利用多層編碼配合 DCCP 形成與 TCP 友善的網路電話

中文摘要

壅塞控制是網路管理的重大問題。目前的網路應用程式多半使用 TCP 或 UDP 這兩個傳輸協議來傳遞資料。TCP 具有壅塞控制機制可以隨著網路狀況調整資料傳送速率，但其重送機制所導致的時間延遲不利於時效性的網路服務。而 UDP 的傳輸速率多半是在傳輸之前先行設定，在資料的傳輸過程中不再改變，對於網路壅塞並無任何調節作用，不利於網路之和諧共用。因此 DCCP 這種具有壅塞控制機制的不可靠傳輸協議被提出，期望取代 UDP 成為不可靠傳輸的主流協議。網路電話大部分使用 UDP 作為傳輸層協定，UDP 不具壅塞控制機制，有礙於網路之和諧共用，而且網路的品質也會因為網路壅塞而遺失封包導致品質受損，要達成網路電話在網路壅塞時有壅塞控制能力，且能有好的通話品質，必須要根據網路的狀況調整封包傳送速度或是封包大小。

本研究利用多層編碼配合 DCCP 形成與 TCP 友善的網路電話。壅塞控制必須針對網路中不同程度的壅塞作出適當程度的反應，因此我們改進語音編解碼器，設計出多層語音編碼並設計出可以與之配合的 DCCP，讓 DCCP 依照網路的狀況送出不同層級的語音封包。本研究透過實際網路的實驗環境中評估以 CBR over UDP、Flexible Bit-Rate 以及 Scalable Codec 三種方式傳輸網路電話封包的效能。並也評估 Scalable Codec VoIP 與 TCP 同時存在於頻寬不足的網路中，對於頻寬競爭能力的表現。實驗結果顯示在網路壅塞最嚴重的情況下可以和一般的 CBR 配合 UDP 的方法比較封包遺失率達到 40%左右的改善，語音品質評估指標 MOS 達到 1.5 分的改善，與 Flexible Bit-Rate 方法比較封包遺失率也達到 8%以上的改善，MOS 達到 1 分的改善，讓整體網路狀況更為穩定，並因為封包遺失的減少讓語音品質提升。而在和 TCP 頻寬競爭實驗中可看出，在頻寬不足的情況下，此研究提出的方法可以和 TCP 友善且公平的競爭頻寬。

TCP-friendly VoIP By Scalable Codec over DCCP

Abstract

Congestion control is one of the major problems of network management. Most current network applications use either TCP or UDP to transport data. TCP is equipped with a congestion control mechanism but is not suitable for real-time multimedia applications due to its instability of delay time. On the other hand, UDP fixes its data rate and doesn't change it during the period of transmission even when the network is congested. Under this circumstance, DCCP, which is an unreliable transport protocol but has a congestion control mechanism, is proposed to replace UDP to support real-time network applications such as VoIP. Our previous study showed that a flexible bit rate CODEC to support VoIP over DCCP can effectively control network congestions while maintaining a good voice quality. However, it has an implementation issue yet to be addressed: it requires a bidirectional interaction between DCCP and CODEC.

This thesis proposes to use a scalable CODEC approach to support flexible bit rate VoIP over DCCP. The CODEC sends the entire spectrum of input voice stream to DCCP. DCCP then selects the appropriate voice activation level to compose output stream according to the measured network status, which is feedbacked from the receiver side. The interaction between DCCP and CODEC is avoid. The proposed scheme was evaluated in a real local area network against two other protocols under various VoIP environments, CBR over UDP and Flexible Bit-Rate. Experimental results show that the proposed scheme can outperform CBR over VoIP in the most serious network congestion (under our lab configuration) by 40% in packet loss rate

and 1.5 in MOS. It can outperform Flexible Bit-rate over VoIP by 8% in packet loss rate and 1.0 in MOS. Finally,the fairness test shows that our scheme can coexist with TCP with a fairness index higher than 95% even when network is congested.

誌謝辭

誠摯感謝連耀南教授在研究過程中不辭辛勞的授與研究知識以及提點缺失。衷心的感謝親愛的家人、朋友以及同學，在就讀碩士研究所期間對我的支持、討論以及陪伴。

林耿誠 2013

目錄

圖目錄

表目錄

第一章 緒論

網路應用程式會期望能夠得到足夠的頻寬才能順利的完成其所肩負的任務，但是除了要求足夠的頻寬，也要必須要講求頻寬取得的公平性，如此才能避免網路壅塞的情形發生。網路層許多路由機制以及路由器等網路環節也都盡力避免產生網路壅塞的狀況。傳輸層傳輸協定負責從傳送端傳送封包到接收端，並負責控制封包傳送速率。目前大部分網路應用程式都使用 UDP 或 TCP 做為傳輸協議來傳送封包。

UDP 傳輸協定不具有壅塞控制機制，並且不保證封包是否到達， UDP 的封包傳送率由上層應用程式指定，在傳輸過程並不隨著網路狀況改變，因此其對於網路壅塞沒有調節作用。

TCP 傳輸協定則具有壅塞控制機制，此外具有確認封包、連線逾時、重送、CWND、AIMD 等機制，透過接收端回傳的確認封包可以得知封包是否到達，也可以藉此估算網路狀況；當網路狀況改變時，TCP 會透過壅塞控制的機制改變傳送封包的速率，以配合傳送端和接收端之間的網路負載。因此 TCP 可以避免網路壅塞加劇， 維持網路的和諧共用。

當網路中使用 TCP 做為傳輸協定的網路應用程式佔大多數時，因為 TCP 可以隨著網路的狀況調整傳送封包的速度，可以適度的控制網路壅塞的狀況，因此，TCP 協定在網路和諧共用中扮演一個重要的調節閥作用。但是隨著網路應用越來越多元，許多新的多媒體網路應用程式如雨後春筍般不斷的出現，例如網路

電話以及視訊實況轉播，此類具時效性的網路應用程式大部分是使用 UDP 做為傳輸協定，當網路中使用 UDP 作為傳輸協定的應用程式佔多數時，TCP 的調節功能將大為減弱，網路壅塞將再成為夢魘。

同樣是非連接導向並且有壅塞控制機制的數據壅塞控制協定(DCCP)因而被提出，希望能取代 UDP 成為時效性網路服務所使用的傳輸協定[19]。不過在我們之前的研究中顯示，DCCP 仍有其缺點，DCCP 的頻寬競爭能力無法和 UDP 或是 TCP 公平競爭頻寬，以及 DCCP 使用 Buffer 暫緩傳送封包的方式調整傳送速率也不適合用於時效性的網路應用程式。

本研究主要研究的對象是網路應用程式中的網路電話，希望可以讓網路電話使用具壅塞控制功能的傳輸協定以及在網路頻寬不足時可以維持語音品質，可歸納為以下兩個研究問題：如何設計供網路電話使用的具有壅塞控制功能以及公平競爭網路頻寬的傳輸協定？如何在網路頻寬不足時可以維持較好的語音品質？

1.1 DCCP

隨著網路應用的多元普及，多媒體應用發展盛興，將可能有大量利用網路即時傳送影音資訊的網路程式，因此需要有一套講究即時性並且提供壅塞控制機制的傳輸協議，以避免壅塞崩潰的發生。網際網路工程工作小組(Internet Engineering Task Force, IETF)因此發展出數據壅塞控制協議(Datagram Congestion Control Protocol, DCCP)。DCCP 定義於[18]，是有壅塞控制功能的不可靠傳輸協議，沒有重送機制的設計，在 DCCP 在封包標頭中記錄著連續的編號，藉此偵測封包遺失的情形來達成壅塞控制，另外 DCCP 提供多種壅塞控制機制，在傳輸開始時可由使用者透過壅塞控制編號來做選擇。

1.2 VoIP over DCCP

VoIP 通常使用 UDP 作為傳輸層協定，UDP 不具壅塞控制機制，不利於網路和諧共用，如果讓 VoIP 採用具有壅塞控制功能的 TCP 作為傳輸協定，可以達成讓 VoIP 有壅塞控制功能的目的，但是因為 TCP 保證封包一定都要到達，所以有 ACK 以及重送的機制。但網路電話可以容忍一定程度的封包遺失，且有較嚴格的時效性（延遲時間不得多於 300ms），因此重送遺失的封包，非但可能因失去時效性而作廢，且非屬接收端重建語音所必需，因此不須採用重送機制。

網路電話對於語音抵達時間有一定的要求，語音封包從發送端傳送到接收端的時間要在一定的時間內(例如 300ms)，如果超過此時限，將可能無法達到即時對話互動的目的。由於要達到語音通話的即時性，網路電話的封包傳遞時間是分秒必爭的，因此發送端要盡快的將音訊轉成封包並送出於網路上。因此，目前大部分的網路電話都採用 UDP 作為傳輸協議，但 UDP 使用固定的封包傳送率，沒有壅塞控制機制，有礙於網路之和諧共用。

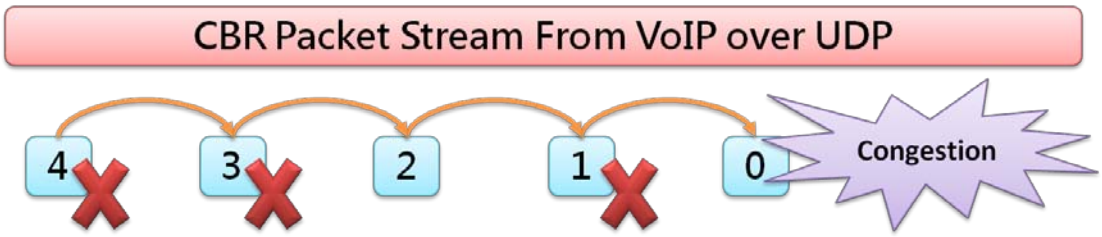

圖 1 網路電話使用 CBR 配合 UDP 在網路壅塞時的狀況

以圖 1 為例，VoIP 使用 Constant Bit-rate(CBR) 的編解碼器並使用 UDP 做為傳輸協定，以固定的速率向網路送出封包。在網路壅塞時，並未依照網路狀況調整速率，會造成封包遺失率增加，影響語音品質。

DCCP 有壅塞控制功能，且不需要保證封包到達，因此可以考慮採用 DCCP 做為網路電話的傳輸層協定。如果改而採用 DCCP 作為網路電話的傳輸層協定，雖然可以達到壅塞控制的目的，但是，是否可以維持網路電話所需的傳輸效能？

以及在網路頻寬不足時時是否可以維持語音通話品質？是本文要研究的問題。

如果 VoIP 使用 Constrant Bit-rate 的編解碼器並配合 DCCP 做為傳輸協定會有以下缺點：

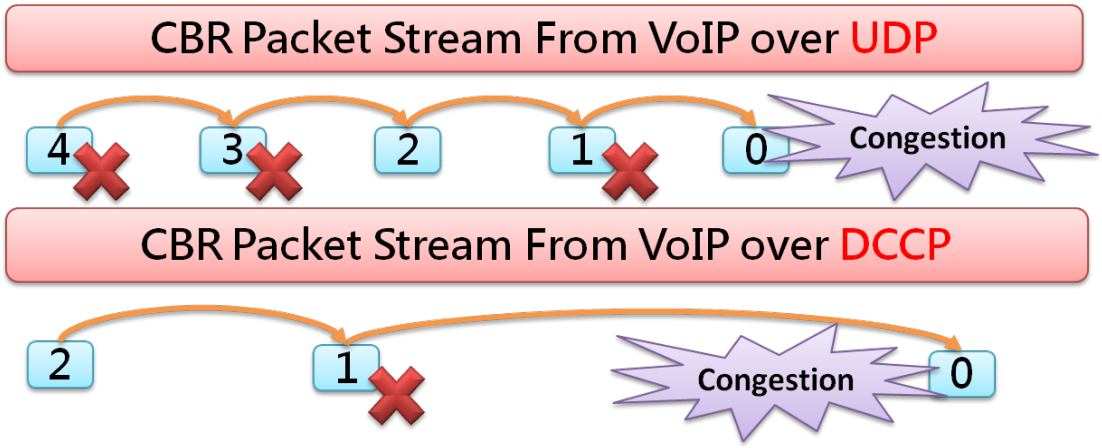

圖 2 VoIP 使用 CBR 並配合 DCCP 的缺點

如圖 2， VoIP 的傳送端在收到音訊後，會盡快封裝成封包送到網路以降低延遲時間。但 DCCP 的壅塞控制機制，是透過調整封包發送間隔來降低傳送速率，當間隔時間多於封包的 Inter-arrived time，語音封包會被暫存於 buffer 之中，稍後再行送出，因而增加了延遲時間，若到達接收端的時間超過一定時限(300ms)，封包就被捨棄，大幅降低語音品質。因此，此種將封包暫存於 Buffer 的機制，有礙於 VoIP 的語音品質，必須改弦易轍。

1.3 VoIP with Flexible Bit-rate

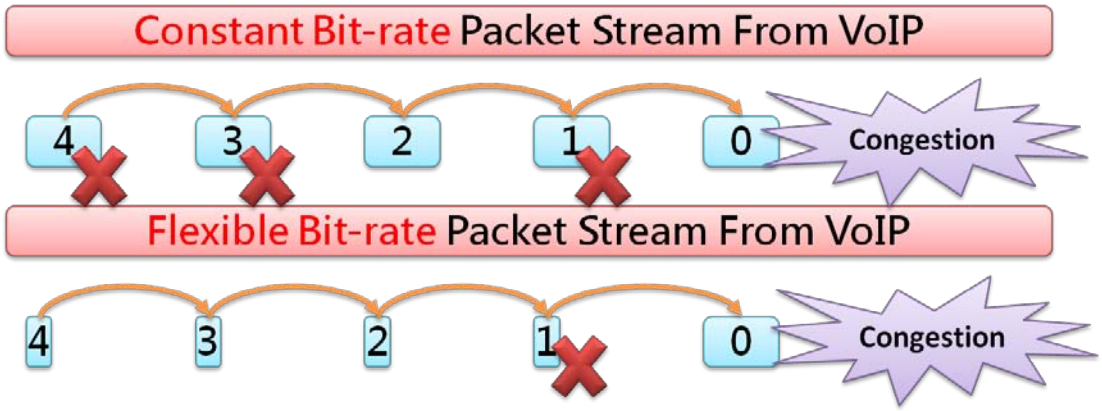

圖 3 VoIP 使用 Flexible Bit-rate 的編解碼器的優點

我們在先前的研究中已提出利用 Flexible Bit-rate 的方式取代一般 VoIP 所使用的 Constant Bit-rate，如圖 3，當 VoIP 改採用 Flexible Bit-rate 的方式時，當網路壅塞發生，不會將封包暫存於 Buffer 中，而是改成降低 Codec 的 Bit-rate 來縮減封包大小，在不拉長延遲時間的情況下，達成降低傳送率的效果，如此，可以減少逾時的封包，並協助減緩網路壅塞的狀況。

