國立交通大學

網路工程研究所

碩 士 論 文

以頻寬整合縮短行動裝置之檔案傳輸時間

Reduce File Transfer Latency by Bandwidth Aggregation for Mobile Devices

研 究 生：楊邵軒

指導教授：易志偉 教授

中 華 民 國 九 十 九 年 六 月

以 頻 寬 整 合 縮 短 行 動 裝 置 之 檔 案 傳 輸 時 間
Reduce File Transfer Latency by Bandwidth Aggregation for Mobile Devices

研　究　生：楊邵軒　　　　Student：Shau-Shiuan Yang

指導教授：易志偉　　　　Advisor：Chih-Wei Yi

國 立 交 通 大 學
網 路 工 程 研 究 所
碩 士 論 文

A Thesis

Submitted to Institute of Network Engineering

College of Computer Science

National Chiao Tung University

in partial Fulfillment of the Requirements

for the Degree of

Master

in

Computer Science

June 2010

Hsinchu, Taiwan, Republic of China

中華民國九十九年六月

以頻寬整合縮短行動裝置之檔案傳輸時間

學生：楊邵軒　　　　　　　　　　指導教授：易志偉 教授

國立交通大學

網路工程研究所

摘要

目前行動裝置配備有多種無線介面，可供人們連線上網際網路。但人們為了省電的目的，通常只有其中一個無線介面被啟用。因此在過去的文獻中，對於同時啟用多種無線網路介面進行傳輸的想法並未被深刻的探討過。然而在許多的應用中如車載網路，能量的消耗已不再成為絕對主要的考慮，使用多重介面傳輸來增加傳送速度則成為一種選擇。

在這篇論文中我們著手研究同時啟用多個無線介面來改善傳輸效率。傳統的檔案傳輸協定中雖然有多重連線的設計但沒有處理同時啟用多種無線傳輸介面的做法，所以我們首先提出了 *Multipath File Transfer Protocol* 來減少檔案傳輸時間。然而在考慮網路狀況變動的情形下，我們提出了 *Adaptive Multipath File Transfer Protocol*。為了更有效率使用無線介面頻寬，我們進而提出了結合 UDP 傳送及網路編碼的檔案傳輸協定 *NC-Based Multipath File Transfer Protocol*，此檔案傳輸協定更可以進一步的減少檔案傳輸時間。實驗環境設定在 netbook 透過 WiFi 和 3G 網路下載遠端伺服器的檔案。

我們透過實作比較這些檔案傳輸協定。而實驗的結果證實了同時啟用多種無線網路介面進行傳輸是可行的。而在我們的實驗中，我們所提出的協定可以減少40%的原本的 FTP 的檔案傳輸時間。

關鍵字：異質性網路，MFTP， AMFTP， NCFTP，網路編碼，WiFi，3G

Reduce File Transfer Latency by Bandwidth Aggregation for Mobile Devices

Student：Shau-Shiuan Yang Advisor： Dr. Chih-Wei Yi

Institute of Network Engineering

National Chiao Tung University

Abstract

In the past, although mobile devices were equipped with multiple radio interfaces, for the sake of power saving, only one was activated for data transmission. The idea of concurrent transmission via multiple radio interfaces has not been seriously studied. However, power consumption no longer is a problem in many application scenarios, e.g. VANETs. In this work, we investigate the performance improvement of concurrent file transfer over two heterogeneous radio networks, e.g. WiFi and 3G. The traditional File Transfer Protocol is modified to utilize two heterogeneous radio connections. We propose *Multipath File Transfer Protocol* to reduce the file transfer latency. Then, considering the network variation, we propose *Adaptive Multipath File Transfer Protocol*. Final, we propose *NC-Based Multipath File Transfer Protocol* to further raise the performance. The experiments are executed over the Internet and a 3G data network to measure the latency and average bandwidth. Our results show that it is possible to integrate the bandwidth of both radio networks. The file transfer latency of our proposed protocols is 60% of file transfer latency of FTP via single interface.

Keywords：Heterogeneous Networks, MFTP, AMFTP, NCFTP, Network Coding, WiFi, 3G

誌　　　　謝

首先，誠摯的感謝我的指導教授，易志偉教授。在兩年來給我的指導跟幫助，並且提供良好的實驗室環境，讓我得以順利完成此篇論文。

此外，我要感謝我的指導學長莊宜達，在研究上提供很多經驗與建議。另外，也感謝NOL的全體同學，在這兩年來給我的鼓勵與包容。

最後，我要感謝我的父母以及所有關心我的人對我的關懷與期許，使我在最低潮的時候可以挺過來，度過最困難的時光，謝謝每一位幫助過我的人。因為有你們的幫忙跟鼓勵，讓我在這兩年的時間學到很多也經歷很多，再次由衷的感謝你們。謝謝！

楊邵軒 於

國立交通大學網路工程研究所碩士班

中華民國九十九年七月

Contents

1 Introduction **1**

2 Preliminaries **5**

2.1 File Transfer Protocol 5

2.2 UDP-based File Transfer Protocol 8

2.3 Related Works 10

2.4 Network Coding Concept 11

3 Latency Analysis **14**

3.1 Latency Preliminaries 14

3.2 Real Latency Analysis 16

3.3 Ideal Latency Analysis 18

3.4 Latency Simulation Result 19

4 Multipath File Transfer Protocol **21**

4.1 Multipath File Transfer Protocol 21

4.2 Adaptive Multipath File Transfer Protocol 23

4.3 NC-Based Multipath File Transfer Protocol 26

5 Implementations and Empirical Results **31**

5.1 Packet Latency Analysis 32

5.2 Parameters Choice of NCFTP 34

5.2.1 ITT$_k$ mechiansm 34

5.2.2 WT mechiansm 36

5.2.3 WIN mechiansm 38

5.3 File Latency Analysis 39

6 Conclusions **44**

List of Figures

2.1 File partition and sliding windows in FTP. 6

2.2 TCP sliding window and selective-repeat operation 7

2.3 File partition and session in UFTP. 8

2.4 UFTP operation 9

3.1 CDF of latency for 1000 packets without consider ACK. 20

5.1 The architecture of the experiment environment 32

5.2 pdf of packet transfer latency of TCP over WiFi and 3G 33

5.3 The received rate via UDP of WiFi and 3G in five ITT scenarios 34

5.4 pdf of packet transfer latency of UDP over WiFi 35

5.5 pdf of packet transfer latency of UDP over 3G 36

5.6 The average file transfer latency of five scenarios 40

5.7 c.d.f of file transfer latency under WiFi with 256KB restriction 42

5.8 c.d.f of file transfer latency under WiFi with 192KB restriction 43

5.9 c.d.f of file transfer latency under WiFi with 128KB restriction 43

List of Tables

5.1 The initial change rate experiment of ITT 36

5.2 The initial condition experiment of ITT 37

5.3 The initial change rate experiment of WT 37

5.4 The initial change rate experiment of WIN 38

5.5 The increasing condition experiment of WIN 38

5.6 The decreasing condition experiment of WIN 39

5.7 The average latency and bandwidth for a file of 1MB. 41

Chapter 1

Introduction

Mobile devices, such as smartphones, PDAs, netbooks, notebooks, etc., have become part of modern life and are one of the must-have gears. For example, the penetration rate of smartphones reached 14% worldwide, 21% in US, and 8.6% in Taiwan, and the rate keeps going up [1] [2]. As part of pervasive computing, mobile devices can support various innovative services such as location-based services and augmented reality. To support these applications, a large amount of data are needed to be transmitted via wireless communications. However, insufficient wireless bandwidth may limit possible developments. In this work, we speedup file download by aggregating the bandwidth provided by heterogeneous networks and evaluate the improvement in term of file transfer latency that is usually the first user-noticing performance metrics.

To provide seamless network access, most mobile devices equip with multiple interfaces, e.g., WiFi, UMTs, WiMAX. If more than one wireless networks are available, for the sake of energy saving, only one interface will be activated, and the most choice is the one with the best signal quality. However, mobile users may experience fluctuations in service quality due to vertical and/or horizontal handoff and also contention from coexisting users. In addition, activating only one interface may waste the bandwidth that can be contributed by other idle interfaces. Therefore, it is worthy to investigate the possibility of aggregating bandwidth provided by multiple communication interfaces.

There are challenges on the implementation of bandwidth aggregation on mobile devices. First, activating multiple interfaces increases energy consumption and creates a concern on device lifetime. However, in some applications, e.g. VANET, energy consumption is no longer a problem, and due to the advancement of power management and battery technologies, energy consumption is not a major concern compared to network bandwidth. Second, although heterogeneous interfaces use difference communication techniques, the interference may exist between collocated antennas. However, according to our experiment results, this doesn't degenerate the performance on bandwidth aggregation. Third, activating multiple interfaces also produces data sequence problems. Receivers need to handle this problem especially for TCP-based applications. So, applications should carefully handle this problem.

