DCCP 是否能完全取代 UDP？

連耀南　　丁諭祺
國立政治大學資訊科學系
{lien, g9714}@cs.nccu.edu.tw

摘要

數據擁塞控制協議(Datagram Congestion Control Protocol, DCCP)的設計初衷在於加入壅塞控制機制，以取代 UDP 成為不可靠傳輸的主流協議，期望能減低網路的壅塞情形。但網路應用程式從 UDP 改為使用 DCCP 傳輸資料，是否能維持原本所需的傳輸效能則是一個值得探討的課題。本研究利用 NS-2 網路模擬器，模擬 DCCP 協定下的網路電話與多條 TCP 連線競爭網路頻寬；我們的研究顯示，當網路中存在多條 TCP 連線時，DCCP 將競爭不到足夠網路頻寬，面對 TCP 的頻寬競爭，DCCP 下的網路電話將無法有穩定的表現。

關鍵詞：DCCP、TCP、UDP、VoIP。

Abstract

DCCP is proposed to replace UDP for its ability of congestion control while maintaining its promptness by ignoring lost packets as UDP does. The network would suffer less congestion. However, whether the applications that switch from UDP to DCCP can maintain their needed performance or not is a big question. This paper investigates this problem by evaluating DCCP based VoIP vs. a variety of TCPs using NS-2 simulator. Our study shows that DCCP has a disadvantage in competing network bandwidth with existing TCPs. DCCP based VoIP may perform poorly in facing the competition of TCP.

Keywords: DCCP, TCP, UDP, VoIP

1. 前言

大多數的網路應用程式都希望能取得適當且公平的網路頻寬來完成它們的工作，另一方面，網路中的路由器等各個網路環節也盡力的避免造成網路壅塞的情形發生。傳輸層傳輸協議除了負責在傳送端和接收端之間傳輸資料之外，對於傳輸速率的控制也扮演了相當重要的角色，目前的網路應用程式多半使用 UDP 和 TCP 這兩個傳輸協議來傳遞資料。

UDP 定義於[RFC 768]，是一種 IP 網路上非連線導向的傳輸協議，不保證資料能夠完整的送達，因此傳輸的速度較快，但缺乏可靠性。在 UDP 中沒有確認封包、重送機制和連線逾時的概念，當資料被送出之後，發送端無法得知資料是否成功的被送達目的地；所以 UDP 無法測量傳送端與接收端之間的頻寬大小，更不可能藉由調整自身的發送速率來適應網路環境。因此 UDP 的傳輸速率多半是在傳輸之前先行設定，在整個資料的傳輸過程中不再改變。講究時效性的網路應用程式多使用 UDP 作為傳輸協議，因為封包延遲遠比封包遺失所造成的影響更為嚴重。

TCP 定義於[RFC 793]，是一種 IP 網路上連線導向的通訊協定，提供可靠的資料傳輸。由於硬體的先天條件限制，在網路中傳輸的封包都有可能遺失，為了保證 TCP 送出的封包能完整送達，接收端必須發送 ACK 封包來做確認，所以 ACK 和重送機制是 TCP 能提供可靠的封包傳輸最重要的機制。

另一方面，壅塞控制在 TCP 中扮演了調整傳輸速率避免網路壅塞的重要角色。大部分的 TCP 是以滑動視窗為基礎的傳輸協定，它藉由調整視窗的大小來做流量的控制，並利用收到 ACK 與否來判斷是否傳輸成功。其擁塞控制主要是藉由擁塞視窗(Congestion Window)來做速度的控制，發送端可以透過 ACK 封包來推斷傳送端與接收端之間的網路情況來改變擁塞視窗的大小。

TCP 在兩個終端節點之間提供可靠的資料傳輸，但它無法即時得知整體網路情況；唯一的方法是藉由封包傳遞時間與封包傳輸的成功與否來推測整體網路的情況，所以不同的 TCP 版本都透過這些資訊來估計兩個終端節點之間的可用頻寬，並適當的調整傳輸速率。然而許多因素會影響估計網路頻寬的準確性，例如網路的穩定性和節點之間的路徑長度，所以大多數的 TCP 版本都難免要不斷的調整傳輸速率來適應網路環境，不幸的是不同版本的 TCP 無法公平地分享網路頻寬。越快或越準確偵測到網路壅塞情形的 TCP 版本所能競爭到的網路頻寬越小。以 TCP Vegas 為例，其調整傳輸速率較其他的 TCP 版本快速且準確，但卻無法在網路世界立足[2]。