1.4 Scalabe Codec over DCCP

欲維持網路電話的品質，必須要根據網路的狀況調整 data rate。此研究希望解決兩個研究問題：設計供網路電話使用的具有壅塞控制功能以及公平競爭網路頻寬的傳輸協定？如何在網路壅塞狀況時可以維持應有的品質？藉由之前章節的討論，我們發現可以考慮採用 DCCP 作為傳輸層協定來達成壅塞控制功能，並且也希望結合 Flexible Bit-rate 的優點，讓 VoIP 在網路壅塞時可以有好的語音品質。但要將 DCCP 配合 Flexible Bit-rate 存在有實際的困難。因此，此研究提出 Scalable Codec 配合 DCCP 的方式。

如圖 4 所示，網路七層架構依層次區分，VoIP 位於 Application 層，之後的層次是 Transport 層、Network 層，DCCP 位於 Transport 層。

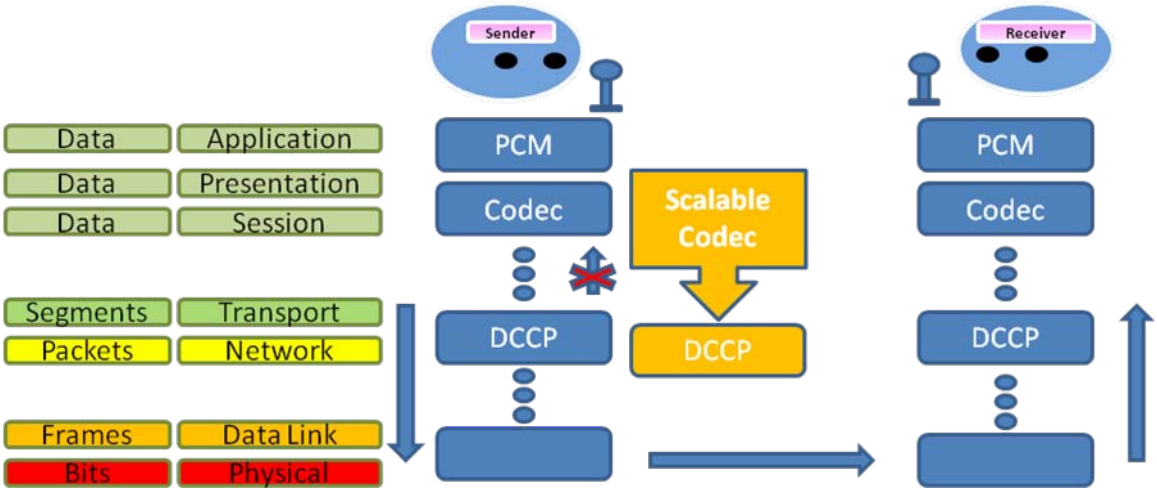

圖 4 VoIP 使用 Scalable Codec 的優點

網路七層資訊的控制是單向的，VoIP 在應用層將語音資訊送給位於傳輸層的 DCCP，但 DCCP 無法向上傳任何控制訊號，讓 VoIP 調節 Codec 的 Bit-rate。Scalable Codec 將編碼的語音封包分層次，將各層的語音封包全部傳給 DCCP，讓 DCCP 挑選要傳送的封包。

1.5 研究目的與方法

本研究將提出利用多層編碼配合 DCCP 的方法，解決此研究探討的兩個主要問題：如何讓網路電話使用具有壅塞控制功能的傳輸協定？如何讓網路電話在網路壅塞的狀況下可以維持好的語音品質？並以實驗的方式驗證所提出方法的效能。

在之前章節的討論中，我們發現網路電話採用 DCCP 作為傳輸層協定來達成壅塞控制功能時，最好搭配 Flexible Bit-rate 來控制資料傳輸率，並維持應有的通話品質，不過因為語音編碼與傳輸協定位於網路七層中不同的階層，且階層之間的傳輸是單向的，而造成 DCCP 知道網路壅塞狀況卻無法告知編碼器做配合的窘境。本研究提出的 Scalable Codec 配合 DCCP 傳輸協定的方法，必須考慮到語音編碼器配合傳輸協定的困難，將語音編碼器的所有層次語音編碼位元都編碼並傳給傳輸層 ，讓傳輸層依照網路狀況選擇適合網路狀況層次的封包之後再做

傳送。如此，可以不需要 DCCP 與編碼器之間的互動。

1.6 論文結構

接下來的章節將照以下安排。第二章介紹壅塞控制以及語音編碼相關背景知識與研究。第三章將介紹 Scalable Codec 和可和其配合的 DCCP 的設計。第四章實驗效能評估。第五章結論與未來研究。

第二章 背景與相關研究

2.1 網路壅塞

在 1986 年 10 月的網路壅塞造成的壅塞崩潰(Congestion Collapse)，讓當時 LBL(Lawrence Berkeley Laboratories)到 UC Berkeley 之間的網路頻寬從 32Kbps 下降到只剩 40bps[17]。發生壅塞崩潰是因為早期的 TCP 版本在壅塞控制的設計上有缺陷，當網路壅塞時，從發送端送出封包後，如果沒有在一定的時間(重送逾時(Retransmission Timeout)內從接收端收到 ACK 確認封包(ACK，Acknowledgement Packet)，就會認為是封包遺失(Packet loss)事件並且啟動重送機制，但發送端沒有在重送逾時時間內接收到來自接收端的確認封包，並非全都是因為封包遺失，而有可能是封包在傳送的過程中遭遇到網路壅塞的情況，使得封包延遲(Packet Delay)時間拉長，早期的 TCP 版本在這些情況下會持續重送封包，在網路壅塞的狀況下傳輸協定應該要降低傳送速率，透過降低傳送速率來減緩網路壅塞的狀況，可是當時的 TCP 版本在網路壅塞時應該降低傳送速率可是卻沒有降低傳送速率，最後使得網路壅塞情況越來越嚴重而造成壅塞崩潰，由此可知好的壅塞控制機制設計對於網際網路是非常重要的[1]。

2.2 壅塞控制

在 1986 年 Nagle 在[22]中提出壅塞控制的概念，由此開始，就一直有關於壅塞控制機制的研究出現,在 1988 年 Jacobson 在[17]中詳細討論了緩啟動(Slow Start）、快速重傳(Fast Retransmit)、壅塞迴避(Congestion Avoidance)等壅塞控制的方法,而緩啟動、快速重傳、壅塞避免等機制是目前最常用的 AIMD(Additive Increase Multiplicative Decrease)壅塞控制方法的基礎。TCP Tahoe、TCP Reno、TCP NewReno、TCP SACK 等傳輸協定都是以 AIMD 壅塞控制方法做為參照準則的，Fall 等人透過模擬分析了這些不同版本的傳輸層協定的壅塞控制機制的效能[7]。

當發生壅塞崩潰的現象後，許多研究學者注意到壅塞控制機制的重要性，並陸續有許多 TCP 壅塞控制機制的版本被提出，而 UDP 這類不可靠傳輸協定也有學者提出 DCCP 這樣有許多版本的壅塞控制機的不可靠傳輸協定，在眾多壅塞控制機制版本中如何判斷壅塞控制機制版本的好壞並選擇到合適的機制是很重要的，壅塞壅塞控制必須要在網路發生壅塞時即時偵測到網路壅塞的狀況，並採取合適的壅塞控制反應。以期望能達到頻寬公平分配以及程式執行品質最佳化的平衡點。

在如今網路普及，資訊技術多元的時代，多媒體網路應用也蓬勃發展，大部分的多媒體網路應用都會以即時串流的方式傳送，應用程式是透過 UDP 傳輸資料，此類網路應用程式不需要 TCP 的可靠傳輸的特性，例如網路電話以及網路實況轉播等等。但由於 UDP 缺乏壅塞控制機制，這些應用程式必須針對網路壅塞的情況自行設計應對方式,也因此慢慢的有針對不可靠傳輸進行壅塞控制的傳輸層傳輸協議被提出。

2.3 TCP 壅塞控制機制

當 TCP 是使用端點到端點的壅塞控制，利用各傳送端依照所能偵測到的網路訊號，判斷網路壅塞程度，並限制將資料流送出的速率。一般而言，網路壅塞造成在網路傳送封包過程的路由器等網路設備的緩衝區無法存放過多的封包，而將封包捨棄，是造成封包遺失的最大原因，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 AIMD 的方式調整傳輸速率，有緩啟動、壅塞迴避等狀態來避免網路壅塞[17]，TCP Tahoe 以及 TCP Reno 是最早開始以這些方式做為壅塞機制的 TCP[7]。

2.3.1 TCP Tahoe 以及 Reno

在 TCP 的連線兩端各擁有接收緩衝區、傳送緩衝區、以及維持許多變數(LastByteSent、LastByteAcked、CWND、RWND)，在壅塞控制方面 TCP 是利用傳送端維護的壅塞視窗(CWND)變數，來限制傳送端已送出但尚未確認是否到達接收端的封包的總數上限，藉由 CWND 來控制傳輸速率。

當 TCP 經過三次握手機制(three-way handshake)建立連線後，TCP 會進入緩啟動機制[25]。緩啟動機制執行的方式是，當 TCP 的接收端收到一個封包，必須要回覆包含收到的封包序號的 ACK 封包給傳送端，在接收端每一次接收到 ACK 封包，會回送一個 ACK 封包，而在傳送端每一次接收到 ACK 封包時，傳送端會將維護的 CWND 變數增加一個 MSS，CWND 增加後可一次傳出更多的封包，而下次收到的 ACK 封包以及增加的 CWND 大小就會倍增，傳送速率會以指數性成長。直到發生逾時(Timeout)表示有遺失事件發生，CWND 會降至初始值，再以指數性成長至 ssthresh (slow start threshold)。

在上述狀況之後，當 CWND 成長至 ssthresh，如果還繼續倍增可能會造成壅塞，因此 TCP 將會進入壅塞避免階段，傳送端會以比較保守的方式增加傳輸速

率，在壅塞避免的狀態下每回封包來回時間(Round-Trip Time，RTT)中只會將 CWND 增加一個 MSS，會改以較緩慢的線性增加傳送速度。直到再度發生逾時事件時，壅塞避免會和緩啟動採取相同的行為，將 CWND 降至初始值，並將 ssthresh 更新為發生遺失事件時 CWND 值的一半。

如果接收端沒有依序收到封包序號，而是收到跳序的封包時，接收端會回覆 duplicate ACK 封包，duplicate ACK 封包是再次傳送前一個封包序號給傳送端，來告知傳送端所收到的封包並不正常。TCP Tahoe 和 Reno 對於傳送端接收到三個 duplicate ACK 以不同的方式處理。TCP Tahoe 在接收到三個 duplicate ACK 所指示的遺失事件之後，會和由於逾時事件所指示的遺失事件一樣，將 CWND 降為初始值並進入緩啟動狀態。而 TCP Reno 收到三個 duplicate ACK 所指示的遺失事件會進入快速復原(Fast Recovery)狀態，將快速重傳(Fast Retransmit) 機制，每收到一次造成 TCP 進入快速復原狀態的遺失區段 duplicate ACK，CWND 便增加 1MSS。當遺失區段的 ACK 抵達時，降低 CWND 後進入擁塞避免階段。但是如果等到逾時所指示的遺失事件發生時仍然沒有收到遺失區段的 ACK，TCP Reno 將會進入緩啟動狀態。

2.3.2 其他版本 TCP

TCP 尚有 Vegas 、 NewReno 和 SACK 等版本。 Vegas 透過觀察 RTT 預測封包有可能遺失，封包 RTT 越長代表網路中路由器壅塞狀況越嚴重，在偵測即將發生封包遺失時預先以線性方式降低速率[3]。在無線網路這類特殊網路環境，網路壅塞可能不是造成封包遺失的主因，壅塞控制機制可能會造成傳輸速度緩慢[11][23]，TCP NewReno 和 SACK 等版本被提出法來改善這些問題[7][8]。

2.4 DCCP 壅塞控制機制

DCCP 是傳輸層傳輸協定，DCCP 實現具壅塞控制的不可靠雙向單點傳播連結。詳細地說，DCCP 不保證封包的送達，提供可靠的握手機制(handshakes)來達成連線的建立與拆除，以及有多個可以選擇的壅塞控制機制。

DCCP 的連結實作上是雙向的，資料可以從連結兩端中任一端傳給另一端。因此也代表資料封包以及確認封包可以同時在連結中的兩個方向傳送。但是，在邏輯上 DCCP 會將連結分為兩個分開的單向連結，稱為半連結(half-connections)。每個半連結又分別定義一端為半連結傳送端(HC-Sender)以及半連結接收端(HC-Receiver)。

DCCP 提供了不同的壅塞控制方式供使用者選擇，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制，但並不會包含重送機制，是一個不可靠的傳輸協議[19]。DCCP 傳送類似 TCP 的 ACK 封包的 DCCP-Ack 或是 DCCP-DataAck 等類型的回饋封包來回饋接收端的訊息，讓傳送端可以透過回饋封包測得網路狀況並做壅塞控制。DCCP-Ack 為純粹的 ACK 訊息的封包類型，DCCP-DataAck 則是傳送 DCCP-Data 時順便夾帶 Ack 訊息的封包類型。DCCP-Ack 的訊息依照不同壅塞控制機制而有不同，可能包含的有接收到的封包序號或是接收端傳送 Ack 前最後收到的封包序號、封包遺失率、ECN 標記等。

DCCP 目前定義了 CCID2(Congestion Control Identifier 2)與 CCID3(Congestion Control Identifier 3)兩種不同的壅塞控制方法：

2.4.1 CCID 2: TCP-Like

DCCP 的 CCID2 壅塞控制機制，又名為 TCP-like Congestion Control，壅塞控制方法與 TCP 類似[9]。和 TCP 相同使用 AIMD(Additive Increase Multiplicative Decrease)的壅塞控制方法，同樣使用緩啟動(Slow Start)、壅塞迴避(Congestion Avoidance)和逾時(Timeout)等機制，同樣維護 CWND、ssthresh 等變數來控制傳送速率。不過 DCCP 設計為不可靠協定，所以和 TCP 不同在於沒有重送(Retransmission)機制。此外 CCID2 也針對 ACK 封包進行壅塞控制。由於 CCID2 是以 AIMD 方式做壅塞控制，會造成傳送速率太突然的改變，因此並不適合需要傳輸速率比較平穩的網路多媒體即時應用程式。

2.4.2 CCID 3: TFRC (TCP-Friendly Rate Control)

DCCP 的 CCID3 壅塞控制機制，又名為 TCP-Friendly Rate Control (TFRC)，是以定義於[12]的 TFRC 傳送速率控制演算法為基礎的壅塞控制機制[10]。因為 CCID3 的設計理念是想要有一個與 TCP 友善，並且可以去除像 TCP 或是 TCP-like 那樣傳輸速率劇烈變化的情形，因此才用 TFRC 壅塞控制機制。TFRC 不是使用 CWND，而是以公式(1)利用測得的網路參數來計算傳送端所應該使用的傳輸速率，網路參數中最主要的兩個是 RTT(Round Trip Time)以及封包遺失率(Packet Loss Rate)，利用計算出的 X 值做為控制傳輸速率的限制。