File transfer latency is the duration between the moment users initial a file transfer task and the moment the task is completed. It should be the most noticeable performance metrics from user aspects. Compared to bandwidth, file transfer latency is a better metrics to evaluate the performance of bandwidth aggregation. In this paper, we focus on how to reduce the file download latency for a mobile device equipped with heterogeneous interfaces.

As we know, FTP supports multi-session connections to fully utilize the bandwidth between servers and clients, but multi-interface scenarios are not considered. So, the first protocol proposed in this work is a variation of the multi-session FTP. In the design, files are divided into several segments each of which will be transmitted by an individual interface, and the length of each segment is proportional to the transport capacity of the corresponding interface such that all interfaces can complete their task expectedly at the same. In other words, we minimize the expectation of file transfer latency. The proposed protocol is called Multipath File Transfer Protocol and abbreviated as MFTP. However, the interface transport capacity is not static. To further reduce the latency, we propose another protocol called Adaptive Multipath File Transfer Protocol, abbreviated as AMFTP, that dynamically dispatches sub-tasks to interfaces according to real time statistics.

Both MFTP and AMFTP are TCP-based protocols. However, there are some limitations on the TCP protocol. For example, the sliding window protocol may restrict the utility of network capacity. In order to further raise the utility of network capacity and decrease the file transfer latency, we apply random linear network coding technique to transmitted packets via UDP channels. The proposed algorithm is called Network Coding Based File Transfer Protocol, abbreviated as NCFTP. Theoretically, applying network coding technique, we only don't need to exactly identify which packets are received but only the number of coded packets is received. BTW, network coding also can provide the functionality of forward error correction. This is one reason why network coding packets can be transmitted via UDP channels.

The proposed protocols are implemented in Java language. Experiments are executed on a window-based netbook that is equipped with a built-in WiFi interface and a plugged-in USB 3G interface and a file server that is connected to the Internet by a high-speed Ethernet interface. Files are downloaded from the server to the netbook via either the WiFi interface connecting to the Internet or the 3G interface over a commercial 3G network, or simultaneously both. Various empirical data are measured in the realistic network environments. The outcomes show that bandwidth aggregation can effectively reduce file transfer latency. Besides empirical results, we also give analysis on file transfer latency for two possible designs. One represents an achievable improvement and is corresponding to the MFTP scenario. The other represents the best possible improvement and is corresponding to the NCFTP scenario. We derive the CDF of file transfer latency based on a lossy channel model. Our analytical results are verified by simulations.

Chapter 2

Preliminaries

In this chapter, we introduce some useful protocols and related works of aggregating heterogeneous networks. We also introduce the concept of network coding, it is used in our proposed protocol. First, we will introduce the most famous protocol for file transfer that is file transfer protocol.

2.1 File Transfer Protocol

File Transfer Protocol (FTP) that transmits files via Transmission Control Protocol (TCP) is one of the most popular network applications. FTP partitions files into segments of 1400 bytes except the last one of each file whose length can be less than 1400 bytes. Each segment is prefixed a header at most 20-bytes to form a TCP packet. So, the size of the TCP packets is at most 1420 bytes. Flow control and data integrity are provided by TCP. TCP adopt sliding window size for flow control. In TCP, the sliding window protocol is used for flow control, and packet integrity checking and retransmission for lost packets are used to guarantee reliability.

Each TCP packet has a unique sequence number, and must be explicitly or implicitly ACKed by the receiver. The sliding window size is the maximal allowance in the difference of the sequence number of the last sent packet and the oldest un-ACKed packet. See Fig 2.1. The window size would change in terms of network conditions. If the sliding window

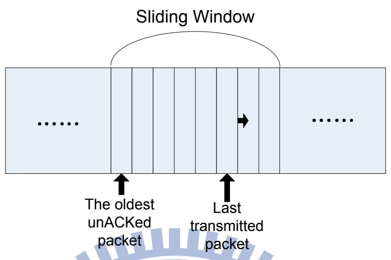

Figure 2.1: File partition and sliding windows in FTP.

become full or a timeout occurs, the window size become smaller. If all packets are ACKed in time, the window size may be increased. However, there is a upper bound for the window size. The packet transmission rate is controlled by the sliding window protocol.

The receiver inform sent the receiving of packets by ACK. The lost packets will be retransmitted according to either go-back N protocol or selective-repeat protocol. In the go-back N protocol, as the sender sends a packet, a timer is set to wait the ACK for this packet. The receiver sends an ACK back for each packet received in order. Out of order packets are dropped and no ACK is sent back. If a packet has not been ACKed and timeout happens, the sender retransmits this packet and all packets behind it. In addition, the sliding window is full, the sender retransmits all packets after the oldest packet that has not been ACKed.

In the selective-repeat protocol, similarly, as the sender send a packet, a timer is set to wait the ACK for the packet. The receiver sends an ACK back for each received packet even out of order. The sender retransmits a packet only if no ACK is received before timeout occurs or when the sliding window is full. Fig 2.2 illustrates an example of the selective-repeat protocol. In the example, the window size is 3. First, PKT0, PKT1 and PKT2 can

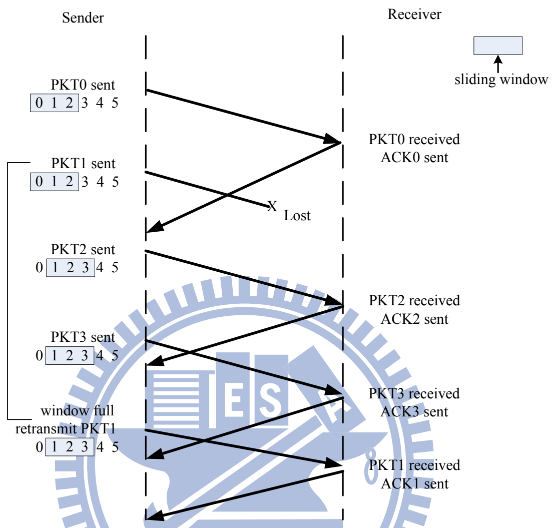

Figure 2.2: TCP sliding window and selective-repeat operation

be sent even without receiving ACKs. As PKT0 is received, ACK0 is sent back. After ACK0 is received by the sender, the window moves forward and PKT3 can be sent. Assume, PKT1 is sent but lost. So, there is no ACK1, and the sender keep sending packets until the sliding window is full or timeout occurs. Assume PKT2 and PKT3 are sent and ACK2 and ACK3 are received. When the timeout of ACK1 occurs, the sender only needs to retransmit PKT1.

Compare to the selective-repeat protocol, the buffer size needed by the go-back N protocol is smaller. However, the selective-repeat protocol has better bandwidth utilization.

2.2 UDP-based File Transfer Protocol

Instead of TCP, the UDP-based File Transfer Protocol (UFTP) use User Datagram Protocol (UDP) for data transmission. UFTP uses one TCP connection and one UDP connection. The TCP connection is for exchanging command/control packets, and the UDP connection is for transmitting data packets. UFTP partitions files into 1472-bytes segments except the last one of each file that can be less than 1472 bytes. A UDP header is 8 bytes, so the UDP data packets usually are in the size of 1480 bytes. UDP is a connectionless datagram protocol without hand-shaking, data ordering and integrity checking. Packets may arrive out of order, appear duplicated or miss without notice. UDP does not provide mechanisms to solve these problems. Applications need to handle these problems by themselves. Furthermore, UDP does not have a flow control mechanism, either. The advantage is that bandwidth can be fully utilization but it may cause buffer overflow, network congestion, and even throughput decreasing.

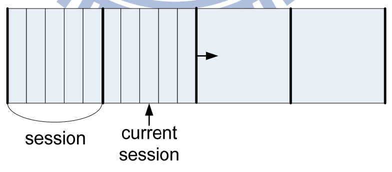

Figure 2.3: File partition and session in UFTP.

UFTP groups number of successive segments together, called a session, and sends them in batch. The session size is decided by application layer. See Fig 2.3. Each segment is assigned a unique sequence number by UFTP. The data integrity is provided in application layer session by session. The details can be found in [3]. Before the sender transmits packets, it sends a packet-tracking table to the receiver. The packet-tracking table is used to trace which packets are lost. The unique sequence number of packets is used for reordering packets and identifying lost packets. The packets of a session are sent one by one until all packets of the session are received. If packets are lost, a list of lost packets, called NAK, is sent back to ask the sender to resend the missing packets. Only if all packets of a session are received, the next session starts. If all session are received by receiver, the file transmission is finish.

