responsibility of the cloud server is to provide file storage, and to store the tracking information in the database. There are two different sets of data in the cloud database: 1) track first and 2) recover first. More details about this will be covered in the later sections.

4.1.2 Data Tracking in LAN

The Local Area Network (LAN) is typically used for device communication when the Internet is unavailable or not needed. The Peer-to-Peer technology was used to setup a network in the non-Internet environment. After comparing the different technologies, we found that Wi-Fi Direct could meet most of our requirements.

The device in the proposed Wi-Fi Direct network has three primary roles of: the local master, the group owner (GO), and the client devices. Each role has different functionalities, as shown in figure 4.2:

- Local Master

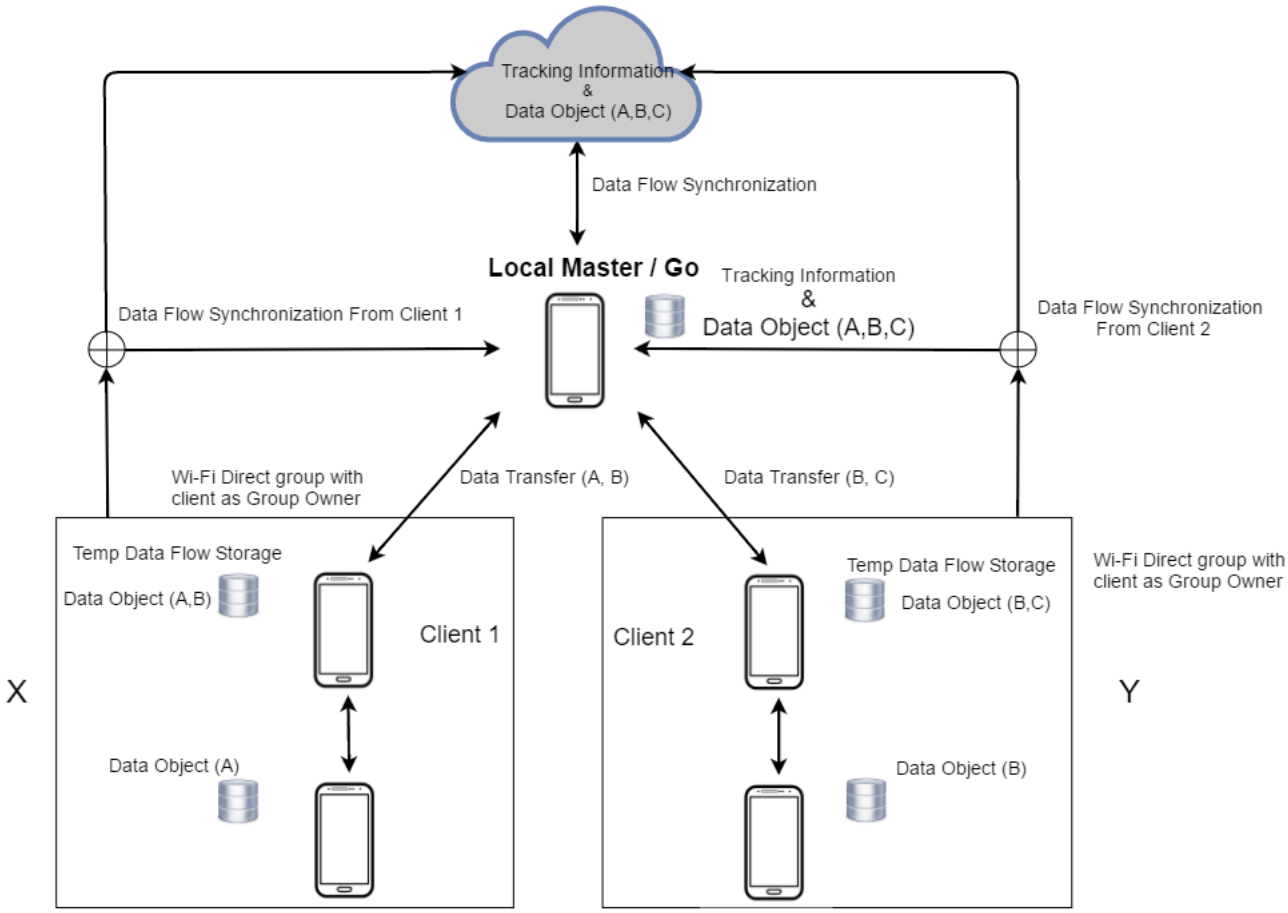

Figure 4.2: Basic Tracking Architecture Design

The local master's job is to provide the file access when the cloud server is not available. The local master has the same file objects and tracking information as the cloud server. The tracking information will be synchronized by the synchronization mechanism whenever the connection between the local master and cloud server is reestablished.

- Client Devices

  The client devices serve as slave devices in the group. The client devices can retrieve a file from the local master when the cloud server is not available. Moreover, each client can also send the received files to other devices with two restrictions:

  1. The client device of the local master can only send the file to one single device once, and
  2. The device which received the file from a client device instead of the local master cannot broadcast the file again - they can only read.

  The goal of these constraints is to protect data privacy as well as the consistency of tracking information.

- Group Owner

There are two types of group owner (GO) devices In the proposed architecture.

1. The GO is the local master. When the local master connects to client devices, it acts as the GO in the network. The GO has the authorization to send the file objects as well as manage the file flow in the network for later synchronization.
2. The GO is a client device. This type of GO is depicted as client devices 1 and 2 in figure 4.2. There is one and only one GO in a single group due to the design of Android Wi-Fi Direct architecture. After a client device disconnects from the local master, it can be the GO in other groups (group X and Y in figure 4.2). However, the restriction of the client devices, mentioned above, are applied. Moreover, this type of GO device keeps tracking the data flow locally for tracking information updates when the connection to the cloud server or the local master is available. More details will be covered in the next section.

4.2 Architectures Details and Functionality

Different architecture designs are necessary for the different roles of the devices in the system. Each of the architecture design follows the standard Model-View-Controller (MVC) pattern.

- Group Owner Architecture

  As stated in the section 4.1.2, there are two types of GO, and each of them has different functionality and design.

  1. Local Master as GO (figure 4.3)
     - Model

       The local master's model has a File Storage Database (FSD) to store all the file documents. Moreover, it also has a File Track Storage (FTS) which is defined as the master version in the two-tier replication, and To Be Updated Transaction (TBUT) which is demonstrated as a tentative version in the two-tier replication. In FTS, it stores all the data tracking records in a key-value NoSQL database. There are two primary purposes of FTS:
       * Track Data (Track-first)

         In this group of FTS, the name of file object acts as the unique key of a data attribute. The information of the device, which requested the corresponding file, is stored in the value field. Every time a client device retrieves a file, the Tracking Manager triggers the push (the file is retrieved the first time) or update (the file has been requested before) event on the database. This database mainly focuses on solving the problem "What is where." The local master can easily find out which data objects have been requested, and also the detail information about the devices by generating the tracking information. The

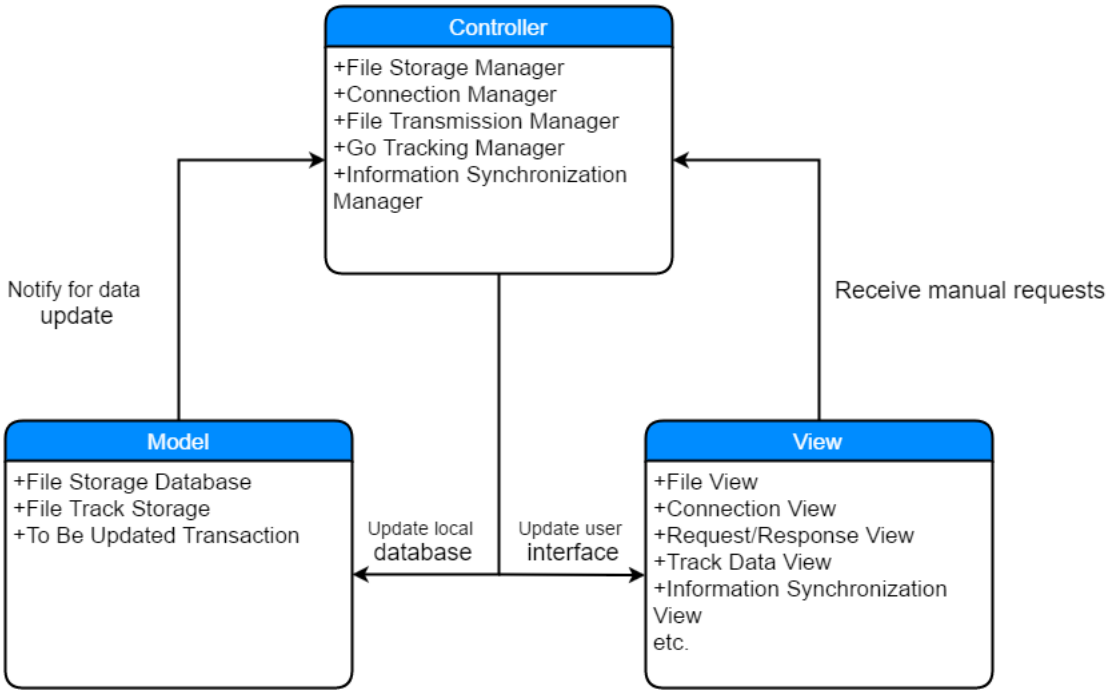

Figure 4.3: Local Master Model-View-Controller Architecture

format of the data structure is as follows: File Name:{Timestamp:{device information}}. The data structure is a nested JSON object which contains three levels:

- First level: file Name. The file name is a nested JSON object which uses the name of the requested file as the key, and the second level as the value.
- Second level: timestamp. The timestamp is the JSON object which is contained under the first level. It uses the current timestamp as the key, and the device information as the value. The purpose of this level is to avoid data conflict by using different timestamp.
- Third level: device information. This level stores all of the information of the device that has requested the file, which is stored in the first level.

* Recover Data (Recover-first)

The second group of FTS is used to solve the problem "Who has what." It can help the local master to recover the file objects on the client devices. The MAC address of the client device is stored as the key field, and the file information stored as the value. A new data set is generated in the database when a device retrieves a file object from the host device. The format of this data set is as follows: Device MAC:{Timestamp:{file information}}. The data structure also has three levels of JSON objects:

- First level: device MAC. In this level, it takes the MAC address of a mobile device as the key, and a JSON object as the value.
- Second level: timestamp. This level is the same as the second level of Track-first, but

it takes the file information as the value.

· Third level: file information. This level stores all the files that have been requested by the device which is denoted in the first level.

Additionally, the TBUT collects all of tracking information that has been generated since the device has disconnected from the cloud server. The data structure is the same as the FTS.

– View

The view object handles all the mobile interfaces and human interactions. Since the client is allowed to receive and view the file locally, it has

* File View (for the user to browse files),
* Request View (for sending out requests), and
* Connection View (for connection setup between the client and the host devices).

The local master also has an interface view that is used to browse the file tracking information, such as the list of devices on which the files are stored since it has the authorization to track data. A synchronization view is necessary as well because the tracking information needs to be updated on both the local master and cloud server after they are reconnected.

– Controller

This object accepts the user's input as events. The controller notifies the corresponding functions to run as soon as there is an event from the views. There are five leading managers to take care of different user interactions in this architecture:

* Storage Manager (FSM),
* Connection Manager (CM),
* Data Transmission Manager (DTM),
* Local Master Tracking Manager (LMTM), and
* Information Synchronization Manager (ISM).

The FSM manages all the file storing on the device. The FSM notifies the FSD to store the objects in the local directory once a file is received. The CM is used to deal with the Wi-Fi Direct connection/disconnection events between two devices. The DTM manages the requests and response between the client and master device. The LMTM is used to deal with all of the data tracking events. It updates File Track Storage whenever there is a file transaction in the network. Moreover, the local master also has Information Synchronization Manager (ISM) to ensure the consistency of tracking information on the local master and cloud server. The primary duty of ISM is to generate the To Be Updated Transaction whenever there is a file flow after disconnection. The synchronization details will be covered in the workflow section.