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

X = 傳送速率，單位為 bytes/second。
s = 平均封包大小，單位為 bytes。
p = 封包遺失率，範圍是 0 到 1.0 之間。
RTT = 封包來回時間，單位為 second。

t_RTO = TCP Retransmission Timeout 時間，單位為 second。
b = 單一個 TCP ACK 所確認的封包數量。

RTT(Round Trip Time)以及封包遺失率(Packet Loss Rate)等網路參數必須透過接收端回傳包含回饋訊息的封包給接收端才能到。TFRC 壅塞控制機制剛開始同樣採用 TCP 的緩啟動狀態，在每個封包來回時間都倍增傳送速率，讓連線快速取得頻寬。直到發生封包遺失事件後，接收端會回傳包含回饋訊息的封包給傳送端，告知封包遺失以及相關網路參數。傳送端就會結束緩啟動狀態，並以得到的網路參數，根據公式(1)限制傳送速率。

TFRC 所傳送的資料封包中會包含序號，接收端接收到封包後透過封包的序號可以判斷封包是否有遺失，並計算封包遺失率，而在接收端傳送回饋封包時夾帶計算出的封包遺失率給傳送端。另外 CCID3 的設計理念是為了有平緩的傳送速率，因此在一個 RTT 以內的封包遺失都視為同一個封包遺失事件。

在計算封包來回時間方面，如果傳送端有記錄每個封包的序號以及其時間戳記(timestamp)，則接收端傳送給傳送端的回饋封包中只需要夾帶最後收到的封包序號，以及該封包到達時和傳送該回饋封包時之間的時間延遲即可。如果傳送端沒有記錄每個封包的序號以及其時間戳記(timestamp)，則需要再傳送資料封包時，也傳送傳送時的時間戳記(timestamp)，而接收端傳送回饋封包時再將時間戳記(timestamp)傳送回去。

2.5 網路電話

當聲音經由麥克風擷取後，經由類比/數位訊號轉換器轉換為 PCM 格式的語音數位訊號，PCM 的語音訊號因為資料格式太大，在頻寬有限的情況下，不適合直接藉由網路傳送，因此，通常會將 PCM 格式的語音訊號交由語音編碼器(Encoder)，經由壓縮演算法將語音訊號以音框(Frame)為單位壓縮，再將縮小後的音框加上相關標頭後封裝成數據封包，交由傳輸層傳輸協定傳輸到網路，接收

端傳輸層接收到數據封包後，去除封包標頭後交由語音解碼器(Decoder)解壓縮還原成 PCM 語音格式的數位音訊，之後再透過數位/類比訊號轉換器轉換成類比訊號由喇叭播出，如圖 5 所示。

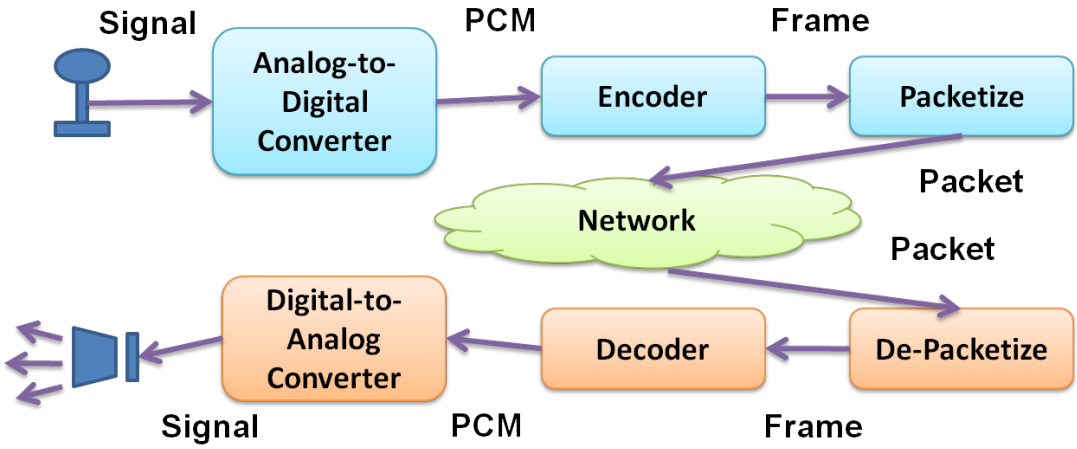

圖 5 VoIP 的語音封包產生流程

2.5.1 聲音取樣以及編碼

發話者震動聲帶產生音波，音波再經過口腔與鼻腔所構成的聲道產生共振而發出聲音，電腦硬體透過麥克風收音，將收到的聲音透過不同強弱幅度變化的電壓訊號表示，必須再將類比的電壓訊號轉換為數位訊號。

為了減少網路頻寬的浪費以及加快傳輸的速度，通常會將所轉換的初步數位訊號透過編碼壓縮後，才傳送到網路。每次的編碼壓縮過程，會從依序取得一段數位音訊訊號，再透過編碼器編碼後，壓縮成較小的訊框(Frame)，通常一個訊框是 10 到 30ms 的語音訊號。編碼類型可分為三種：波形編碼、參數編碼與混合編碼，如表 1 所示。常用於網路電話的 iLBC 編碼器以及此研究所使用的 Speex 編碼器都以混合編碼的 CELP 編碼方式為基礎。

依照不同編碼器所選擇的不同壓縮演算法，壓縮之後的語音品質以及訊框大小會有所差異。通常壓縮率越高，所產生的封包越小，但是音質較差，不過所需的頻寬也可較小。而壓縮率越低，所產生的封包越大，但是音質較佳，不過所需

的頻寬也要比較多。現今常見的語音編碼器有：G.723.1、G.729a、iLBC、Speex[15][16][2][27]等，表 2 是幾個常見編碼器的參數。

表 1 編碼類型

	波形編碼	參數編碼	混合編碼
原理	編碼每個取樣值	使用數學模型 以音框為單位	合成作分析(AbS)
優點	保持原始信號的波形 系統結構簡單 合成音質佳	位元率低	波形編碼的高音質 參數編碼的高壓縮率
缺點	位元率偏高	系統結構較複雜 合成音質差	
舉例	PCM(Pulse-Code Modulation)	LPC(Linear Predictive Coding)	CELP (Code-Excited Linear Prediction)

表 2 各種常見語音壓縮編碼的參數

Codec	Sampling Rate	Frame Size	Bit-rate
G. 723. 1	8kHz	30ms	5. 3/6. 3Kbps
G. 729	8kHz	10ms	8Kbps
iLBC	8kHz	30ms/20ms	13. 3/15Kbps
Speex	8kHz/16kHz	20ms	2. 15~44Kbps

2.5.2 語音封包

網路電話應用程式會將語音編碼編碼出的語音訊框封裝(Packetization)為封包，網路電話應用程式可能自行設計所需要加入的封包標頭，或是透過 RTP 等協定加入 RTP 等協定制定的標頭後，才傳給傳輸層交由傳輸層傳遞。網路電話

在傳輸層通常是使用 UDP 做為傳輸協定，在加入 UDP 標頭後，再傳給網路層。透過 IP 協定包裝為 IP 封包後，透過 IP 資訊可在網路傳送到達目的位置。

2.5.3 影響語音品質的因素

語音品質和通話設備、編碼器或網路狀況等因素有關。不同的通話設備會因為設備的好壞而造成語音品質之差異，而喇叭發出聲音後也有可能再被同一端的麥可風收入產生回音。不同的編碼演算法，也會造成編碼解碼後不同程度的語音失真。因為網路電話是透過網路傳輸，網路的狀況並不像一般電話線路這麼穩定，可能會有封包延遲(Packet Delay)、封包遺失(Packet Loss)、Jitter 等狀況，因為網路的不同狀況也會對於語音有不同程度的影響並降低語音品質。

2.6 網路語音品質評量指標

網路語音品質因其評估的標準較為特殊，目前常見方式可分為主觀（Subjective）與客觀（Objective）兩種。主觀的語音評估方式必須透過受測者實際聆聽，可參考 ITU-T 所制定的 P.800 環境標準及方式，在一定標準的環境中，讓多位受測者聆聽聲音並判定聲音 MOS(Mean Opinion Score)，最後將所有分數平均。客觀的語音評估方式則可參考 ITU-T 所制定的 E-Model。

2.6.1 Mean Opinion Score

目前常見於評估網路語音品質的指標是 ITUT 所制定 MOS(Mean Opinion Score)分數，由高至低分為最好和最壞，可分為 Excellent(5)、Good(4)、Fair(3)、Poor(2)及 Bad (1)等五個等級[13]，如表 3。

表 3 Mean Opinion Score 定義

MOS	Quality	Impairment
5	Excellent	Imperceptible
4	Good	Perceptible but not annoying
3	Fair	Slightly annoying
2	Poor	Annoying
1	Bad	Very annoying

2.6.2 E-Model

ITUT 所制定的 E-Model 是以客觀的方式來評估網路電話品質，語音傳輸過程中可能有許多原因會造成語音品質降低，E-Model 詳細列出可能影響語音品質的項目，並訂定計算公式以及各項目對於語音影響程度的標準[14]。E-Model 透過 R-factor 的公式可以客觀的計算出該語音的分數

$$R = Ro - Is - Id - Ie + A \quad (2)$$

在公式(2)中 Ro 代表沒有網路及設備影響的狀況下只有基本電流訊號轉換相關的基本信噪比。 Is 代表聲音訊號可能會出現的損傷，例如 STMR (Side tone masking rating)、RLR(Receiving loudness rating)等。Id 代表傳輸延遲相關的因素，例如 network delay、codec-related delay 和 playout buffer delay 等。Ie 代表了語音壓縮所造成的語音品質下降和語音透過播放設備放出所造成的失真。A 代表基於對語音質量的期望進行的補償。E-Model 計算出的 R-factor 可再透過公式(3)轉換成 MOS，表 4 列出 R-Factor 與 MOS 之間的分數對應。

$$MOS = 1 + 0.035 * R + 7 * 10^{6} * R\,(R - 60)(100 - R) \quad (3)$$

表 4 R-factors 與 MOS 之對應

R-factor	Quality	MOS
90<R<100	Excellent	4.34 ~ 4.5
80<R<90	Good	4.03 ~ 4.34
70<R<80	Fair	3.60 ~ 4.03
60<R<70	Poor	3.10 ~ 3.60
50<R<60	Bad	2.58 ~ 3.10

2.7 具壅塞控制功能之 VoIP 的相關研究

EVCCM: An Efficient VOIP Congestion Control Mechanism[4]：

該篇研究描述網路電話科技已被廣泛的採用在可以建立網站應用程式溝通的中介系統，例如網路視訊會議系統(WebEx、Gotomeeting)、IP 為基礎的電話服務中心、網路聊天室。如果同時太多使用者登入網路電話服務，網路電話很容易受到網路壅塞影響品質，尤其是同時傳送視訊的電話。網路電話傳送通常是透過 UDP 來實做。因為 UDP 無法確認封包是否到達，因為網路壅塞造成封包遺失、封包延遲，甚至服務中斷，不良的服務品質都會傷害服務提供商的商譽。該篇研究應用賽局理論提出機制的設計，應用在網路電話中介軟體管理來維持壅塞的通話。利用此模型，使用者可以在價格以及品質之間有好的選擇。服務提供商根據使用者的選擇來最大化服務提供商的利益(例如在線使用者的人數、使用費率、系統資源重複利用)，藉此來防止網路壅塞並改善網路使用效能。該篇論文利用使用者選擇所要的價格方案後才作整體的規劃，並沒有依照整體網路狀況來做壅塞控制，並且該篇研究著重於使用者以及提供商之間利益的最佳化，而不是著重於整體的壅塞控制以及頻寬的公平分配。

Adaptive Rate Control for Aggregated VoIP Traffic[24][6]：

該篇研究提出一個設計，可以動態控制採用合適的語音品質來協助壅塞控制，此架構設計在發話方與受話方之間的通信閘(gateway)建立起 XCP(eXplicit Control Protocol)傳輸協議來計算傳送端與接收端之間的網路頻寬。當通話建立時，在傳送端與接收端之間的通信閘(gateway)建立 XCP 連線，並利用 XCP 估計可用頻寬，通信閘將傳送端傳來的語音串流依照 XCP 所估計的頻寬，採用合適的語音壓縮方式進行壓縮後傳送至接收端的通信閘。接收端通信閘接收到壓縮過後的語

音封包後，經過解壓縮再傳給接收端。該篇研究利用 NS2 模擬器模擬結果來證明所提出的設計可以有較高的語音品質以及頻寬利用率，並減少網路壅塞。該篇研究透過在通信閘(gateway) 額外建立的 XCP 來評估網路狀況，並在通信閘做壓縮與解壓縮，因此相對的必須在通信閘有較高的建置成本。

A Mathematical Model of the Skype VoIP Congestion Control Algorithm[21]：該篇研究描述網路電話是日益重要的網路應用程式。該篇研究主要是要推導出 Skype 網路電話的壅塞控制機制的數學模型。該研究分析觀察，當 Skype 要避免網路不穩定時，Skype 的動態傳送速率、平衡點的穩定性以及頻寬利用的效能。結果顯示在網路壅塞時，Skype 的傳送速率以及封包大小會依照封包遺失率而變動，調整到配合所能得到的頻寬。該研究只針對 Skype 對於網路壅塞時的壅塞控制反應提出一個數學模型來觀察是否符合，對於 Skype 實際上是以控制速率或是改變語音封包大小並沒有實際的證實，並且研究實驗只證明模型有符合 Skype 壅塞控制的反應，並沒有研究該壅塞控制在網路壅塞時語音的品質，以及頻寬利用的公平性。

Porting VoIP applications to DCCP[26]:
討論如何讓 VoIP 的應用程式使用 DCCP 通訊協定。以 Linphone 為主將其做修改，讓其可以使用 DCCP 協定。DCCP 連線流程和 TCP 相當類似，可分為三種狀態， Initiation, Data transfer and Termination 。 要將 Linphone 改以 DCCP 協定做傳送，必須做以下三點的修改。1. 因為 DCCP 傳送串流是單向的。因此想要將 Linphone 以 DCCP 的方式傳送 RTP 串流，必須建立兩個 RTP sessions. 2. 此外，DCCP 是連結導向的，所以這和 Linphone 中原本以 UDP 非連接導向的傳送協定不同。Server 中要加入 accept(…)功能，用來等待 client 呼叫 connect(…)來建立連線。 3. socket 建立時必須以 DCCP 為傳送協定，並且傳送的參數也要