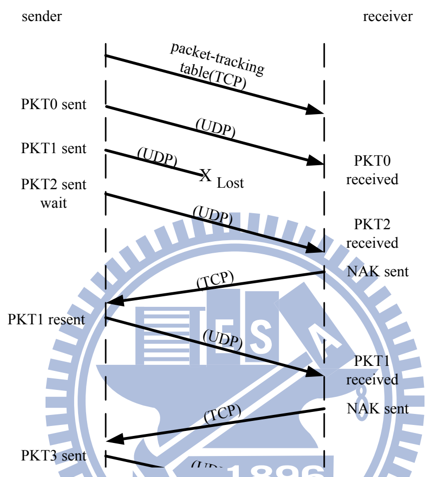

Figure 2.4: UFTP operation

An example of UFTP is illustrated in Fig 2.4. In this example, the session size is 3. The sender sends PKT0, PKT1 and PKT2 to the receiver. After finishing sending a session, the sender set a timer to wait NAK. The receiver misses PKT1, so it sends a NAK to ask the sender to resend PKT1. The sender receives the NAK to resend PKT1. After the receiver receives PKT1, the sender would start next session.

2.3 Related Works

The idea of bandwidth aggregate can be found in the early 90's, e.g., the PPP Multilink Protocol that was originally described in RFC 1717 and was updated in RFC 1990 [4].

To provide seamless network access, most mobile devices equip with multiple interfaces, e.g., WiFi, UMTs, WiMAX. If more than one wireless networks are available, for the sake of energy saving, only one interface will be activated, and the most choice is the one with the best signal quality. At the same time, there are many other mobile devices within the same area to access the resources. So, load balancing is needed to optimize system performance. In addition, vertical handoff and horizontal handoff happen as mobile devices moving from here to there. In such a case, mobile users may experience fluctuations in service quality. Moreover, the idea of activating only the best interface for communication may waste the bandwidth that can be contributed by idle interfaces. Therefore, it is worthy to investigate the possibility of aggregating bandwidth provided by multiple communication interfaces.

In literature, there were a lot of research works on the integration of heterogeneous networks. Some research contributions are architecture design and QoS supporting. In [5], a bandwidth adaptation algorithm was proposed to provide adaptive QoS support by WLAN and cellular networks. Some focus on handoff and load balancing in integrated heterogeneous networks. The handoff problems considers when and how to switch from one network to another to provide smooth transition and consistent network services. In [6], an energy-efficient scan policy algorithm was proposed to minimize the energy consumed by handoff scan operation. The load balance problem considers how to distribute subscribers among heterogeneous networks such that users can have qualified server and traffic can be fairly distributed.

However, there are some researches focused on the performance of integration heterogeneous networks. In [7], it proposed a 3-stage greedy algorithm on video streaming with heterogeneous networks, but it mainly distributes packets via heterogeneous networks under the deadline constraints. And in [8], it proposed an Earliest Delivery Path First algorithm to ensure packets arrived under the deadline constraints. However, there are some different ideas to deal with packets reordering. For example, in [9], it proposed a multilink proxy idea to handle the packets delivery. There is a research considered on real-time video applications in [10]. In order to assure the quality of service, it proposed a packet scheduling scheme to deal with the packets reordering issue and try to reduce the reordering delay. In [11], it focused on IP packet reordering for heterogeneous networks. It also proposed an mechanism of dividing and scheduling packets if the devices with resource constraints. However, these researches are not considered the file transfer latency as the most important criterion. In this paper, we consider the file transfer latency as the important criterion, and we reduce the latency as many as possible.

2.4 Network Coding Concept

The concept of network coding was first introduced in the pioneering work by Ahlswede *et al.* (2000) [12], which is proposed to solve multicast capacity problem by properly mixing information from different sources at intermediate nodes. After that, many works on theoretical designing efficient coding schemes to improve network capacity were published. Li *et al.* (2003) [13] proposed a linear network coding scheme, which regards a block of data as a vector over a certain finite field and allows a node to apply linear transformation to a vector before transmitting, for multicast in direct acyclic networks that can provide a throughput for each individual destination. [14]. In [15], Koetter and Médard proposed an algebraic approach for polynomial time encoding and decoding algorithms. Following this framework, Ho *et al.* [16] proposed a random linear network coding in which network nodes independently and randomly select linear mapping from inputs onto output links over a certain finite field.

Recently, researchers focus on implementing network coding into practices. In [17], Katti *et al.* proposed a network coding techniques for wirless mesh networks, called COPE, that was implemented as a protocol layer inserted between the IP and MAC layers. COPE was latter extended to OASIS by Wang *et al* [18], which further brought opportunistic information dissemination that tries to mix information that may provide additional throughput for receivers in future transmissions.

Gkantsidis and Rodriguez [19] proposed a randomized and distributed large content distribution solution that adopted the concept of random linear network coding and utilized network broadcasting. When a source node wants to share a large file to other nodes, the source node divides the file into equal size segments and broadcasts enough number of linearly independent combinations of these segments into the network. If one node receives enough linearly independent combinations, it can recover the original file. To increase the independence among linear combinations, instead of purely relaying received combinations, intermediate nodes may broadcast a new linear combination from received combinations.

In this paper, we adopt the idea from random linear network coding to further improve the data transfer efficiency in term of latency by integrating network coding into file transfer protocol. To see possible improvement, we will first give an analysis on latency

Chapter 3

Latency Analysis

In this section, we consider two possible thoughts to integrate channel bandwidth contributed by multiple independent interfaces for downloading files. One is a practical solution that is compatible with the existing FTP and represents achievable performance improvement. The other is an ideal solution that represents the theoretical upper bound for performance improvement. First of all, we introduce the notations and models used in the following discussion.

3.1 Latency Preliminaries

Let X denote a random variable (abbreviated as RV) of the number of successes in a sequence of n independent and identical Bernoulli trials of success probability p. The probability mass function (abbreviated as pmf) of $X = s$ for $s = 0, 1, \cdots, n$ follows the binomial distribution with parameters n and p, denoted as $f_B(s;n,p)$, and is given by

$$\Pr[X=s]=f_B(s;n,p)=\binom{n}{s}p^s(1-p)^{n-s}.$$

The cumulative distribution function (abbreviated as cdf) of X can be given in terms of the regularized incomplete beta function

$$\begin{aligned}\Pr[X\le s]=F_B(s;n,p)&=\sum_{i=0}^{\lfloor s\rfloor}f_B(i;n,p)\\&=I_{1-p}(n-s,s+1)\\&=(n-s)\binom{n}{s}\int_0^{1-p}t^{n-s-1}(1-t)^s\,dt,\end{aligned}$$

where

$$\begin{aligned}I_x(a,b)&=\frac{B(x;a,b)}{B(a,b)};\\B(a,b)&=\int_0^1 t^{a-1}(1-t)^{b-1}\,dt\text{ (the beta function);}\\B(x;a,b)&=\int_0^x t^{a-1}(1-t)^{b-1}\,dt\text{ (the incomplete beta function).}\end{aligned}$$

Readers can refer to [20] and [21] for details.

Let Y be an RV denote the number of failures in a series of Bernoulli trials of success probability p before s successes. The pmf of $Y=r$ for $r=0,1,2,\cdots$ follows the negative binomial distribution with parameters s and p, denoted as $f_{NB}(r;s,p)$, and is given by

$$\Pr[Y=r]=f_{NB}(r;s,p)=\binom{r+s-1}{r}p^s(1-p)^r.$$

The cdf of Y can be given in terms of the regularized incomplete beta function

$$\begin{aligned}\Pr\left[Y \le x\right] &= F_{NB}\left(r; s, p\right) = \sum_{i=0}^{\lfloor x \rfloor} f_{NB}\left(i; s, p\right)\\ &= I_p\left(s, r+1\right).\end{aligned}$$

Readers can refer to [22] and [23] for details.

There is a simple relation between the negative binomial distribution and binomial distribution. Let Y_s be a RV following the negative binomial distribution with parameters s and p, and X_n be a RV following the binomial distribution with parameters n and p. We have

$$F_{NB}\left(r; s, p\right) = 1 - F_B\left(s-1; r+s, p\right)$$

and

$$\Pr\left[Y_s \le r\right] = \Pr\left[X_{r+s} \ge s\right].$$

In other words,

$$\begin{aligned}&\Pr\left[\text{there are at most } r \text{ failures before } s \text{ successes}\right]\\ &= \Pr\left[\text{there are at least } s \text{ successes after } r+s \text{ trials}\right].\end{aligned}$$

3.2 Real Latency Analysis

We assume there are K independent lossy channels between the sender and receiver. Consider there are totally N packets to be transmitted and all packets are of the same size. For $k = 1, \cdots, K$, let S_k denote the number of packets that can be transmitted via the k-th channel per second, and every packet transmitted via the k-th channel is independently with a success probability p_k. So, $S_k p_k$ is the expected transport capacity of the k-th channel. Let N_k denote the number of packets to be transmitted via the k-th channel. The expected transmission duration of the k-th channel is $\frac{N_k}{S_k p_k}$. Let T_i denote the transmission duration of the k-th channel. $T = \max_{k=1}^{K} T_i$ is the transmission latency. Note that once a packet is added into to the buffer of a channel, it can't be dequeued from the buffer.