2. Client Architecture (figure 4.4)

– Model

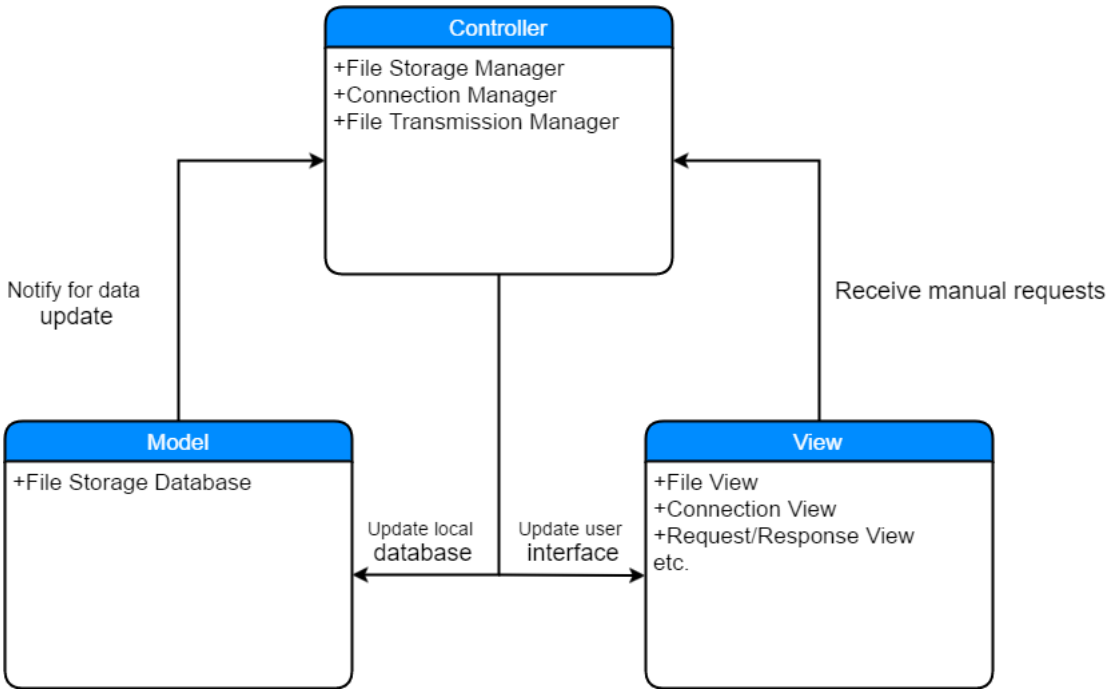

Figure 4.4: Client Model-View-Controller Architecture

The model deals with the data store and manipulation in the system. For a client device, the model only has a file storage Database (FSD) that stores all of the file objects received from the host devices.

- View

  The view object of client contains three primary interfaces: File View (for the user to browse files), Request View (for sending out requests), and Connection View (for connection setup between client and GO or cloud server).

- Controller

  There are four leading managers to take care of different user interactions: File Storage Manager (FSM), Connection Manager (CM), Data Transmission Manager (DTM) and Information Synchronization Manager (ISM). These four managers have the same functions as the local master's.

3. Client device as GO architecture (client 1 and 2 in figure 4.2) in figure 4.5.

   - Model

     The model for this type of GO is slightly different from the local master. Similarly, this model has the FSD to store all the files received. It only has the TBUT data structure to store all the offline file transactions that have performed rather than the permanent File Track Storage on the local master. The data in the TBUT will be wiped by certain events which will be explained later.

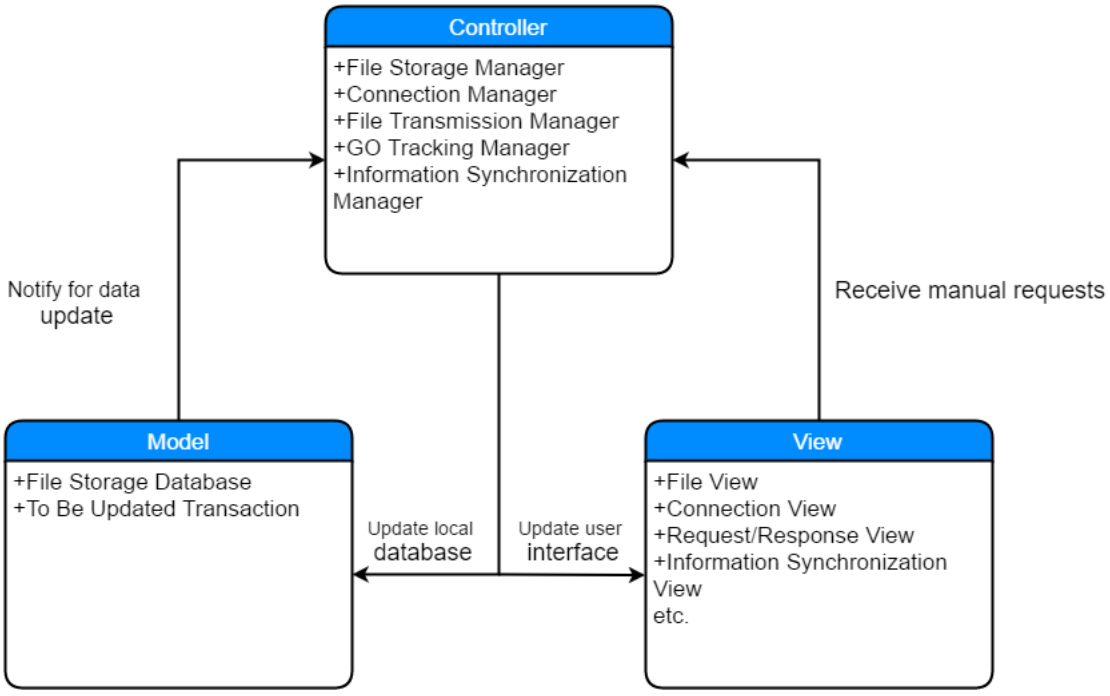

Figure 4.5: Client as GO Model-View-Controller Architecture

- View

  With the exception of the Track Data View, the views for the client device as the GO are almost identical to the local master. Since the track information on this device is temporary, the Track Data View is not needed under this situation.

- Controller

  The controller for this GO is the same as the controller for the local master. Additionally, the information synchronization manager needs to deal with the clearance of the TBUT.

- Cloud Server Architecture

  The basic architecture of cloud server is the same as the local master since they share the same models and controllers. The only difference between the basic architecture of the cloud server and the local master is the view. In the cloud server, it does not need the mobile interfaces to interact. On the other hand, the cloud has a web interface for the administrator to manage data.

4.3 File Track Workflow

In reviewing the concept of mobile computing in section 3.1, it was found that the two major mobile device communication states are connected and disconnected. The file tracking for disconnected devices is the key challenge in this research. For example, the local master or a client can be disconnected from the cloud server due to the intermittent Internet connection; a client device can be disconnected from the local master when

it is out of the network range. Therefore, in the proposed architecture, both of connected and disconnected scenarios were taken into consideration. In this section, the workflow of the architecture will be explained.

4.3.1 Cloud Server is Connected

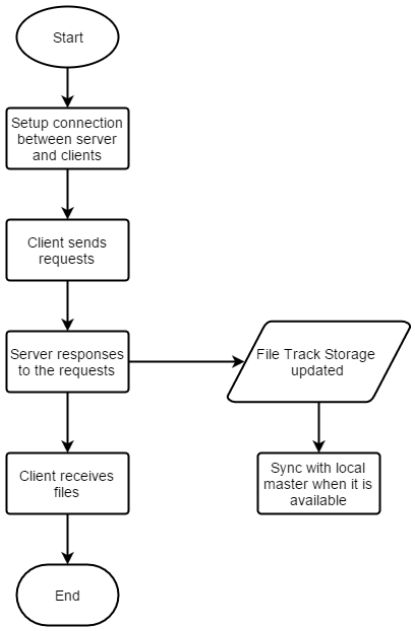

Figure 4.6: Cloud Server File Tracking Workflow

The file tracking process when the cloud server and client devices are connected is straightforward, as demonstrated in figure 4.6. The first step in the procedure is to set up a connection between the cloud and clients. As soon as the connection is established, the client device can send a retrieve request to the server. Once the server finds the matched file object, it sends the file back to the client device, Meanwhile, the File Track Storage records all of the tracking information: track first and recover first. The details of track data and recover data were explained in the model of local master architecture in section 4.2. The client can browse the file locally via the file view after. Simultaneously, the tracking information is stored in the TBUT of the cloud server. Whenever, the local master is connected, the tracking information will be synchronized.

4.3.2 Cloud Server is not Connected

This section explains how the architecture handles the situation when the cloud is not available. The local master acts as the host device in the mobile network and records all the tracking information.

4.3.2.1 Local Master can be Connected

The client can retrieve files via the local master as long as they are within the range of Wi-Fi Direct network in the case of the cloud server being offline. In this situation, the challenge becomes tracking information's

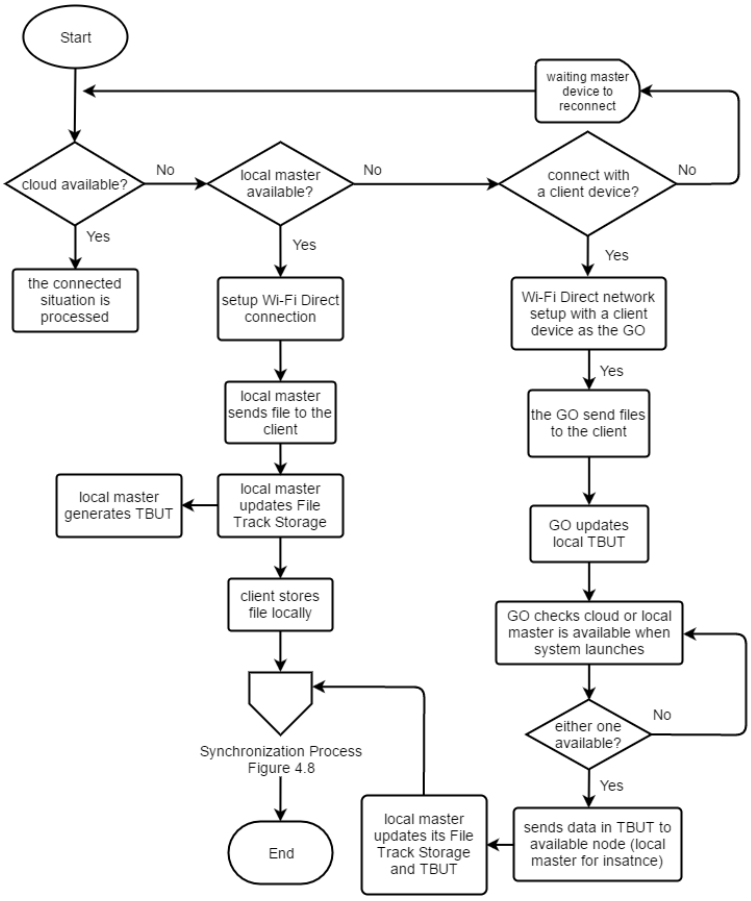

Figure 4.7: Local Master File Tracking Workflow

integrity and consistency in the intermittent network, as discussed in Chapter 2. The major responsibility of the local master is to keep file available as well as maintain its local file and tracking information consistent with the cloud server. There are several steps that need to be accomplished in order to ensure the consistency and integrity of tracking information (demonstrated in figure 4.7):

1. Local master retrieves all the files and tracking information at the first connection with the cloud server.

2. When a client needs to retrieve a file remotely, it will check if the cloud server is available. If the connection can be established, then the architecture will handle the file tracking in the connected situation as described in section 4.3.1. Otherwise, the client checks if its distance between the local master is within the coverage of Wi-Fi Direct. If the local master can be reached, then the system follows the next steps, otherwise the situation where a client device is the GO (explained in the 4.3.2.2) will be applied.

3. The local master can send desired files to the clients as soon as the local master and client devices are connected. Meanwhile, the local master's Tracking Manager triggers an event which notifies the File Track Storage to push or update the tracking data objects depends on the file request appearance. The

To Be Updated Transaction, which records all the tracking information in offline condition, will also be generated.

The following procedures describe the tracking information synchronization when the cloud server and local master are reconnected again. The workflow is presented in figure 4.8.