從 UDP 類型的參數改為 DCCP 所需的參數。

Congestion Control Enabled VoIP by Flexible Bit-rate[20]：

網路壅塞是在設計通訊協定時必須考慮到的項目，該研究中我們提出 Flexible Bit-rate 的方式達到網路電話壅塞控制。在軟體層接收端測得連續封包接收的時間間隔，一般狀況下將會在固定的時間收到語音封包。如有網路壅塞發生，則間隔時間將不固定，Flexible Bit-rate Algorithm 將會啟動。Flexible Bit-rate Algorithm 分成兩個主要的部分，一個部分是當壅塞狀況發生時依照壅塞狀況去降低 Bit-rate，如圖 7 演算法所示。另一個部分是當壅塞狀況消失時，將會將逐步增加回復 Bit-rate，如圖 8 演算法所示。

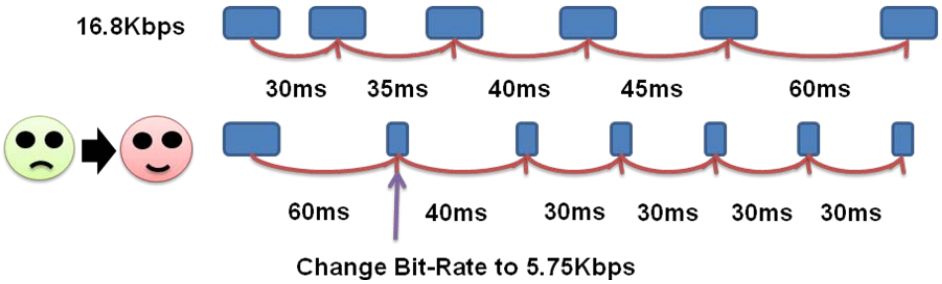

圖 6 調整 bit-rate 以適應不同的網路情況

when receive a new packet at T_i, *Then* $t_{new} = T_i - T_{i-1}$
if $t_{new} \geq \bar{t} + 3\sigma_t$ *and congestion frequency* $\geq \alpha$
    send decrease bitrate command to sender
    congestion frequency $= 0$
else if $t_{new} \geq \bar{t} + 3\sigma_t$
    congestion frequency $=$ *congestion frequency* $+ 1$
else
    congestion frequency $= 0$
    calculate new $\bar{t}$ *and new* σ_t
fi

圖 7 Flexible Bit-rate 降低 - Bit-rate 演算法

```
when sender entry low bitrate state
until [bitrate_now = bitrate_start]
      do
   Sleep(β)
   increase bitrate
done
```

圖 8 Flexible Bit-rate - 提升 Bit-rate 演算法

第三章 Scalable Codec over DCCP

3.1 需求分析與設計目標

即時傳輸的網路應用程式通常在一定的時間間隔傳送封包，現今網路電話會在通話前設定一定的速率，並固定 Bit-rate，通話過程不再改變封包傳送速率或是封包大小。因此在網路頻寬不足時，對網路壅塞沒有任何調節作用，並且會因此有大量的封包遺失。

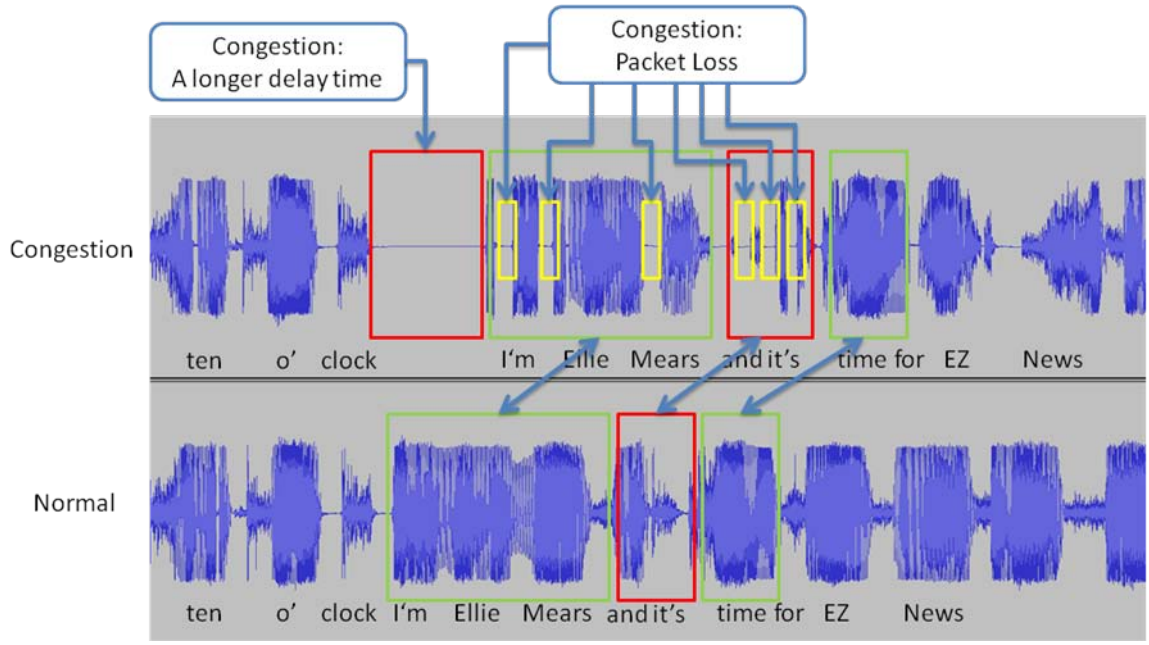

圖 9 網路壅塞時造成造成封包延遲時加拉長或是封包遺失

如圖 9 中所示，當網路壅塞發生時，可能造成封包延遲時間拉長或是封包遺失的狀況，封包延遲時間拉長將會造成電話接收端延後聽到語音，或是因逾時而捨棄封包，而封包遺失會造成通話斷斷續續，並且通話內容部分遺失。由此可

知減緩網路壅塞的狀況對於提升語音品質有很大的幫助。

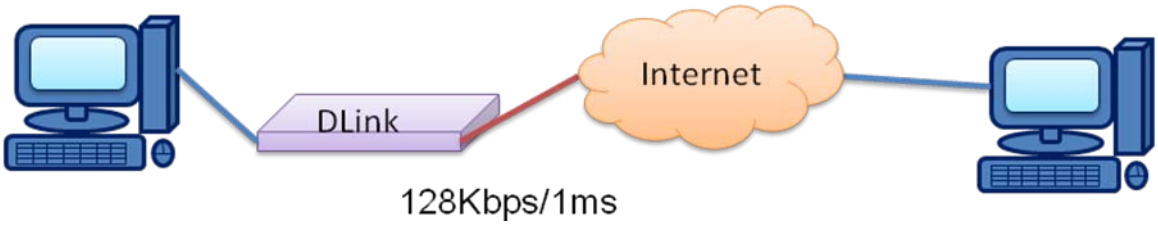

圖 10　觀察網路壅塞時採用不同 Bit-rate 的語音 MOS 變化的實驗網路拓樸

我們觀察在頻寬瓶頸為 128Kbps 的網路連線中，傳送 10 通網路電話，通話長度約為 15 秒，並分別以不同 Bit-rate 傳送語音，觀察通話品質指標 MOS 的變化。如表 5 所示，我們可以看出，在網路頻寬不足時，採用較低 Bit-rate 的語音品質會比採用高 Bit-rate 的語音品質佳。

表 5 在網路頻寬不足時，採用不同 Bit-rate 的語音 MOS 變化。

Sender	Receiver	Bit-rate(Kbps)	Packet Loss Rate	品質(MOS)
451	288	42.2	0.3614	1.475
451	385	27.8	0.1463	2.233
451	428	12.8	0.0509	2.79
451	451	5.75	0	3.289

3.2 Scalable Codec over DCCP 基本觀念

我們提出多層編碼搭配 DCCP 作為傳輸協定，在通話中多層編碼將所有層次的語音編碼都一併傳送給 DCCP，DCCP 再依照網路的狀況傳送合適速率的語音封包，以保持網路電話的語音品質。DCCP 利用網路的 RTT 以及封包遺失率偵測網路壅塞的情況，當網路壅塞時啟動壅塞控制機制，依照網路狀況選擇較合適的語音層次，縮小封包可以舒緩頻寬，等到網路回復順暢時再回復較好的語音，提升通話品質。

3.3 Speex

Speex 是屬於混合編碼類型的語音編碼器，以 CELP(Code-Excited Linear Prediction)的方式編碼語音。CELP 中其中一個主要編碼概念是 Analysis-By-Synthesis(AbS)。AbS 代表的意思是，編碼過程(Analysis)在編碼後會先解碼(Synthesis)，再將解碼結果與原訊號以感知濾波器得到 gain，如此反覆數次，在有限次內盡量最佳化 gain，以此來減少編解碼後與原訊號之間的誤差。CELP 利用線性預測編碼(Linear Prediction Coding，LPC)數學模型表示波形，經過計算得到的數學模型參數會再經過量化(quantization)步驟，利用碼簿(codebook)比對要量化數值與碼簿中相對應的數值，量化後只需紀錄碼簿中位置的編號即可壓縮所需傳送的資料量大小。

由於只以 LPC 模型編碼波形會遺失很多波形的特徵，聲音會過度失真，因此 CELP 除了 LPC 過程外，還會執行音 Pitch Prediction 以及計算 Excitation 等步驟。Pitch Prediction 指的是，由於語音訊號會有週期性的特色，利用此特色可以對於編碼後語音的音高做調整，讓編碼後的語音音高較接近原來的音高。扣除 LPC 以及 Pitch Prediction 以外所無法得到的音波資訊，CELP 將透過計算 Innovation

signal 來記錄，因此 CELP 最後編碼結果中完整的 Innovation signal 會是佔據最多位元的一個項目。

編碼過後會有 LSP、OL pitch、OL pitch gain、OL Exc gain、Fine pitch、Pitch gain、Innovation gain、Innovation VQ、Excitation gain、Excitation VQ 等項目的參數。在不同參數之間以及每個參數位元大小之間做取捨可以得到不同語音品質以及不同訊框大小的語音訊號。

Speex 的 narrowband 模式輸入訊號是採用 8 kHz smapling rate 的音訊格式，也就是每秒取樣 8000 個音波訊號，每個取樣點以 16 位元來記錄。Narrowband 模式每次編碼的 frame size 是 20ms，也就表示每次編碼將使用 160 個取樣點(8000*20/1000)。

Speex 將每次編碼的範圍細分為 frame 以及 sub-frame。以 narrowband 模式為例，一個 frame 包含 160 個取樣點，編碼過程會將一個 frame 再分為 4 個 sub-frame，也就是每 40 個取樣點分為一個 sub-frame。編碼時會先以整個 frame 為單位做編碼分析，之後再細部的以每個 sub-frame 為單位做編碼分析。

以 frame 為單位做編碼分析後所產生的項目有 LSP、OL pitch、OL pitch gain 以及 OL Exc gain。LSP 項目在做量化時分別可量化為 30 bits 以及 18 bits 兩種，30 bits 可用於較好的語音層次，18 bits 可用於較差的語音層次。OL pitch、OL pitch gain 以及 OL Exc gain 則分別為 7 bits、4 bits 以及 5 bits。

以 sub-frame 為單位做編碼分析後所產生的項目有 Fine pitch、Pitch gain、Innovation gain、Innovation VQ。Fine pitch 編碼後大小為 7 bits，而與它相關的 Pitch gain 項目則可量化為 7 bits(用於較好的語音層次)或是 5 bits(用於較差的語音層次)。Innovation gain 則依不同層次大小範圍在 0 到 3 bits 之間。其中大小變化最大的 Innovation VQ 項目，在編碼時是使用 sub-vector 的概念，也就是表示將 sub-frame 再細分 sub-vector 分別做量化。例如，設 sub-vector 的大小為 20 個取樣，就會相對應的使用 32 entries 的 codebook(其索引值大小為 5 bits)，所以每

個 sub-frame 的 Innovation VQ 項目就會以 10 bits 表示。設 sub-vector 的大小為 5 個取樣，就會相對應的使用 256 entries 的 codebook(其索引值大小為 8 bits)，所以每個 sub-frame 的 Innovation VQ 項目就會以 64 bits 表示，詳細的 narrowband 模式位元分配如表 6 所示。

表 6 Speex narrowband mode Bit allocation

Parameter	Update rate	0	1	2	3	4	5	6	7	8
Wideband bit	frame	1	1	1	1	1	1	1	1	1
Mode ID	frame	4	4	4	4	4	4	4	4	4
LSP	frame	0	18	18	18	18	30	30	30	18
OL pitch	frame	0	7	7	0	0	0	0	0	7
OL pitch gain	frame	0	4	0	0	0	0	0	0	4
OL Exc gain	frame	0	5	5	5	5	5	5	5	5
Fine pitch	sub-frame	0	0	0	7	7	7	7	7	0
Pitch gain	sub-frame	0	0	5	5	5	7	7	7	0
Innovation gain	sub-frame	0	1	0	1	1	3	3	3	0
Innovation VQ	sub-frame	0	0	16	20	35	48	64	96	10
Total	frame	5	43	119	160	220	300	364	492	79

Speex 的 wideband 模式輸入訊號是採用 16 kHz smapling rate 的音訊格式，也就是每秒取樣 16000 個音波訊號，每個取樣點以 16 位元來記錄。Wideband 模式每次編碼的 frame size 同樣採用 20ms，也就表示每次編碼將使用 320 個取樣點(16000*20/1000)。

Wideband模式在進行編碼前會先利用Quadrature Mirror Filter(QMF)將輸入訊號分為兩個bands，利用QMF的low pass filter H_0 以及high pass filter H_1 將訊號分為low band以及high band。Low band的訊號依舊維持語音波形的特性，Wideband模式中利用先前所介紹的narrowband的編碼方式編碼low band訊號。High band則因為沒有語音波形的某些特性，和low band使用不同的編碼方式。

Wideband 模式中經由 QMF 分離出的 High band 訊號，因為失去了語音訊號的 pitch 特性，所以經過編碼後的項目只有 LSP、Excitation gain 以及 Excitation VQ，Wideband 模式的 LSP 以及 Excitation 的編碼方式與 narrowband 的類似，詳細的 wideband 模式位元分配如表 7 所示。

表 7 Speex wideband mode Bit allocation

	Parameter	Update rate	0	1	2	3	4	5	6	7	8	9	10
Low band	Wideband bit	frame	1	1	1	1	1	1	1	1	1	1	1
	Mode ID	frame	4	4	4	4	4	4	4	4	4	4	4
	LSP	frame	18	18	18	18	18	30	30	30	30	30	30
	OL pitch	frame	7	7	7	0	0	0	0	0	0	0	0
	OL pitch gain	frame	4	4	0	0	0	0	0	0	0	0	0
	OL Exc gain	frame	5	5	5	5	5	5	5	5	5	5	5
	Fine pitch	sub-frame	0	0	0	7	7	7	7	7	7	7	7
	Pitch gain	sub-frame	0	0	5	5	5	7	7	7	7	7	7
	Innovation gain	sub-frame	1	0	0	1	1	3	3	3	3	3	3
	Innovation VQ	sub-frame	0	10	16	20	35	48	48	64	64	96	96
High band	Wideband bit	frame	1	1	1	1	1	1	1	1	1	1	1
	Mode ID	frame	3	3	3	3	3	3	3	3	3	3	3
	LSP	frame	12	12	12	12	12	12	12	12	12	12	12
	Excitation gain	sub-frame	5	5	5	5	5	5	4	4	4	4	4
	Excitation VQ	sub-frame	0	0	0	0	0	0	20	20	40	40	80
	Total	frame	79	115	155	196	256	336	412	476	556	684	844