Intuitively, if the bandwidth of each channel is fully utilized and all channels have the same transmission duration, the latency is minimized. So, we distribute packets to channels proportionally to their transport capacity such that all channels have the same expected transmission duration. Thus,

$$N_k = \frac{S_k p_k}{\sum_{i=1}^{K} S_i p_i} N, \tag{3.1}$$

and the CDF of the latency can be derived as follows

$$\begin{aligned}
&\Pr\left[N \text{ packets can be transmitted in time } T\right] \\
&= \prod_{k=1}^{K} \Pr\left[N_k \text{ packets can be transmitted via the } k\text{-th channel in time } T\right] \\
&= \prod_{k=1}^{K} F_{NB}\left(\lfloor TS_k \rfloor - N_k; N_k, p_k\right) \\
&= \prod_{k=1}^{K} \left(1 - F_B\left(N_k - 1; \lfloor TS_k \rfloor, p_k\right)\right).
\end{aligned} \tag{3.2}$$

where $F_{NB}(\cdot; s, p)$ is the CDF of the negative binomial distribution with parameters s and p, and $F_B(\cdot; s, p)$ is the CDF of the binomial distribution with parameters s and p.

3.3 Ideal Latency Analysis

Although the expected transmission durations of all channels are the same, the real transmission durations are dynamic and different from channels to channels. Applying dynamic dispatching, we can further reduce the latency. In addition, applying the random linear network coding technique, we don't need to distinguish packets are transmitted via which channels, and theoretically, the receiver only needs totally N coded packets to decode the original N native packets. In such a case, the bandwidth of the K channels can be really treated as one. For $k = 1, 2, \cdots, K$, let X_k denote the number of packets received via the k-th channel. $X = \sum_{i=1}^{K} X_i$ is the total number of packets received via the K channels. X_k is an RV following the binomial distribution with parameter $\lfloor TS_k \rfloor$ and p_k, and the PMF of X can be given by the convolution of the PMFs of $X_1, X_2, \cdots, X_K$. The CDF of the improved file transfer latency can be derived as follows

$$
\begin{aligned}
&\Pr\left[N \text{ packets can be received in time } T\right] \\
&= \Pr\left[\sum_{i=1}^{K} X_i \geq N\right] = 1 - \sum_{n=1}^{N-1} \Pr\left[\sum_{i=1}^{K} X_i = n\right] \\
&= 1 - \sum_{n=1}^{N-1} \sum_{n_1+\cdots+n_K=n} \prod_{i=1}^{K} f_B\left(n_i; \lfloor TS_i \rfloor, p_i\right). \\
&= 1 - \sum_{n_1+\cdots+n_K \leq N-1} \prod_{i=1}^{K} f_B\left(n_i; \lfloor TS_i \rfloor, p_i\right). \qquad (3.3)
\end{aligned}
$$

Note that this ideal scenario optimizes file transfer latency.

3.4 Latency Simulation Result

A simple simulation is executed to verify the correctness of our analysis. Consider a file of 1.33MB is transferred and each data blocks is of 1400 bytes. Therefore, 1000 packets will be transmitted. There are three independent lossy channels between the file server and the portable device. The bandwidths of these channels are 1Mb/s, 1.2Mb/s, and 1Mb/s, respectively. Each packet can be successfully transmitted with the probability 0.9, 0.7, and 0.9, respectively. For simplicity, we assume the feedback is perfect. In other words, there is no cost for acknowledgement and the result is known immediately.

First, Eq. (3.1) is applied to distribute packets among channels. This heuristic is denoted as H1 and will be implemented in Section 4. Next, with the help of such as network coding, 1000 packets in total received by the receiver is enough to recover the file. This heuristic is denoted as H2. A protocol will be designed based on this heuristic in Section 4.3. In Fig 3.1, the green curve with cross marks is the CDF of the latency of H1, and the blue curve with triangle marks is the theoretical CDF given by Eq. (3.2); the black curve with circle marks is the CDF of the latency of H2, and the red curve with square marks is the theoretical CDF given by Eq. (3.3). The simulation outcomes agree with the analytical results. We can see latency can be reduced by adaptively packet dispatching.

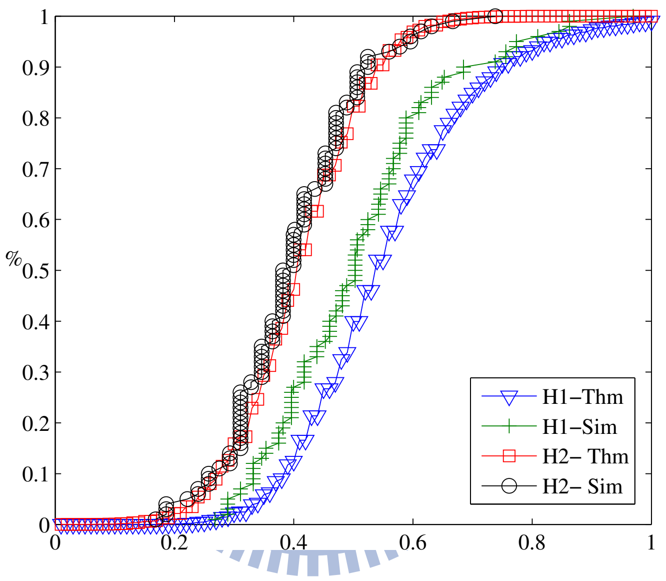

Figure 3.1: CDF of latency for 1000 packets without consider ACK.

Chapter 4

Multipath File Transfer Protocol

In this chapter, we propose three protocols for file transfer. First, we introduce the detail of Multipath File Transfer Protocol. Then, we consider the factor of fluctuated of network bandwidth, we propose Adaptive Multipath File Transfer Protocol. Finally, we further rise the utility of network bandwidth, we combine network coding concept and transferring packets via UDP connections. The proposed protocol is called NC-Based Multipath File Transfer Protocol.

4.1 Multipath File Transfer Protocol

File Transfer Protocol (FTP) on top of Transmission Control Protocol (TCP) is one of the most popular network applications. FTP partitions files into 1400-bytes data blocks except the last one of each file whose length can be less than 1400 bytes. Each data block is prefixed a TCP head no longer than 20 bytes to form a TCP packet. TCP provides flow control and data integrity for FTP. In TCP, flow control is implemented by the sliding window protocol, and data integrity is guaranteed by packet integrity checking and lost packet retransmission.

FTP has supported concurrent multi-session file transfer to fully utilize the bandwidth between senders and receivers. Our first thought is to design a multi-interface file transfer protocol based on the multi-session feature such that the enhancement is compatible with the original FTP. The proposed algorithm opens one data session, or called a data channel, on each interface. Files are divided into segments, and one segment will be transmitted by one data session. The size of each segment is proportional to the bandwidth of its corresponding interface. Each data channel acts like in the original FTP. The proposed protocol is called Multipath File Transfer Protocol, and abbreviated as MFTP. Since MFTP is compatible with FTP, only the client-side psuedocode will be given.

Assume there are K interfaces available, and totally N packets are going to be transmitted. For $k = 1, \cdots, K$, let B_k be the bandwidth of the k-th interface, and $N_k = N \times B_k / \sum_{i=1}^{K} B_i$ be the number of packets to be transmitted by the k-th interface. The psuedocode of MFTP is given in ALGO 1. To implement MFTP, interface information and bandwidth information are needed. The interface information are provided by a profile file, and the bandwidth information can be obtained either by probing the connections or from historical statistic data. A separate control session, or called a control channel, is opened for exchanging control information such as the interface information, file information, and segment information, etc.

ALGO 1 MFTP for the receiver

```
$B_k$: the bandwidth of the $k$-the interface
$N$: the total number of data blocks
$T_0 = 0$ and $B = \sum_{k=1}^{K} B_k$
for $k = 1$ to $K$ do
    $T_k = T_{k-1} + N \times B_k / B$
end for
for all $k = 1$ to $K$ do {parallel}
    Receive data blocks $T_{k-1} + 1$ to $T_k$ via the $k$-th interface
end for
```

4.2 Adaptive Multipath File Transfer Protocol

MFTP minimizes the expected file transfer latency based on the assumption of static channel bandwidth. However, end-to-end bandwidth fluctuates according to network conditions. A channel whose bandwidth drops may dominate the latency performance. If we apply adaptive partition strategies, the latency performance can be improved. An adaptive heuristic comes up to deal with dynamic bandwidth fluctuation. In the design, files are partitioned into M segments each of which is composed of S data blocks except the last one. The number of data blocks assigned to each channel is calculated one segment by one segment. If one interface reminds less than R data blocks to be received, the bandwidth of all channels are re-evaluated, and the receiving of the next segment is arranged. The number of data blocks to be received by each channel is calculated based on current channel bandwidth and also the number of data blocks that has not received. This procedure is iteratively executed until the last segment is scheduled. The proposed file transfer protocol is called Adaptive Multipath File Transfer Protocol, abbreviated as AMFTP, is depicted in ALGO 2.