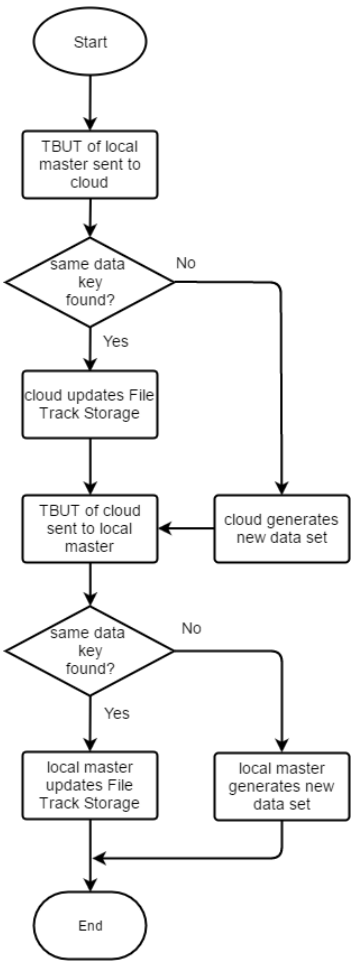

Figure 4.8: File Tracking Information Synchronization Workflow

4. The architecture checks the TBUT on each of these two nodes as soon as the local master is reconnected to the cloud server. The following steps will be processed if either one is not empty; otherwise, an empty TBUT means there is no file transaction during the disconnected time.

   There are two situations where file transactions can occur when the local master and cloud server are disconnected:

   (a) Either the local master or the cloud server has transferred files to other devices during disconnection period. In this case, it does not matter which one did the transaction as the solution is the same. The local master will be an example explained below.

i. The local master's data objects in TBUT of are sent to the cloud server. In TBUT, there are two main key-value data sets:

- The key is file's unique id, and the value is the device information.
- Conversely, the second data set takes the device's MAC address as the key and the file information as the value.

Each of the two key-value data sets have the timestamp attribute in order to avoid data conflicts during the synchronization process.

ii. The cloud server compares the incoming data objects' first level, which is the key, with those keys in the File Track Storage of the cloud.

iii. If the same key is found, then the corresponding data in the cloud server's FTS will be updated.

iv. On the other hand, the cloud needs to push a new set of data in TFS if there is no identical data object key is found.

(b) Both of the local master and cloud server transferred files to clients during the disconnection. In this case, the following processes will be performed:

i. The local master's data objects in TBUT are sent to the cloud,

ii. The cloud server compares the incoming data objects' keys with those in its File Track Storage,

iii. If an identical data object is found, the corresponding data in the cloud server's FTS will be updated,

iv. The TBUT data of cloud is sent to the local master, and

v. TFS of each master hosts updates its data based on the received data.

Device MAC	File Name	Actions	Time
client 1	File A	Retrieved	2:00

Table 4.1: Recovery-first of client's TBUT

File Name	Device MAC	Actions	Time
File A	client 1	Retrieved	2:00

Table 4.2: Track-first of client's TBUT

The medical reports example will be used to explain these processes in details. The hospital's cloud server stores the medical files of its patients Alex, Bob, and Charlie. The local master device downloads the patients' files immediately as soon as the cloud generates their files. One day, Alex experiences a medical emergency. However, the Internet service in the area is down for maintenance, and the cloud server cannot be accessed. So the doctor, Emily, downloads Alex's file from the local master to her tablet for analysis. At

this time, the MAC address of Emily's devices is recorded in File Track Storage of local master and the To Be Updated Transaction is generated (shown in table 4.1 and 4.2.) When the maintenance of the could server is finished, the local master sends the data objects in TBUT to the cloud which contains two main data sets:

- Alex's file name is the key attribute, and the information of doctor's device are stored as values. These values are stored in order to track where the data is located.
- The MAC address of the physician's device is the key attribute, and Alex's file information are the value field. The data stored are used for recovering data from the doctor's device if it is lost.

After the cloud receives the data from the local master, it looks up its TFS to find out if it contains the data objects which take Alex's file or the MAC address of Emily's device as keys. In the case that nothing is found, the cloud creates a set of new data objects in the File Track Storage. However, if there is a data object found in the cloud's TFS, the existing data objects will be updated.

4.3.2.2 Local Master can not be connected

The previous steps explained how the file flow is tracked when the local master is the GO of Wi-Fi Direct network. However, the connection between clients and the local master cannot be persistent. A device may not be able to connect to the local master when it is out of the range, or the local master is down. In this case, neither cloud nor local master is available. Therefore, another scenario, where the client device of the local master acts as the GO, is taken into consideration.

1. If a client device needs to retrieve a file when the local master and the cloud are both offline, the user will need to check if the nearby devices have the desired file. If any nearby devices have the desired file, then the next few steps will proceed; otherwise, the client device waits until the local master or cloud is back online.
2. The nearby device establishes a Wi-Fi Direct network, and acts as the GO; the requested device can join the group as a slave.
3. The GO can send a file to one slave device in the network with certain limitations, as mentioned in section 4.1.2.
4. The TBUT of the GO stores the tracking information.
5. Every time it launches, the system checks if the local master or the cloud server is available. If either of the local master or the cloud server is found, then the data objects in GO's TBUT are sent to the available node. An available node, for example, is the local master.
6. The local master compares the incoming data objects' keys with those in its File Track Storage.
7. If the same key is found, then the corresponding data in the local master's FTS will be updated.

8. If there is no identical data object key is found in TFS, then the local master needs to push a new set to TFS.

9. The TBUT of local master records the incoming tracking data for synchronization.

10. After the local master generate its TBUT, the synchronization processes, as described in step (4) of 4.3.2.1 will be executed in order to ensure the tracking information consistency in the cloud and local master.

The continuous medical report example is demonstrated to explain this situation more clearly. The previous example describes the scenario when the local master is available. However, when Emily goes back to her apartment, the distance between her device and the local master in the hospital is too far to establish a connection. But Emily's neighbor, David, is an expert on Alex's disease. Emily uses her tablet to transfers Alex's medical report to David so that he can help her treat Alex. However, due to hospital's privacy policy, David cannot share this report with anyone else. The next day, when Emily goes back to the hospital, the file transaction that was made last night is synchronized to the local master as soon as she launches the system.

4.4 Network Bandwidth

As mentioned in Chapter 2 and 3, the mobile network bandwidth is constrained due to intermittent connectivity, and it is a challenge to this research. Therefore, maximizing the efficient use of the limited bandwidth resource is valuable.

In the proposed architecture, the efficient bandwidth consumption is achieved by applying three aspects:

- Connection technology. After comparing different connection technologies, Wi-Fi Direct became the first choice because of its high transfer speed, which is nearly ten times faster than Bluetooth. Moreover, the maximum number of devices that can be connected in the network is more flexible.

- Appropriate synchronization patterns. As described in section 3.5.2 and 3.5.3, there are several transfer and storage patterns that occur during the synchronization process. In this architecture, the timestamp pattern is more suitable. Meanwhile, we proposed a data set called To Be Updated Transaction, which is based on the two-tire replication, so that only the necessary data is sent. Sending only the data necessary for the synchronization can help to reduce the data flow in synchronization procedures.

- Lightweight web service protocol. In the proposed architecture, we used the RESTful web service which is a more lightweight protocol as it supports multiple message format. It reduced the data load sent between devices.

4.5 Summary

In this chapter, we designed an architecture which can achieve the following objectives:

1. The mobile devices can be connected in the Internet and non-Internet environment,
2. Ensuring file tracking information consistency in mobile networks,
3. Integrity of file tracking information is achieved in mobile networks, and
4. Improve network bandwidth by using appropriate web service protocol, connection technologies, and synchronization patterns.

To ensure the tracking information consistency, the architecture constructs a replication mechanism based on the two-tier replication model which supports data update when the device is offline.

For the data tracking, two data storage patterns were designed: track first and recover first. In the track first mode, the architecture deals with the question "What is where" to improve the data management and supervise. In the recover first mode, the architecture focuses on the question "Who has what." The solution to this question can help the user to recover data from lost devices.

To overcome the bandwidth limitation of mobile networks, the architecture uses Wi-Fi Direct for device connection in the non-Internet network, the RESTful web service for data transmission, and timestamp synchronization pattern to reduce the amount of data transferred.

Chapter 5

IMPLEMENTATION

There are three different implementation designs in the architecture: the local master, the client device, and the cloud server in the architecture. Each of these three roles follows the Model-View-Controller design pattern. In this chapter, it covers how the different components are integrated and implemented.

5.1 Local Master Design

The local master is the local host device while the cloud is not available. The model and controller are the most important parts. The details of implementation are as following:

5.1.1 Model of the Local Master

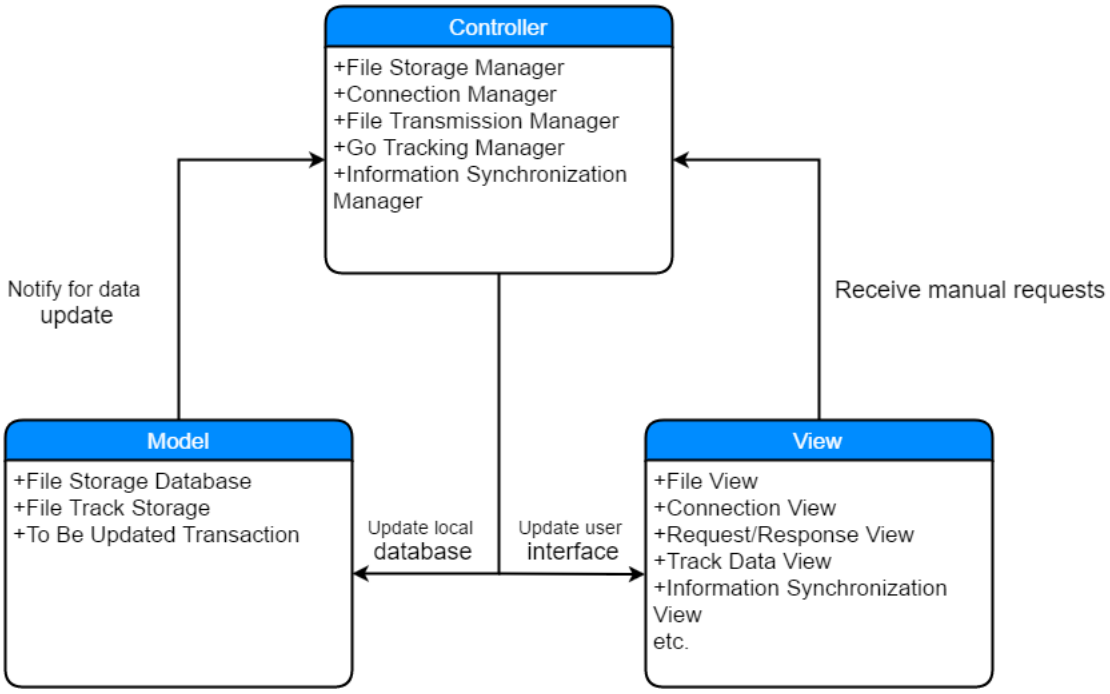

Figure 5.1: Local Master Model-View-Controller Architecture

In figure 5.1, there are three main data objects in the model: File Track Storage (master version), To Be

Updated Transaction (tentative version), and File Storage. The FTS stores all of the tracking information that is identical to the information on the cloud server. The TBUT stores the most updated information that has not yet been synchronized. Both the FTS and TBUT are stored in the Couchbase Lite NoSQL database locally. The file objects are stored in the application's directory on the device. There are several procedures to store the file tracking information based on the file transactions, they are:

1. Create File Track Storage database document.

```
public void createLocalTrackDocument()
{
        localTrackFileDocument = new Document(localDatabase,"LocalMasterTrack");
        Log.d("local track document", "local track document created with id:"
        +localTrackFileDocument.getId());
}
```

   This class defines a local database document with the ID "LocalMasterTrack." The localDatabase is an instance of the local Couchbase Lite database. The code snippet is called at the first time the user runs the application.

2. The local TBUT database document is created to store all the tentative data versions.

```
public void generateLocalTbut()
{
        Document localTbut =  new Document(localDatabase,localTBUTId);
        Log.d("create local tbut","local tbut is created with id:" + localTbut.getId()
}
```

   This class defines a TBUT database with the ID "LocalTBUTId". It uses the same database instance as the FTS database document. It is also called at the first time the user launches the application.

3. Define and store the data attributes for the FTS and the TBUT database.

   The data objects are stored as a key-value data structure in which each key is mapped to a single value. The values, in this case, are nested JSON objects which contain multiple JSON strings. The code below demonstrates the steps of storing tracking information to the FTS. The infoJson stores all the tracking information, including action type, device MAC address, timestamp, and file name. This JSON object is then stored as the value in the timeKey JSON, which takes timestamp as the key. Afterward, the timeKey is put into the value fields of the deviceKey JSON, which takes device MAC address as the key. Finally, the deviceKey JSON is passed to the recoverKey HashMap data structure that uses "Recover First" as the static key. After the data object is stored in the Map data structure, it will then be passed to the local database document by providing the document Id.

```
Map<String, Object> FTS = new HashMap<String, Object>();
    JSONObject deviceKey = new JSONObject();
    JSONObject timeKey = new JSONObject();
    JSONObject infoJson = new JSONObject();
    try{
        infoJson.put("Action", "test");
        infoJson.put("Device MAC Address", "device");
        infoJson.put("Timestamp", "20111111");
        infoJson.put("File Name", "Test File");
        timeKey.put("20111111",testJson);
        deviceKey.put("device",timeTest);
        FTS.put("Recover_First",deviceTest);
         localDatabase.getDocument(localTrackDocumentId).
    putProperties(FTS);
    localDatabase.getDocument(localTBUTId).
    putProperties(FTS);

    }catch (JSONException e){
        e.printStackTrace();
    }
```

The data pushing in TBUT follows the same process as the code snippet above but with a different document ID.