在競爭網路頻寬的課題中，UDP 應該為此負起最大的責任，因為它無法調整傳輸速率來適應網路之變化。當 UDP 的使用量僅占整體網路的一小部分時，尚可容忍；但近年來，多媒體串流網路服務的用量激增佔據大量網路頻寬，例如網路實況轉播、網路電台、網路電話等服務，UDP 這種極具頻寬侵略性的行為已不容忽視，因此 DCCP 這種具有壅塞控制機制的傳輸協議被提出，期望取代 UDP 成為不可靠傳輸的主流協議[4]。

進行重大變革之前必先釐清一個問題，DCCP 真能取代 UDP 嗎？是否有些尚未發現的隱憂？本研究利用 NS-2 模擬器進行實驗來回答這個問題；首先以網路電話為例，在多條 TCP 數據流的競爭之下，使用 DCCP 傳輸網路電話測試是否能分享到公平的頻寬，另外測試以 DCCP 為基礎的網路電話是否能維持通話品質。

2. 相關研究

2.1 DCCP

DCCP 定義於[RFC 4340]，是個連線導向的傳輸層傳輸協議[4]。DCCP 包含了連線的建立與拆除、ECN、壅塞控制等，它提供了不同的壅塞控制方式供使用者選擇，但並不包含重送機制，是一個不可靠的傳輸協議。DCCP 和 TCP 相當類似，透過 ACK 封包來確認資料是否送達接收端，ACK 訊息中包含了發送端送出的封包是否到達目的地，以及標頭中是否含有 ECN 標記。在連線建立的時候，使用者可透過壅塞控制編號(CCID)選擇不同的壅塞控制機。

2.2 TCP 的壅塞控制

由於網路並非完美無缺，少部分的封包可能在傳遞的過程中遺失。一般來說網路壅塞是造成封包遺失最大的原因，此時傳輸協議必須要盡快做出反應降低傳送速率以舒緩網路的壅塞。

TCP 使用遞增倍減(AIMD)的方式調整傳輸速率，另外還有慢啟動、壅塞迴避等狀態來避免網路壅塞[9]，這兩個方式最早出現在 TCP Tahoe 和 Reno 版本中[5]。

TCP Tahoe 和 Reno

在每條 TCP 建立的連線之中，發送端利用壅塞視窗之控制來限制兩個終端節點之間未經確認的封包總數上限，亦即已送出但尚未到達的封包總數之上限，藉此來控制傳輸速率。當接收端收到一個封包，便會回覆一個 ACK 給發送端，告知目前所收到最新的封包序號；當發送端收到 ACK 之後，便再送出一個封包，壅塞控制窗口內的未被確認封包總數維持於窗口上限。若接收端收到跳序的封包時，它會送回一個 duplicate ACK 給傳送端，藉以回報異常狀況。

TCP 在連結建立的開始或是逾時產生後會進入慢啟動機制[11]。接收端每收到一個封包，便回送一個 ACK，而傳送端在接收到每一個 ACK 後便增加擁塞視窗之大小，以倍數的方式遞增直到產生封包遺失，當接收端發生連線逾時，CWND (Congestion Window)會降至初始值，再以倍數成長至 threshold，此時便進入擁塞避免階段。

此因既已偵測到傳輸速率之極限，理當調整減緩 CWND 的擴張速度。當 CWND 增長至 threshold 之後每接到一個 ACK 後 CWND 只增加一格，這個過程持續不斷的進行，直到擁塞產生後，縮減 CWND 並將 threshold 減半。而 TCP Tahoe 和 Reno 對於收到 duplicate ACK 有著不同的處理方式。

TCP Tahoe 處理收到三個 duplicate ACK 的方式和處理連線逾時相同，而 Reno 則認為等到 Timeout 發生後再進行重傳常會耗時過久，影響效能，因此加上了 Fast Retransmit 機制，當傳送端收到三個 duplicate ACK 後，便假設此重複 ACK 號碼之下一個封包已經遺失，立即重傳該封包，並進入 Fast Recovery，不需等到 Timeout。

