國立臺灣大學電機資訊學院資訊網路多媒體研究所

碩士論文

Graduate Institute of Networking and Multimedia
College of Electrical Engineering and Computer Science
National Taiwan University

Master Thesis

NFC 智慧多媒體分享平台

Intelligent Media Content Sharing System via NFC

劉昱承

Yu-Cheng Liou

指導教授：陳彥仰 博士

Advisor: Mike Y. Chen, Ph.D.

中華民國 100 年 7 月

July, 2011

誌謝

呼！要感謝的真的太多人，碩班兩年的時光過的真是快阿！首先感謝陳彥仰老師,碩一升碩二暑假的時候能推薦我去 IBM 實習,往年暑假都是在娛樂中度過，那年接觸到公司後讓自己盲目的人生有了方向，也特別感謝老師給我十分大的自由，能自由的選擇自己想做的事情。

接下來感謝 HTC 的爽大和西門哥，感謝你們教導我 Android Framework 層以下的東西；也感謝 IBM 的馬克先生，讓我能淺嘗做研究的快樂。

還有實驗室的夥伴們，雖然平常比較少待在實驗室，但偶爾的實驗室嘴砲時間都能聽到一些很有趣的八卦,畢業之後,我一定會常常回來的！也特別感謝 NFC 的 Leader 公里哥，在我走投無路的時候讓我加入了 NFC 這個 Project，今天也才有這篇論文的產生，也祝你未來職場能很順利！

感謝 SZT 的一群損友們，碩一的時候每天沒日沒夜的做線上遊戲的研究，不過大家也都能順利畢業也是很不容易！畢業後大家可能相聚的時間也會變少很多。特別感謝摸摸和阿哈，你們讓我對畢業充滿信心 XD

最後感謝我的家人，和一直很支持我的亞兒～在我趕論文很暴躁的時候都能體諒我，不跟我計較，也都能時時刻刻的陪著我，嘻嘻等你要寫論文的時候我也來支持你一下！

還有好多好多好多，希望有一天我也有機會能幫忙到你們！謝謝大家☺

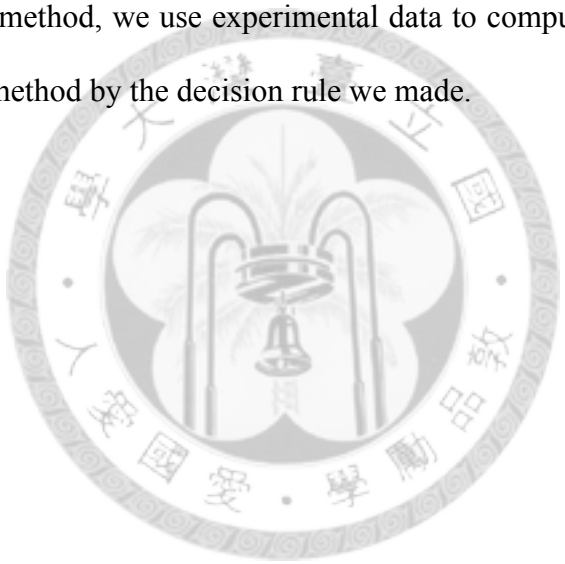
中文摘要

現在行動裝置如平板電腦、智慧型手機有很多應用程式能分享彼此的照片或影片，但使用者常常不知道現在的網路狀況該選擇哪一個應用程式才能傳或是傳的快。我們提出一個智慧型網路分享平台，考量使用者易用性，透過 NFC 來交換雙方網路設定，再利用我們事先建立好的傳輸時間預測模型預測各種連線的傳輸時間再加上各種連線的設定時間，選擇最好的連線方式。選擇好後自動設定雙方的網路裝置並開始傳送檔案。

關鍵字：NFC、Throughput prediction model、Measurement、Media sharing

ABSTRACT

Current mobile devices such as mobile phones and tablet PC have a lot of methods to share media content such as Wi-Fi, Bluetooth, and Internet…etc. We propose an intelligent file sharing system. This system eases the transfer of files between two mobile devices by accessing the best wireless connection available, considering quality and security. We use NFC, which has zero setup time, to exchange both network settings, then programmatically construct a link to transfer the media content. In order to determine the best method, we use experimental data to compute the quality of each method then select a method by the decision rule we made.



CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1 Introduction

1.1 Background

1.1.1 Media Content Sharing

隨著智慧型手機和平板電腦的普及，使用者在很多場合(如生日派對，公司開會，出外旅遊...等等)都會有跟身邊的人分享多媒體檔案的需求。如果我們只想和一些現在就在身邊的人分享，現在人的做法可能還是透過緩慢的上傳速度上傳到社群網站，然後設定一大堆權限，然後接收者連進社群網站後還得花時間和金錢下載剛剛分享者上傳的照片到自己的手機中。

於是我們想到，為什麼明明要分享的人就在身邊，為什麼我們要把檔案丟到數公里，甚至數萬公里的伺服器後再從遠方拿回來呢？我們應該可以利用現在手機很多的網路介面例如藍芽，Wi-Fi 直接做點對點的傳輸吧？到底什麼時候該用藍芽？什麼時候該用 Wi-Fi？使用者完全沒有任何的資訊能決定該用哪種傳輸方式，只能盲目的重複嘗試直到成功為止，我們能不能幫他簡化這個步驟呢？於是開啟了我們這個研究。

1.1.2 Near Field Communication

NFC 是一種短距離的高頻無線通訊技術，允許電子設備之間進行非接觸式點對點資料傳輸，在 10 公分內交換資料。這個技術由 Radio-Frequency Identification（RFID）演變而來，傳輸速度最大支持 848kbps，共有主動和被動模式。

從 Android 2.3 開始，開放了 NFC 相關的功能，雖然現在搭配 NFC 晶片的手機並不多，但在 Google 積極推動 Google Wallet 下，各大廠如 hTC、Sony Ericsson 等等也紛紛表態即將要推出具有 NFC 晶片的智慧型手機。根據 Juniper Research 研

究表示，在 2014 年五隻智慧型手機中就有一支會有 NFC 功能[11]，未來支援 NFC 的智慧型手機想必會越來越多。

1.2 Problem Definition and Proposed Solution

現在既有的分享方式大致上分為兩類，一類是透過既有的 Infrastructure 來傳送檔案，如透過一些網際網路上的服務來分享資訊像是 Facebook、Dropbox、Flickr、Whatsapp 等等，或是透過既有的 Wi-Fi Access Point 來傳送資訊。而另一類則是透過 Ad-hoc 的方式直接傳輸，如 Bluetooth、Wi-Fi Hotspot 等等直接傳輸的方法。但要使用者要選擇一種傳輸方式時會遇到幾個問題。第一個就是每一個分享方式都有他的基本限制或是麻煩的地方，第一類的方式像是 Whatsapp、Facebook 等網際網路上的服務需要兩隻手機都能連上網際網路，而 Wi-Fi AP 需要兩隻手機都連上同一個 Access Point，使用者只能在不斷的嘗試中找出一個能用的傳輸方式。而第二類的傳輸方式設定又過於繁瑣，對沒有相關知識或經驗的使用者常常會不知道如何使用。

第二個問題是使用者沒有足夠的資訊能判斷現在哪種連線方式能有最好的效率，使用者只知道他想傳送五張照片，或是兩部影片，但不知道這五張照片會傳送多久，或不知道這樣的連線方式是否安全，是否會讓自己的私密資訊曝光等等。為了解決這些問題，我們想要在各種不同的連線狀態下能自動的找出一種使用者操作負擔最少，卻又傳的最快速又安全的連線方式。

我們提出了一個透過 NFC 來建立連線的多媒體資訊分享系統，利用 NFC 較短的設定時間和短距離安全傳輸的特性快速的交換雙方的連線狀況，再配合我們利用理論與實驗數據推得的傳輸時間預測模型比較各種不同連線方式的效能，選出

最佳的連線方式之後，自動調整連線設定來交換多媒體資訊。

1.3 Contribution

這篇論文有以下貢獻，第一我們透過 NFC 開發一個多媒體資訊分享系統，他能考量速度和安全自動找出一個符合現在網路狀態的無線傳輸方式來傳送檔案，且使用者不需要輸入和設定任何東西。第二我們實際在手機平台上測試各種網路介面的傳輸效率，進而推論模型且驗證其準確度。

1.4 Thesis Organization

論文章節規劃如下：Chapter 2 介紹建立預測模型相關的題目，和一些在智慧型手機上現有的智慧型多媒體分享應用程式。Chapter 3 詳細的描述我們系統的架構和每一個模組實作的方式。Chapter 4 說明測量的方法，和傳輸時間預測模型的建立。Chapter 5 開始分析預測模型的正確率和使用者測試的結果。Chapter 6 提出結論和一些未來可能的方向。

Chapter 2 Related Work

2.1 NFC Interaction

NFC 互動是一種很直覺的操作方式，近年來，很多研究透過 NFC 用手機直接和大螢幕互動，Hardy 和 Rukzio[1]提出 Touch & Interact 的互動方式讓使用者拿起手機直接和大螢幕互動，並透過使用者調查得知 Touch & Interact 這種使用者介面（User Interface, UI）在速度和使用者滿意程度都超過手機傳統的使用者介面。

接下來 Broll 和 Hausen[3]提出實體使用者介面（Physical User Interface）將很多 NFC 標籤（Tag）貼在菜單上或著地圖上讓使用者透過手機直接和實體物體互動，使用者意見也都較偏好實體使用者介面。接下來這種技術廣泛的應用在各種不同領域，包括貼標籤在紙琴鍵後面把手機當成樂器[4]、把網路存取的資訊放在標籤裡讓使用者 Touch 就能上網[6]等等。

這樣實體的介面不但能讓使用者更輕易且更直覺的操作系統，也能讓年紀較長的使用者能輕鬆的接觸科技[7]。

根據上述研究，我們可以知道 NFC 所提供的實體使用者介面能提供使用者更直覺的操作方式，因此我們想利用 NFC 的直覺的操作來讓使用者互相交換多媒體資訊。

2.2 Throughput Prediction Model

Henty 和 Rappaport[8]在找出在 802.11b 環境的物理層（Physical Layer）訊號強度、干擾和使用者數量等特性對應用層（Application Layer）傳輸吞吐量（Throughput）的影響，並推導出吞吐量和訊噪比（Signal-to-Noise Ratio, SNR）的經驗模型（Empirical Model）。