5.1.2 Controller of Local Master

In this component, the controller of the local master contains all the event listeners that will be triggered once the specific actions are performed. There are five managers on the local master that handle all the events.

File Storage Manager

The file storage manager deals with the file download actions. It is triggered once the file download event has been performed.

```
protected String doInBackground(Void... params) {
        FirebaseStorage mFirebaseStorage = FirebaseStorage.getInstance();
        final File localFile = new File(
        Environment.getExternalStorageDirectory().
```

```
getPath()+"/"+ getActivity().
getPackageName()+"/"+fileName);
StorageReference storageRef = mFirebaseStorage.
getReferenceFromUrl(
"gs://test-c4fe2.appspot.com/" + fileName);
storageRef.getFile(localFile).addOnSuccessListener(new OnSuccessListener<FileDownlo
@Override
public void onSuccess(FileDownloadTask.TaskSnapshot taskSnapshot) {
        SimpleDateFormat formatter =
        new SimpleDateFormat("yyyy-MM-dd_HH-mm-ss_SSS");
        String dateString = formatter.
        format(new java.util.Date());
             }
         }).addOnFailureListener(new OnFailureListener() {
             @Override
             public void onFailure(@NonNull Exception e) {
                 Log.d("main", "file downloads failed");
             }
         });
         return localFile.getAbsolutePath();
}
```

Once the file storage manager has been triggered, the code snippet above is activated. A Firebase storage instance is created when the code snippet first runs. The Firebase instance is used to retrieve the database reference on the cloud server. Afterward, a GET request is sent to the Firebase to obtain the desired file by providing the file name. Once the file is targeted successfully, the file is then stored under the application directory.

Connection Manager (CM)

The connection manager deals with the Wi-Fi direct connection while the cloud is not available. The architecture implements the WifiP2pManager to enable the P2P service in each device. After the devices are connected, all the device information, such as MAC address and IP address can be retrieved via the WifiP2pInfo.

File Transmission Manager (FTM)

The file transmission manager is performed when the file is transferred between the client device and the local master. The file will be transferred via the socket stream to the client devices once the action is triggered

Local Master Tracking Information Manager (TIM)

The TIM of the local master ensures all the file flows between the local master and client devices can be recorded precisely. After a file is transferred to a client device, the local master manipulates the codes snippet demonstrated in the third step of section 5.1.1.

Information Synchronization Manager (ISM)

The ISM deals with the tracking information synchronization between the local master and the cloud server. The ISM is triggered once the connection between the local master and cloud is reestablished. There are two cases that are considered:

1. The local master sent files to client devices during the disconnection period.

   In this case, the ISM of local master sends its local TBUT (tentative version) to the cloud server, and the cloud server updates its File Track Storage.

2. The cloud sent files to the client devices during disconnection period.

   Under this scenario, the local master retrieves TBUT data objects from the cloud. The implementations are shown below:

```
if(localDatabase.getDocument(localTrackDocumentId) != null) {
    if(localDatabase.getDocument(localTrackDocumentId).getProperties() != null){
       j = new JSONObject(localDatabase.getDocument(localTrackDocumentId).
       getProperties().toString());
      }
}
myRef.child("TBUT").addListenerForSingleValueEvent(new ValueEventListener (){
     @Override
     public void onDataChange(DataSnapshot dataSnapshot) {
        if (dataSnapshot.exists() && (dataSnapshot.hasChild("Recover_First")
        || dataSnapshot.hasChild("Track_First"))) {
           try {
                String json = new Gson().toJson(dataSnapshot.child("Recover_First").
                getValue());
                JSONObject fromTbutJson = new JSONObject(json);
                JSONObject toLocalDeviceJSON = null;
                Iterator deviceTimeIterator = null;
                JSONObject recoverTempJson = fromTbutJson;
                Iterator recoverIterator = fromTbutJson.keys();
```

```
  while (recoverIterator.hasNext()) {
      String deviceKey = (String) recoverIterator.next();
      toLocalDeviceJSON = recoverTempJson.getJSONObject(deviceKey);
      deviceTimeIterator = toLocalDeviceJSON.keys();
      if(j != null) {
         if (((JSONObject) j.get("Recover_First")).has(deviceKey)) {
            while (deviceTimeIterator.hasNext()) {
                  String Timekey = (String) deviceTimeIterator.next();
                  ((JSONObject)j.get("Recover_First")).
                  getJSONObject(deviceKey).put(Timekey,
                  toLocalDeviceJSON.get(Timekey));
            }
          } else {
            while (deviceTimeIterator.hasNext()) {
                   String Timekey = (String) deviceTimeIterator.next();
                   timeJson.put(Timekey, toLocalDeviceJSON.
                   getJSONObject(Timekey));
            }
           ((JSONObject) j.get("Recover_First")).put(deviceKey, timeJson);
          }
      }else{
         JSONObject tempDeviceJson = new JSONObject();
         JSONObject tempTimeJson = new JSONObject();
         while (deviceTimeIterator.hasNext()) {
             String Timekey = (String) deviceTimeIterator.next();
             tempTimeJson.put(Timekey,toLocalDeviceJSON.get(Timekey));
         }
         tempDeviceJson.put(deviceKey,tempTimeJson);
         JSONObject r = new JSONObject();
         r.put("Recover_First",tempDeviceJson);
         j = new JSONObject(r.toString());
      }
   }
} catch (JSONException e) {
       e.printStackTrace();
}
```

```
Map<String, Object> combine = new HashMap<String, Object>();
combine.put("Recover_First",j.get("Recover_First").toString());
localDatabase.getDocument(localTrackDocumentId).putProperties(combine);
```

The code snippet above first determines if there is any tracking information in the database document of the local master. If the database is not null, then the local database's data objects will be stored in a JSON object. After this has been completed, there are a few steps to transfer the TBUT from the cloud to the local master's tracking database:

(a) The local master reads the data objects from the cloud's TBUT, and stores everything in a JSON object called "fromTbutJson".

(b) The system goes through the received data objects and the local tracking information to check if there is any identical object (device's MAC address in the shown codes) is found. In the case where an identical key is found, the existing data objects will be updated; otherwise, if the local tracking information doesn't contain any of the TBUT's keys, a new data set will then be pushed to the local tracking database.

5.2 Client Device Design

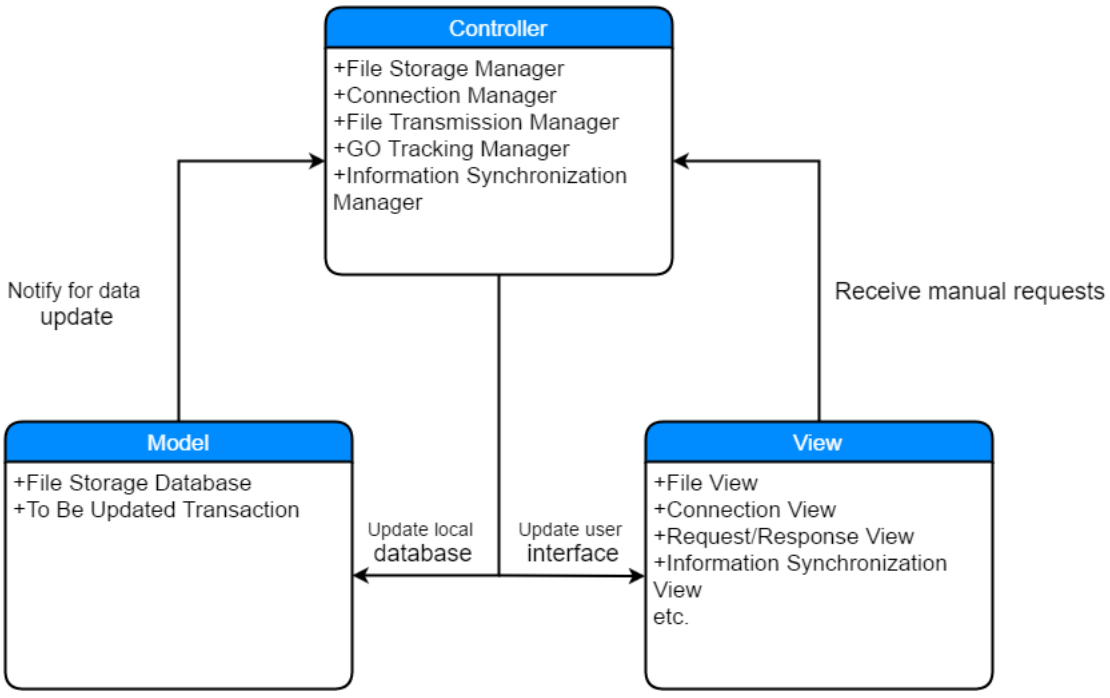

Figure 5.2: Client as GO Model-View-Controller Architecture

As described in the Chapter 4, the architecture of the client device is a sub-component of the local master. The client device contains fewer functions than the local master device.

As shown in figure 5.2, the client device only has two data storage: file storage and TBUT. The file storage and TBUT use the same implementations as the local master. The controller in the client device design shares some common managers with the local master, such as File Storage Manager, Connection Manager, and File Transmission Manager. The only difference between the controller of the local master and the client device is the Information Synchronization Manager. In the client device, the synchronization manager updates its TBUT to the local master or the cloud server. The update is a one way transmission (client to the host). Conversely, the local master's ISM updates the TBUT to the cloud or receives new data objects from the cloud, which is a two way communication (local master to the cloud or cloud to the local master.)

5.3 Cloud Server

The architecture's cloud server is hosted on the Firebase mobile platform because of its high scalability and availability. The Firebase Realtime Database, which is a NoSQL data storage, is used to store all the tracking information as well as the TBUT information. The Firebase Storage, which is backed by the Google Cloud Storage, handles all of the file storage on the server.

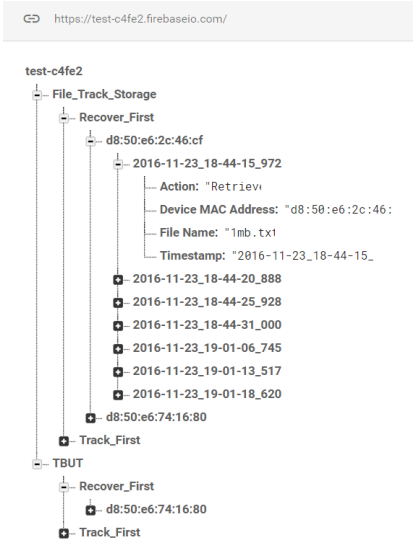

Figure 5.3: JSON objects in the Firebase database

The data on the Firebase Realtime Database, stored as a JSON object, can synchronize to each connected devices in real-time. Figure 5.3 presents the database structure on the cloud server. Both the data storage components, File Track Storage and TBUT, have the same structure as those in the local master. The File Track Storage has two children paths: Recover First and Track First. Each of these paths are used for different purposes and is described in Chapter 4. The Recover First takes the device MAC as the key to store the tracking information. Track First, on the other hand, uses the file name as the key for data objects. In order to implement the database, the system follows the code snippet below:

```
myRef.child("File_Track_Storage").child("Track_First").
    child("File Name").setValue(mFileName);
myRef.child("File_Track_Storage").child("Track_First").
    child("Device MAC Address").setValue(macAddress);
myRef.child("File_Track_Storage").child("Track_First").
    child("Action").setValue("Retrieve");
myRef.child("File_Track_Storage").child("Track_First").
    child("Timestamp").setValue(dateString);
```

The meRef is the Firebase database instance that is acquired by calling the Firebase.getReference() function. The myRef.child)() function can retrieve the reference of the given location if any exists; otherwise the function will create a new path.

As shown in figure 5.4, file uploading on the Firebase is straightforward. The administrator can simply update the file to the storage by clicking the Upload File button. For file downloading, however, the client side needs to provide the URL shown in the figure 5.4 with the additional file name.