在 Fast Retransmit 之後，所進入的狀態可以是擁塞避免階段，而非慢啟動階段，此機制稱為 Fast Recovery，當進入擁塞避免階段時，壅塞視窗會在每個 RTT 一次加一個 Maximum Segment Size (MSS)；倍減是當又發生網路擁塞事件後，擁塞視窗之大小會減半。要是仍然沒有收到 ACK，TCP Reno 將會視為連線逾時，並進入慢啟動狀態。

TCP Vegas

TCP Vegas 利用封包來回時間(Round-Trip Time)來調整 CWND[2]，並使用遞增的方式來增加 CWND 的大小，雖然這種測量網路頻寬的方式較為準確，但相較於其他 TCP 版本，TCP Vegas 對於頻寬的競爭力較為薄弱，所以不受一般人青睞。

Other TCP

NewReno 和 SACK 的出現是用來改善並增進 Reno 的效能[5, 6]。在特殊的網路環境中，例如無線網路，封包遺失的主因不再是網路壅塞，原本的壅塞控制機制反而會造成傳輸速度緩慢，因此有許多不同的 TCP 版本被提出法來改善這些問題。

2.3 DCCP 的壅塞控制

DCCP 目前定義了 CCID2 與 CCID3 兩種不同的壅塞控制方法。

CCID 2: TCP-Like 壅塞控制[7]

CCID2 制定的壅塞控制機制是和 TCP 相當類似的，其中也有幾個重要的狀態，分別是慢啟動(Slow Start)、壅塞迴避(Congestion Avoidance)和重送逾時(Retransmission Timeout)，最大的不同在於沒有重送(Retransmission)機制。如同 TCP 一般，在 CCID2 中利用壅塞視窗(Congestion Windows)配合遞增倍減(AIMD)的演算法來控制傳輸速率，另外 CCID2 也針對 ACK 封包進行壅塞控制。

CCID 3: TFRC 壅塞控制[8]

TFRC 壅塞控制方式不是使用 CWND，而是以(1)所列計算式來估計並控制發送端的傳輸速率，主要透過 RTT(Round Trip Time)與封包遺失(Packet Loss)作為依據，並以計算出的值為限制，控制傳輸速率。

$$X = \frac{s}{R*\sqrt{\frac{2*b*p}{3}} + (t\_RTO*(3*\sqrt{\frac{3*b*p}{8}}*p*(1+32*p^2)))} \quad (1)$$

因不使用 CWND 調整傳輸速率，使得 TFRC 對於網路情況的反應較為緩慢，乍看之下適合用來傳輸網路電話。

2.4 頻寬競爭公平性

雖然大多數傳輸協議都盡力避免造成網路壅塞，但因為各種原因，它們無法在相同的網路環境下公平地分享整體頻寬，某些 TCP 版本對於頻寬的競爭能力較佳，有些版本則較差，造成了頻寬競爭公平性的問題。在研究[1,3]指出，即使是相同的 TCP 版本仍會有不公平的情況發生，取決於發送端和接收端之間的 end-to-end delay。

更糟的是，目前即時性網路應用程式多半使用 UDP 作為傳輸協議，而 UDP 缺乏壅塞控制機制，因此當應用程式的傳輸行為開始後，UDP 便不再改變其傳輸速率。當整個網路中 TCP 的比例較大時，UDP 這種行為是能被容忍的，根據網路發展至今十幾年的經驗，TCP 的壅塞控制機制往往能調整好自身的傳輸速率，避免壅塞崩潰的情形發生。但隨著多媒體網路應用的興起，網路中會有越來越多 UDP 的應用程式，例如 VoIP 和 IPTV 等等威脅到網路的穩定，讓我們不得不重視這項問題。

2.5 VoIP

所謂的 VoIP 就是使用網際網路來傳語音資料的技術，其作法是將語音從麥克風擷取後，由類比訊號轉為數位訊號，經由壓縮演算法封裝為數據封包，透過網路傳輸後接收端將數據封包解壓縮後，再轉換成為類比訊號由喇叭播出。