ALGO 2 AMFTP for the receiver

```
$B_k$: the bandwidth of the $k$-th interface.
$R_k$: the number of residual data blocks to be received via the $k$-th interface.
Divide the file into $S_1, \cdots, S_M$ segments each of which has $S$ data blocks.
Initialize $R_k = 0$ for all $k = 1, \cdots, K$
for $i = 1$ to $M$ do
  Calculate $N_k$ for all $k = 1, \cdots, K$ such that $\min_{N_1+\cdots+N_K=S} \max_{1\le k\le K} \frac{R_k+N_k}{B_k}$
  $T_0 = S \times (k-1)$ and $B = \sum_{k=1}^{K} B_k$
  for $k = 1$ to $K$ do
    $T_k = T_{k-1} + S \times B_k/B$
  end for
  for all $k = 1$ to $K$ do
    if $N_k \neq 0$ then
      Data blocks $T_{k-1}+1$ to $T_k$ is scheduled to be received via the $k$-th interface.
    end if
  end for
  Update $B_k$ and $R_k$ for all $k = 1$ to $K$ as one of those interfaces has less than $R$ data blocks to be received.
end for
```

4.3 NC-Based Multipath File Transfer Protocol

Files are divided into equal-length data blocks, and a certain number of data blocks are grouped together and transmitted in a batch. Each group of data blocks is called a coding set. A linear combination of data blocks in a coding set is called a coded (data) block. In the random linear network coding scheme, the coefficients for encoding data blocks are generated randomly. Instead of the native data blocks, the coded blocks as well as the encoding coefficients are transmitted. Due to the linear independency of random vectors, in most cases, the data blocks in a coding set can be decoded if the same number of coded blocks as of the native data blocks are received no matter how many packets or which packets are lost. In some sense, random linear network coding provides the functionality of forward error correction, and this is why coded blocks can be transmitted via lossy UDP channels.

However, UDP does not provide the functionality of data integrity and flow control. Just like UFTP, additional mechanisms are needed to guarantee data integrity and provide flow control. Data integrity is guaranteed by the forward error correction of random linear network coding, and flow control is provided by a sliding window mechanism and a packet dispatching mechanism. In addition, an adaptive waiting heuristic is used at the receiver side. The proposed protocol is called NC-based File Transfer Protocol or NCFTP in short. Details are given below.

Let $csSIZE$ denote the size of coding sets, i.e. the number of data blocks in a coding set. ITT_k, called the Inter Transmission Time of the k-th channel, is the minimal waiting time between two successive packets added into the buffer of the the k-th channel. ITT_k is a variable parameter for flow control. The receiver needs $csSIZE$ coded blocks to decode a code set and 1 more coded block for checking integrity. Since UDP channels are lossy, more than $csSIZE$ coded blocks will be added into channel buffers. Let p_k be the probability of successful transmission of the k-th channel, and N_k be the number of data packets added into the k-th channel. Then, $\sum_{k=1}^{K} p_k N_k \geq csSIZE$ is the condition to stop adding data packets into the buffers. Note that besides the UDP data channels, a TCP control channel is maintained for exchanging commands and control messages, transmitting lost data packets, etc.

After adding enough number of coded blocks into the buffers, the sender will wait a reply from the receiver. At the receiver side, the receiver sets a countdown timer to WT for receiving coded blocks. As the time is up, the receiver tries to decode received coded blocks by applying Gaussian elimination. If the coding set can be decoded, an ACK is replied; otherwise, a NAK is sent back to indicate which data blocks are needed. If a NAK is received, the sender will transmit the indicated data block via the TCP connection. Therefore, the coding set can be decoded and verified, and the receiver send and an ACK back. After the sender receives an ACK from the receiver, the transmission of the coding set is completed. At the same time, the receiver also does statistics to adjust the waiting time WT and inter transmission time ITT_k.

1. If the packet lost rate during the last half of the waiting time is significantly higher than the rate during the first half of the waiting time, WT will be increased. On the other hand, if most packets are received earlier than the timeout of WT, WT will be

decreased.

2. If the packet loss rate of the k-th channel is above a threshold and not due to the insufficiency of waiting time, ITT_k will be increased. On the other hand, if the packet loss rate is below a threshold, ITT_k will be decreased.

At the same time, to prevent wasting bandwidth on waiting the NAK, the sender keeps going to send the following coding sets. On the other hand, to prevent network congestion, the number of ongoing coding sets must be controlled. Therefore, a sliding window like the one used in TCP is adopted for limiting the number of ongoing code sets. Let $minWIN$, $maxWIN$, and WIN, respectively, denote the minimum sliding window size, maximum sliding window size , and current sliding windows. WIN is the maximal number of ongoing coding set before receiving an ACK. When the last unACKed coding set is ACKed, the coding set window advances. WIN increases as network conditions become better, and decrease as network conditions become worse.

The sender part of NCFTP is depicted in ALGO 3, and the receiver part is in ALGO 4.

ALGO 3 NCFTP for the sender

```
$CS_1, CS_2, \ldots, CS_N$: coding sets to be transmitted.
$ACK = i = 0$
Initialize $ITT_k$ and $p_k$, for $k = 1, 2, \ldots, K$, and $csWIN$, and $csWIN$.
repeat
  if $i < ACK + csWIN$ then
    $i = i + 1$
    for all $k = 1$ to $K$ do {parallel}
      $N_k = 0$
      repeat
        Add a coded packet of $CS_i$ to the buffer of the $k$-th channel every $ITT_k$ seconds.
        $N_k = N_k + 1$
      until $\sum_{k=1}^{K} p_k N_k \geq csSIZE$
    end for
  end if
  if receive an ACK then
    Update $ITT_k$ and $p_k$, for $k = 1, 2, \ldots, K$, and $csWIN$.
  end if
  if receive a NAK then
    Send the requested native data blocks to the receiver by a TCP channel.
  end if
until $ACK = N$
```

ALGO 4 NCFTP for the receiver

```
$N$: the total number of coding sets.
repeat
  if a coded packet $P$ is received then
    Assume $P$ belongs to the coding set $CS_i$.
    if $P$ is the first received coded packet of $CS_i$ then
      Set the count down timer $T_i = WT$.
    end if
    Decode $CS_i$.
    if $CS_i$ is completely decoded. then
      Send $ACK_i$, clear timer $T_i$, and update $WT$.
    end if
  end if
  if a timer $Tj$ is up then
    Send $NAK_j$ with a list of requested native data blocks.
    Clear time $T_j$, and update $WT$.
  end if
until All coding sets are received.
```

Chapter 5

Implementations and Empirical Results

We implement the proposed protocols in Java language. A netbook equipped with a built-in WiFi interface and a plugged-in USB 3G interface downloads files from a file server connected to the Internet by a high-speed Ethernet interface. The WiFi interface connects to an AP backhauled to the Internet and the 3G interface also connects to the Internet over a commercial 3G network. We use a bandwidth control software to limit the bandwidth of WiFi interface in order to have a fair comparison between the 3G and WiFi interfaces. The bandwidth of WiFi interface is set to 128KB/s, 192KB/s and 256KB/s. The architecture of the experiment environment is in Fig. 5.1. There are three download scenarios: (a) files are downloaded via the WiFi interface; (b) files are downloaded via the 3G interface; (c) files are downloaded concurrently via both interfaces. The data provided in this section are

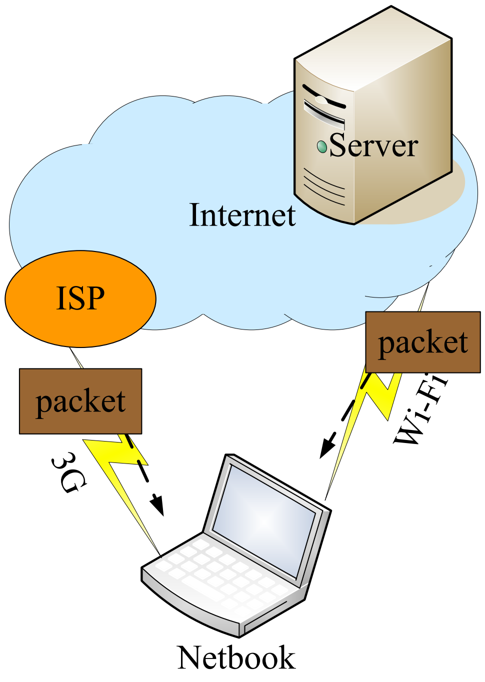

Figure 5.1: The architecture of the experiment environment

measured in realistic network environments.