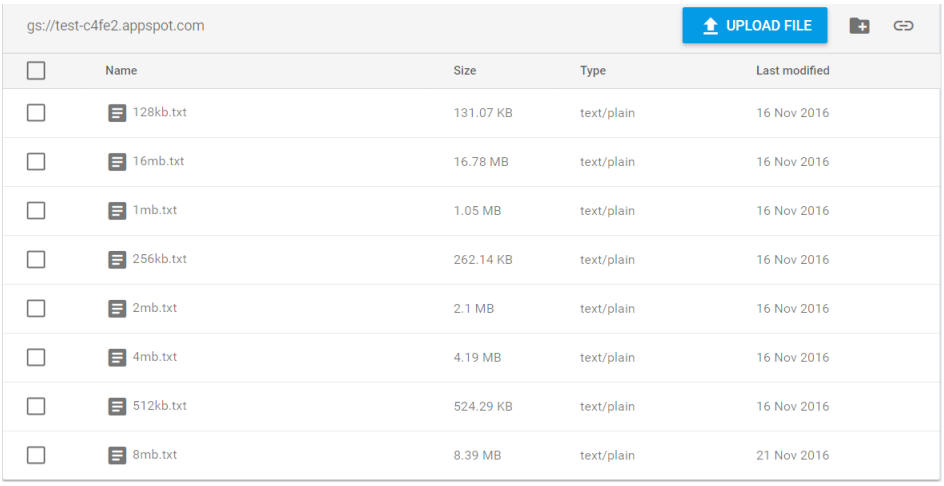

Figure 5.4: Firebase file storage

5.4 Conclusion

In this chapter, the details of architecture implementation have been explained based on different roles of devices in the system: the local master, the client device, and the cloud. The core part of the architecture is the local master. The model of the local master consists of two main parts: TBUT and File Track Storage. Each of these parts stores tracking information for different purposes. Each manager in the controller of the local master handles different action events. The file storage manager deals with all the file objects downloads, the connection manager deals with the Wi-Fi direct connection event, the file transmission manager is triggered when a file is transferred between the host device and client devices, the tracking information manager is used

to ensure the integrity of the tracking information, and the information synchronization manager handles the information synchronization process between the local master and the cloud server. For the cloud server design, the Firebase realtime database and file storage were used. The tracking information is stored in the database, which is constructed as two parts that consists of track first and recover first. The file storage keeps all the file objects available.

Chapter 6

EXPERIMENT

In this chapter, the proposed architecture's functionality and performance was evaluated. In order to address the challenges described in Chapter 2 as well as examining the architecture's feasibility in various circumstances, the following aspects were assessed: file transmission performance, tracking information consistency and integrity, and synchronization mechanism performance.

- The file transmission performance experiment examines the device connection performance in the non-Internet environment.
- The information consistency and integrity experiment verifies the effectiveness of the designed data tracking architecture. By doing this experiment, the data consistency challenge can be verified.
- The data synchronization performance experiment tests the amount of data transferred between multiple mobile devices when a synchronization event was triggered. The result of the experiment was then compared to the basic synchronization strategy. The purpose of this experiment is to examine the network bandwidth efficiency in the architecture.

6.1 Experiment Setup

This research proposes a new architecture that can be embedded in the mobile network whether the Internet is available or not. The Android mobile peer-to-peer (Android Wi-Fi Direct) network was implemented for the non-Internet environment. Conversely, the Firebase Cloud Service was applied as the backend server for the Internet situation. To evaluate different functionality and performance, various devices were needed:

Hardware Components	Specification Details
Platform (OS)	Android 5.1
CPU	Quad-core 1.5 GHz Krait
Storage Capacity	32 GB (2 GB RAM)
Wireless Communication	Wi-Fi 802.11 a/b/g/n, dual-band, Wi-Fi Direct

Table 6.1: Hardware specification of Nexus 7 (local master)

- Nexus 7 (Local Master)
- Nexus 7 (Client device)

Hardware Components	Specification Details
Platform (OS)	Android 5.1
CPU	Quad-core 1.5 GHz Krait
Storage Capacity	16 GB (2 GB RAM)
Wireless Communication	Wi-Fi 802.11 a/b/g/n, dual-band, Wi-Fi Direct

Table 6.2: Hardware specification of Nexus 7 (client device)

- Samsung Galaxy Note II (Client device)

Hardware Components	Specification Details
Platform (OS)	Android 4.4
CPU	Quad-core 1.6 GHz Cortex-A9
Storage Capacity	16 GB (2 GB RAM)
Wireless Communication	Wi-Fi 802.11 a/b/g/n, dual-band, Wi-Fi Direct

Table 6.3: Hardware specification of Samsung Note II (client device)

Each device represents different roles in the network: local master, the client devices, and the cloud server. As described before, the local master acts as host device when the cloud server is not available. The local master contains the same file objects as the cloud server. The client devices retrieve files from either the cloud server or the local master depends on the availability of the Internet.

6.2 File Transmission Performance

As described in chapter 2 and 3, mobile bandwidth is a major challenge because of the high mobility and intermittent connection of mobile devices. Wi-Fi Direct follows the standard Wi-Fi speed which can be up to 250 Mb/s. The file transmission speed in the Android P2P network was examined in this experiment.

Experiment

The bandwidth was reviewed by sending the different size of the files between the client device and the local master. In this experiment, the relationship between the size of the archive and one-way transmission time in the network was analyzed. Figure 6.1 shows the experiment demonstration. To calculate the length of

time, a timer was triggered when the file was sent out and then ended as soon as the client device finished downloading.

Figure 6.1: File Transmission among multiple devices

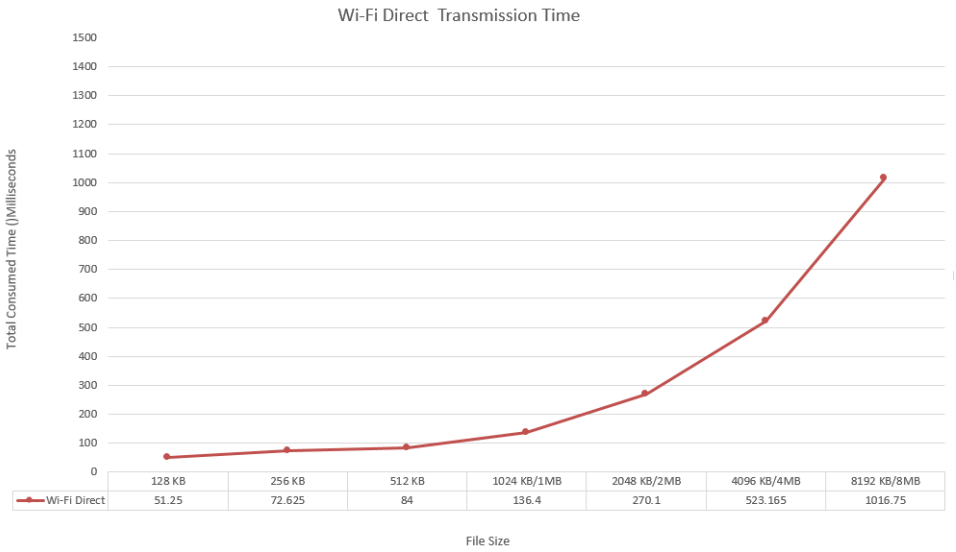

Figure 6.2: File Transmission Time

Result

Figure 6.2 shows a proportional relationship between the file size and transmission time. The file required more time to transfer as the file size increased. However, when the experimental data is compared to the theoretical Wi-Fi direct speed in figure 6.3. The blue bar stands for the transmission time of the proposed architecture; on the other hand, the red bar means the theoretical time. The standard speed is almost eight-times faster than the examined data. After taking several factors into consideration, it was suspected that the file writing time could be the reason that caused this difference. After running a benchmark test on the experiment device, the average writing time was concluded and is shown in table 6.4. Figure 6.4 shows the relationship between the theoretical transmission time and actual transmission time - this was arrived at by subtracting the writing time from the experimental transmission time. The difference in transmission time reduces as the size of file increases. This reduction in transmission time means that as the file size gets larger, the experiment transmission time approaches the theoretical time infinitely. This result means that Wi-Fi

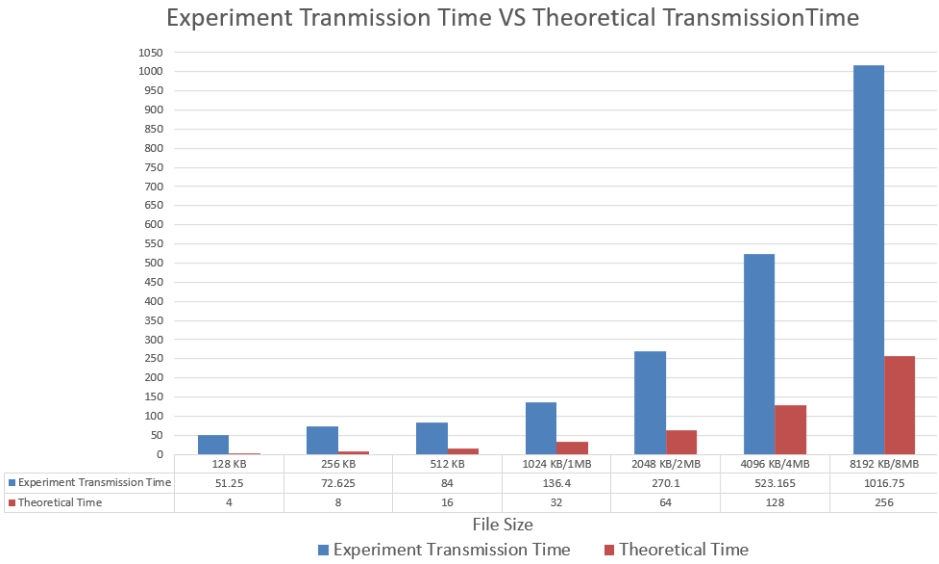

Figure 6.3: Experiment time VS theoretical time

File Size	Writing Time (Milliseconds)	Wi-Fi Standard Time (Milliseconds)
128 KB	6.834	4
256 KB	13.678	8
512 KB	27.35	16
1024 KB	54.70	32
2048 KB	109.41	64
4096 KB	218.82	128
8192 KB	437.64	256

Table 6.4: Device File Writing Time

direct performs faster in the architecture than the Bluetooth, whose theoretical speed is 25 Mb/s.

6.3 Tracking Information Consistency and Integrity

The architecture's primary element is the tracking mechanism. Both the local master and cloud server have the capacity to store tracking information locally. The tracking information can be synchronized instantaneously when these two nodes are connected. However, either host may update its tracking data during the disconnection period since the Internet service can be down for unforeseen reasons. Therefore, the data consistency and integrity need to be verified under this circumstance. In this section, the tracking mechanism was evaluated in several situations:

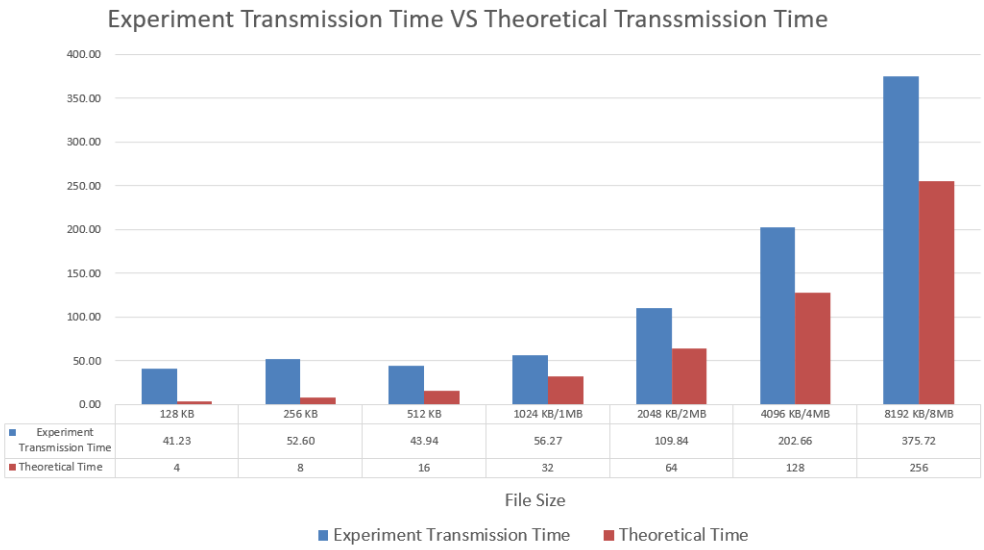

Figure 6.4: File Transmission Time without file writing time

Experiment

- The cloud server is available
  - Case one: two devices retrieve different files from the cloud server. as shown in figure 6.5.