Rodrig 和 Reis[9]透過實際測量 SIGCOMM 2004 會議的 802.11 無線網路訊號分析傳輸時的各種消耗（Overhead），他把消耗分成三個種類：資料（重傳）、控制（RTS、CTS 等）、管理（Beacon、認證等），並探討重傳發生的原因。

我們參考上列的相關研究在智慧型手機上建造一個傳輸時間預測模型。

2.3 Application

在現在智慧型手機熱門的 iOS App Store 和 Android Market 都有很多在近距離分享檔案的應用程式，我們找一些其中比較有代表性的拿出來分析與比較。

Bump[12]、Hoccer[13]都是擁有跨平台特性的 App，他的目的是要讓使用者能很簡易的用一些直覺的動作來交換檔案。Fig. 2-1 是 Hoccer 發佈 API 的概略圖，Bump 也是相同概念，手機會定期跟雲端主機更新他自己的環境，包括 GPS 座標、Wi-Fi 狀態等等，系統會自動將相似環境的手機群組起來。接下來就只要定義傳送和接收的動作，如 Bump 互相碰撞當傳送動作、Hoccer 丟球當傳送動作而接球當接收動作。接收和傳送動作配對成功後傳送端會把資料先傳到雲端伺服器上在轉給接收端。

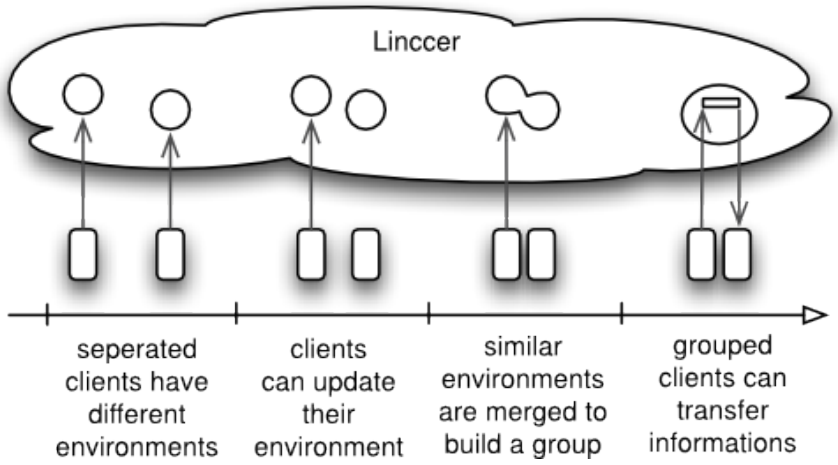


Fig. 2-1　Linccer API Overview

這類型應用程式的好處是因為配對是由伺服器端來處理，因此使用者不需要額外的設定，只要能連上網際網路，就可以互相傳輸檔案，但美中不足的是網際網路的狀況很多時候是無法控制的，加上很多地方網路上傳速率不高，傳送需要花不少的時間。

Bluetooth Transfer[14]是用藍牙裝置配對後用藍芽連線傳送檔案，不需要任何的 Infrastructure 就可以傳送檔案，但缺點是要使用藍芽裝置需要較多的設定，和藍芽傳送速率較為低，在傳送大型檔案的時候會花費較多的時間。

File Expert[15]是直接在傳送端手機上執行一個 HTTP 伺服器或是 FTP 伺服器，對方可以透過瀏覽器直接下載傳送端手機中的檔案，這樣的優點是不論是使用網際網路或是連上 Wi-Fi AP 或是利用 Wi-Fi Hotspot 都可以傳輸檔案，但相對的使用者無法判斷接收端手機是否可以連結到傳送端手機所顯示的位置，例如傳送端手機顯示他已經開啟 HTTP 伺服器，請連線至 http://192.168.0.118:8080，但是如果接收端和傳送端不在同一個 AP 下的話，其實是無法拿到檔案的。

總和以上的各種服務，每種服務都有適合使用的地方和不適合使用的地方，我們希望整合這些服務的特色，讓系統幫使用者判斷現在該使用什麼樣的服務並自動的建立連線來傳送檔案。

Chapter 3　System Design and Implementation

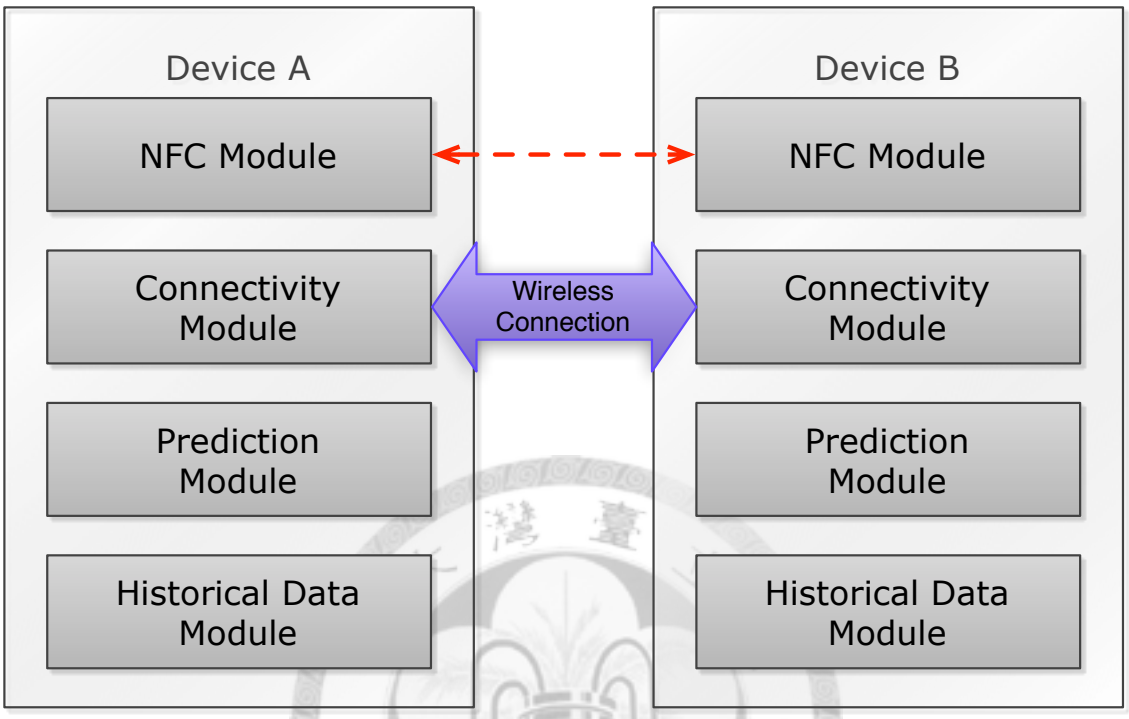


Fig. 3-1　System overview

我們把系統大略分為四個部分 Fig. 3-1，Connectivity Module 負責管理各種不同的無線裝置的狀態(如 Wi-Fi、Bluetooth…等等)；NFC Module 主要是控制 NFC 晶片交換兩裝置的網路設定和溝通傳輸的方法與參數；而 Prediction Module 是利用我們預先利用理論和實驗建立的模型來預測現在各種無線通訊方法傳送檔案需要的時間，Historical Data Module 會自動抓取各個時候每個裝置狀態改變如藍牙裝置開啟、連接某 Wi-Fi AP 等等所花費的時間並存到資料庫中，還有建立一個回饋機制讓使用我們系統後會把傳輸時間送回 Historical Data Module 來調整參數，讓預測模型慢慢適合這隻手機。

由於我們設計的最優先的考量就是盡量減低使用者的操作次數，所以我們會盡量減少 NFC 的連線次數。而 NFC 需要非常靠近（距離 3~5 公分）才能做資料交

換，對使用者來說，兩隻手機要一直保持非常靠近的狀態是一件很麻煩的事，因此我們設計的第二優先的就是盡量減低兩隻手機靠近的時間。因此我們根據這兩個原則設計了以下的架構 Fig. 3-2。

首先傳送端會檢查自己網路狀態，然後把要傳送檔案的大小和剛剛檢查的網路狀態利用 NFC 送至接收端；接收端接收完後，會快速的檢查自己的網路狀態，並把自己和對方的網路狀態傳送到 Prediction Module，利用建置好的模型計算出各種連線的時間，再同時設定自己的網路組態和利用 NFC 回傳網路命令；而傳送端接收到網路命令後就會根據命令設定自己的網路組態，並開啟連線開始傳輸。