首先麥克風將聲音轉換成為有強弱幅度變化的類比電壓信號，經由數位轉換成為數位信號，才能由電腦處理，一般取樣頻率為 8000 赫茲。在取樣過後，電腦將數位訊號進行編碼，通常會加入壓縮的動作來減少網路頻寬的消耗，編碼時從取樣後的數位音訊訊號逐次截取一小段，經由編碼器(Codec)編碼，每一段資訊稱為一個訊框，包含 10 毫秒到 30 毫秒的音訊訊號，將一個或多個訊框封裝成為封包後才交由網路傳遞，一般網路電話應用程式會將編碼過後的資料封裝成 UDP 封包，再加上 IP 標頭後成為一個 IP 封包，當一段語音資料封裝成為 IP 封包後，由 IP 封包標頭中所記錄的目的地位址資訊，即能透過網路傳送至目的地。

編碼的訊框包含 10 毫秒到 30 毫秒的音訊訊號，換言之 VoIP 每秒鐘所送出的訊框數量是固定的，就像 CBR(Constant-bit-rate)應用程式一般。此外，使用者對於聲音的延遲是非常敏感的，聲音由嘴巴發出，傳到收聽者的耳朵之間隔，必須控制在 300 至 400 毫秒以下，因此重送遺失的封包並無意義，因為當重送的封包到達接收端時，早已超過了人耳能接受的延遲時間。

在如此嚴格的要求之下，對於發送端來說幾乎沒有時間餘裕將封包儲存於 buffer 中，當封包產生後必須馬上送出封包。在這種條件之下，我們預期透過調整封包發送間隔的壅塞控制機制(此法必須暫緩送出封包)，將無法適用在 VoIP 之上。

3. 實驗

在這個章節中，我們將使用 NS-2 網路模擬器[12]並裝上 Mattsson 所提供的 DCCP 模組[10]來進行實驗，評估 DCCP 在頻寬競爭中的表現。這些實驗結果將解答本論文要探討的問題：DCCP 是否可以取代 UDP 作為即時性網路應用程式的傳輸協議？首先我們將測試 DCCP 是否能和 TCP 公平地分享頻寬(或需求的頻寬)，並測試是否能維持網路電話的通話品質。

3.1 實驗設計

圖 1 為實驗中所使用的網路拓樸，此外在中間的鏈結設置 0.25Mbps 的頻寬與 50 毫秒的延遲模擬越洋的網路連結，用以作為網路的瓶頸點，其他的節點之間設置 10Mbps 的頻寬與 10 毫秒的網路延遲。

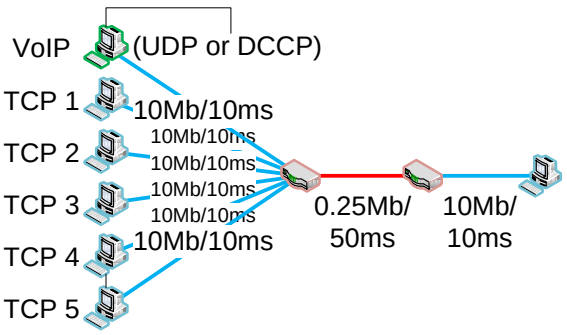


圖 1 實驗網路拓樸

實驗中將模擬 UDP 和 DCCP(CCID3)作為比較。在第一個情境中，我們使用 UDP(或 DCCP)傳輸一通網路電話，每隔 10 秒鐘增加 1 條 TCP 資料流進入網路，直到網路中有 5 條 TCP 資料流；在第二個情境中，實驗開始時網路中就存在 5 條 TCP 資料流，網路電話將在實驗的第 20 秒進入直到第 80 秒結束。這個實驗將使用五種不同版本的 TCP，分別是 Tahoe、Reno、NewReno、SACK 以及 Vegas。

表 1 中列出所有的實驗參數，另外(1)是在 DCCP CCID3 中用來估計 TCP 吞吐率的方程式，其詳細參數列於表 2。

表 1 實驗參數

Parameters	Value
VoIP Packet Size	80 Bytes
VoIP Inter-Packet Interval	30 ms
TCP Versions	Tahoe, Reno, NewReno, SACK, Vegas
TCP Packet Size	1460 bytes
Router Buffer Size	20 packets
Buffer Management Scheme	DropTail
Link Bandwidth	1.5~10 Mbps
Number of VoIP session	1
Number of TCP session	5