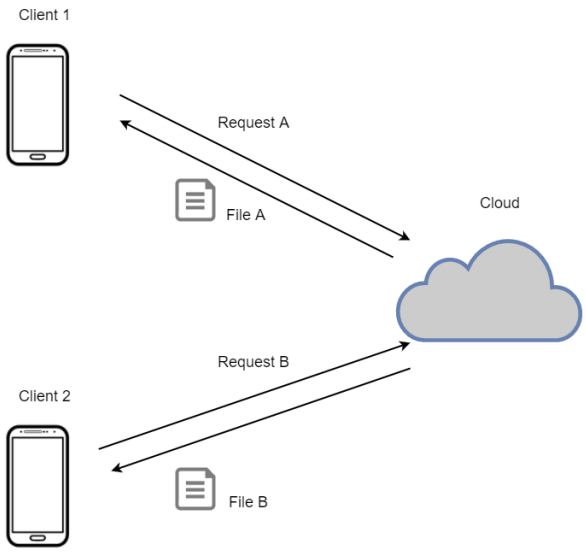

Figure 6.5: Different file retrieval

  - Case two: two devices retrieve the same file from the server, as shown in figure 6.6.

The tracking information on the local master and the cloud was verified after the experiment was carried out for these two cases.

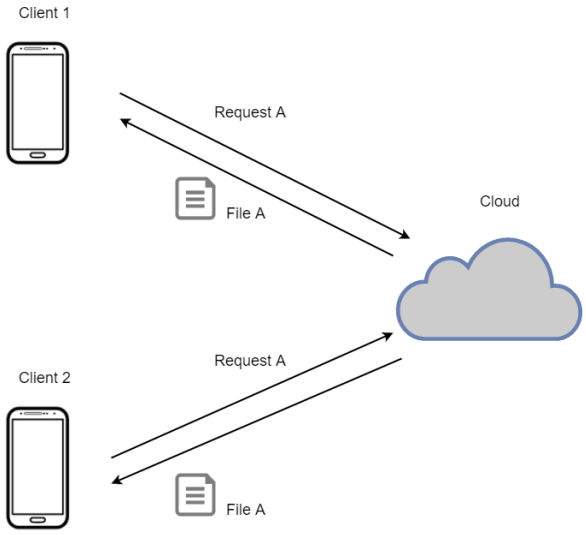

Figure 6.6: Same file retrieval

- The cloud server is unavailable
  - Case one: two client devices connect to the local master, and retrieve different files from it. The procedures is identical to the one shown in figure 6.5.
  - Case two: two client devices connect to the local master, and retrieve the same file from it. The process follows the same steps as shown in figure 6.6.

  The local master connected to the cloud server for data synchronization after each of these two cases were completed. The tracking information for the local master and the cloud server will then be compared.

- Neither the cloud server nor the local master is available.

  One client device can be the group owner of a Wi-Fi Direct network when the local master and cloud are not available. The client device can transfer a file to its slave device, as shown in figure 6.7. There were three steps in this experiment:

  1. The GO device sends several files to its client devices,
  2. The GO reconnects to the local master or the cloud server, whichever one is available, and
  3. Synchronize the tracking information on the local master and the cloud server when they are reconnected.

  The data integrity and consistency of the tracking information on the local master and cloud server were then verified.

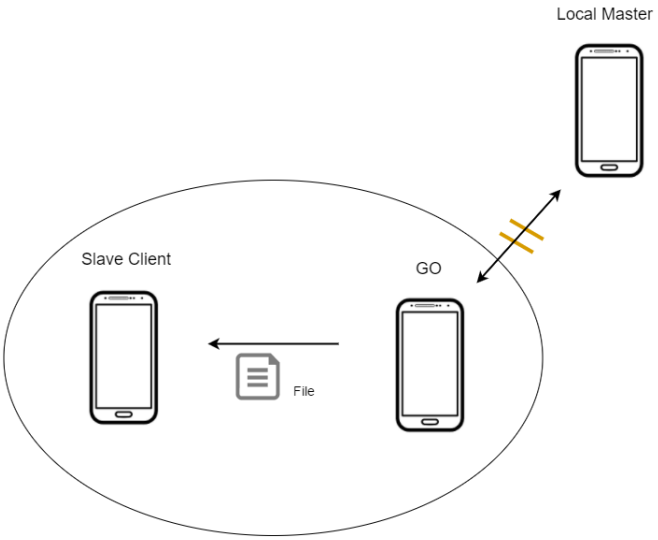

Figure 6.7: Client of local master acts as group owner in a network

Result

- The cloud server is available
  - Case one: Retrieve different files
    - Integrity results

      There were two client devices (Nexus 7 and Samsung Note II) in the experiment. The Samsung Note II with the MAC address: 90:18:7C:20:23:F5 retrieved the file named "256kb.txt" from the cloud, and Nexus 7 with the address of d8:50:e6:74:16:80 requested the file called "128kb.txt". After inspecting the cloud's tracking data objects, as shown on the left-hand side of figure 6.8, we concluded that the implemented architecture ensured tracking data integrity in this case.

Figure 6.8: File Tracking Storage comparison while retrieving different files

* Consistency results

  The data objects, in figure 6.8, were compared to examine the consistency of tracking information. The left-hand side shows the cloud server's database, while the right-hand side demonstrates the JSON object of the local master's local tracking information. The JSON object was generated based on the To Be Updated Transaction of the cloud after the synchronization event was triggered, and the red contexts represents the keys of each nested data. The architecture ensured the data consistency in this scenario by matching the data objects on the two sides.

– Case two: Retrieve same files

Figure 6.9: File Tracking Storage comparison while retrieving same file

* Integrity results

  The two client devices sent requests to the cloud to retrieve the same file by providing the file name from the user interface. The left hand side of figure 6.9 presents the data objects on the cloud server. The data integrity is guaranteed by manually verifying the data contents.

* Consistency results

  The data consistency experiment was identical to the case one experiment. The data values of each key were duplicated by comparing the data objects on the cloud (left-hand side) and local master (right-hand side), in figure 6.9. Therefore, the data consistency was ensured in this case.

- The cloud server is not available

  As described in the experiment section, the processes to examine the data integrity and consistency were identical whether the cloud was available or not. The only dissimilarity was the host device. In this section, the local master acted as the master device which sent files to clients. The experiment results were analyzed as following:

  – Case one: Retrieve different files from the local master

* Integrity results

  Two client devices retrieved different files, named: "1mb.txt" and "2mb.txt", from the local master. The tracking information was stored as a JSON object, figure 6.10. The JSON object indicated that the two file transactions had been recorded accurately in the File Track Storage of the local master with the desired formats and values.

Figure 6.10: File Tracking Storage comparison while retrieving different files from local master

* Consistency results

  In the consistency test, the local master updated the file transactions to the cloud's File Track Storage. The latest data objects on the server side were presented as the left-hand portion of the figure 6.10. The information consistency was verified by comparing the data objects of the cloud and local master.

– Case two: Retrieve same files from the local master

* Integrity results

Figure 6.11: File Tracking Storage comparison while retrieving the same from local master

As before, the integrity test was verified by inspecting the File Track Storage of the host devices. The right-hand side of the figure 6.11 presents the JSON object of the local master's tracking data. All of the information had been stored correctly.

* Consistency results

  To examine the data consistency, the local master updated its tracking information to the cloud as before. After reviewing the File Track Storage of both host devices, figure 6.11, the data objects were considered consistent.

- Neither the cloud server nor the local master is available

  In this experiment, a client device from the previous two tests acted as the group owner in a Wi-Fi Direct network. The client device could send files to its slave device in the group. Two situations were simulated after the file transactions: the GO reconnected to the local master first, or the cloud server first. The synchronization procedures are slightly different for these two situations. If the local master was reconnected first, it generated the TBUT, which could then be sent to the cloud server. On the other hand, if the cloud was available first, the local master would synchronize its File Track Storage based on the server's TBUT. The details are shown below:

  – Reconnected to the local master first

    1. Local master sends a file named "256kb.txt" to a client device. The data objects in the local master's File Track Storage is shown in the figure 6.12a.

{Recover_First=
{"da:50:e6:74:16:80":
{"2016-11-22_20-30-16_033":
{"Action":"Retrieve","Device MAC Address":"da:50:e6:74:16:80",
"File Name":"256kb_txt","Timestamp":"2016-11-22_20-30-16_033"}}},

Track_First=
{"256kb_txt":
{"2016-11-22_20-30-16_033":
{"Action":"Retrieve","Device MAC Address":"da:50:e6:74:16:80",
"File Name":"256kb_txt","Timestamp":"2016-11-22_20-30-16_033"}}}}

(a) File Track Storage for the local master

{Recover_First=
{"92:18:7c:20:23:f5":
{"2016-11-22_20-32-12_073":
{"Action":"Retrieve","Device MAC Address":"92:18:7c:20:23:f5",
"File Name":"256kb.txt","Timestamp":"2016-11-22_20-32-12_073"},
"2016-11-22_20-32-48_169":
{"Action":"Retrieve","Device MAC Address":"92:18:7c:20:23:f5",
"File Name":"256kb.txt","Timestamp":"2016-11-22_20-32-48_169"}}},

Track_First=
{"256kb_txt":
{"2016-11-22_20-32-12_073":
{"Action":"Retrieve","Device MAC Address":"92:18:7c:20:23:f5",
"File Name":"256kb_txt","Timestamp":"2016-11-22_20-32-12_073"},
"2016-11-22_20-32-48_169":
{"Action":"Retrieve","Device MAC Address":"92:18:7c:20:23:f5",
"File Name":"256kb_txt","Timestamp":"2016-11-22_20-32-48_169"}}}}

(b) TBUT for a client device

Figure 6.12: File Transactions among multiple client devices

    2. The client device disconnected with the local master and acted as the group owner of another Wi-Fi Direct network. The client device sent the file received from the local master to its slave device. Afterward, the file flow was recorded in its temporary TBUT, as shown in figure 6.12b.

3. The simulations of reconnecting the local master and the client devices were performed. The client device updated its TBUT to the local master once the re-connection was finished. This action generated some data objects to the TBUT and File Track Storage of the local master. The demonstration is presented in figure 6.13. In this case, the data objects of this two storage are the same because the local master did not synchronize its tracking information with the cloud after the step 1.

{Recover_First=
{"da:50:e6:74:16:80":
{"2016-11-22_20-30-16_033":{"Action":"Retrieve","Device MAC Address":"da:50:e6:74:16:80","File Name":"256kb.txt","Timestamp":"2016-11-22_20-30-16_033"}},
"92:18:7c:20:23:f5":
{"2016-11-22_20-32-12_073":{"Action":"Retrieve","Device MAC Address":"92:18:7c:20:23:f5","File Name":"256kb.txt","Timestamp":"2016-11-22_20-32-12_073"},
"2016-11-22_20-32-48_169":{"Action":"Retrieve","Device MAC Address":"92:18:7c:20:23:f5","File Name":"256kb.txt","Timestamp":"2016-11-22_20-32-48_169"}}},

Track_First=
{"256kb_txt":
{"2016-11-22_20-30-16_033":{"Action":"Retrieve","Device MAC Address":"da:50:e6:74:16:80","File Name":"256kb_txt","Timestamp":"2016-11-22_20-30-16_033"},
"2016-11-22_20-32-12_073":{"Action":"Retrieve","Device MAC Address":"92:18:7c:20:23:f5","File Name":"256kb_txt","Timestamp":"2016-11-22_20-32-12_073"},
"2016-11-22_20-32-48_169":{"Action":"Retrieve","Device MAC Address":"92:18:7c:20:23:f5","File Name":"256kb_txt","Timestamp":"2016-11-22_20-32-48_169"}}}}

(a) File Track Storage of the local master

{Recover_First=
{"da:50:e6:74:16:80":
{"2016-11-22_20-30-16_033":{"Action":"Retrieve","Device MAC Address":"da:50:e6:74:16:80","File Name":"256kb.txt","Timestamp":"2016-11-22_20-30-16_033"}},
"92:18:7c:20:23:f5":
{"2016-11-22_20-32-12_073":{"Action":"Retrieve","Device MAC Address":"92:18:7c:20:23:f5","File Name":"256kb.txt","Timestamp":"2016-11-22_20-32-12_073"},
"2016-11-22_20-32-48_169":{"Action":"Retrieve","Device MAC Address":"92:18:7c:20:23:f5","File Name":"256kb.txt","Timestamp":"2016-11-22_20-32-48_169"}}},

Track_First=
{"256kb_txt":
{"2016-11-22_20-30-16_033":{"Action":"Retrieve","Device MAC Address":"da:50:e6:74:16:80","File Name":"256kb_txt","Timestamp":"2016-11-22_20-30-16_033"},
"2016-11-22_20-32-12_073":{"Action":"Retrieve","Device MAC Address":"92:18:7c:20:23:f5","File Name":"256kb_txt","Timestamp":"2016-11-22_20-32-12_073"},
"2016-11-22_20-32-48_169":{"Action":"Retrieve","Device MAC Address":"92:18:7c:20:23:f5","File Name":"256kb_txt","Timestamp":"2016-11-22_20-32-48_169"}}}}

(b) TBUT of the local master

Figure 6.13: File Transactions among multiple client devices

4. After running the synchronization procedure, the tracking information on the cloud and local master were verified by analyzing the figure 6.13a and 6.14.