下面幾小節，我們會詳細介紹每個模組的詳細功能和在 Android API-10 實作的方法和遇到的挑戰。

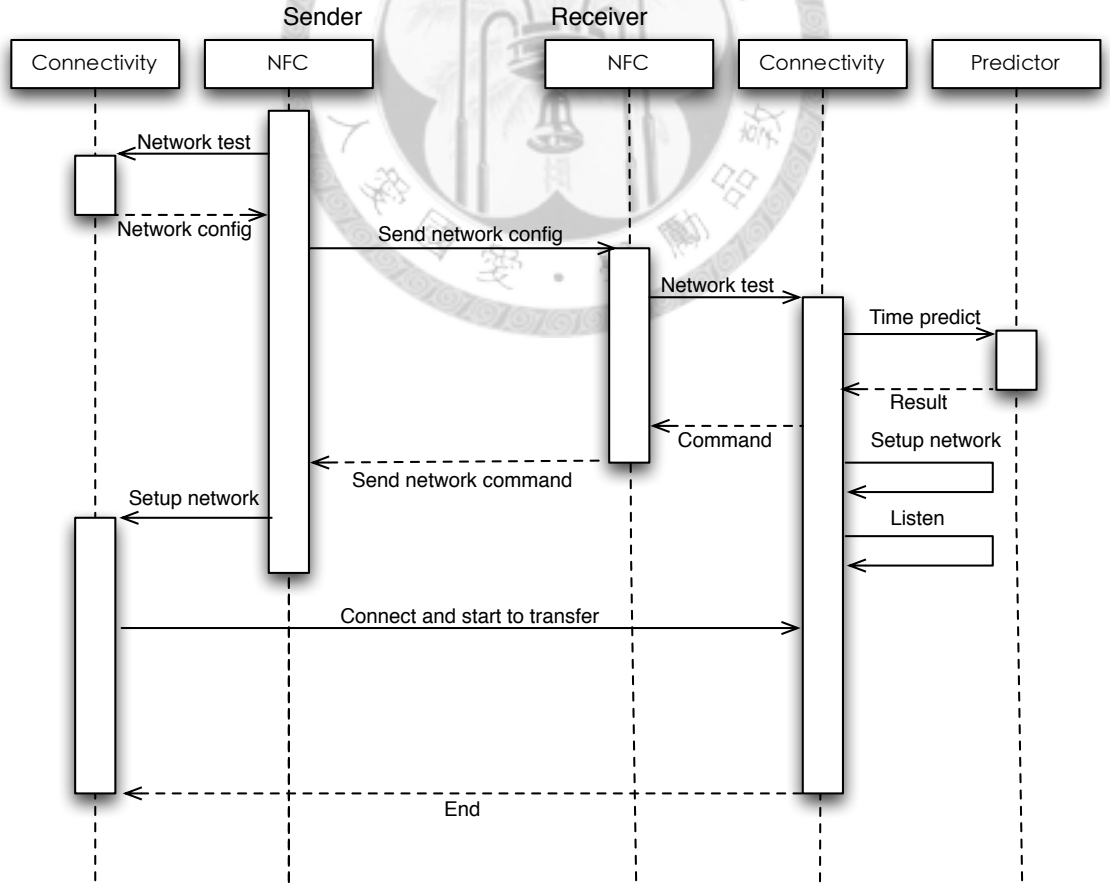


Fig. 3-2　Sequence diagram

3.1 NFC Module

3.1.1 Design

在設計 NFC 功能的時候由於我們需要雙方溝通並選擇最好的連線方式，所以我們至少需要一來一回雙向的傳輸，才能夠達成雙方溝通的要求。而要雙向溝通有兩種方式,第一種是雙方都先測試好雙方的網路狀況,然後在建立 NFC 的連線，這樣的好處是只要一次連線就可以知道自己和對方的網路狀況，而碰觸後也不用花時間在進行網路測試所以接觸的時間非常短，但是缺點是接收端得停止手邊正在做的工作開啟我們的應用程式，並做好網路測試後才能開始傳輸；而第二種是只有傳送端先做好網路測試，然後把傳送端的網路狀態透過 NFC 傳送給接收端，而接收端接收到訊息後在做快速的網路測試並回傳，優點是接收端不需要在開啟程式，系統接收到 NFC 訊息後會自動開啟，但缺點是由於接觸後還需要進行網路測試等相關工作，因此接觸的時間會較為久。

比較兩種方式後，由於使用者體驗是我們所重視的，我們不想讓接收端有使用者要接收個檔案還需要開啟程式才能接收，又因為經過我們縮短接收端的網路測試時間，所以第二種方式的接觸時間也不到 1 秒，因此我們系統決定使用第二種方式來實作。

3.1.2 Implementation

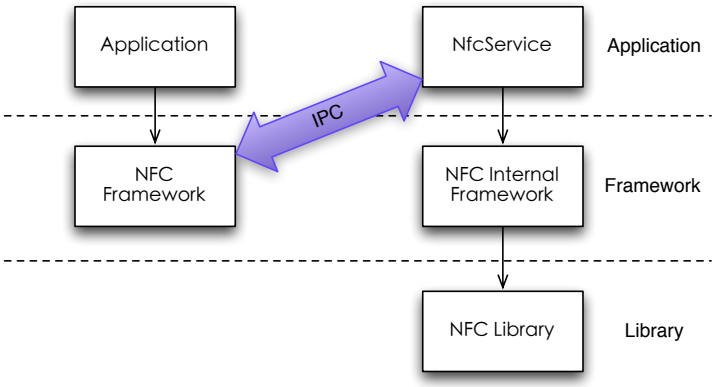


Fig. 3-3 NFC Framework in Android

在目前 Android API-10 版本中，NFC 的簡略架構如 Fig. 3-3。作業系統在打開 NFC 裝置的時候會自動執行 *NfcService* 來負責管理 NFC 相關的功能。而 NFC 現在提供的 API 是透過 Inter-Process Communication (IPC)來操控 *NfcService*。但現在 API 提供的點對點傳輸功能只有 *enableForegroundNdefPush* 和 *enableForegroundNdefDispatch* 這兩個函數來做，前者是告訴 *NfcService* 下次和其他 NFC 手機建立連線的時候要傳送什麼資訊，而後者是告訴 *NfcService* 下次接收到 NFC 某特定格式的訊息時用我們的應用程式來開啟。

由於我們設計時的第二種方式是接收端接收到 NFC 訊息後做網路測試，然後回傳網路設定給傳送端。因為 *enableForegroundNdefPush* 是設定在兩隻手機狀態從斷線到連線時要傳送的資料，所以我們必須要先將兩隻手機靠近建立第一次連線，然後接收端測試好網路後在將兩隻手機分開讓連線斷掉後然後再靠近讓他連線，這行為非常的麻煩，因此我們想要在直接操控 *NfcService* 讓兩隻手機在連線狀態的時候，也能簡單的傳送接收資料。

因此我們透過 Android Interface Definition Language (AIDL) 直接控制

NfcService 中的 Logical Link Control Protocol (LLCP) Socket，在雙方在連線狀態下直接傳送訊息到對方手機中，這樣我們就能達到只要接觸一次，也不必遠離就可以完成雙方溝通訊息的功能。

3.2 Connectivity Manager

3.2.1 Wi-Fi Module Design

Nexus S 的 Wi-Fi 支援 AP 和 hotspot 模式，但這兩個模式不能同時存在，所以若現在的 Wi-Fi 是 AP 模式的話必須先關閉 Wi-Fi 裝置在重新打開成 hotspot 模式，而同理若是要從 hotspot 模式轉回 ap 模式也需要關掉裝置在打開。

Table. 3-1 Connect and Accept Decision

Role	Connect End	Accept End
Hotspot Mode	連接 Hotspot 的一端	打開 Hotspot 的一端
AP Mode – Existed SSID	換 SSID 的一端	不用換 SSID 的一端
AP Mode – New SSID	傳送端	接收端

若是兩端經過溝通過後決定使用 Wi-Fi 來傳輸資料，兩端的 Wi-Fi 模組一個要做連線發起端(執行 socket connect 命令)；一個要做連線接收端(執行 socket accept 命令)，而在不同的狀況下則會有不同的情況 Table. 3-1。值得一提的是兩端決定要一起換到一個新的 AP 下的話，由於 NFC 連線在溝通完命令就斷掉了，所以兩端沒有任何溝通的管道，唯一確定的只有他們連上同一個 AP。這時候有兩個方法可以選擇。

第一個是通知使用者再將兩隻手機靠近透過 NFC 來傳，優點是很簡單又安全，只要直接把新拿到的 IP 利用 NFC 傳給對方，對方接受到就可以建立連線，但缺點

是要讓使用者有多一次的操作，這違反了我們設計的第一原則－**盡量減少 NFC 連線次數**；另外也可以交換完命令後不要把手機分開，直到連線確定建立後再分開，但是網路設定的時間有時候會長達 5 秒以上，加上第一次溝通的時間，等於是要要求使用者兩隻手機要彼此靠近將近 10 秒的時間，這違反了我們設計的第二個原則－**盡量減低手機靠近的時間**。

第二個則是我們想利用 Multicast 的機制來告訴其他人自己新拿到的 IP 位址。首先先讓兩隻手機連上某個相同的 Multicast group，然後一端接收，另一端每隔一秒發送一個 UDP 封包，當接收端接受到 Multicast 的封包時，自然而然也就可以拿到發送端的 IP 位址了。但是這樣子做會有安全上的問題，如果有惡意攻擊者也用同樣的 Port 做 Multicast，並且他的封包比正常發送端的封包來的還要早，那接收端就會把檔案傳到惡意攻擊者的 IP 位址了。

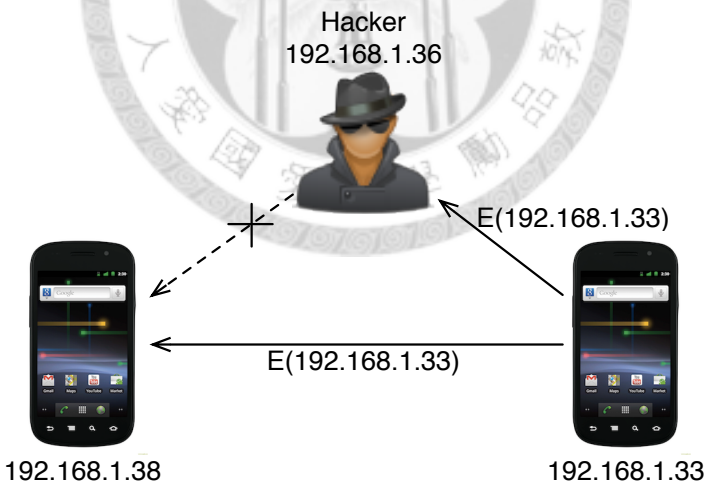


Fig. 3-4　Simple Authentication

因此我們需要有個判斷機制Fig. 3-4 來判斷此封包是正常發送端傳送的還是惡意攻擊者傳送的。我們把 Multicast 的封包內容存經過加密的發送端 IP，然後 Key 用 NFC 來交換，接收端接收到封包後就把封包的內容用 Key 來解密，然後再拿解

密後的 IP 位址和封包的來源 IP 位址做比對，如果比對一致就代表這是正常發送端送來的封包，也就可以安心的傳送檔案過去了。

3.2.2 Wi-Fi Module Implementation

在實作部分，我們使用 Android SDK 的 *WifiManager* 類別去控制 Wi-Fi 裝置，透過這個類別，能簡單的關閉開啟裝置、連上特定的 SSID、掃描附近的 AP 等等，可以完成所有 AP 模式的功能。但在 hotspot 模式需要一台手機開啟 hotspot，可是目前 Android 的版本的原始碼[16]中，控制 hotspot 的功能被隱藏起來了 Fig. 3-5，也就是說開發者無透過 API 直接操控 hotspot。但是我們能透過 Java reflection 的方式來存取 hotspot 相關的功能。因此我們寫了 *WifiAPManager* 這個類別來控制 Hotspot 的開關和 SSID 和加密等相關的設定。

```
/**
 * Start AccessPoint mode with the specified
 * configuration. If the radio is already running in
 * AP mode, update the new configuration
 * Note that starting in access point mode disables station
 * mode operation
 * @param wifiConfig SSID, security and channel details as
 *        part of WifiConfiguration
 * @return {@code true} if the operation succeeds, {@code false} otherwise
 *
 * @hide Dont open up yet
 */
public boolean setWifiApEnabled(WifiConfiguration wifiConfig, boolean enabled) {
    try {
        return mService.setWifiApEnabled(wifiConfig, enabled);
    } catch (RemoteException e) {
        return false;
    }
}
```