5.1 Packet Latency Analysis

We first give some analyze the packet transfer latency of WiFi and 3G between client and server through implemented FTP. In this experiment, 1000 packets are transmitted from the server to the receiver. The packets size is 1400 byte. The WiFi bandwidth is set to 128KB/s by software. In order to accurate measure the packet transfer latency, we will control the inter transmission time between packets by packets. Because we want to eliminate the time of packets in the buffer, we write 15 packets into the buffer per second. To measure packet transfer latency, the clocks of the two computers are synchronized. The server places a time stamp in each packet. As the client receives a packet, it calculates the time difference between the current time and the time stamp in the packet. We collect packet transfer latency of FTP by WiFi and 3G only. Fig. 5.2 illustrates the empirical pdf of TCP connections over WiFi and 3G interfaces. The packet transfer latency distribution is analogous normal distribution. Most packet transfer latencies are shorter than 80ms in TCP. We also do the measurement

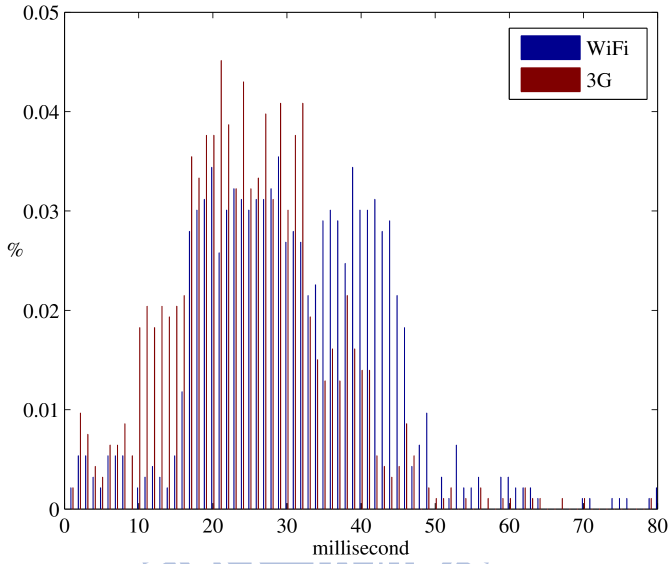

Figure 5.2: pdf of packet transfer latency of TCP over WiFi and 3G

for UDP connection. However, since UDP does not provide flow control, we figure out a proper ITT first. In order to consider the performance of NCFTP, we test some possible ITT. The probability of successful deliver rate under different ITT is given in Fig. 5.3. The ITT choose 3ms and 5ms for WiFi and 3G respectively. The PDF of transfer latency of UDP connection is illustrated in Fig. 5.4 and 5.5.

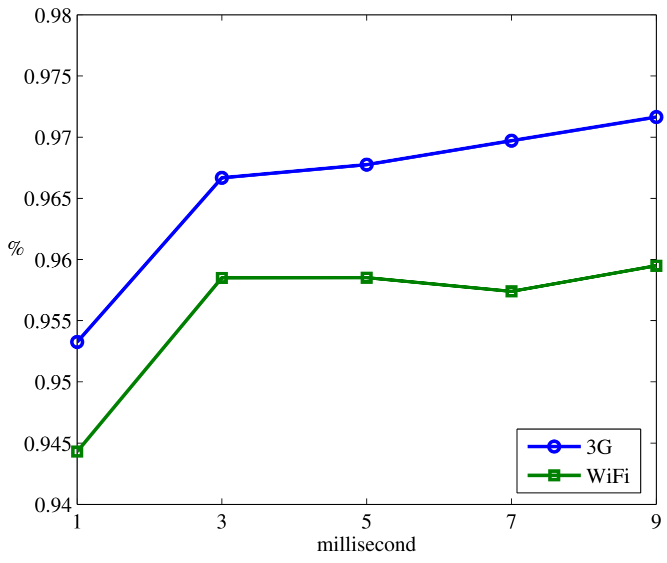

Figure 5.3: The received rate via UDP of WiFi and 3G in five ITT scenarios

5.2 Parameters Choice of NCFTP

Next, we will choose the parameters for NCFTP via the experiments. In NCFTP, we should maintain flow control in the application layer. The flow control follows the previous section designed. There are three parameters adjusted with time including ITT_k, WT, and WIN. In next subsections, we will introduce our mechanisms of parameters in NCFTP. We choose these parameters of NCFTP via the experiments. The receiver sends a download request to the sender, then the sender sends a file to the receiver. The size of file is 1MB.

5.2.1 ITT$_k$ mechiansm

When the congestion occurs, there are many successively packets lost in the pattern of received packets in the receiver side. ITT must become longer to avoid too many packets

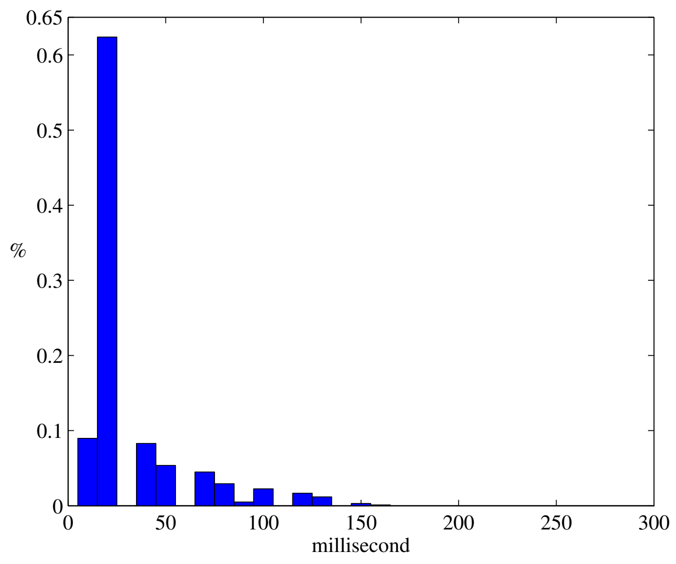

Figure 5.4: pdf of packet transfer latency of UDP over WiFi

inserting into the buffer of the sender. If the successively lost packets in the receiver side does not too many, ITT_k can become shorter to reduce inter transmission time. ITT_k is adjusted by Eq. (5.1). Let s_lost, $inITT$, $deITT$, $maxITT$, $minITT$, $congestion$, and $lower$, respectively, denote the number of successively lost packets in a coding set, increasing rate of ITT, decreasing rate of ITT, maximum Inter Transmission Time, minimum Inter Transmission Time, the congestion occur, and lower network utility.

$$ITT_k = \begin{cases} \min(inITT \times ITT_k, maxITT) & \text{if } s\_lost \geq congestion \wedge ITT_k < maxITT; \\ \max(deITT \times ITT_k, minITT) & \text{if } s\_lost < lower \wedge ITT_k > minITT; \end{cases} \tag{5.1}$$

We find the parameters of Eq. (5.1) via the experiments. First, we fix the *congestion* and *lower*, and test the file transfer latency of NCFTP in the different *inITT* and *deITT*. The results are given in Table 5.1, We choose 1.2 and 0.8 as *inITT* and *deITT*.

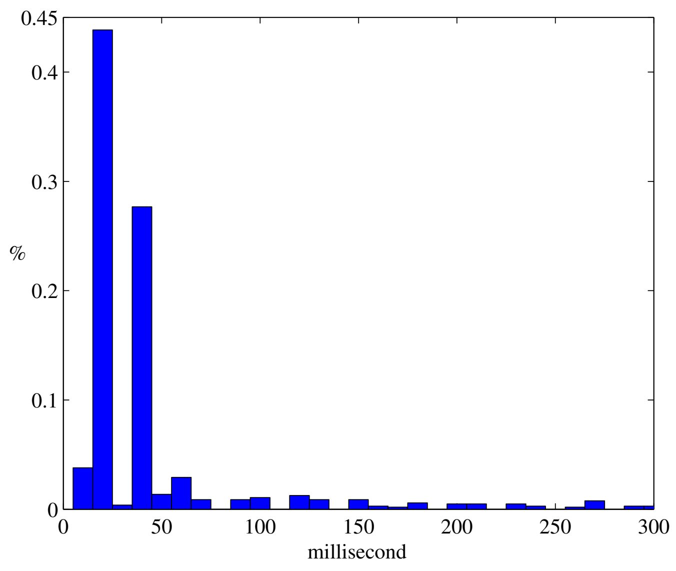

Figure 5.5: pdf of packet transfer latency of UDP over 3G

Table 5.1: The initial change rate experiment of ITT

deITT/inITT	0.9/1.1	0.8/1.2	0.6/1.4
Time(s)	8.065	8.019	8.363

Similarly, we fix *inITT* and *deITT*, and test the file transfer latency of NCFTP in the different *congestion* and *lower*. The results are given in Table 5.2. Then, we choose 8 and 4 as *congestion* and *lower*.