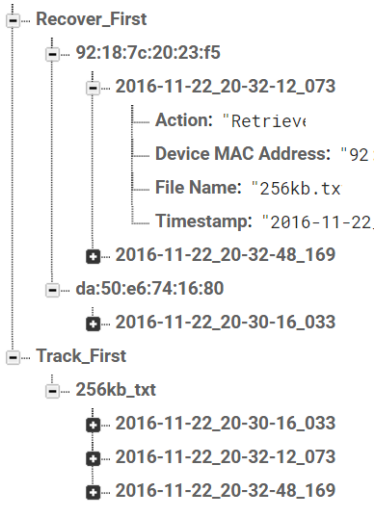

Figure 6.14: Cloud's database after synchronization

– Reconnect to the cloud server first

In this case, steps of file transaction are the same as previous situation. The only difference is the synchronization procedure. In the last scenario, the updates were sent from the local master to cloud. Nevertheless, in this circumstance, the cloud server needs to send its TBUT (figure 6.15) to the local master. It was generated from the TBUT (figure 6.16) of client device which was the GO of a Wi-Fi direct network. To check if the local master updates all tracking information accurately, the data objects in the local master's File Track Storage were compared (figure 6.17).

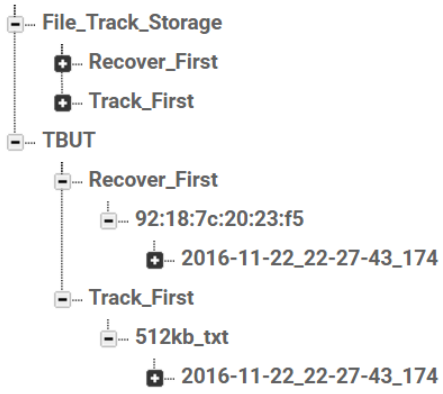

Figure 6.15: Cloud's TBUT from a client device

{Recover_First=
{"92:18:7c:20:23:f5":
{"2016-11-22_22-27-43_174":
{"Action":"Retrieve","Device MAC Address":"92:18:7c:20:23:f5",
"File Name":"512kb.txt","Timestamp":"2016-11-22_22-27-43_174"}}},

Track_First=
{"512kb_txt":
{"2016-11-22_22-27-43_174":
{"Action":"Retrieve","Device MAC Address":"92:18:7c:20:23:f5",
"File Name":"512kb_txt","Timestamp":"2016-11-22_22-27-43_174"}}}}

Figure 6.16: Clients's local TBUT

To sum up, the experiments that have been done in this section examined the proposed architecture from various circumstances. Most of the connection situations have been taken into consideration. For each of the experiments, the architecture was found to have ensured the integrity and consistency of tracking information in the Internet and non-Internet environment.

(a) FTS on local master before sync

(b) FTS on local master after sync

Figure 6.17: File Track Storage of local master

6.4 Data Synchronization Performance

The consistency of tracking information was the core part of this research. Additionally, the performance of the synchronization procedure was critical because it could affect the bandwidth performance of mobile devices.

Experiment

Normally, the performance of the architecture was determined by the size of the tracking data that needs to be updated. Therefore, the following experiments were repeated with the different number of file requests: 10 requests, 20 requests, 40 requests, and 80 requests.

1. The local master sends ten files to the client devices while it is disconnected from the cloud.

2. Reconnect the local master and the cloud server, and synchronize the tracking information.

3. Disconnect the local master and cloud again, then sends twenty files from the local master to the client devices.

4. Repeats the previous steps until the number of file requests reaches to 80.

Meanwhile, the same experiment was applied to a basic synchronization strategy. This synchronization algorithm deleted all of the local tracking information first and then retrieves every piece of tracking information from the connected node. The sizes of the updated data objects in both strategies were recorded and compared. The details are presented in the results.

Results

In the experiments, the performance was examined by calculating the size of transferred data objects. In figure 6.18, the blue bar represents the size of total downloaded data of the basic synchronization strategy, the red stands for the examined architecture, and the green bar indicates the difference of downloaded data size between these two strategies. As the number of file requests increased during the disconnection time, the size of downloaded data of the basic strategy rose dramatically. However, the proposed architecture's synchronization mechanism takes up half of the bandwidth, meaning that the more files transferred, the more bandwidth the architecture saves. This space saving can lead to the reduction of synchronizing time. Thus, the proposed synchronization mechanism can provide a more efficient procedure for data updates.

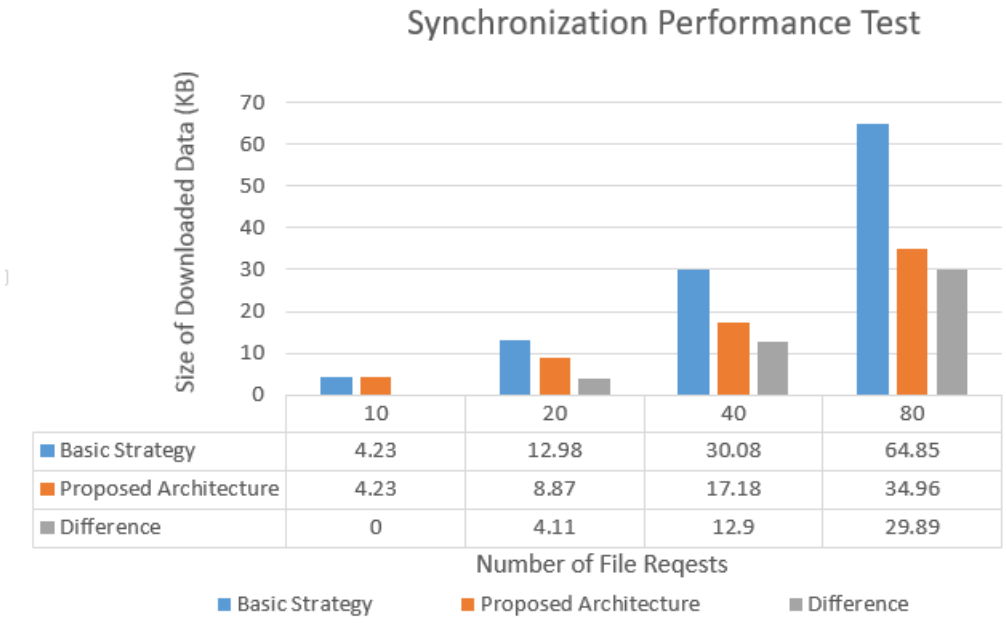

Figure 6.18: Synchronization performance comparison

6.5 Conclusion

To sum up, by examining our proposed architecture based on the above experiments, we verified that the architecture resolved all the challenges described in Chapter 2. The file transmission performance experiment tested the transmission performance of the proposed architecture. Even though the data I used was not as fast as the theoretical Wi-Fi direct speed, I found that as the size of the file transferred was increased, the transmission rate of proposed architecture also increased. Moreover, the proposed architecture's rate was found to be infinitely close to the standard speed if the file was large enough. By verifying the data integrity and consistency, the tracking information on local master and cloud server was considered identical. Several

different situations have been exposed in the experiments. In each of these cases, the tracking information was fully recorded and synchronized without exceptions. In the last experiment, I examined the performance of synchronization mechanism in the architecture. The synchronization mechanism provided a more efficient procedure that can reduce the redundant of data updates.

Chapter 7

SUMMARY AND CONTRIBUTION

7.1 Summary

With the dramatic rise of mobile computing development, personal smart devices have become an essential part of people's daily lives. This integration with daily life has led to millions of mobile applications developed. A large amount of data, such as contact list, photos, and emails just to name a few, are generated through the use of these applications. Furthermore, people are able to send these data to any devices at anytime via the Internet. Under this circumstance, data management is utterly essential.

This research focused on solving file tracking problem in mobile networks. The client side of the architecture was based on the Android platform, and the file storage and database storage were hosted on the Firebase cloud platform. Generally, the application will work regardless of the availability of the Internet. When the Internet connection was available, the mobile device could then send requests to the cloud server directly to retrieve files. The cloud would then store the tracking information in its database. However, when the cloud server was not available, a mobile device acted as the local master to handle clients' requests. Each client device was able to connect to the local master via the Wi-Fi direct technology. All of the file tracking information in this situation was stored on the local master, and then was synchronized with the cloud server via the designed synchronization mechanism. When neither the cloud nor the local master was available, all the file transactions were stored on the group owner of the present Wi-Fi direct network. These transactions were then synchronized with either the local master or the cloud. Therefore, the tracking information's consistency was ensured from different situations.

Moreover, to overcome the mobile device limitations mentioned in Chapter 2, the designed architecture contained following main components:

- The Wi-Fi Peer-to-Peer technology was applied in the architecture in order to ensure the device connection in the non-Internet network. The Wi-Fi P2P technology has standard Wi-Fi transfer speed, with a maximum speed up to 250 Mb/s.

- A synchronization mechanism was embedded into the architecture in order to overcome the tracking information consistency problem in the intermittent connectivity environment. Different synchronization procedures were applied based on the network situations.

- The RESTful protocol and the timestamp transfer pattern were implemented to solve the bandwidth constraint of the mobile device. RESTful is a lightweight web service protocol. It can pass light message formats such as JSON and plain text. Additionally, the timestamp transfer pattern only transfers data changed since the last synchronization among the mobile devices.

There were three factors that were analyzed in the experiments: file transmission performance, tracking consistency and integrity, and synchronization mechanism performance. Through the examination of these three factors, the proposed architecture was verified from three essential aspects: transmission rate, data consistency and data integrity, and synchronization bandwidth.

- Transmission rate

  Different files were sent from the local master to client device. The relationship between the transmission time and file size were compared. The experimental time approached to the theoretical data infinitely as the size of file getting larger.

- Data consistency and integrity

  There are several situations that were considered in the experiments in order to verify the effectiveness of the architecture:

  - The cloud server was available
    1. Two devices retrieved the different files.
    2. Two devices retrieved the same file.
  - The cloud server was not available
    1. Two devices retrieved the different files from the local master.
    2. Two devices retrieved the same file from the local master.
  - Neither the cloud server nor the local master was available

    In this case, one client device acted as group owner of a Wi-Fi direct network. The client device sent its retrieved files to the other device. Afterwards, the client device needed to reconnect to the host device for information synchronization.
    1. The group owner reconnected to the local master first.
    2. The group owner reconnected to the cloud server first.

  The data consistency and integrity were guaranteed after performing each of the experiments.

- Synchronization mechanism performance

  Multiple files were sent from the host device to the client devices to verify the performance of the architecture. After comparison it with the basic synchronization mechanism, the architecture only transferred the most updated data.

7.2 Contribution

Though there are already some data management systems available, most of them are only supported by desktop computers. For those management systems that are compatible with mobile devices, the users have to connect their devices to the Internet in order to share data with other personal devices. The performance of this kind of architecture can be crucial when the Internet connection is unavailable. Therefore, in this research, a new architecture was designed and implemented. The architecture supports data transaction and tracking in both the Internet and non-Internet environment. By enabling the file transaction in the non-Internet environment, the user would be able to share files without being constrained to the availability of the Internet. Moreover, by providing the functionality of the file tracking, the chance of file loss can be reduced; also, the file transactions that have been made in the network can be retrieved so that the administrator can improve the management efficiency. For instance, by applying the file tracking architecture to the medical example mentioned in Chapter 2, if the doctor has the proposed architecture implemented on his tablet, he can simply logins the hospital's cloud server to retrieve the file transactions that were made on his tablet. By doing so, the doctor wouldn't waste time to look for the tablet and all of his efforts can be retained.