3.4 Scalable Codec 設計

我們以 Speex 編碼器[27]為例，進行設計並實驗來驗證本方法之效能，Speex 為開源語音編解碼器，可直接取得其開源碼並再做修改，由先前章節介紹我們了解 Speex 本身也已經有許多不同大小的 bit-rate 可供選擇，符合我們實驗的要求。我們選定具有多種 Bit-rate 的 Speex 做為 Scalable Codec 的設計原型，為了讓傳

送端能依據所感受到的網路壅塞程度，選擇較適合該程度的壅塞控制反應，codec 必須有足夠多的層次供給壅塞控制機制做選擇。我們以 Speex 的 wideband mode 為雛形做修改，結合 narrowband mode 並去除重複的層次，將原本的 Speex 的 wideband mode 的 11 層次再做層次細分，細分後我們將語音層次的範圍訂為 0 到 Max_level，目前將語音編碼器實做細分為 18 個層次。此外，一般網路電話常用的 iLBC 的位元率為 13.3Kbps 以及 15Kbps，因此，我們規劃以 15Kbps 的層次 9 的語音編碼封包為基準，並稱呼該層次為 Base_rate，將所有層次分為 0~Base_rate 的低位元率層次，10~Max_level 的高位元率層次。當偵測到壅塞發生時，立刻改選擇 Base_rate 的語音編碼封包，並且持續偵測網路狀況，如壅塞程度又加劇，則還可持續遞減選擇 Base_rate 層次以下的語音編碼。當網路持續穩定沒有壅塞訊號時表示網路頻寬足夠，可嘗試遞增慢慢提高語音品質最高可選到層次 17 位元率為 42.2Kbps 的高位元率的語音編碼。

表 8 Speex Quality 0~4

Quality	Bit-rate(Kbps)	Packet Size (Payload only)
0	3.95	10 Byte
1	5.75	15 Byte
2	7.75	20 Byte
3	9.8	25 Byte
4	12.8	32 Byte

表 9 Scalable Codec 0~Base_rate

Quality	Bit-rate(Kbps)	Packet Size (Payload only)
0	2.15	6 Byte
1	3.95	10 Byte
2	5.75	15 Byte
3	5.95	15 Byte
4	7.75	20 Byte
5	8	20 Byte
6	9.8	25 Byte
7	11	28 Byte
8	12.8	32 Byte
9	15	38 Byte

表 10 Speex Quality 5~10

Quality	Bit-rate(Kbps)	Packet Size (Payload only)
5	16.8	42 Byte
6	20.6	52 Byte
7	23.8	60 Byte
8	27.8	70 Byte
9	34.2	86 Byte
10	42.2	106 Byte

表 11 Scalable Codec 10~Max_level

Quality	Bit-rate(Kbps)	Packet Size (Payload only)
10	16.8	42 Byte
11	18.2	46 Byte
12	20.6	52 Byte
13	23.8	60 Byte
14	24.6	62 Byte
15	27.8	70 Byte
16	34.2	86 Byte
17	42.2	106 Byte

3.5 演算法設計

3.5.1 壅塞偵測

此研究利用 DCCP 的壅塞控制機制偵測網路壅塞程度，並依照不同的壅塞程度選擇相對應的語音編碼層次，來協助紓緩網路壅塞，並提升壅塞狀況下的語音品質。DCCP 可以選擇的壅塞控制機制中的 CCID3 被建議用來傳送多媒體網路串流封包。CCID3 又名 TCP-Friendly Rate Control(TFRC)，透過 RTT (Round Trip Time) 與封包遺失率(Packet Loss Rate)等參數計算傳輸速率。

DCCP 傳送端傳送封包時會在封包中加入封包序號以及 Timestamp。每傳送一個封包，封包序號會遞增一，並規定序號的範圍大小要夠大，才不會造成接收端所以記錄的歷史序號出現重複的狀況。Timestamp 則是說明該封包是何時傳送

的，在接收端可以藉由 Timestamp 來判斷遺失封包的傳送時間，並計算遺失封包是否屬於同一個遺失事件。如果傳送端沒有記錄每個封包的 Timestamp，則傳送端會利用傳送給接收端後，接收端再回覆確認封包中夾帶的原本的封包序號以及 Timestamp 來計算封包來回時間。如傳送端有記錄每個封包序號及其相對應的 Timestamp，則接收端則不須要將 Timestamp 再包含於確認封包中，傳送端接收到確認封包後即可利用確認封包到達時間減去原本記錄的 Timestamp 計算得到封包來回時間。

CCID3 的壅塞控制機制中，接收端會回傳 Feedback Packet 給傳送端，Feedback Packet 中包含 t_recvdata、t_delay、X_recv 以及 P 等欄位。t_recvdata 指的是發出 Feedback Packet 前收到最後一個封包的 Timestamp。t_delay 則是指發出 Feedback Packet 前收到最後一個封包的到達時間和目前傳送 Feedback 的時間之間的時間延遲。X_recv 則表示從前一個 Feedback Packet 到目前為止接收端估計的資料傳送率。P 則代表與封包遺失率相關的 Loss Event Rate。傳送端在接收到接收端的 Feedback Packet 後會計算封包來回時間。首先會計算該次收到 Feedback Packet 的封包來回時間採樣 R_sample，公式如下：

$$R\_sample = (t\_now - t\_recvdata) - t\_delay \quad (4)$$

並利用指數加權移動平均(Exponential Weighted Moving Average, EWMA)的概念，計算平均 Run Trip Time 如公式(5)：

$$RTT = q*RTT + (1-q)*R\_sample \quad (5)$$

q:權重建議值為 0.9

利用公式(5)，我們可以計算出較為平滑的封包來回時間，圖 11 是一個實驗顯示在壅塞發生時 RTT 的變化，我們在網路瓶頸為 128Kps 的網路中，每 10 秒增加 1 通電話，直到網路中同時存在 10 通電話後每隔 10 秒結束 1 通電話。隨著通話數的增加，頻寬將逐漸不足，網路壅塞也會隨之發生。我們記錄封包的

RTT 以及遺失率，觀察其變化。

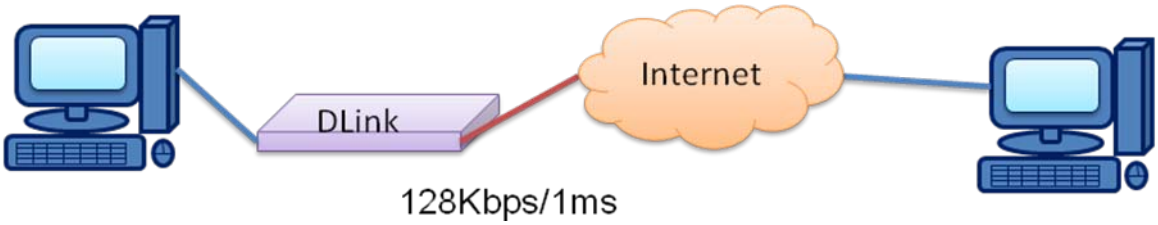

圖 11　觀察網路壅塞時 RTT 及封包遺失率的變化的實驗網路拓撲

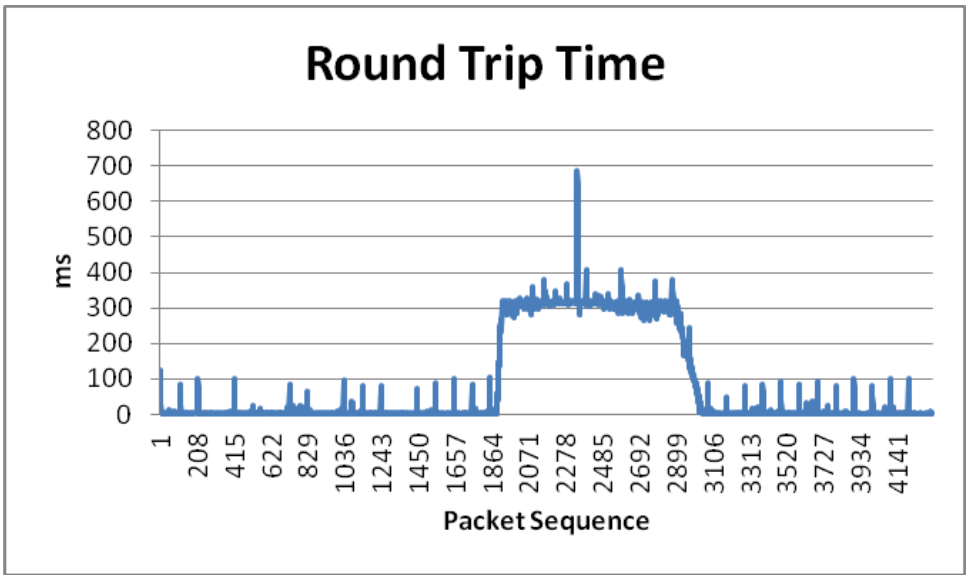

圖 12　網路壅塞時封包來回時間的變化

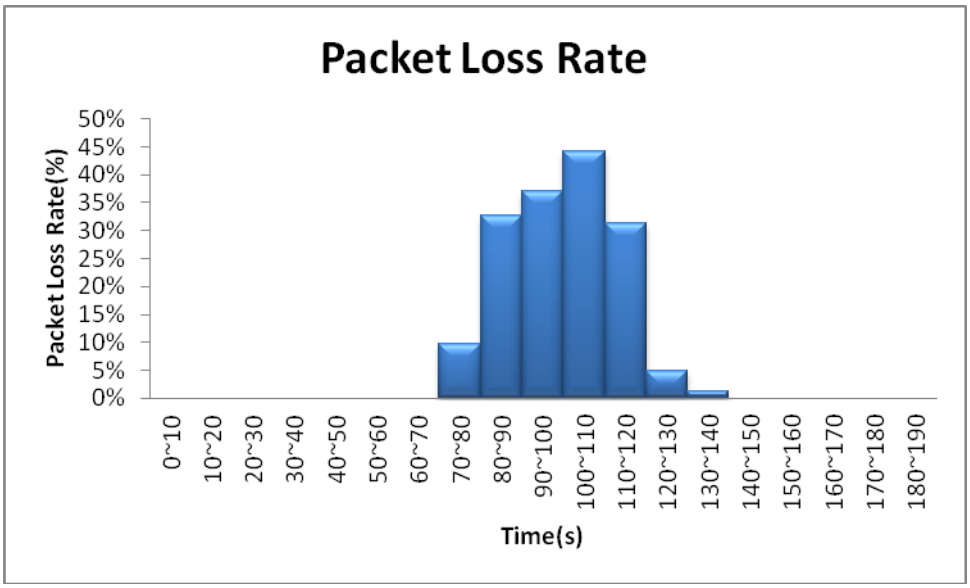

圖 13　網路壅塞時封包遺失率的變化

如圖 12、圖 13，可以看出隨著通話數的增加，網路發生壅塞狀況，RTT

以及封包遺失率也隨之增長，而當網路壅塞狀況舒緩，網路恢復正常時 RTT 以及封包遺失率也隨之減少，可以看出封包傳送到接收端時，RTT 應該在一個固定的時間範圍之內；當頻寬不足時，RTT 會拉長，偵測到此現象時可視為網路壅塞。

我們假設假設 RTT 的時間為t_i，且服從常態分配，則 RTT 可用$t_1, t_2, \cdots, t_N$表示，其平均值為$\bar{t}$(6)，標準差為σ_t(7)，並且為避免重複計算標準差，我們可以將(7)推導成(8)。依照常態分配之經驗法則，我們可以估計出：下一個 RTT 時間t_i在$[\bar{t}-\sigma_t, \bar{t}+\sigma_t]$範圍之機率為 68.3%。下一個 RTT 時間$t_i$在$[\bar{t}-2\sigma_t, \bar{t}+2\sigma_t]$範圍之機率為 95.4%。下一個 RTT 時間$t_i$在$[\bar{t}-3\sigma_t, \bar{t}+3\sigma_t]$範圍之機率為 99.7%。

$$\bar{t} = \frac{1}{N}\sum_{i=1}^{N} t_i \tag{6}$$

$$\sigma_t = \sqrt{\frac{1}{N-1}\sum_{i=1}^{n}(t_i - \bar{t})^2} \tag{7}$$

$$\sum_{i=1}^{N}(t_i - \bar{t})^2 = \sum_{i=1}^{N}({t_i}^2 - 2t_i\bar{t} + \bar{t}^2) = \sum_{i=1}^{N}{t_i}^2 - 2\bar{t}\sum_{i=1}^{N}t_i + \sum_{i=1}^{N}\bar{t}^2$$

$$= \sum_{i=1}^{N}{t_i}^2 - 2N\bar{t}^2 + N\bar{t}^2 = \sum_{i=1}^{N}{t_i}^2 - N\bar{t}^2 \tag{8}$$

$$\sigma_t = \sqrt{\frac{1}{N-1}(\sum_{i=1}^{N}{t_i}^2 - N\bar{t}^2)}$$

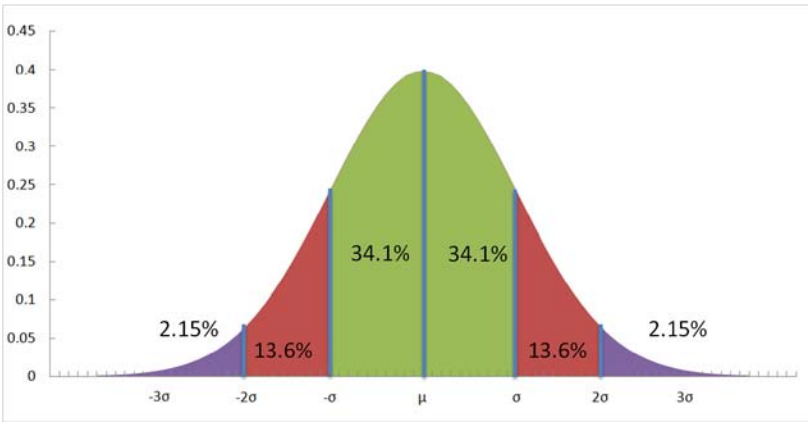

圖 14　常態分配之經驗法則

3.5.2 High Delay Response 演算法

利用常態分配之經驗法則我們可以推測，如果 RTT 大於$[\bar{t}+3\sigma_t]$則判斷為網路壅塞，我們訂定調降 Bit-rate 的門檻值 HDThresh(High Delay Threshold)。如果連續 HDThresh 個封包之 RTT 大於$[\bar{t}+3\sigma_t]$，且所選擇的語音編碼層次高於 Base_rate，則將層次立即降低至 Base_rate，如果所選擇的語音編碼層次已經低於 Base_rate 且不低於最低層次，則只需降低一個層次，演算法如圖 15、圖 16 所示。

```
double[] RTTHistroy = new double[HistorySize];
If no feedback has been received before
    R = R_sample;
Else
    R = q*R + (1-q)*R_sample;
    add R to RTTHistroy;
End If
If R > average(RTTHistroy)+3*stdev(RTTHistroy) and HDCount > HDThresh
    If codec_level > Base_rate
        Set the codec_level as Base_rate;
    Else If codec_level > the_lowest_level
        codec_level - -;
    End If
    HDCount = 0;
Else If R > average(RTTHistroy)+3*stdev(RTTHistroy)
    HDCount + +;
Else
    HDCount = 0;
End If
```