Fig. 3-5 Android source code of *setWifiApEnabled* method

3.2.3 Bluetooth Module Design

藍牙連線為了安全性的考量，在第一次連線時需要進行配對(Pairing)的動作。再藍牙 2.1 版過後提出了一個新的功能：Secure Simple Pairing，這個新的安全配對機制有四種方式：Numeric Comparison，Just Works，Out Of Band 和 Passkey Entry，現在手機上藍芽配對一般採取 Numeric Comparison 的方式也就是雙方手機都會顯

示一個六位數的數字來讓使用者來確認配對;而由於我們有 NFC 這個額外的連線,因此我們可以使用 Out Of Band 這種方式,利用 NFC 連線來交換機密資料來配對。

另外一般的配對都需要透過使用者手動確認配對，但因為我們系統是由 NFC 來發起連線的，而 NFC 只能在 3~5 公分內交換資料，所以若要建立連線，需要使用者確認讓對方的手機靠近才可進行傳輸，也就是說使用者再建立 NFC 連線的時候使用者已經確認過了，所以我們希望能跳過藍芽配對的動作減少使用者的負擔。

3.2.4 Bluetooth Module Implementation

在實作部分，Android API-10 有提供 *createRfcommSocketToServiceRecord* 和 *createInsecureRfcommSocketToServiceRecord* 的兩種連線方式，前者需要雙方經過事先的配對才能進行傳輸；後者不需要事先的配對，但在要連線的時候會進行一次性的配對，所以需要花費額外的時間。

這兩個連線方式都是需要透過一個 Universally Unique Identifier (UUID)來選定傳輸的頻道。連線接收端會先在 Service Discovery Protocol (SDP)的資料庫中註冊一個 UUID;而傳送端在建立連線的時候會在接收端的 SDP 資料庫搜尋這個 UUID,找到後雙方會溝通這次連線使用的頻道，然後開始連線。

所以在連線建立前我們會隨機產生一個 UUID，在把這個 UUID 透過 NFC 來傳輸，由於 UUID 是一個 128-bit 長度的數字，所以也不用擔心會被猜到的問題，在雙方溝通 UUID 後就可以開始進行資料傳輸了。

3.2.5 Network Measure Module Design and Implementation

在 Network Measure Module 中，我們先檢查各種裝置的開啟關閉狀態，若是

該裝置開啟，則會查詢該裝置的相關資訊，如藍牙 MAC 位址、Wi-Fi 的 IP 位址、Mobile Network 的 IP 位址等等。而比較不同的是，若是採用 Wi-Fi AP 的連線方式，雙方需要溝通要使用哪一個 AP 來傳送檔案，最簡單的方式就是雙方把儲存在手機裡的所有網路設定傳給對方，然後取聯集即可，但是這樣會有一個隱私權的問題，就是對方可以知道你曾經去過哪些地方。因此我們會先由一支手機先掃描附近的 AP 有哪些，然後在這些掃描的結果中再取得雙方以儲存無線網路設定的交集，這樣就不會知道對方去過哪裡，也可以溝通雙方可以使用的 AP 有哪些了。

而實作部分，根據上述文章，為了讓兩隻手機接觸時間越短越好，接收端網路測試時間要越短越好，而掃描 AP 需要花費一些時間，因此我們讓傳送端在傳送前先掃描附近的 AP，再將結果傳送至接收端，接收端只要花非常少的時間查詢那些結果有沒有在自己以儲存的網路設定資料庫中即可。

3.3 Historical Data Module

3.3.1 Setup Time Database

因為每支手機的硬體裝置都不一樣，所以我們不能只用一支手機測量到的設定時間當成每隻手機的設定時間，因此我們利用一個 *BroadcastReceiver* 來接收在日常生活每次裝置狀態改變的時候所花費的時間，如開啟藍牙裝置的時間，或是關閉 Wi-Fi 裝置的時間等等。

由於連接每個 AP 的時間也不盡相同，所以我們也同樣的用一支 *BroadcastReceiver* 來蒐集連接不同 AP 所花費的時間。

我們將這些資料透過 SQLite 存放在本機端中，供 Prediction Module 使用。

3.3.2 Transmission Time Database

在 Ad-hoc 的連線模式如藍牙、Wi-Fi Hotspot 等等，因為變數較少，傳輸時間變動並不大，所以不太需要重新計算模型參數；但是在 Infrastructure 的連線模式，如 Wi-Fi、3G 網路等等因為變數非常多，所以我們必須要建立回饋機制，記錄系統每一次的傳輸的時間，在每次傳輸完後重新迴歸該種類的模型。

3.4 Prediction Module

Prediction Module 主要計算每種連線方式的設定時間和傳輸時間的總和。

3.4.1 Setup Time

計算各種連線方式的設定時間很簡單，只要從 Historical Data Module 的資料庫中查詢該裝置的狀態改變時間和連接時間，再加起來即為該連線方式所需的設定時間。而若是在資料庫中查詢不到該資料，則使用我們在 Nexus S 測量到的數據為預設值。

3.4.2 Transmission Time

在建設傳輸時間預測模型（Transmission Time Prediction Model）的時候由於我們的情境只會在兩隻手機相近的時候傳送資料，所以在 Ad-hoc 模式（藍牙、Wi-Fi hotspot 和 NFC Direct）的時候變數較少，訊號強度對傳輸時間的影響較少，所以唯一的變數就是傳送檔案的大小。

但是在 Infrastructure 模式（Wi-Fi AP）因為雙方是連結到遠方的 AP 來傳輸資料，所以會多了幾個變數，首先是訊號強度（Received Signal Strength Indication, RSSI）的大小，訊號強度會影響封包重傳機率，訊號強度越小則封包重傳機率會越高；第二個是 AP 的忙碌程度，有越多的使用者在使用這個 AP，重傳的機率也

會越高[8][9]，雖然現在可以利用 ping 的方式取得在這個 AP 下有多少個使用者，但要花很多時間，而且也無法取得有多少使用者正在透過這個 AP 傳輸資料。因此我們的模型會只先考慮訊號強度的因素。

Chapter 4 Measurement

4.1 Measurement Setup

4.1.1 Platform

為了確定在各種不同的手機的差異性，我們選擇了兩隻手機來做實驗規格如 Table. 4-1，另外這兩隻手機的無線網路晶片都是採用 BCM4329 規格如 Table. 4-2，另外在測試 Wi-Fi AP 模式時，我們也會測試不只一種 AP 詳細的規格如 Table. 4-7。

Table. 4-1 Device Used in Network Performance Measurements

Model	Nexus S[19]	Nexus One[20]
Processor	Cortex A8 processor 1GHz	Qualcomm QSD8250 1GHz
Memory	512MB	512MB
Operating System	Android 2.3.4	Android 2.3.3
Wireless Chip	Broadcom BCM4329	Broadcom BCM4329
NFC	NXP PN544	Not Support

Table. 4-2 Wi-Fi and Bluetooth Chip in Nexus One & Nexus S

Chip	BCM4329
Wi-Fi Standard	802.11 a/b/g/n
Wi-Fi Max Data Rate	802.11n 72Mbps
Wi-Fi Output Power	2.4GHz 18dBm, 5GHz 15dBm
Bluetooth Standard	Bluetooth 2.1+EDR
Bluetooth Max Date Rate	3Mbps

4.1.2 Test Data

為了符合真實的情況，我們實際上測試了 5 隻 Android 的手機，計算記憶卡中的所有圖片共 1370 張和影片共 41 部的平均大小，圖片平均是 933.99KB，影片是 15194.57KB，因此我們建立大小從 1B、2B、4B 一直到 32MB(2^{25}B)充滿 0123456789 的文字檔案作為測試檔案。另外在測試 Wi-Fi 訊號強度對吞吐量的影響時我們使用 4MB 的檔案來做測試。

4.1.3 Measurement Tool

我們在 Java framework 層寫了一個可以接受不同連線的 socket client/server 程式，這個 socket 可以接受 BluetoothSocket(Bluetooth)，LlcpSocket(NFC) 和 TCP/IP(Wi-Fi AP or hotspot) 的 socket，在真實情況下傳檔通常都是採用 connection-oriented 的傳輸協定，以確保封包不會掉，所以我們在 NFC 和 Bluetooth 和 Wi-Fi 都是採用這樣的傳輸協定來傳送檔案。

4.2 Measure Method

4.2.1 Setup Time on Different Devices

為了確定各種裝置的開啟和關閉時間，我們實際測試開關藍牙和 Wi-Fi 和 NFC 裝置和開關 Wi-Fi Hotspot 的時間，每樣裝置開啟 100 次關閉 100 次，記錄下來。

4.2.2 Encryption and Decryption Algorithm on Android Platform

為了確定各種不同加解密演算法在 Android 平台上的執行效率，我們比較各種比較常用的對稱式加密演算法（AES-128、AES-256、DES 和 Triple-DES）在 Android 平台上實際加解密一個 4MB 的檔案 10 次，測量其平均時間。

4.2.3 Transfer Time vs. File Size

裝置擺設如 Fig. 4-1，兩隻手機間隔 50 公分，差不多是人與人之間的距離，若是使用 Wi-Fi AP 模式則 AP 放置在離手機 200 公分遠的地方。

Fig. 4-1 Experimental layout

傳送端會不斷重複傳送各種大小的檔案到接收端，接收端接收完傳送端傳來的檔案後會回傳接收到的檔案大小給傳送端，傳送端計算傳送第一個緩衝(Buffer)前到接收到接收端回傳的檔案大小即為這次傳輸的傳輸時間，每次實驗共傳輸 10 次，每次傳輸完成會休息 3000 毫秒。

4.2.4 Wi-Fi Throughput vs. RSSI

在測量訊號強度對吞吐量的影響時，我們採用和檔案大小實驗一樣的配置，在傳送前先測試訊號強度，把手機慢慢遠離 AP，從-30dBm 一次減少 5dBm 一直到-80dBm 為止，每個訊號強度重複傳 10 次，分別記錄傳輸前的訊號強度和這次傳輸的傳輸時間。