Chapter 8

FUTURE WORK

The proposed architecture achieves file tracking in both the Internet and non-Internet environment. However, the architecture can be improved in four ways:

- Multiple file formats

  Currently, the only file format supported by the architecture is the image file, which can be downloaded and browsed in the application. Support for the pdf and other common file formats have not yet been implemented. By achieving this feature, the architecture will compatible with more document types.

- Data tracking in the IoT application

  The IoT, which stands for the Internet of Things, aims to make physical devices (i.e. sensors) available in the virtual world so that they can be connected by the Internet. These sensors continuously receive and generate huge amounts of data for analysis, which can cause an increase in the number of data errors. In this scenario, the proposed architecture can be modified and integrated into IoT applications to track data flows. By doing so, the administrator will be able to figure out the source of the data problem as soon as a warning appears.

- Cross-platform

  In the architecture, the mobile device only supports the Android platform. As mentioned in Chapter 3, there are many other operating systems. In the future, a cross-platform application should be designed and implemented.

- Security

  The user has to acquire the authorization from the Firebase cloud server in order to establish connection to the database and file storage of the cloud in the present architecture. However, the local database and file objects of the mobile devices have not been encrypted. The next step of the architecture's development is to construct an encryption algorithm to ensure the security of the local files.

References

[1] Androutsellis-Theotokis, S. and Spinellis, D. (2004). A survey of peer-to-peer content distribution technologies. *ACM computing surveys (CSUR)*, 36(4):335–371.

[2] Baatz, S., Frank, M., Kuhl, C., Martini, P., and Scholz, C. (2001). Adaptive scatternet support for bluetooth using sniff mode. In *Local Computer Networks, 2001. Proceedings. LCN 2001. 26th Annual IEEE Conference on*, pages 112–120. IEEE.

[3] BluetoothSIG. Bluetooth low energy. https://www.bluetooth.com/what-is-bluetooth-technology/bluetooth-technology-basics/low-energy. Accessed May, 2016.

[4] BluetoothSIG (2010). Specification of the bluetooth system 4.0. https://www.bluetooth.org/docman/handlers/downloaddoc.ashx?doc_id=229737. Accessed May, 2016.

[5] BluetoothSIG (2014). Specification of the bluetooth system 4.2. https://www.bluetooth.org/DocMan/handlers/DownloadDoc.ashx?doc_id=286439&_ga=1.184284000.737790313.1467059345. Accessed May, 2016.

[6] Bonaventure, O. (2011). *Computer Networking: Principles, Protocols, and Practice*. The Saylor Foundation.

[7] Buckle, C. (2016). Digital consumers own 3.64 connected devices. http://www.globalwebindex.net/blog/digital-consumers-own-3.64-connected-devices. Accessed June, 2016.

[8] Buyya, R., Yeo, C. S., Venugopal, S., Broberg, J., and Brandic, I. (2009). Cloud computing and emerging it platforms: Vision, hype, and reality for delivering computing as the 5th utility. *Future Generation computer systems*, 25(6):599–616.

[9] Camps-Mur, D., Garcia-Saavedra, A., and Serrano, P. (2013). Device-to-device communications with wi-fi direct: overview and experimentation. *IEEE wireless communications*, 20(3):96–104.

[10] Casetti, C., Chiasserini, C. F., Pelle, L. C., Del Valle, C., Duan, Y., and Giaccone, P. (2015a). Content-centric routing in wi-fi direct multi-group networks. In *World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2015 IEEE 16th International Symposium on a*, pages 1–9. IEEE.

[11] Casetti, C., Chiasserini, C. F., Pelle, L. C., Del Valle, C., Duan, Y., and Giaccone, P. (2015b). A demonstration for content delivery on wi-fi direct enabled devices. In *World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2015 IEEE 16th International Symposium on a*, pages 1–2. IEEE.

[12] Cattell, R. (2011). Scalable sql and nosql data stores. *Acm Sigmod Record*, 39(4):12–27.

[13] Cisco. Internet of things (iot). http://www.cisco.com/c/en/us/solutions/internet-of-things/overview.html? Accessed May, 2016.

[14] Cisco. Simple mail transfer protocol. http://www.cisco.com/c/en/us/td/docs/ios/sw_upgrades/interlink/r2_0/user/ugsmtp.pdf. Accessed May, 2016.

[15] Codd, E. F. (1971). Normalized data base structure: A brief tutorial. In *Proceedings of the 1971 ACM SIGFIDET (now SIGMOD) Workshop on Data Description, Access and Control*, pages 1–17. ACM.

[16] Crockford, D. (2006). Json: The fat-free alternative to xml. http://www.json.org/fatfree.html. Accessed May, 2016.

[17] Date, C. J. (2006). *An introduction to database systems*. Pearson Education India.

[18] Deepak, G. and Pradeep, B. (2012). Challenging issues and limitations of mobile computing. *Int. J. Computer Technology & Applications*, 3(1):179.

[19] Elkstein, M. Learn rest: A tutorial. http://rest.elkstein.org/. Accessed May, 2016.

[20] Fielding, R. T. (2000). *Architectural styles and the design of network-based software architectures*. PhD thesis, University of California, Irvine.

[21] Flanders, J. (2009). Service station-more on rest. https://msdn.microsoft.com/en-us/magazine/dd942839.aspx. Accessed May, 2016.

[22] Fu, W. (2014). Data synchronization in a network-volatile mobile ecosystem.

[23] Ghosh, D. (2009). Peer to peer network for mobile communication. *Int. J. of Recent Trends in Engineering and Technology*, 1(2).

[24] Gilbert, S. and Lynch, N. (2002). Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services. *ACM SIGACT News*, 33(2):51–59.

[25] Gray, J., Helland, P., O'Neil, P., and Shasha, D. (1996). The dangers of replication and a solution. *ACM SIGMOD Record*, 25(2):173–182.

[26] Hadjigeorgiou, C. et al. (2013). Rdbms vs nosql: Performance and scaling comparison.

[27] Hathaway, M. E. and E.Savage, J. (2012). Duties for internet service providers. http://belfercenter.ksg.harvard.edu/files/cyberdialogue2012_hathaway-savage.pdf.

[28] Huang, A. and Rudolph, L. (2005). Bluetooth for programmers. *Massachusetts Institute of Technology, Cambridge*.

[29] Hughes-Systique. Wi-fi direct white paper. http://hsc.com/Portals/0/Uploads/Articles/WFD_Technology_Whitepaper_v_1.7635035318321315728.pdf. Accessed May, 2016.

[30] IBM (2015). Why nosql? https://cloudant.com/wp-content/uploads/Why_NoSQL_IBM_Cloudant.pdf. Accessed May, 2016.

[31] IBM-Software. Native, web or hybrid mobile-app development. ftp://public.dhe.ibm.com/software/pdf/mobile-enterprise/WSW14182USEN.pdf. Accessed May, 2016.

[32] Imam, A. A., Basri, S., and Ahmad, R. (2015). Data synchronization between mobile devices and server-side databases: A systematic literature review. *Journal of Theoretical and Applied Information Technology*, 81(2):364.

[33] Javvin-Technologies (2005). *Network Protocols Handbook*. Javvin Technologies Inc. Accessed May, 2016.

[34] Lionbridge. Mobile web apps vs. mobile native apps:how to make the right choice. WWW.LIONBRIDGE.COM/FILES/2012/.../LIONBRIDGE-WP_MOBILEAPPS2.PDF. Accessed May, 2016.

[35] Maly, R. J., Mischke, J., Kurtansky, P., and Stiller, B. (2003). Comparison of centralized (client-server) and decentralized (peer-to-peer) networking. *Semester thesis, ETH Zurich, Zurich, Switzerland*, pages 1–12.

[36] McCormick, Z. and Schmidt, D. C. (2012). Data synchronization patterns in mobile application design. In *Proceedings of the 19th Conference on Pattern Languages of Programs*, page 12. The Hillside Group.

[37] Mell, P. and Grance, T. (2011). The nist definition of cloud computing.

[38] Mughees, M. (2014). Data migration from standard sql to nosql.

[39] Oracle (2011). Oracle nosql database. http://www.oracle.com/technetwork/database/nosqldb/learnmore/nosql-database-498041.pdf. Accessed May, 2016.

[40] Pautasso, C., Zimmermann, O., and Leymann, F. (2008). Restful web services vs. big'web services: making the right architectural decision. In *Proceedings of the 17th international conference on World Wide Web*, pages 805–814. ACM.

[41] Pitoura, E. and Chrysanthis, P. K. (2007). Caching and replication in mobile data management. *IEEE Data Eng. Bull.*, 30(3):13–20.

[42] Qi, H. and Gani, A. (2012). Research on mobile cloud computing: Review, trend and perspectives. In *Digital Information and Communication Technology and it's Applications (DICTAP), 2012 Second International Conference on*, pages 195–202. ieee.

[43] Rajamohan, P., Thinaharan, M. R., et al. (2014). Performance analysis and comparison of wireless protocols standards in wpan-bluetooth and wlan-wi-fi. *International Journal of Scientific Engineering and Technology*, 3(12).

[44] Service-Architecture. Representational state transfer (rest). http://www.service-architecture.com/articles/web-services/representational_state_transfer_rest.html. Accessed May, 2016.

[45] Service-Architecture. Soap. http://www.service-architecture.com/articles/web-services/soap.html. Accessed May, 2016.

[46] Sharma, A. and Kansal, V. (2011). Replication management and optimistic replication challenges in mobile environment. *International Journal of Database Management Systems*, 3(4):81.

[47] Sharma, V. and Dave, M. (2012). Sql and nosql databases. *International Journal of Advanced Research in Computer Science and Software Engineering*, 2(8).

[48] Stage, A. (2005). Synchronization and replication in the context of mobile applications. *May*, 30:1–16.

[49] Suda, B. (2003). Soap web services. *Retrieved June*, 29:2010.

[50] Sundar, S., Kumar, M. K., Selvinpremkumar, P., and Chinnadurai, M. (2012). Voice over ip via bluetooth/wi-fi peer to peer. In *Advances in Engineering, Science and Management (ICAESM), 2012 International Conference on*, pages 828–837. IEEE.

[51] Tan, G., Miu, A., Guttag, J., Balakrishnan, H., et al. (2001). Forming scatternets from bluetooth personal area networks. *Massachusetts Institute of Techonology, http://lcs. mit. edu/, Tech. Rep. MIT-LCS-TR-826*.

[52] Tutorialspoint. Data communication and computer network. http://www.tutorialspoint.com/data_communication_computer_network/data_communication_computer_network_tutorial.pdf. Accessed May, 2016.

[53] Tutorialspoint. Mobile computing - brief overview. http://www.tutorialspoint.com/mobile_computing/mobile_computing_overview.htm. Accessed May, 2016.

[54] Tutorialspoint. Web service description language. http://www.tutorialspoint.com/wsdl/wsdl_tutorial.pdf. Accessed May, 2016.

[55] Tutorialspoint. What are web services? http://www.tutorialspoint.com/webservices/what_are_web_services.htm. Accessed May, 2016.

[56] U.S.Robotics. Wireless lan networking white paper. http://support.usr.com/download/whitepapers/wireless-wp.pdf. Accessed May, 2016.

[57] Van Renesse, R. and Tanenbaum, A. S. (1988). Voting with ghosts. In *Distributed Computing Systems, 1988., 8th International Conference on*, pages 456–462. IEEE.

[58] Voorsluys, W., Broberg, J., and Buyya, R. (2011). Introduction to cloud computing. *Cloud computing: Principles and paradigms*, pages 1–44.

[59] W3C-Working-Group (2004). Web services architecture. https://www.w3.org/TR/ws-arch/#id2260892. Accessed April, 2016.

[60] Wi-Fi-Alliance. Wi-fi direct. http://www.wi-fi.org/discover-wi-fi/wi-fi-direct. Accessed May, 2016.

[61] Wi-Fi-Alliance. Wi-fi protected setup specification. http://v1ron.ru/downloads/docs/Wi-Fi%20Protected%20Setup%20Specification%201.0h.pdf. Accessed May, 2016.

[62] Wi-Fi-Alliance (2010). Wi-fi peer-to-peer (p2p) technical specification. *www. wi-fi. org/Wi-Fi_ Direct. php*. Accessed May, 2016.

[63] Zahariev, Z. and Damyanov, G. (2015). Data synchronization on mobile device. https://www.ucviden.dk/student-portal/files/31917172/Documentation_Final.pdf.