圖 15　High Delay Response 演算法(pseudo code)

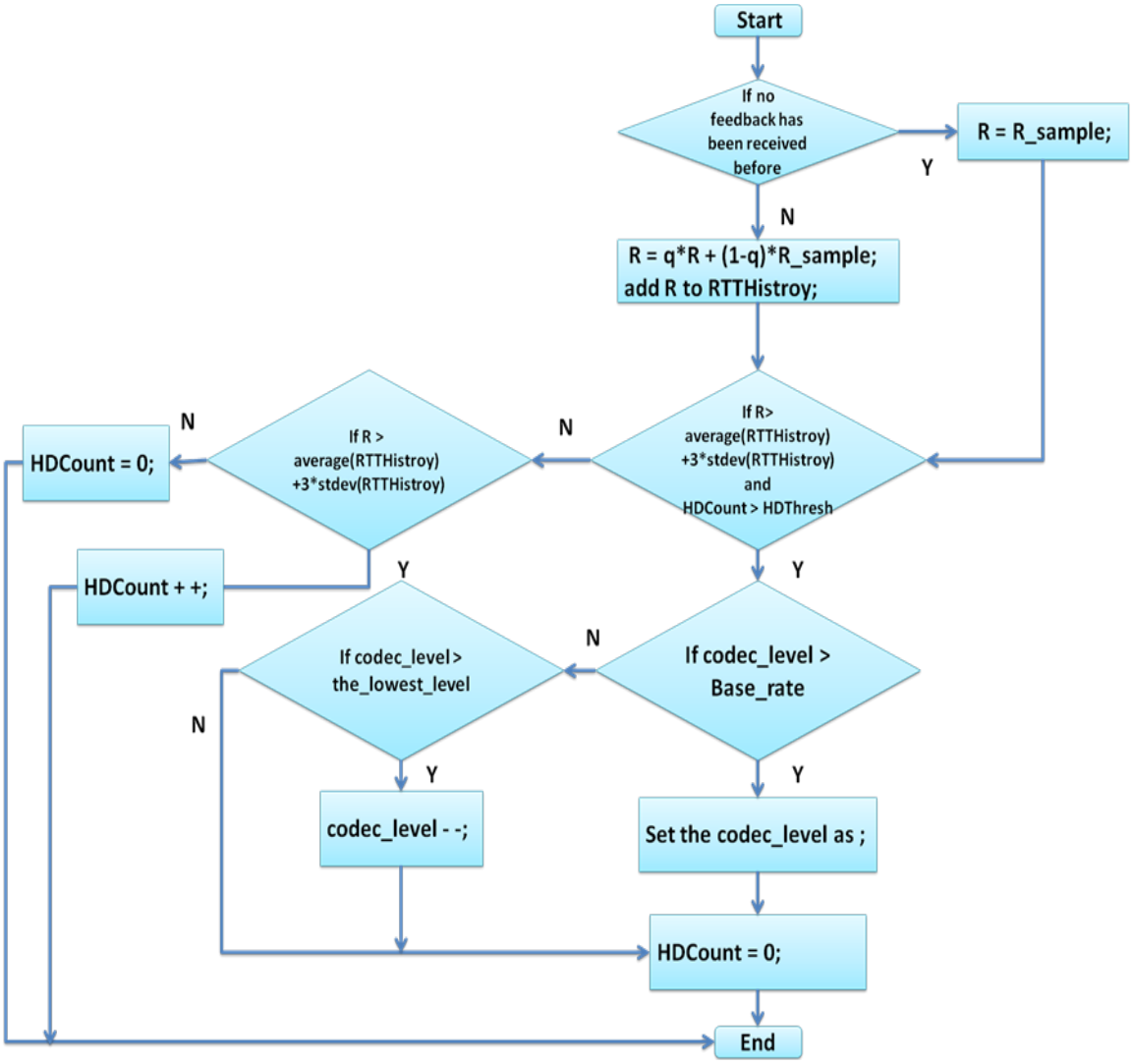

圖 16　High Delay Response 演算法流程圖

3.5.3 High Loss Response 演算法

TFRC 利用 RTT 與封包遺失率可以立即得知網路狀況,再利用 RTT 以及封包遺失率等參數，透過公式做傳送速率的限制。因為 DCCP 可以即時的偵測到網路的狀況，並迅速調降封包傳輸率，因此傳輸速率會下降的比其他傳輸協定快速。TCP 傳輸協定則是要等到 Timeout 才降低速率。因此使用 TFRC 的 DCCP 無法取得合理的頻寬。

由以上的描述我們可以知道 DCCP 在得知壅塞發生時的快速反應並不適切，當 DCCP 偵測到網路狀況時，立即限制傳輸速率。因為 DCCP 較快降低傳送速率，網路狀況得到舒緩，但 TCP 因此沒有感受到網路的壅塞因而持續增加 CWND，會因此造成 DCCP 持續退讓而其他協定沒有感受到的後果。

如果要避免 DCCP 對於網路頻寬競爭不合理的狀況，我們設想當網路壅塞況狀發生時，DCCP 可以得知網路狀況，但是持續保持比適切速率較高一些，就可以維持網路擁塞的訊號,而其他傳輸協定偵測到網路壅塞的狀況後也會因此做調整。並且雖然網路保持輕微的壅塞狀況，但對於可以忍受少量封包遺失的網路應用程式(例如網路電話)而言並不影響。

由以上論述，我們設定 HLThresh(High Loss Threshold)，HLThresh 必須依照應用程式類型所能忍受的封包遺失率訂定。當封包遺失率小於 HLThresh 時表示尚能忍受少量封包遺失的壅塞程度，並不做壅塞控制反應，讓網路持續保持小壅塞訊息，讓 TCP 可以感受到網路壅塞的狀況並降緩其傳送封包的數量。當封包遺失率大於 HLThresh 時表示，壅塞程度已經讓封包遺失量到達已經不能忍受的程度了，將採取壅塞控制反應，演算法如圖 17、圖 18 所示。

```
Receive feedback Packet and get the p(loss event rate)
If p >HLThresh
    If codec_level > Base_rate
        Set the codec_level as Base_rate;
    Else If codec_level > last_level
        codec_level - -;
    End If
End If
```

圖 17　High Loss Response 演算法(pseudo code)

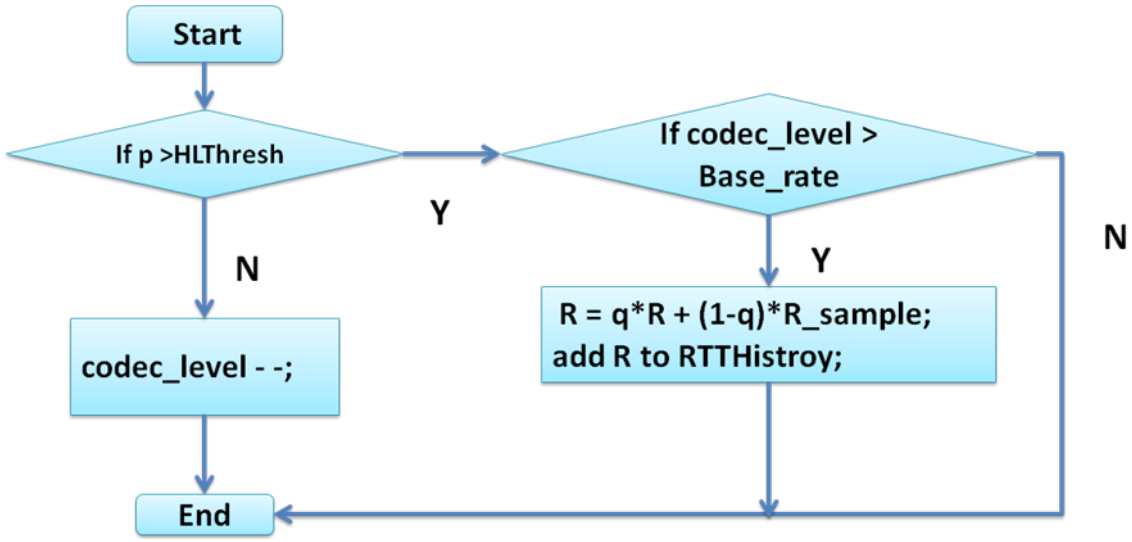

圖 18　High Loss Response 演算法流程圖

3.5.4 Low Delay Response 演算法

利用常態分配之經驗法則我們可以推測，如果 RTT 小於$[\bar{t}-3\sigma_t]$則判斷為網路壅塞狀況舒緩，我們訂定調升 Bit-rate 的門檻值 LDThresh（Low Delay Threshold），如果連續 LDThresh 個封包之 RTT 小於 LDThresh，則將 Bit-rate 提升一個層次，演算法如圖 19、圖 20 所示。

```
double[] RTTHistroy = new double[HistorySize];
If no feedback has been received before
    R = R_sample;
Else
    R = q*R + (1-q)*R_sample;
    add R to RTTHistroy;
End If
If R < average(RTTHistroy)-3*stdev(RTTHistroy) and LDCount > LDThresh
    If codec_level < the_highest_level
        codec_level + +;
    End If
    LDCount = 0;
Else If R > average(RTTHistroy)-3*stdev(RTTHistroy)
    LDCount + +;
Else
    LDCount = 0;
End If
```

圖 19　Low Delay Response 演算法（ pseudo code ）

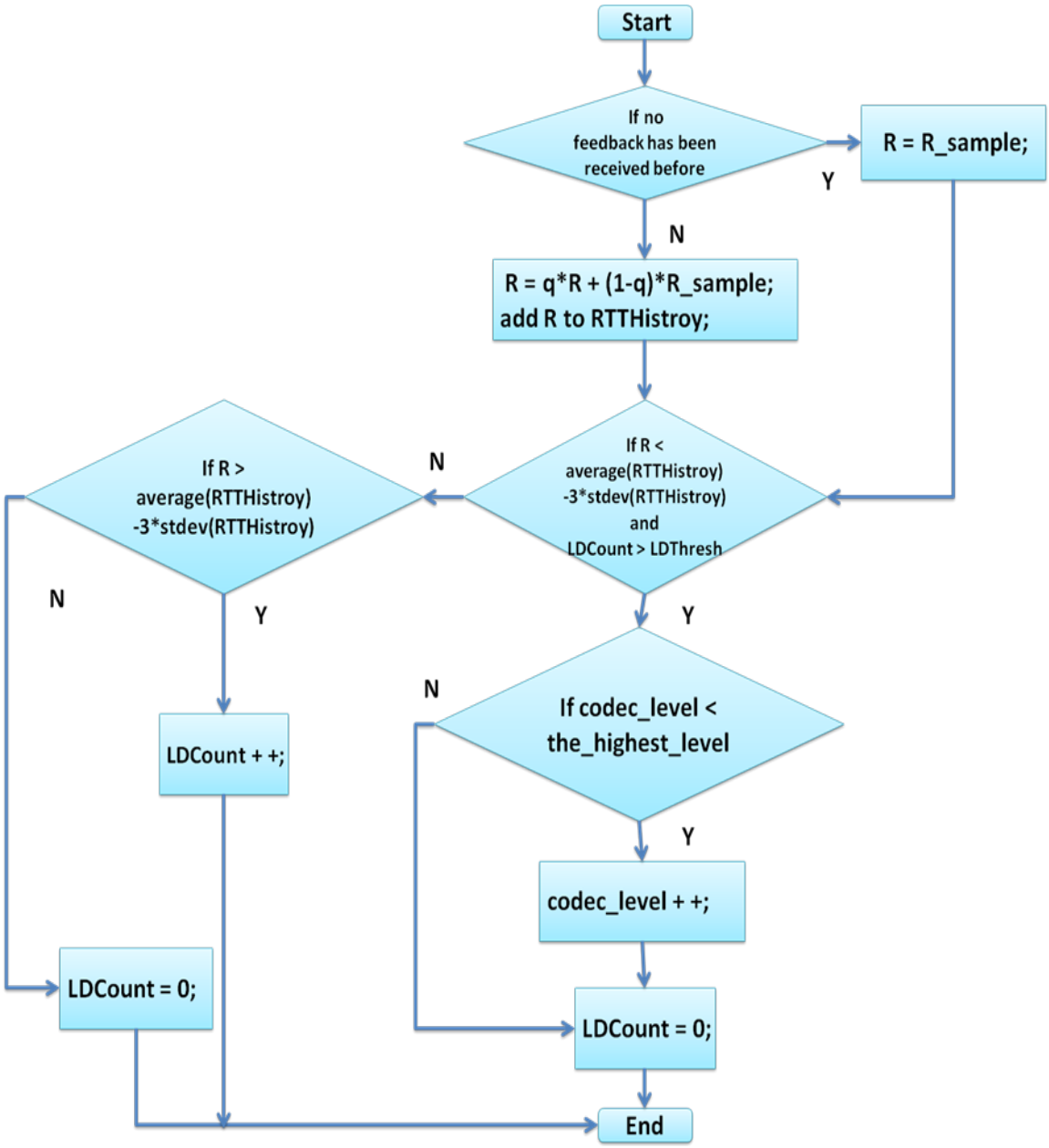

圖 20　Low Delay Response 演算法流程圖

第四章 實驗與效能評估

本研究在實際網路上進行實驗以評估本系統的效能。實驗一中，假設網路中僅使用同一種傳輸協定，以 MOS 與 E-Model 作為評量標準，評估(CBR over UDP)以及 Flexible Bit-rate 以及 Scalabe Codec 三種方式在不同的網路壅塞程度之下的通話品質。實驗二中假設同時存在使用 TCP 傳輸協定的應用程式，評估 Scalable Codec 搭配 DCCP 對於頻寬的競爭能力，亦即公平性。

4.1 評估指標

MOS(Mean Opinion Score)是直接利用人耳對於語音的感覺，並給予 1 到 5 分的分數。但有限的人力難以進行大量的評估工作，故大部分採用 E-model，以網路的工程參數，如 Packet Delay、Packet Loss、Jitter 和經編碼器(Codec)壓縮後的失真程度來計算效能，並換算成 MOS。