5.2.2 WT mechiansm

When *WT* is too short, the number of received packets of the second half is much less than the first half. We consider *WT* too short to let packets lost, and *WT* adjusts longer to wait for packets. If the received number of the first half and the second half are almost the

Table 5.2: The initial condition experiment of ITT

lower/congestion	3/6	4/8	5/10
Time(s)	8.303	8.161	8.557

same, it means that WT can become shorter to reduce the unnecessary waiting time. Eq. (5.2) is what we used to adjust WT in our implementation. Let fh, sh, $inWT$, $deWT$, $maxWT$, and $minWT$, respectively, denote the number of received packets in the first half, the number of received packets in the second half, increasing rate of WT, decreasing rate of ITT, maximum waiting time, and minimum waiting time.

$$WT = \begin{cases} \min(inWT \times WT, maxWT) & \text{if } fh > sh \times 2 \wedge WT < maxWT; \\ \max(deWT \times WT, minWT) & \text{if } sh > fh \times \frac{3}{4} \wedge WT > minWT; \end{cases} \tag{5.2}$$

Table 5.3: The initial change rate experiment of WT

deWT/inWT	0.9/1.1	0.8/1.2	0.7/1.3	0.6/1.4	0.5/1.5
Time(s)	8.56	8.314	8.411	8.518	9.14

We test the file transfer latency in different $inWT$ and $deWT$, and fix the other parameters of Eq. (5.2). The results are given in Table 5.3. So, we choose 1.2 and 0.8 as $inWT$ and $deWT$.

5.2.3 WIN mechiansm

When p is bigger than a certain rate, it means that the bandwidth of interfaces may not be fully used. So, WIN can become bigger to send more coding sets into the networks. If p is smaller than the lower bound, it means too many packets inserted into the networks. WIN should become shorter to avoid dropping packets. Eq. (5.3) is what we used to adjust WIN in our implementation. Let p, $frate$, $srate$, $maxWIN$, and $minWIN$, respectively, denote the successful transmission probability of the receiver, the rate of speedup signal, the rate of slow down signal, maximum window size, and minimum window size.

$$WIN = \begin{cases} \min(WIN + inWIN, maxWIN) & \text{if } p > frate \wedge WIN < maxWIN; \\ \max(WIN - deWIN, minWIN) & \text{if } p \leq srate \wedge WIN > minWIN; \end{cases} \tag{5.3}$$

Table 5.4: The initial change rate experiment of WIN

inWIN/deWIN	2/2	4/4	8/8	WIN / (WIN/2)	3WIN / (3WIN/4)
Time(s)	8.053	8.117	8.365	8.313	8.679

Table 5.5: The increasing condition experiment of WIN

fsign	0.95	0.85	0.75	0.65
Time(s)	8.5	8.136	8.077	8.384

We test the file transfer latency of $inWIN$ and $deWIN$, and fix the other parameters of Eq. (5.3). The results are given in Table 5.4. We choose +2 and -2 as $inWIN$ and $deWIN$.

Table 5.6: The decreasing condition experiment of WIN

ssign	0.5	0.6	0.7
Time(s)	8.069	8.156	8.491

We fix $inWIN$, $deWIN$, and $srate$, and test the file transfer latency in different $frate$.The result are given in Table 5.5. Then, we fix $inWIN$, $deWIN$, and $frate$, and test the file transfer latency in different $srate$. The results are given in Table 5.6. We choose 0.75 and 0.5 as $frate$ and $srate$.

5.3 File Latency Analysis

To compare the performance of MFTP, AMFTP, and NCFTP, a file of size 1MB is downloaded from the server. Five transmission scenarios are considered in these experiments, including 1) Download one file via FTP by 3G only; 2) Download one file via FTP by WiFi only; 3) Download one file via MFTP by 3G and WiFi concurrently; 4) Download one file via AMFTP by 3G and WiFi concurrently; 5) Download one file via NCFTP by 3G and WiFi concurrently. We execute each scenario steps 100 times under limited WiFi bandwidth respectively 128KB/s, 192KB/s and 256KB/s. The file transfer latency of each transfer is recorded.

In Fig. 5.6, the average file transfer latency of our proposed protocols are shorter than FTP via WiFi only and 3G only. The average file transfer latency of MFTP, AMFTP, and NCFTP under WiFi with 128KB/s are 11.654s, 11.355s, and 11.074s respectively. The average file transfer latency of MFTP are 57.07% and 52.36% of transfer latency of using WiFi only and 3G only. The ratio of AMFTP are 55.6% and 51.01% respectively. The ratio of NCFTP are only 54.23% and 49.76%. The average file transfer latency and bandwidth are given in Table 5.7.

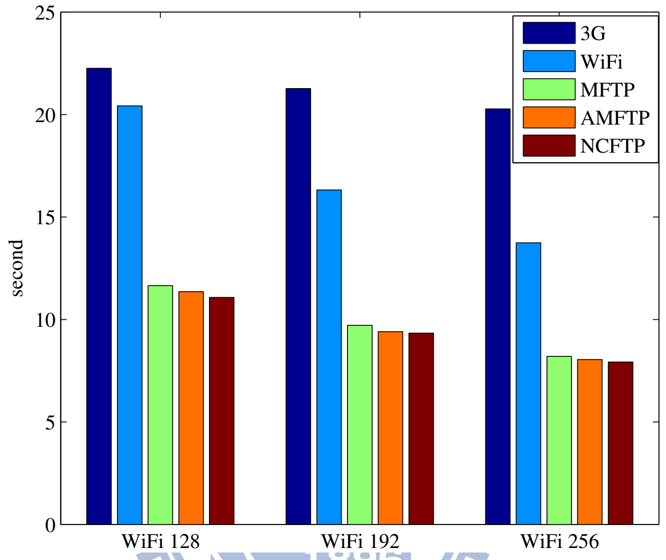

Figure 5.6: The average file transfer latency of five scenarios

We can see the performance of AMFTP better than MFTP, and the performance of NCFTP is better than AMFTP. The file transfer latency of NCFTP is 97.43% of AMFTP, and the transfer latency of AMFTP is 97.53% of MFTP. So NCFTP reduce the file transfer latency more than AMFTP. Similar results can be found in the experiments with WiFi bandwidth 192KB/s and 256KB/s. The results verify the performance of our proposed protocols. Our proposed protocols are successful shortened the file transfer latency. When the bandwidth of each interfaces are closed, the latency of our proposed protocols are nearly

Table 5.7: The average latency and bandwidth for a file of 1MB.

WiFi with bandwidth 128KB/s					
	3G	WiFi	MFTP	AMFTP	NCFTP
Time	22.25	20.42	11.65	11.36	11.07
KB/s	46.01	50.15	87.87	90.18	92.46

WiFi with bandwidth 192 KB/s					
	3G	WiFi	MFTP	AMFTP	NCFTP
Time	21.27	16.32	9.71	9.40	9.34
KB/s	48.15	62.74	105.38	108.88	109.69

WiFi with bandwidth 256KB/s					
	3G	WiFi	MFTP	AMFTP	NCFTP
Time	20.28	13.74	8.20	8.05	7.93
KB/s	50.51	74.51	124.89	127.26	129.17

the half latency of FTP via WiFi only or 3G only. We can see that there are no big problems for a mobile device to activate both radio interfaces. The bandwidth of MFTP and AMFTP are roughly equal to the sum of the bandwidth of the 3G and WiFi interfaces, but the bandwidth of NCFTP can further improve the performance. In Fig. 5.9, Fig. 5.8, and Fig. 5.7, the distribution of file transfer latency of FTP, MFTP, AMFTP and NCFTP are similar with our proposed math model in Fig. 3.1. It adequately verifies the truth of the math model. We find a phenomenon in Fig. 5.9 and Fig. 5.7. When the bandwidths of interfaces are lower, the improvement of transfer latency of NCFTP is much bigger than AMFTP and MFTP.