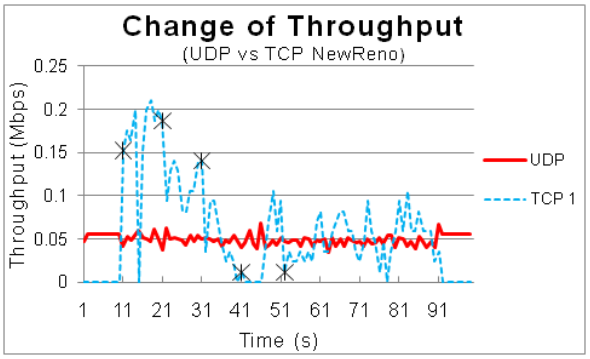
表 2 DCCP CCID3 參數

Parameters		Initial_Value
s	Packet Size	1460 Bytes
R	Round Trip Time	3 sec
P	Loss Event Rate	0-1.0
t_RTO	TCP Retransmission Timeout Value	3 sec
b	# of Packets Acknowledged by a Single TCP ACK	1

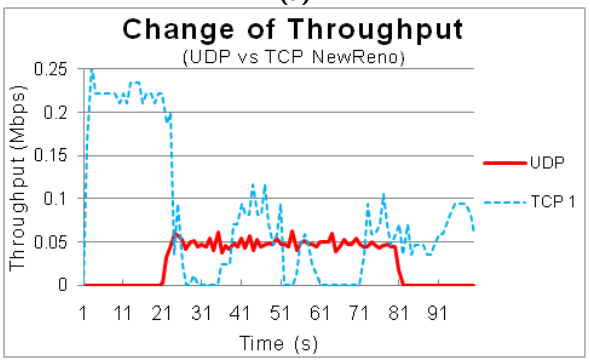
3.2 頻寬競爭

本章節的討論重點在於吞吐率(throughput ratio)的比較。實驗結果發現，UDP 就如我們預期的，在面對各種 TCP 本版的競爭時，都能維持其吞吐率，在圖 2 裡顯示了 UDP 與第一個 TCP 資料流的吞吐率變化，隨著越來越多的 TCP 資料流進入網路中，UDP 依然能維持其吞吐率，但 TCP 1 的吞吐率下降，這表示 TCP 會調整自身的傳輸速率，以適應網路壅塞，但 UDP 並沒有這種機制。

圖 3 裡顯示了 DCCP 與第一個 TCP 資料流的吞吐率變化，隨著後續 TCP 資料流的進入，DCCP 和 TCP 1 都受到影響，顯示出 DCCP 調整了傳輸速率，以適應網路壅塞。但我們可以從圖中發現，DCCP 所爭取到的頻寬，遠小於 TCP NewReno，且在 Tahoe 和 SACK 中也發現到這種現象，甚至在只有一條 TCP 資料流的競爭下，DCCP 所爭取到的頻寬也只有初始值的 30%。

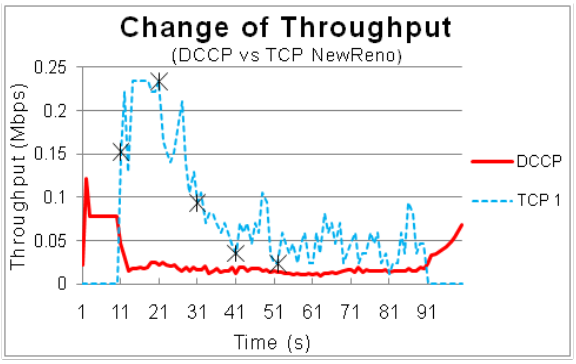


(a)

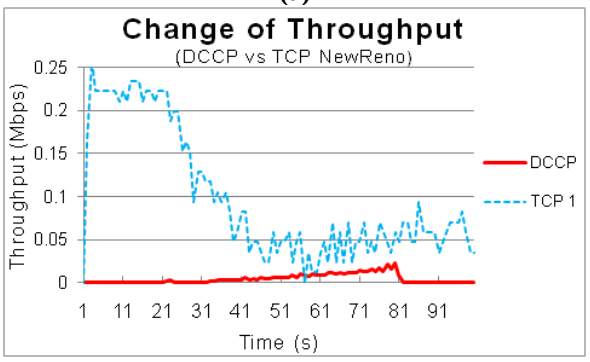


(b)