4.2 實驗環境

在實驗一和實驗二中，我們利用兩部 PC 做為發話端和收話端，作業系統為 Windows 7，實驗程式安裝在 Java eclipse-SDK-3.7.2-win32 上。接收端安裝在政治大學行動計算與網路通訊實驗室並連上台灣學術網路。發送端同樣安裝在政治大學行動計算與網路通訊實驗室，但透過連結 D-Link DIR-600 Wireless 150 Router 作為網路瓶頸點，在控制功能選項中有 QoS 引擎功能可限制連線通過的頻

寬流量，之後再連結台灣學術網路，硬體設備如表 12 所示。

表 12 實驗設備硬體規格

PC1	Processor: Intel(R) Core(TM)2 CPU 6300 @ 1.86GHz 1.87GHz Installed memory(RAM): 2.00 GB System type: Windows 7, 32-bit Operating System
PC2	Processor: AMD Athlon(tm) 64X2 Dual Core Processor 6000+ 3.10GHz Installed memory(RAM): 2.00 GB System type: Windows 7, 32-bit Operating System
Router	D-Link DIR-600 Wireless 150 Router

4.3 實驗一

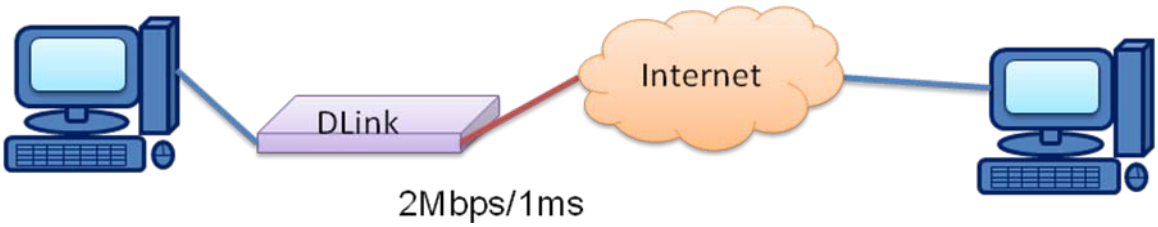

圖 21 實驗一網路拓樸

實驗總共傳送數量為100通VoIP語音通話，因無法透過耳朵實際收聽所有電話，因此我們僅以人耳收聽10通電話作為MOS評分參考，其餘則利用在實驗過程中記錄下封包遺失率及bit-rate的變化透過圖表呈現。

4.3.1 CBR over UDP VoIP 實驗結果與分析

圖 22中記錄著以CBR配合UDP作為VoIP的傳輸協議時的封包遺失率與語音通話品質MOS的變化，圖 23為傳輸bit-rate與MOS的變化。實驗中每隔10秒增加10通VoIP語音通話進入網路，直到網路中有著100通VoIP同時進行傳輸為止，之後每隔10秒鐘減少10通電話，直到結束所有通話。實驗時間共190秒。

實驗時間40~50秒時，網路中共存在著50通網路電話，此時封包遺失率近乎為零，代表網路尚能負載這些資料量。但隨著網路電話通話數增加至60通，封包遺失率逐漸增加，透過封包遺失率的變化我們可以發現，由於UDP沒有依照網路狀況做調整，網路壅塞的情形越來越嚴重，已無法維持整體網路的穩定，語音通話品質也由原來的MOS 4.35分，下降至普通的MOS 2.93分。

當網路電話通話數到達80通時，網路壅塞加劇，封包遺失率超過30%，整體通話品質更降到糟糕的MOS 1.5分；當實驗時間90~100時，同時通話數到達100通時，整體封包遺失率已超過40%，此時的MOS降到幾乎完全無法聽清楚對話內容的1.5之下。

實驗時間100秒後，每隔10秒結束10通VoIP語音通話，直到所有語音通話結束後終止實驗。在這個過程中可以看到因為網路狀況逐漸舒緩，封包遺失率逐步下降，此時仍在通話中的網路電話，其語音通話仍持續有斷話，但是有慢慢減少，直到網路頻寬足夠時才能恢復到原本的品質。

實驗數據詳細記載於表 13、表 14與表 15，表中白色網底是封包遺失率介於0%到10%之間，在VoIP中可能還聽不太出來差異，淺色網底的是封包遺失率介於10%到20%之間，這是VoIP勉強還能容忍的範圍，深色網底的是封包遺失率大於20%，已經嚴重影響到通話品質。

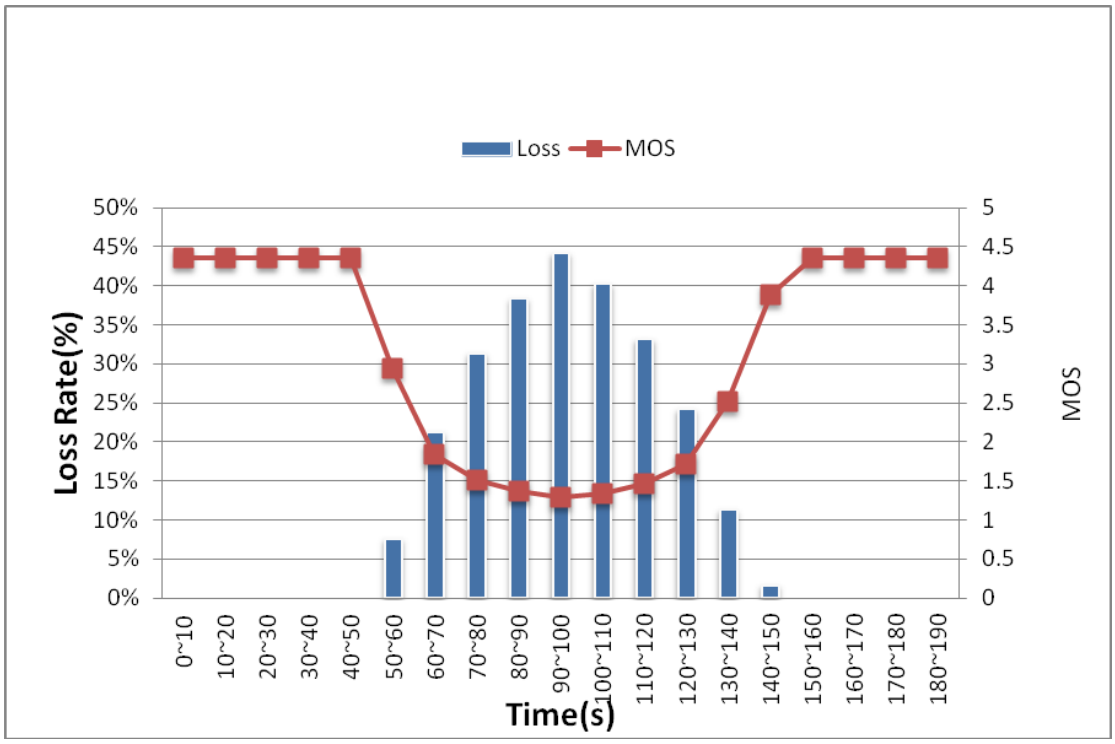

圖 22　Packet Loss Rate 與 MOS 之變化–CBR over UDP

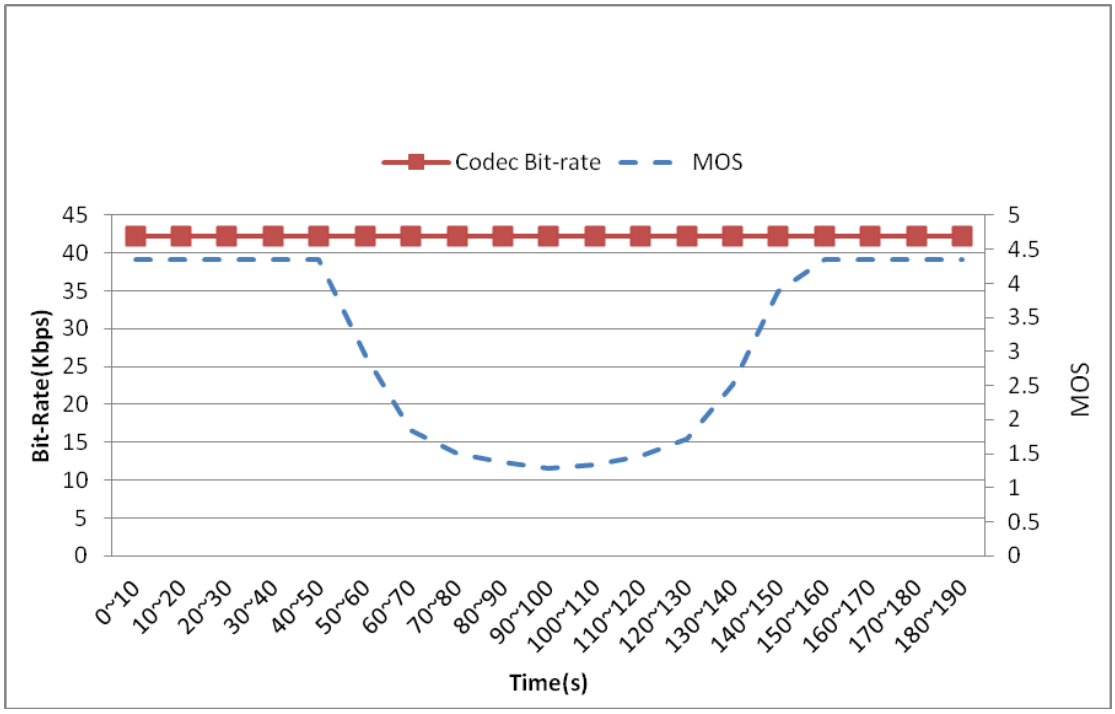

圖 23　Bit–rate 與 MOS 之變化– CBR over UDP

表 13 以 CBR over UDP 傳輸 100 通 VoIP 封包遺失率之變化-PART1

Calls	0~10	10~20	20~30	30~40	40~50	50~60	60~70	70~80	80~90	90~100	100~110	110~120	120~130	130~140	140~150	150~160	160~170	170~180	180~190	Loss
1	0.00	0.00	0.00	0.00	0.00	0.03	0.21	0.31	0.22	0.22	0.30	0.23	0.27	0.17	0.02	0.00	0.00	0.00	0.00	0.10
2	0.00	0.00	0.00	0.00	0.00	0.06	0.17	0.27	0.25	0.30	0.31	0.19	0.13	0.09	0.01	0.00	0.00	0.00	0.00	0.09
3	0.00	0.00	0.00	0.00	0.00	0.00	0.06	0.11	0.18	0.16	0.14	0.07	0.06	0.02	0.00	0.00	0.00	0.00	0.00	0.04
4	0.00	0.00	0.00	0.00	0.00	0.06	0.06	0.15	0.20	0.19	0.19	0.08	0.08	0.03	0.01	0.00	0.00	0.00	0.00	0.06
5	0.00	0.00	0.00	0.00	0.00	0.08	0.11	0.21	0.24	0.29	0.30	0.13	0.11	0.04	0.01	0.00	0.00	0.00	0.00	0.08
6	0.00	0.00	0.00	0.00	0.00	0.09	0.23	0.34	0.34	0.34	0.42	0.24	0.20	0.09	0.02	0.00	0.00	0.00	0.00	0.12
7	0.00	0.00	0.00	0.00	0.00	0.11	0.32	0.47	0.39	0.44	0.54	0.33	0.26	0.16	0.02	0.00	0.00	0.00	0.00	0.16
8	0.00	0.00	0.00	0.00	0.00	0.13	0.42	0.54	0.45	0.47	0.60	0.47	0.39	0.28	0.03	0.00	0.00	0.00	0.00	0.20
9	0.00	0.00	0.00	0.00	0.00	0.12	0.49	0.54	0.46	0.52	0.63	0.48	0.48	0.39	0.06	0.00	0.00	0.00	0.00	0.22
10	0.00	0.00	0.00	0.00	0.00	0.12	0.50	0.59	0.47	0.54	0.73	0.56	0.49	0.44	0.06	0.00	0.00	0.00	0.00	0.24
11		0.00	0.00	0.00	0.00	0.04	0.16	0.29	0.24	0.33	0.35	0.31	0.27	0.15	0.03	0.00	0.00	0.00		0.13
12		0.00	0.00	0.00	0.00	0.04	0.11	0.24	0.33	0.40	0.33	0.27	0.17	0.04	0.00	0.00	0.00	0.00		0.11
13		0.00	0.00	0.00	0.00	0.00	0.07	0.13	0.26	0.30	0.15	0.11	0.06	0.00	0.00	0.00	0.00	0.00		0.06
14		0.00	0.00	0.00	0.00	0.00	0.07	0.14	0.27	0.34	0.24	0.16	0.07	0.02	0.00	0.00	0.00	0.00		0.08
15		0.00	0.00	0.00	0.00	0.00	0.07	0.23	0.31	0.43	0.29	0.22	0.15	0.02	0.01	0.00	0.00	0.00		0.10
16		0.00	0.00	0.00	0.00	0.04	0.12	0.35	0.38	0.50	0.39	0.31	0.21	0.04	0.00	0.00	0.00	0.00		0.14
17		0.00	0.00	0.00	0.00	0.07	0.14	0.44	0.43	0.56	0.52	0.45	0.33	0.07	0.01	0.00	0.00	0.00		0.18
18		0.00	0.00	0.00	0.00	0.09	0.21	0.49	0.50	0.63	0.64	0.56	0.43	0.14	0.02	0.00	0.00	0.00		0.22
19		0.00	0.00	0.00	0.00	0.10	0.29	0.54	0.56	0.69	0.68	0.60	0.49	0.19	0.03	0.00	0.00	0.00		0.25
20		0.00	0.00	0.00	0.00	0.11	0.31	0.52	0.55	0.64	0.66	0.61	0.52	0.22	0.04	0.00	0.00	0.00		0.25
21			0.00	0.00	0.00	0.07	0.24	0.23	0.25	0.34	0.31	0.32	0.30	0.18	0.02	0.00	0.00			0.15
22			0.00	0.00	0.00	0.04	0.08	0.22	0.32	0.46	0.32	0.24	0.13	0.04	0.00	0.00	0.00			0.12
23			0.00	0.00	0.00	0.03	0.06	0.12	0.23	0.30	0.13	0.11	0.06	0.00	0.00	0.00	0.00			0.07
24			0.00	0.00	0.00	0.00	0.06	0.15	0.28	0.37	0.19	0.14	0.08	0.00	0.00	0.00	0.00			0.08
25			0.00	0.00	0.00	0.03	0.11	0.20	0.33	0.42	0.28	0.20	0.10	0.02	0.00	0.00	0.00			0.11
26			0.00	0.00	0.00	0.07	0.20	0.34	0.40	0.52	0.40	0.30	0.17	0.03	0.01	0.00	0.00			0.16
27			0.00	0.00	0.00	0.10	0.29	0.42	0.48	0.61	0.57	0.48	0.28	0.08	0.03	0.00	0.00			0.22
28			0.00	0.00	0.00	0.14	0.40	0.54	0.56	0.63	0.62	0.57	0.33	0.12	0.03	0.00	0.00			0.26
29			0.00	0.00	0.00	0.19	0.45	0.57	0.55	0.67	0.67	0.62	0.40	0.24	0.05	0.00	0.00			0.29
30			0.00	0.00	0.00	0.24	0.50	0.58	0.52	0.63	0.65	0.62	0.47	0.28	0.05	0.00	0.00			0.30
31				0.00	0.00	0.09	0.19	0.28	0.24	0.30	0.38	0.28	0.27	0.18	0.03	0.00				0.17
32				0.00	0.00	0.07	0.13	0.20	0.36	0.36	0.34	0.25	0.15	0.03	0.00	0.00				0.15
33				0.00	0.00	0.00	0.06	0.09	0.24	0.28	0.16	0.11	0.06	0.00	0.00	0.00				0.08
34				0.00	0.00	0.04	0.10	0.13	0.28	0.31	0.23	0.15	0.07	0.00	0.00	0.00				0.10
35				0.00	0.00	0.07	0.16	0.20	0.39	0.39	0.30	0.21	0.09	0.03	0.00	0.00				0.14