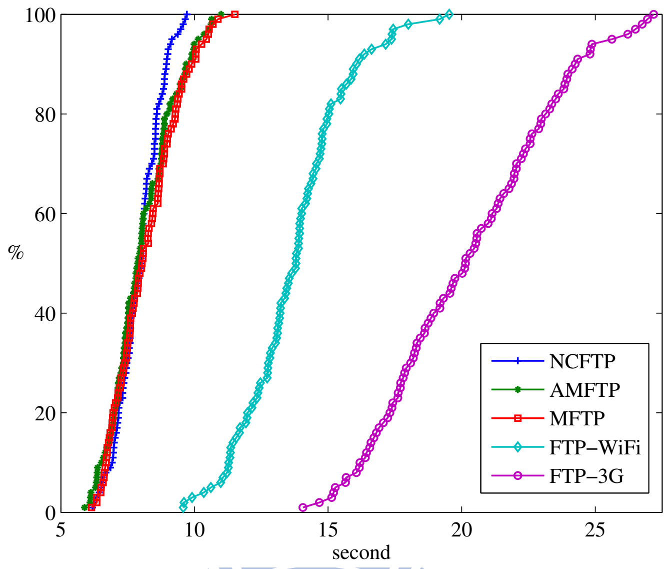

Figure 5.7: c.d.f of file transfer latency under WiFi with 256KB restriction

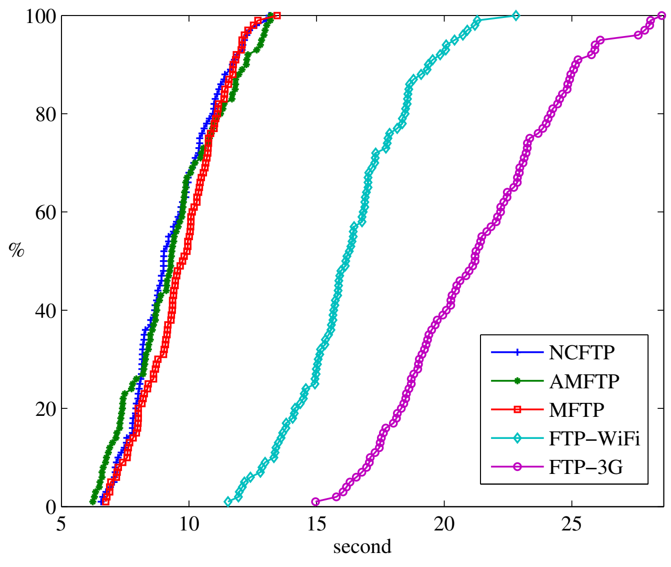

Figure 5.8: c.d.f of file transfer latency under WiFi with 192KB restriction

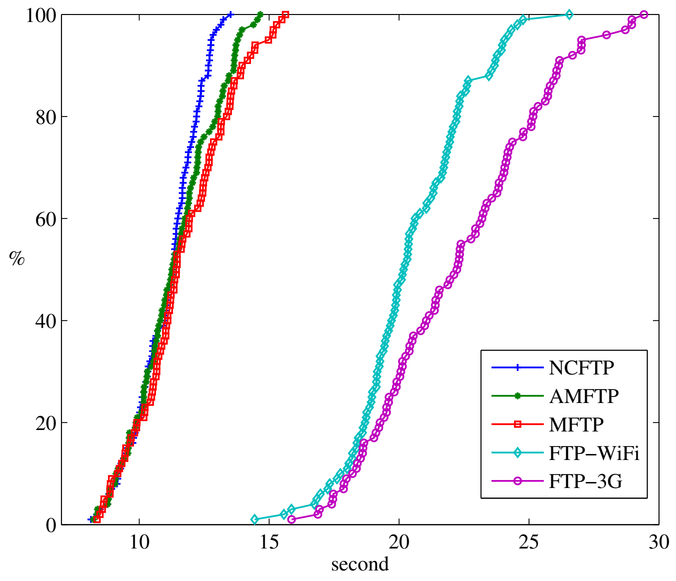

Figure 5.9: c.d.f of file transfer latency under WiFi with 128KB restriction

Chapter 6

Conclusions

In this work, we proposed three file transfer protocols for mobile devices equipped with multiple radio interfaces to reduce file transfer latency by integrating heterogeneous radio resources. We implemented the proposed protocols on a netbook to download files via WiFi and 3G connections, and observed performance improvement. In the experiment results, our proposed protocols are successful shortened the file transfer latency. The file transfer latency of our proposed protocols are almost 60% of transfer latency of FTP via single interface. When the bandwidth of each interfaces are closed, the latency of our proposed protocols are nearly half of transfer latency of FTP via single interface. When the bandwidth MFTP and AMFTP are roughly equal to the sum of the bandwidth of the 3G and WiFi interfaces, but the bandwidth of NCFTP can further improve the performance. The results verify the performance of our proposed protocols. Our proposed protcools are reducing the file transfer latency and raising the bandwidth utility of network interfaces. We also induce the c.d.f math model of practical solution and ideal solution, and verify the truth of the math model. In the future, we hope the proposed protocols be used on file transfer, and we will develop general-purpose protocols not only for file transfer.

Bibliography

[1] [Online]. Available: http://sinwen.com/?p=4231

[2] [Online]. Available: http://www.eettaiwan.com/ART_8800580575_617723_NT_d3c3a65f.HTM

[3] J. Zhang and R. D. McLeod, "A udp-based file transfer protocol with flow control using fuzzy logic approach," in *Electrical and Computer Engineering, 2003. IEEE CCECE 2003. Canadian Conference on*, vol. 2, no. 2, 4-7 May 2003, pp. 827–830.

[4] K. Sklower, B. Lloyd, G. McGregor, D. Carr, and T. Coradetti, "The PPP multilink protocol (MP)," August 1996. [Online]. Available: http://www.rfc-editor.org/rfc/rfc1990.txt

[5] X. G. Wang, G. Min, J. Mellor, and K. Al-Begain, "A qos-based bandwidth management scheme in heterogeneous wireless networks," *International Journal of Simulations*, 2004.

[6] Y. Choi and S. Choi, "Service charge and energy-aware vertical handoff in integrated ieee 802.16e/802.11 networks," in *INFOCOM 2007. 26th IEEE International Conference on Computer Communications.*, 6-12 May 2007.

[7] K. J. Peng and Z. Tsai, "Distortion and cost controlled video streaming in a heterogeneous wireless network environment," in *The 17th Annual IEEE International Symposium on Personal Indoor and Mobile Radio Communications*, 2006, pp. 1–5.

[8] K. Chebrolu and R. Rao, "Bandwidth aggregation for real-time applications in heterogeneous wireless network.pdf," in *Mobile Computing, IEEE Transactions on*, 2006, pp. 388 – 403.

[9] K. Evensen, D. Kaspar, P. Engelstad, A. F. Hansen, C. Griwodz, and P. Halvorsen, "A network-layer proxy for bandwidth aggregation and reduction of ip packet reordering," in *Local Computer Networks 2009. LCN 2009. IEEE 34th Conference on*, 2009, pp. 585 – 592.

[10] J. C. Fernandez, T. Talebt, K. Hashimoto, Y. Nemoto, and N. Kato, "Multi-path scheduling algorithm for real-time video applications in next-generation wireless networks," in *Innovations in Information Technology 2007. IIT 07. 4th International Conference on*, 2007, pp. 73 – 77.

[11] D. Kaspar, K. Evensen, A. F. Hansen, P. Engelstady, P. Halvorsen, and C. Griwodz, "An analysis of the heterogeneity and ip packet reordering over multiple wireless networks," in *Computers and Communications ISCC 2009 IEEE Symposium on*, 2009, pp. 637 – 642.

[12] R. Ahlswede, N. Cai, S.-Y. R. Li, and R. W. Yeung, "Network information flow," *IEEE Transactions on Information Theory*, vol. 46, no. 4, pp. 1204–1216, Jul 2000.

[13] S.-Y. R. Li, R. W. Yeung, and N. Cai, “Linear network coding,” *IEEE Transactions on Information Theory*, vol. 49, pp. 371–381, February 2003.

[14] S. Fong and R. Yeung, “Variable-rate linear network coding,” in *Information Theory Workshop (ITW ’06)*, Oct. 2006, pp. 409–412.

[15] R. Koetter and M. Médard, “An algebraic approach to network coding,” *IEEE/ACM Transactions on Networking*, vol. 11, no. 5, pp. 782–795, Oct. 2003.

[16] T. Ho, R. Koettert, M. Médard, D. R. Karger, and M. Effros, “The benefits of coding over routing in a randomized setting,” in *IEEE International Symposium on Information Theory (ISIT 2003)*, 29 June-4 July 2003, p. 442.

[17] S. Katti, H. Rahul, W. Hu, D. Katabi, M. Médard, and J. Crowcroft, “Xors in the air: Practical wireless network coding,” *IEEE/ACM Transactions on Networking*, vol. 16, no. 3, pp. 497–510, June 2008.

[18] H.-P. Wang, Y.-T. Chuang, C.-W. Yi, Y.-C. Tseng, and P.-C. Liu, “Xor-forwarding for wirelss networks,” in *Global Telecommunications Conferences, (IEEE GLOBECOM 2009)*, 30 November - 4 December 2009.

[19] C. Gkantsidis and P. Rodriguez, “Network coding for large scale content distribution,” in *Proceedings of the 24th Annual Joint Conference of the IEEE Computer and Communications Societies (IEEE INFOCOM 2005)*, vol. 4, March 2005, pp. 2235–2245.

[20] [Online]. Available: http://en.wikipedia.org/wiki/Binomial_distribution

[21] [Online]. Available: http://mathworld.wolfram.com/BinomialDistribution.html

[22] [Online]. Available: http://en.wikipedia.org/wiki/Negative_binomial_distribution

[23] [Online]. Available: http://mathworld.wolfram.com/NegativeBinomialDistribution.html

[24] E. Furman, "On the convolution of the negative binomial random variables," *Statistics & Probability Letters*, vol. 77, no. 2, p. 169V172, 15 January 2007.

[25] C.-L. Hwang and S.-Q. Li, "On input state space reduction and buffer noneffective region," in *Proceedings of the 13th Annual Joint Conference of the IEEE Computer and Communications Societies (IEEE INFOCOM '94)*, vol. 3, 12-16 June 1994.