圖 2 UDP 與 TCP NewReno 吞吐率比較
(a)UDP 先進入(b)NewReno 先進入

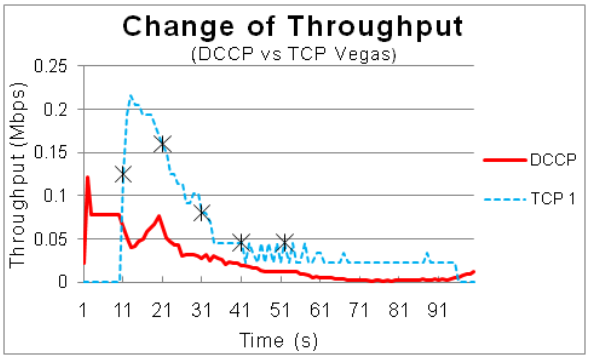


(a)

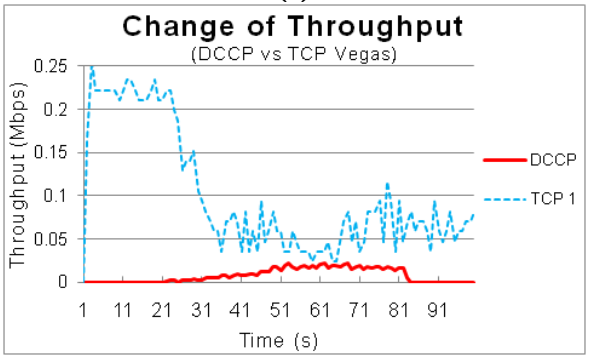


(b)

圖 3 DCCP 與 TCP NewReno 吞吐率比較
(a)DCCP 先進入(b)NewReno 先進入



(a)



(b)

圖 4　DCCP 與 TCP Vegas 吞吐率比較
(a)DCCP 先進入(b) Vegas 先進入

另一方面，TCP Vegas 對於頻寬的侵略性較其他版本來得低，DCCP 所受到的影響較小。在圖 4 中顯示，五條 Vegas 資料流進入網路後，DCCP 所受的影響越來越大，在只有兩條 Vegas 進入網路時，以 DCCP 為基礎的 VoIP 沒有受到嚴重影響，但 VoIP 必須調整其傳輸速率(語音封包大小)來維持通話品質。

總之當 DCCP 面對 TCP Tahoe、Reno、NewReno 和 SACK 的頻寬競爭之下都處於劣勢。表 3 中列出以 DCCP 為基礎的 VoIP 串流在 10-20 秒、21-30 秒、31-40 秒和 41-50 秒之間的平均吞吐率，其中基準值(100%)是在 0-10 秒間網路沒有 TCP 資料流存在網路中的時候。

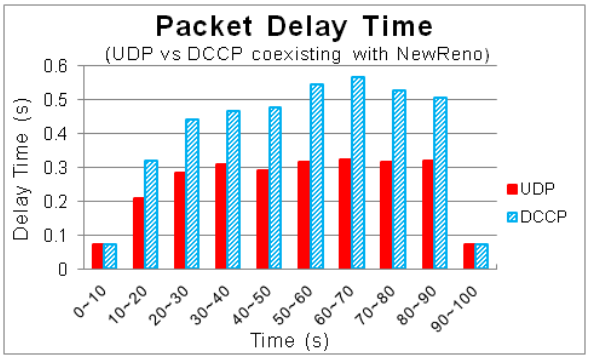
表 3　DCCP 吞吐率(DCCP 先進入,單位%)

Period(sec)	10-20	20-30	30-40	40-50	50-60
vs. Tahoe	25.12	25.23	25.47	21.61	16.59
vs. Reno	29.67	32.01	23.01	16.94	20.21
vs. NewReno	29.79	25.82	20.68	21.26	15.89
vs. SACK	21.73	17.64	17.76	15.77	16.12
vs. Vegas	72.78	52.45	33.18	19.63	12.62