4.3 Measurement Result

4.3.1 Device Setup Time Result

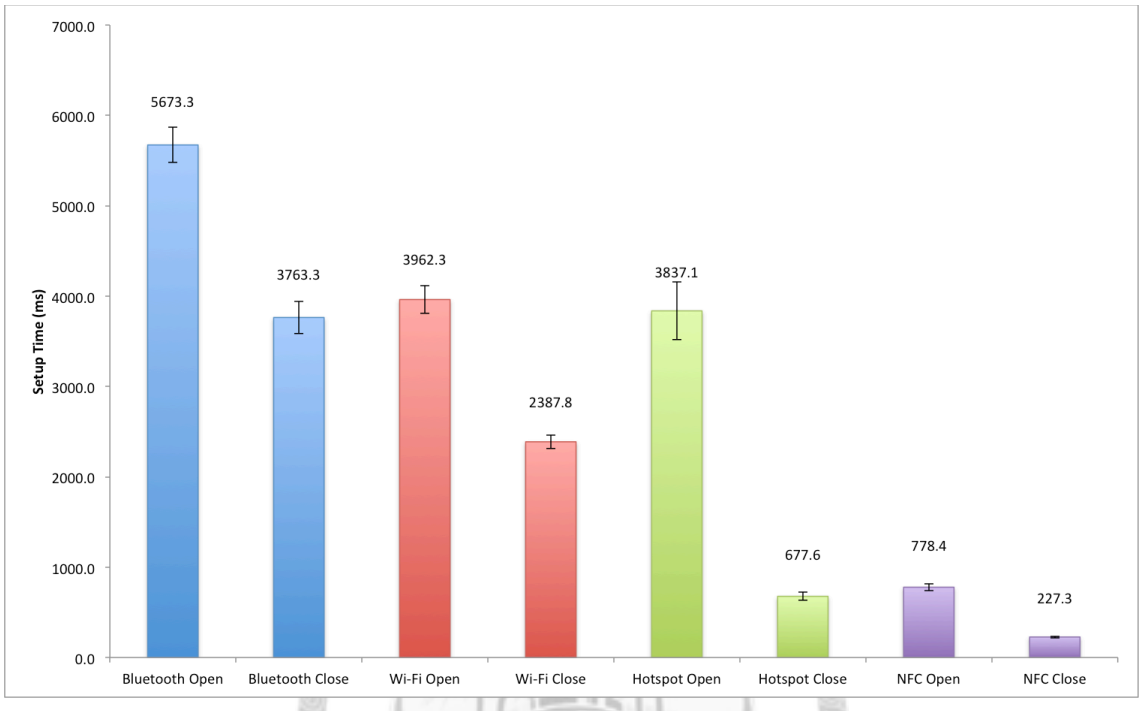


Fig. 4-2 Setup Time on Different Devices

由 Fig. 4-2 可以得知，不管是藍牙或是 Wi-Fi 裝置，改變狀態都需要花費很多的時間。藍牙裝置需要最長的開啟時間(5.6 秒)，而 NFC 裝置只需要不到 1 秒鐘的時間就可以開啟完畢，這也是我們系統為什麼要選擇 NFC 當成初始化連線的媒介的原因之一。

4.3.2 Encryption Algorithm Comparison in Android Platform

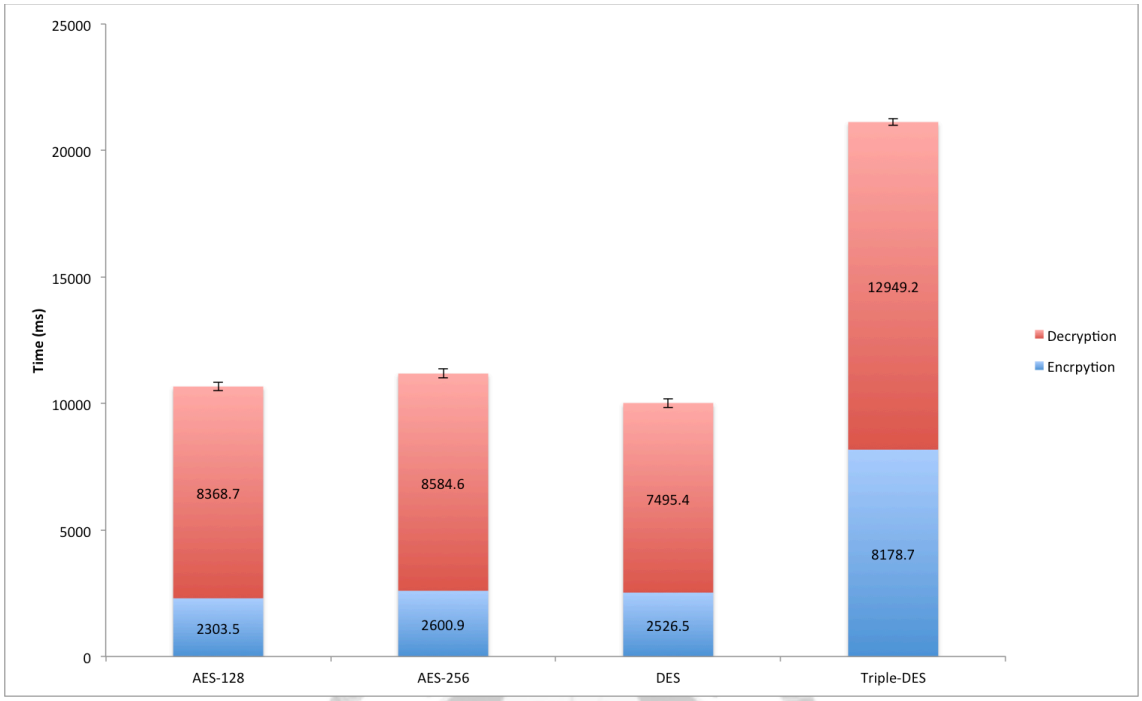


Fig. 4-3 4MB File Encryption and Decryption Time by Different Algorithm

由 Fig. 4-3 可知，最快的是 DES，而 AES-128 和 AES-256 緊接在後，Triple-DES 最慢。但是由於安全上的考量，DES 的金鑰（key）長度只有 64bits，很容易遭人破解。因為 AES-128 和 AES-256 時間只相差 4.8%，但是 AES-256 的安全性卻大大的增加，因此我們系統在不安全的網路環境（如未加密的 Wi-Fi AP 或是不安全的藍牙）會在應用層（Application Layer）採用 AES-256 加解密對傳輸的內容做加解密以確保資料的安全。

4.3.3 NFC Direct Transmission Time Result

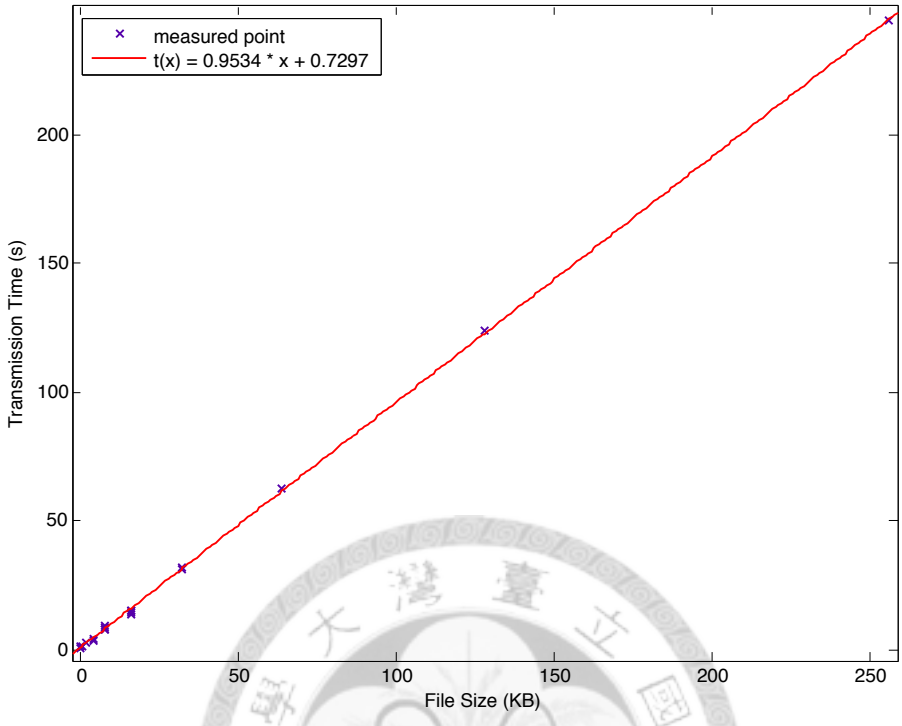


Fig. 4-4 NFC Direct Transmission Time vs. File Size

在測試 NFC 裝置的傳送時間時，我們發現在傳送時間拉長後，常常連線會斷掉，在傳送 256KB 時，能完整傳完的次數很低傳送 10 次只能成功 1 次，在傳送 512KB 以上時傳送成功的機率為 0%，因此推斷可能在 NFC 裝置傳送時有設置最大連線時間，因此我們只能利用 NFC 傳送非常小的檔案。另外根據 Fig. 4-4，NFC 的平均傳輸速率為 8.6kbps，離理論數據 828kbps 有不小的差距，我們推斷可能是 NFC 預設的一個封包大小太小（128byte）所導致。