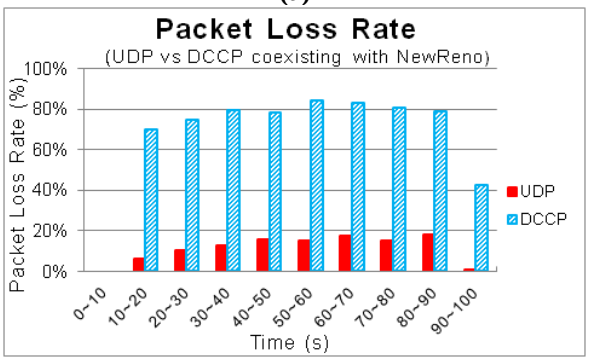
3.3　VoIP 通話品質

本節中討論 VoIP 的通話品質。在圖 5 與圖 6 中顯示，以 UDP 為基礎的 VoIP 通話品質不受影響；而 DCCP 會因越來越多的 TCP 資料流進入網路，通話品質越來越糟。另外表 4 也列出 DCCP 的封包延遲和封包遺失率，其中封包遺失包含送達接收端但時間超過 VoIP 所能容忍的範圍。以 UDP 為基礎的 VoIP 能維持 300 毫秒以下的封包延遲，且在 TCP 競爭下依然能為 20%以下的封包遺失率；DCCP 相較之下就顯得遜色許多，對於 TCP 資料流的競爭，封包延遲時間越來越長，封包遺失率越來越高，甚至超過了 80%，而且當只有一條 TCP 資料流時，吞吐率降至初始值的 30%。

DCCP 在面對 TCP Vegas 的競爭下，情況稍好一點，DCCP 為基礎的 VoIP 在一條 Vegas 的競爭下勉強能維持通話品質，封包延遲時間保持在 200 毫秒下、封包遺失率維持在 27%以下。

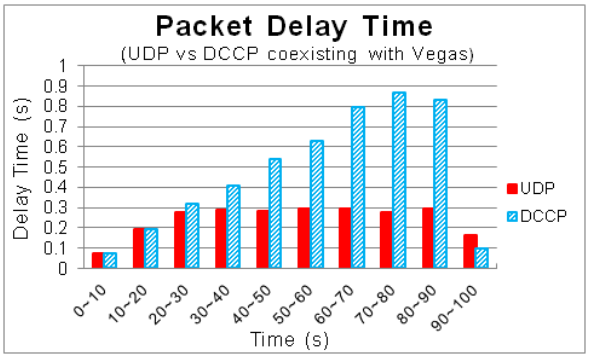


(a)

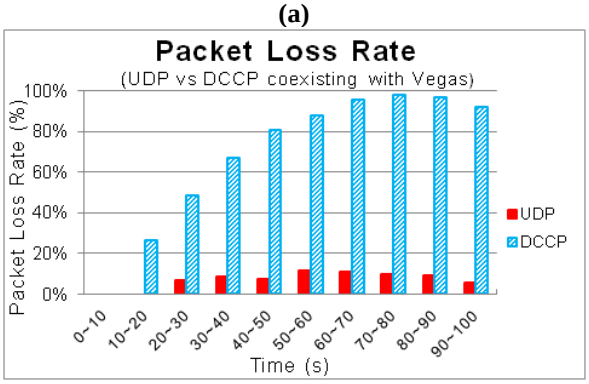


(b)

圖 5　VoIP 通話品質:UDP 和 DCCP 與 NewReno 競爭(a)平均封包延遲時間(b)平均封包遺失率



(a)



(b)

圖 6　VoIP 通話品質:UDP 和 DCCP 與 Vegas 競爭 (a)平均封包延遲時間(b)平均封包遺失率

表 4　DCCP 為基礎的 VoIP 通話品質

Period (sec)	0-10	10-20	20-30	30-40	40-50	50-60
vs. Tahoe						
Throughput Ratio(%)	100	25.12	25.23	25.47	21.61	16.59
Delay Time(ms)	73	395	415	424	470	548
Loss Rate(%)	0	74.86	74.86	75.32	78.09	83.85
vs. Reno						
Throughput Ratio(%)	100	29.67	32.01	23.01	16.94	20.21
Delay Time(ms)	73	319	343	438	498	497
Loss Rate(%)	0	70.13	68.40	77.97	82.81	79.82
vs. NewReno						
Throughput Ratio(%)	100	29.79	25.82	20.68	21.26	15.89
Delay Time(ms)	73	322	441	468	477	545
Loss Rate(%)	0	70.01	74.97	79.58	78.32	84.54
vs. SACK						
Throughput Ratio(%)	100	21.73	17.64	17.76	15.77	16.12
Delay Time(ms)	73	443	490	513	536	554
Loss Rate(%)	0	78.43	82.35	82.70	83.97	84.08
vs. Vegas						
Throughput Ratio(%)	100	72.78	52.45	33.18	19.63	12.62
Delay Time(ms)	73	194	321	412	543	627
Loss Rate(%)	0	26.64	48.79	67.36	80.85	87.89