4.3.4 Bluetooth Transmission Time Result

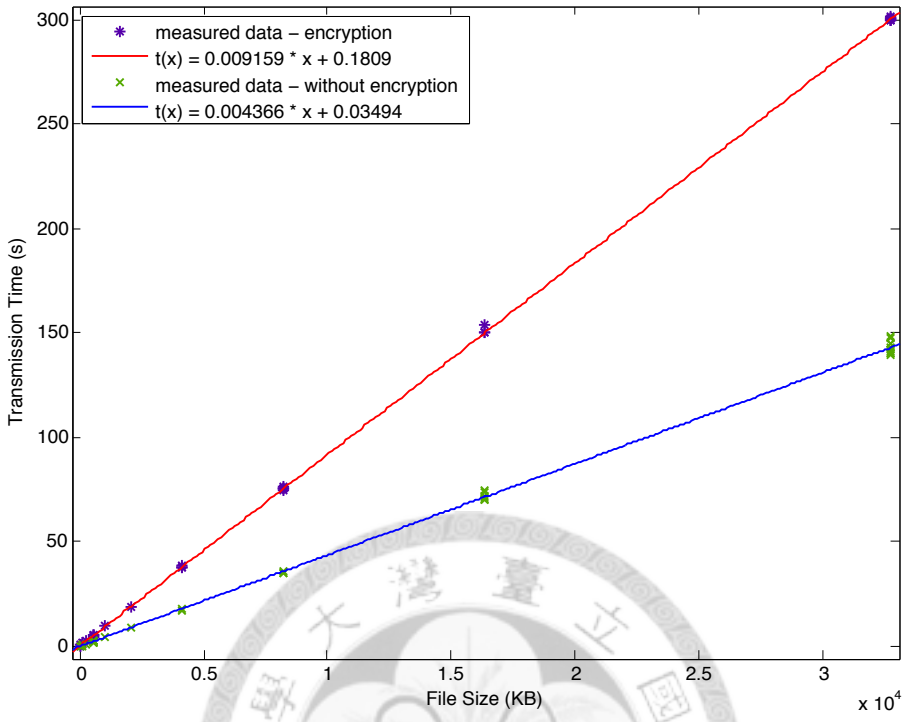


Fig. 4-5 Bluetooth Transmission Time vs. File Size

藍牙裝置因為有不安全模式和安全模式，若兩隻手機在傳送前還沒配對，系統會自動選擇不安全模式來傳輸，在不安全模式下，我們必須在應用層做點對點的家解密。因此我們測試了利用 AES-256 做加密和不用加密兩種方式傳輸的傳輸時間，結果如 Fig. 4-5，沒加密的平均速率是 1832.3kbps，加密的平均速率降為 873.5kbps，速率改變了-52.3%。

4.3.5 Wi-Fi AP Mode Transmission Time Result

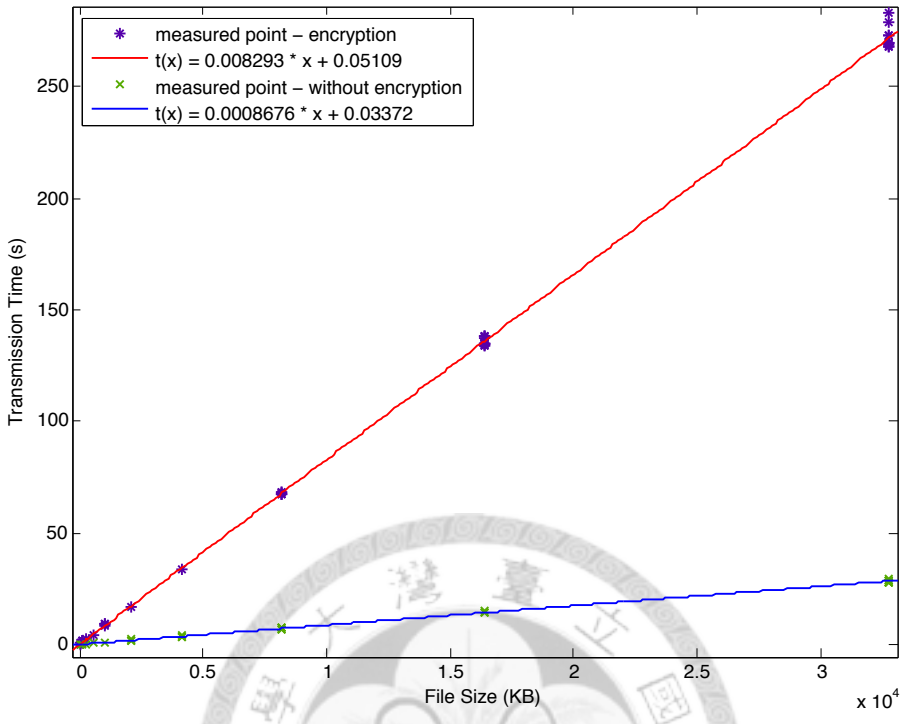


Fig. 4-6 Wi-Fi Transmission Time vs. File Size

我們系統會自動在進入沒加密的 Wi-Fi AP 下，會自動在應用層使用做點對點的加解密，和藍牙相同，我們也測試了使用 AES-256 加解密和不使用加解密的傳輸時間。結果如 Fig. 4-6，未加密的平均傳輸速率是 9220.8kbps，經過加解密後的平均速率為 964.7kbps，傳輸速率改變了-89.5%。我們可以觀察到加解密後的藍牙傳輸平均速率為 873.5 kbps，而加解密後的 Wi-Fi 平均速率為 964.7kbps，僅相差 10%左右，所以可以推得加解密傳輸的傳輸瓶頸在於加解密的速度。

4.3.6 Wi-Fi Throughput vs. RSSI Result

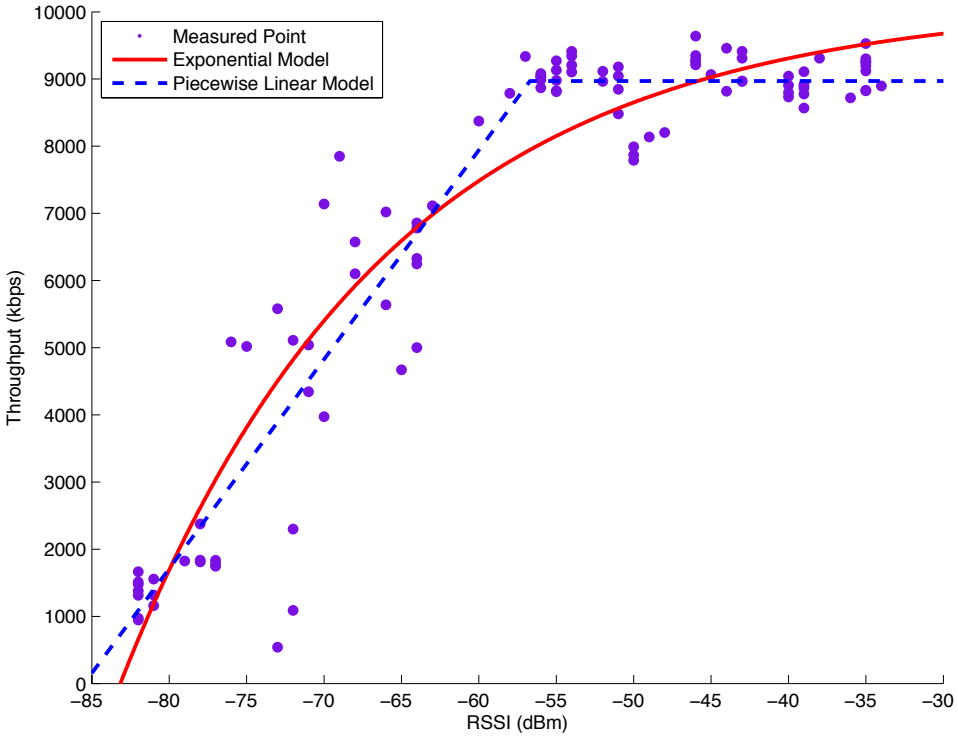


Fig. 4-7 Wi-Fi Throughput vs. RSSI

文獻上[8][9]都是計算吞吐量（Throughput）和 Signal-Noise Ratio（SNR）的關係，由上文所述，我們手機無法直接拿到雜訊（Noise），所以我們實際上用手機測量雜訊大致上為定值，因此我們仍然使用文獻裡[8]的兩種模型，並直接把 SNR 用訊號強度（RSSI）取代（SNR = RSSI – Noise）。

指數模型（Exponential Model）

$$T = T_{max}\left(1 - e^{-A_e \times (RSSI - RSSI_0)}\right)$$

分段線性模型（Piecewise Linear Model）

$$T = \begin{cases} T_{max}, & RSSI \geq RSSI_c \\ A_p \times (RSSI - RSSI_0), & RSSI < RSSI_c \end{cases}$$

我們分別使用這兩種模型迴歸結果如圖 Fig. 4-7，係數 Table. 4-3，經過誤差分析 Table. 4-4 得知，兩種模型的相關係數（Correlation Coefficients）都在 90%左右，

兩種模型都滿符合測量的數據。至於誤差來源可能來自兩個地方，首先是雜訊在實際上並不是一個定值，所以只看訊號強度會造成一定的誤差；第二是我們的訊號強度是測量傳輸前的 Beacon 的訊號強度，但整段時間的訊號強度和雜訊並不會保持在一個定值的狀態，因此可能一開始訊號強度很高（-30dBm）然後我們開始測量，但是後來可能因為某些原因（如人阻擋到訊號或是有其他干擾）訊號可能變為(-60dBm),造成平均吞吐量的下降,但是我們還是會標記這次傳輸是-30dBm，造成一定程度的誤差。

總和實驗結果，分段線性模型較指數模型好一些，因此我們系統採用分段線性模型來修正 Wi-Fi AP 的公式。

Table. 4-3 Coefficient of Throughput Models

Exponential Model			Piecewise Linear Model			
T_{max} (kbps)	A_e (dBm^{-1})	$RSSI_0$ (dBm)	T_{max} (kbps)	A_p (kbps/dBm)	$RSSI_0$ (dBm)	$RSSI_c$ (dBm)
10150	0.058	-83.16	8970	311.46	-85.5	-56.7

Table. 4-4 Statistics of Throughput Models

	μ	σ	R^2
Exponential Model	1.80	988.85	0.89
Piecewise Linear Model	0.09	847.47	0.92

4.3.7 Wi-Fi Hotspot Mode Transmission Time Result

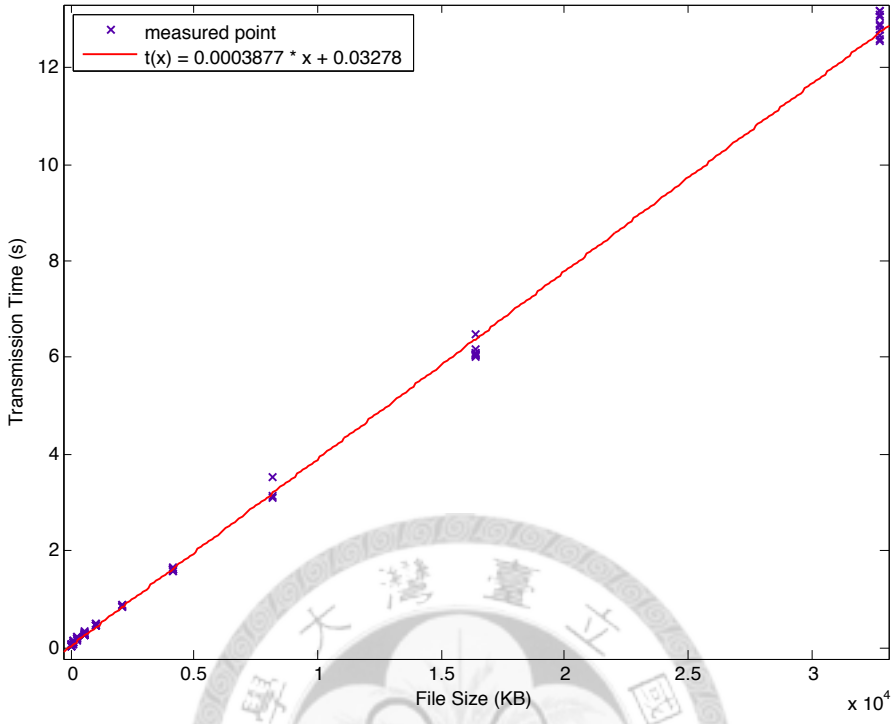


Fig. 4-8 Wi-Fi Hotspot Transmission Time vs. File Size

在 Wi-Fi Hotspot 下，平均傳輸速度為 20634.5kbps 為全部測量中最快的方式，測量的誤差也不會很大，但是如果手機現在已經連到某 AP，那在轉開 Hotspot 的話需要約 9 秒的時間，算是一個很長的時間，因此不一定在所有情形使用 Hotspot 都是最快的。

4.3.8 Summery

每種無線網路協定下都有不同的傳輸速率，速度快到慢分別為 Wi-Fi Hotspot、Wi-Fi AP Original、Bluetooth Original、Wi-Fi AP Encryption、Bluetooth Encryption 最後是 NFC Direct。

每種連線方式都大致上都成線性（R-Square = 0.99），在 Wi-Fi AP 做 Encryption 後的誤差標準差為 1.0499 最大，Wi-Fi Hotspot 為 0.0889 最小。

Table. 4-5 Coefficients and Statistics of Transmission Time Model on Nexus S

		Coefficients		Statistics		
		a ($kBps^{-1}$)	b (s)	μ	σ	R
Bluetooth	Encryption	9.16E-03	3.49E-02	0	2.66E-01	0.99
	Original	4.37E-03	1.81E-01	0	7.10E-01	0.99
Wi-Fi AP	Encryption	8.29E-03	5.11E-02	0	1.05	0.99
	Original	8.68E-04	3.37E-02	0	1.08E-01	0.99
Wi-Fi Hotspot		3.88E-04	3.28E-02	0	8.89E-02	0.99
NFC Direct		9.53E-01	7.30E-01	0	3.53E-01	0.99

4.3.9 Different Between Devices

為了確定傳輸模型在不同裝置上的參數的不同，我們在 Nexus One 也做了同樣的測量，比較結果如 Table. 4-6，在沒有加解密的傳輸方式，兩隻手機的傳輸時間模型斜率(a)差異不大，最大差異為藍牙的-3%，而截距(b)在未加解密的 Wi-Fi 下變化較大-85%，其他都在 20%以下，可知在兩隻手機，在網路傳輸方面的效能其實是差不多的；在經過加解密的處理下，兩隻手機的斜率相差較大分別是藍牙 10%、Wi-Fi 24%，截距的差異則是藍牙-71%、Wi-Fi 338%，由於加解密後的傳輸瓶頸在於加解密的速度，由此可知兩隻手機在加解密上的效能 Nexus S 比 Nexus One 好。

Table. 4-6 Variation of Models between Nexus S (NS) and Nexus One (N1)

		a ($kBps^{-1}$)			b (s)		
		NS	N1	Diff	NS	N1	Diff
Bluetooth	Encryption	9.16E-03	1.01E-02	10%	3.49E-02	1.01E-02	-71%
	Original	4.37E-03	4.23E-03	-3%	1.81E-01	1.50E-01	-17%
Wi-Fi AP	Encryption	8.29E-03	1.03E-02	24%	5.11E-02	2.24E-01	338%
	Original	8.68E-04	8.84E-04	2%	3.37E-02	4.91E-03	-85%
Wi-Fi Hotspot		3.88E-04	3.84E-04	-1%	3.28E-02	3.75E-02	14%

4.3.10 Different Between Access Points

Table. 4-7 Wi-Fi Access Points Used in Network Performance Measurements

Model	A. D-Link DIR-600	B. D-Link DIR-635	C. Edimax BR-6225n
Standard	802.11 b/g/n	802.11 b/g/n	802.11 b/g/n
Max Signal Rates	150Mbps	300Mbps	150Mbps
Transmit Power	16dBm ±1dB	15dBm±2dB	14dBm

我們使用三台 AP，規格如 Table. 4-7，其中 B 有三隻天線，可較有效率的跑 802.11n 的 Multiple Input Multiple Output (MIMO)傳輸協定，最大速度可到達 300Mbps，每一台 AP 跑一次傳輸速度實驗。做實驗時，只會開一台 AP，並都選用頻道 1，而其他兩台 AP 傳輸時都是關掉的狀態。在手機連上 AP 時，由於晶片的限制，802.11n 只能支援 Modulation and Coding Scheme (MCS) 7，也就是 65Mbps，並不能達到 150Mbps 或是 300Mbps 等傳輸速率。

我們從 Fig. 4-9 可以得知 B 的吞吐量較 A 為高，傳輸較有效率，而 C 的傳輸時間分佈十分的散，以致於我們對於這種狀況不太能預測出他的傳輸時間，因此每種種類的 AP 傳輸效能差很多，穩定度也差很多。

在 A 和 B 的比較下，A 的平均吞吐量比 B 大 54.78%。兩種 AP 都很適合線性模型（$R^2 = 0.99$）。

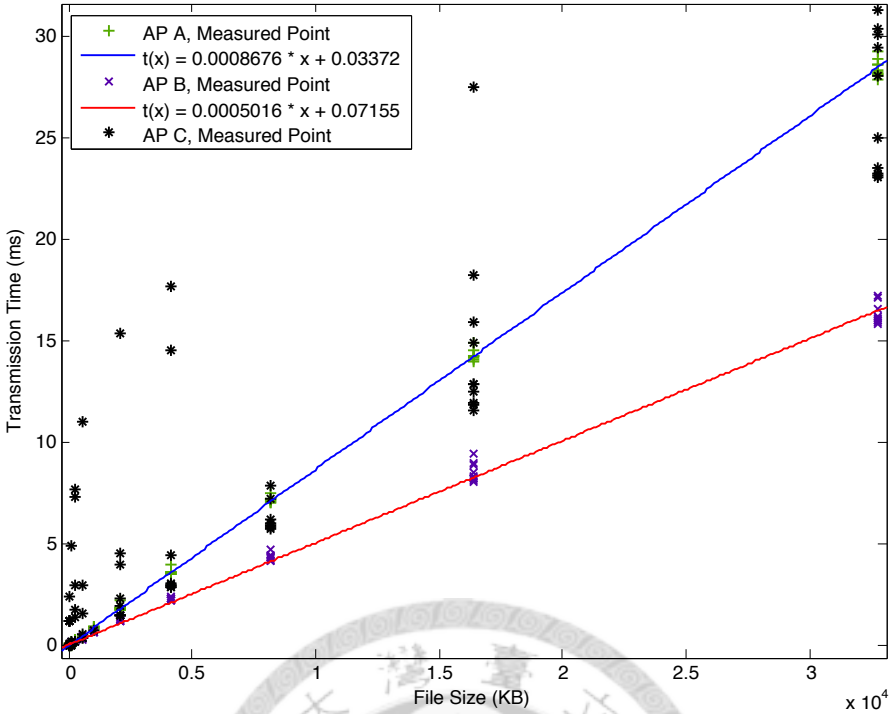


Fig. 4-9 Comparison of Different Access Points

Table. 4-8 Coefficients and Statistics of Different Access Points

	Coefficients		Statistics		
	a ($kBps^{-1}$)	b (s)	μ	σ	R
Access Point A	5.02E-04	3.28E-02	0	1.71E-01	0.99
Access Point B	7.77E-04	2.40E-02	0	1.08E-01	0.99

Chapter 5 Evaluation

Evaluation 我們主要分成兩個部分，第一個部分是分析預測模型的準確度，第二部分是和其他現有系統的比較。

5.1 Evaluation of Transmission Time Prediction Model

為了確定傳輸時間測試模型的正確性，我們做了一個準確度的測試來測試系統實際傳送時的傳輸時間和模型推得的傳輸時間的差距。

5.1.1 Setup

為了測試每種連線方式的我們每種連線方式測試 20 次，分別隨機選取 1~20 張照片或影片使用我們的系統直接傳輸，我們記錄每次傳送前和傳送後的時間為傳輸時間

5.1.2 Result

如 Fig. 5-1、Table. 5-1，在藍牙傳輸模型有最好的準確率，而在 Wi-Fi Hotspot 模式有測量時間會比預測時間長的趨勢（平均長 791 毫秒），我們觀察這些點也呈現線性狀態（R-square = 0.99，$T_{measure} = 1.03 * T_{predict}$），所以也是可預測的，但目前我們仍無法確認造成預測誤差原因為何。

Table. 5-1 Error Analysis of Predicted Model

Method	μ	σ
Bluetooth	-103.7893	251.0525
Wi-Fi AP	906.1157	1.6656×10^3
Wi-Fi Hotspot	-791.8620	960.7010