4. 結語

DCCP 是否能取代 UDP 是個大問題，本研究透過 NS-2 模擬器模擬 DCCP 在各種 TCP 版本之下的表現。研究結果顯示，DCCP 與目前常見的 TCP 版本在頻寬競爭的議題下尚有值得改進的地方。以 DCCP 為基礎的 VoIP 無法維持其傳輸效能，甚至於在 TCP 版本中侵略性最低 Vegas 之競爭下也無法有令人滿意的表現。雖然實驗結果無法代表真實網路中的所有情況，但在我們的實驗拓樸是一個極具典型的越洋網路電話環境，這是目前網路電話最常見的情境。

可變速率的 VoIP 是個不錯的方法之一，此方法需要應用程式的配合來改變語音封包的大小，藉此調整傳輸速率，使用 DCCP 搭配可變速率 VoIP 或許是個值得研究的議題。

打造一個具有壅塞控制機制的傳輸協議供即時性網路應用程式使用，其設計難度遠大於一般非即時性的網路應用程式，因為除了平均吞吐量之外，它必須考量到傳輸時間；這種壅塞控制的關鍵之處可能在於透過應用程式來調整封包大小。

參考文獻

[1] C. Albuquerque, B.J.Vickers, and T. Suda, “Network border patrol: Preventing congestion collapse and promoting fairness in the Internet”, IEEE-ACM Transactions on Networking, vol. 12, no. 4, Dec. 2004, pp. 173-186.

[2] L. S. Brakmo, S. W. O’Malley, and Larry L. Peterson. "TCP Vegas: New Techniques for Congestion Detection and Avoidance, " Proc. Of ACM SIGCOMM, pp. 24-35, Aug. 1994.

[3] D. Chiu and R. Jain, "Analysis of the Increase and Decrease Algorithms for Congestion Avoidance in Computer Networks," Computer Networks and ISDN Systems, vol.1, pp. 1-14, 1989.

[4] Eddie Kohler, Mark Handley, and Sally Floyd, "Designing DCCP: Congestion Control Without Reliability," SIGCOMM’06, Sep., 2006, Pisa, Italy, pp. 27-38.

[5] K. Fall and S. Floyd, "Simulation-based Comparisons of Tahoe, Reno, and SACK TCP," ACM Computer Communication Review, vol. 26, no.3, pp. 5-21, 1996.

[6] S. Floyd and T. Henderson, "The NewReno Modification to TCP's Fast Recovery Algorithm," IETF RFC 2582, 1999.

[7] S. Floyd and E. Kohler, “Profile for DCCP Congestion Control ID 2: TCP-like Congestion Control”, IETF draft-ietf-dccp-ccid2-08, http://www.ietf.org/internet-drafts/draft-ietfdccp-ccid2-10.txt, Mar. 2005.

[8] S. Floyd, E. Kohler, and J. Padhye, “Profile for

DCCP Congestion Control ID 3: TFRC Congestion Control", IETF draft-ietf-dccp-ccid3-11, http://www.ietf.org/internetdrafts/draft-ietf-dccp-ccid3-11.txt, Mar. 2005.

[9] V. Jacobson, "Congestion Avoidance and Control," Proc. of ACM SIGCOMM, pp. 314-329, Aug. 1988.

[10] N. E. Mattsson, "A DCCP module for ns-2", http://epubl.luth.se/1402-1617/2004/175/LTU-EX-04175-SE.pdf, Sep. 2004.

[11] W. Stevens, "TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms," IETF RFC 2001, 1997.

[12] "The Network Simulator - ns-2", http://www.isi.edu/nsnam/ns/.