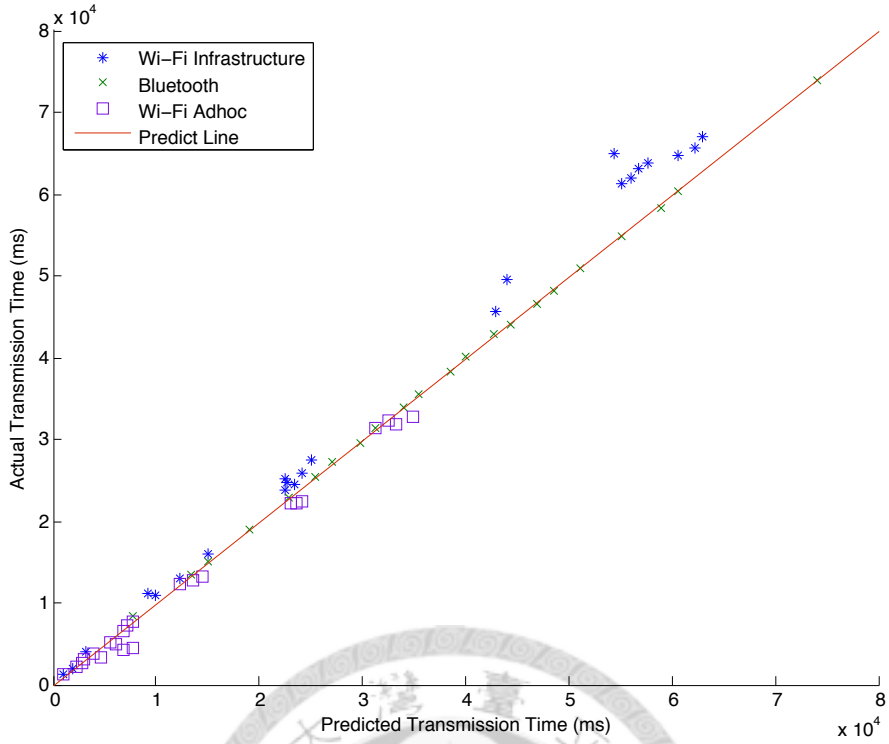


Fig. 5-1　Comparison of Predicted Time and Actual Time

5.2　Evaluation Task Time on Different System

為了比較我們的系統和現有的系統的時間，我們用錄影的方式記錄使用者實際操作各個系統傳送一張照片所花費的時間，我們把花費的時間分為三個部分，選照片時間、連接時間和傳送時間比較 Bump 和 Hoccer 和我們系統的表現。

5.2.1　Setup

受測者是一名 24 歲男性研究生，熟悉智慧型手機操作方式和三種系統的操作方式。兩隻手機都連上 dlink DIR-600 AP，AP 連結中華電信 10M/2M 光纖網路，傳送檔案大小為 946KB，每種系統傳送五次。

5.2.2　Result

測試結果如 Fig. 5-2，我們的系統在三種時間的表現都最好，而雖然 Hoccer 和 Bump 都是雲端服務，但是在傳輸時間卻差了 830%，除了 Bump 會縮小照片尺

寸外，Hoccer 是讓傳送端先把檔案丟到伺服器上，接收端在去伺服器上下載，而 Bump 是伺服器會幫兩隻手機連接起來，直接透過 AP 傳送資料，因此有相當大的差異。

另外 Bump 的連接時間為最長的原因是 Bump 常常會誤判碰撞這個行為，有時候甚至連拿起手機他都認為是在碰撞，雖然他可以調整靈敏度，但是一般使用者也不會知道要調整到怎樣的數字會比較好。

我們的系統因為目的很簡單，所以除了選照片影片不需要太多的設定等其他介面，所以在選擇傳送照片可以花很少的時間就能完成。而在連接的時間由於是用 NFC 來連接，不會有誤判的情況發生。傳輸時間因為直接透過 AP 傳送，所以也是最快的時間。

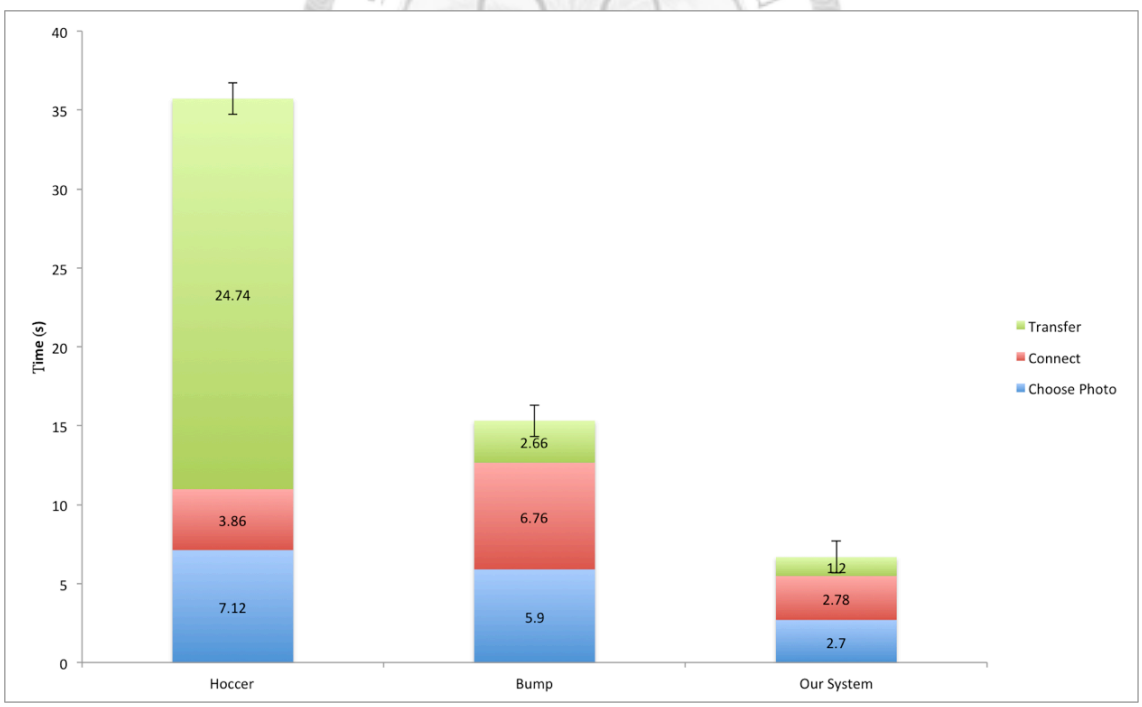


Fig. 5-2　Task time on different system

Chapter 6　Conclusion and Future Work

我們開發了一個多媒體分享系統，可以在任何情況下自動偵測網路環境而選擇最快速且安全的方式來傳送檔案，在使用者測試和現在知名的智慧型分享應用程式比較顯示我們系統不論在選照片，連接和傳送時間都是最快速的。

我們測量了各種無線裝置的傳輸時間，在 NFC 裝置有 8.39kbps，和理論數據 848kbps 相差 10007%，我們預測可能是預設的封包大小只有 128byte 因此傳輸速度降低很多。打開藍牙裝置需要 5.6 秒，平均吞吐量為 1832kbps。打開 Wi-Fi 裝置需要 3.9 秒，平均吞吐量為 9220kbps。打開 Wi-Fi Hotspot 需要 3.8 秒，平均吞吐量最大為 20634kbps。使用 AES-256 進行加解密後，傳輸時間會大幅上升。

我們建造了一個無線傳輸時間模型，能根據不同檔案大小來預測傳輸時間，在 Ad-Hoc 模式藍牙和 Wi-Fi Hotspot 能很準確預測，在 Infrastructure 模式 Wi-Fi 實際傳輸時間比預測時間大 3%，目前還沒確定原因為何。

由於我們現在只建造 Wi-Fi 和藍牙和 NFC 的傳輸模型，對於行動網路（Mobile Network）沒有進行分析，原因是因為行動網路變化比 Wi-Fi 更大，有更多的使用者，更不穩定的傳輸速率，因此預測時間是一個很難的問題，未來希望能對行動網路進行分析。

現在我們的選擇連線策略都是使用時間最短的方式，沒有電源的考量，但在某些情況下，有時候電源會比時間來的重要，像是快沒電的時候，使用者不會希望在打開 Hotspot 來加速電池的消耗，因此未來希望能把電源這個面相也考慮進來。

REFERENCE

[1] Hardy, R., Rukzio, E. Touch & Interact: Touch-based Interaction of Mobile Phones with Displays. In MobileHCI'08, 245-254, 2008.

[2] Khoovirajsingh Seewoonauth, Enrico Rukzio, Robert Hardy, and Paul Holleis. 2009. Touch & connect and touch & select: interacting with a computer by touching it with a mobile phone. In MobileHCI '09, 36-45.

[3] Gregor Broll and Doris Hausen. 2010. Mobile and physical user interfaces for NFC-based mobile interaction with multiple tags. In MobileHCI '10, 132-142.

[4] Tuomo Tuikka. 2009. 'PhonePhone': NFC phone as a musical instrument. In CHI EA '09, 2627-2630.

[5] Gregor Broll, Roman Graebsch, Paul Holleis, and Matthias Wagner. 2010. Touch to play: mobile gaming with dynamic, NFC-based physical user interfaces. In MobileHCI '10, 459-462.

[6] Minna Isomursu, Pekka Isomursu, and Mervi Komulainen-Horneman. 2008. Touch to access the mobile internet. In OZCHI '08, 17-24.

[7] Juha Häikiö, Arto Wallin, Minna Isomursu, Heikki Ailisto, Tapio Matinmikko, and Tua Huomo. 2007. Touch-based user interface for elderly users. In MobileHCI '07, 289-296.

[8] B. E. Henty and T. S. Rappaport, "Throughput Measurements and Empirical Prediction Models for IEEE 802.11b Wireless LAN (WLAN) Installations," Dept. of Electrical and Computer Engineering, Virginia Polytechnic Institute and State University, Technical Report MPRG 01- 08, 2001.

[9] M. Rodrig, C. Reis, R. Mahajan, D. Wetherall, and J. Zahorjan.

Measurement-based Characterization of 802.11 in a Hotspot Setting. In ACM SIGCOMM E-WIND Workshop, 2005.

[10] Dan Halperin, Wenjun Hu, Anmol Sheth, and David Wetherall. Predictable 802.11 Packet Delivery from Wireless Channel Measurements. Proceedings SIGCOMM 2010.

[11] Juniper Research. 1 in 5 Smartphones will have NFC by 2014. http://juniperresearch.com/viewpressrelease.php?pr=239

[12] Bump Technology. http://bu.mp/

[13] Hoccer. http://hoccer.com/

[14] Bluetooth Transfer. http://www.medieval.it/

[15] File Expert. http://www.xageek.com/

[16] Android open source project. http://source.android.com/index.html

[17] Adopted Bluetooth Core Specifications Core Version 2.1 + EDR. https://www.bluetooth.org/Technical/Specifications/adopted.htm

[18] Android Developers. http://developer.android.com/index.html

[19] Nexus S Tech Specs. http://www.google.com/nexus/tech-specs.html#

[20] Nexus One Specs. http://www.htc.com/www/product/nexusone/specification.html