表 14 以 CBR over UDP 傳輸 100 通 VoIP 封包遺失率之變化-PART2

Calls	0~10	10~20	20~30	30~40	40~50	50~60	60~70	70~80	80~90	90~100	100~110	110~120	120~130	130~140	140~150	150~160	160~170	170~180	180~190	Loss
36				0.00	0.00	0.14	0.21	0.32	0.48	0.43	0.43	0.28	0.18	0.04	0.00	0.00				0.19
37				0.00	0.00	0.23	0.31	0.43	0.58	0.51	0.56	0.41	0.31	0.07	0.01	0.00				0.26
38				0.00	0.00	0.30	0.40	0.53	0.61	0.60	0.61	0.52	0.43	0.12	0.01	0.00				0.32
39				0.00	0.00	0.32	0.45	0.63	0.62	0.59	0.67	0.59	0.44	0.15	0.02	0.00				0.34
40				0.00	0.00	0.37	0.52	0.59	0.62	0.63	0.68	0.56	0.52	0.22	0.02	0.00				0.36
41					0.00	0.05	0.20	0.27	0.34	0.29	0.28	0.35	0.27	0.20	0.03					0.21
42					0.00	0.04	0.08	0.21	0.30	0.40	0.34	0.29	0.19	0.06	0.00					0.17
43					0.00	0.00	0.05	0.09	0.22	0.29	0.16	0.13	0.05	0.02	0.00					0.09
44					0.00	0.00	0.07	0.13	0.27	0.33	0.21	0.15	0.08	0.04	0.00					0.12
45					0.00	0.00	0.10	0.18	0.32	0.44	0.33	0.22	0.14	0.05	0.00					0.16
46					0.00	0.04	0.17	0.22	0.44	0.51	0.43	0.31	0.28	0.10	0.00					0.23
47					0.00	0.04	0.23	0.31	0.49	0.59	0.57	0.41	0.37	0.14	0.01					0.29
48					0.00	0.07	0.31	0.43	0.54	0.63	0.59	0.56	0.46	0.24	0.02					0.35
49					0.00	0.08	0.38	0.48	0.54	0.63	0.64	0.61	0.57	0.32	0.04					0.39
50					0.00	0.11	0.35	0.50	0.59	0.69	0.69	0.65	0.53	0.32	0.08					0.41
51						0.06	0.21	0.27	0.29	0.31	0.30	0.29	0.22	0.12						0.23
52						0.02	0.14	0.16	0.36	0.36	0.25	0.27	0.14	0.02						0.19
53						0.02	0.07	0.10	0.27	0.24	0.13	0.11	0.05	0.00						0.11
54						0.02	0.08	0.13	0.31	0.30	0.18	0.16	0.07	0.02						0.14
55						0.00	0.12	0.17	0.37	0.36	0.24	0.22	0.12	0.02						0.18
56						0.02	0.15	0.24	0.45	0.43	0.33	0.31	0.17	0.04						0.24
57						0.03	0.24	0.30	0.57	0.55	0.46	0.50	0.22	0.09						0.33
58						0.04	0.34	0.45	0.64	0.61	0.54	0.59	0.29	0.12						0.40
59						0.04	0.44	0.48	0.59	0.63	0.58	0.62	0.38	0.20						0.44
60						0.05	0.46	0.56	0.59	0.63	0.65	0.62	0.41	0.24						0.47
61							0.27	0.31	0.24	0.31	0.29	0.32	0.17							0.27
62							0.08	0.21	0.32	0.37	0.29	0.21	0.11							0.23
63							0.05	0.09	0.22	0.22	0.17	0.10	0.07							0.13
64							0.06	0.12	0.25	0.29	0.21	0.13	0.09							0.16
65							0.09	0.17	0.30	0.38	0.33	0.19	0.13							0.23
66							0.12	0.31	0.37	0.48	0.45	0.30	0.16							0.31
67							0.13	0.40	0.47	0.57	0.52	0.44	0.20							0.39
68							0.20	0.48	0.52	0.65	0.61	0.49	0.29							0.46
69							0.28	0.54	0.55	0.66	0.64	0.50	0.33							0.50
70							0.34	0.57	0.60	0.65	0.63	0.61	0.37							0.54

表 15 以 CBR over UDP 傳輸 100 通 VoIP 封包遺失率之變化-PART3

Calls	0~10	10~20	20~30	30~40	40~50	50~60	60~70	70~80	80~90	90~100	100~110	110~120	120~130	130~140	140~150	150~160	160~170	170~180	180~190	Loss
71								0.27	0.31	0.39	0.27	0.29								0.31
72								0.13	0.30	0.38	0.34	0.19								0.27
73								0.07	0.25	0.22	0.16	0.06								0.15
74								0.11	0.29	0.28	0.22	0.11								0.20
75								0.16	0.33	0.36	0.30	0.18								0.27
76								0.20	0.43	0.45	0.37	0.23								0.34
77								0.28	0.46	0.52	0.54	0.10								0.38
78								0.36	0.49	0.60	0.61	0.27								0.47
79								0.41	0.61	0.64	0.69	0.53								0.58
80								0.48	0.59	0.67	0.68	0.57								0.60
81									0.23	0.31	0.18									0.24
82									0.20	0.34	0.28									0.27
83									0.10	0.21	0.11									0.14
84									0.14	0.27	0.14									0.18
85									0.19	0.32	0.30									0.27
86									0.26	0.44	0.37									0.36
87									0.32	0.54	0.52									0.46
88									0.43	0.63	0.26									0.44
89									0.44	0.61	0.62									0.56
90									0.45	0.67	0.29									0.47
91										0.47										0.47
92										0.23										0.23
93										0.11										0.11
94										0.19										0.19
95										0.25										0.25
96										0.39										0.39
97										0.47										0.47
98										0.57										0.57
99										0.60										0.60
100										0.58										0.58
AVG	0.00	0.00	0.00	0.00	0.00	0.08	0.21	0.31	0.38	0.44	0.40	0.33	0.24	0.11	0.02	0.00	0.00	0.00	0.00	

4.3.2 Flexible Bit-rate VoIP 實驗結果與分析

圖 24中記錄著以Flexible Bit-rate傳送VoIP時的，封包遺失率與通話品質MOS的變化，圖 25為傳輸bit-rate與MOS的變化。實驗中每隔10秒增加10通VoIP語音通話進入網路，直到網路中有著100通VoIP同時進行傳輸為止，之後每隔10秒鐘減少10通電話，直到結束所有通話。實驗時間共190秒。

實驗初始時，10通VoIP進入網路中，封包遺失率近乎為零，顯示網路頻寬相當充裕；當通話數來到60通時，封包遺失率約在3%左右， MOS 3.8分的通話品質。此時Flexible Bit-rate的壅塞偵測偵測到網路壅塞，降低了Bit-Rate一個層次，不過由於對於壅塞程度的反應不夠適切，當同時通話數由60上升到90通時，持續有封包遺失的增加，不過相較於一般CBR配合UDP的方式，封包遺失率已有明顯改善。當通話數達到100通時，封包遺失率還是持續的上升到超過10%，而MOS也降到通話品質不佳的2分左右。

實驗時間100秒後，每隔10秒結束10通VoIP語音通話，直到所有語音通話結束後終止實驗。在這個過程中可以看到因為網路狀況逐漸舒緩，封包遺失率逐步下降。而Flexible Bit-rate的方法也讓Bit-rate逐步提升，語音品質也在固定時間間隔下慢慢恢復原本的品質。可是因為Flexible Bit-rate是以固定間隔時間恢復語音品質，沒有依照網路的狀況來做恢復，所以在實驗時間100秒到130秒之間，封包遺失率影響語音品質的狀況依舊持續存在。

實驗詳細結果記錄於表 16、表 17與表 18，表中白色網底是封包遺失率介於0%到10%之間，在VoIP中可能還聽不太出來差異，淺色網底的是封包遺失率介於10%到20%之間，這是VoIP勉強還能容忍的範圍，深色網底的是封包遺失率大於20%，已經嚴重影響到通話品質。

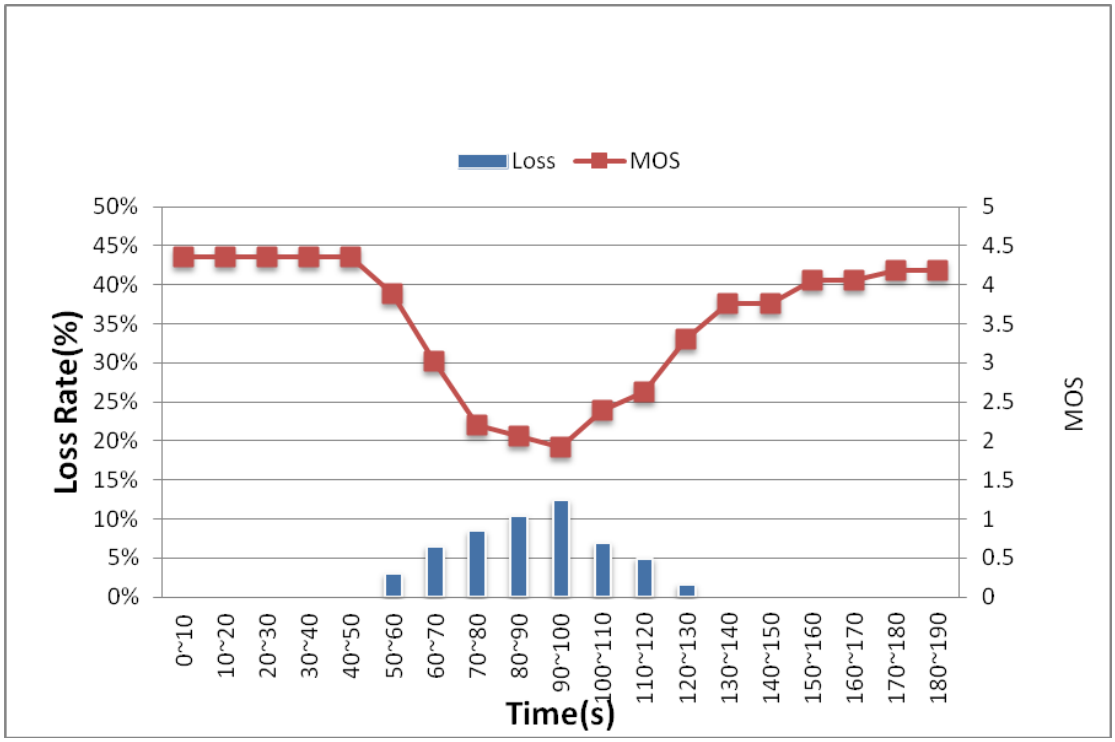

圖 24　Packet Loss Rate 與 MOS 之變化–Flexible Bit-rate

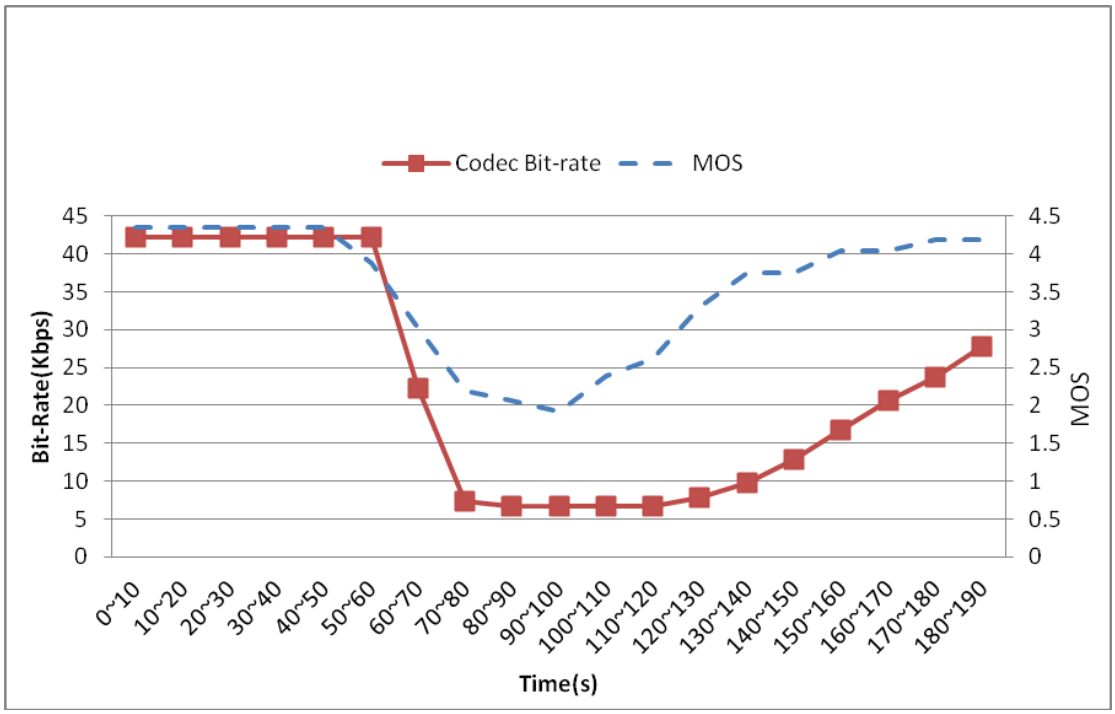

圖 25　Bit–rate 與 MOS 之變化–Flexible Bit-rate

表 16 以 Flexible Bit-Rate 傳輸 100 通 VoIP 封包遺失率之變化-PART1

Calls	0~10	10~20	20~30	30~40	40~50	50~60	60~70	70~80	80~90	90~100	100~110	110~120	120~130	130~140	140~150	150~160	160~170	170~180	180~190	Loss
1	0.00	0.00	0.00	0.00	0.00	0.01	0.08	0.04	0.09	0.10	0.09	0.08	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.03
2	0.00	0.00	0.00	0.00	0.00	0.00	0.06	0.07	0.09	0.09	0.06	0.04	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.02
3	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.09	0.09	0.06	0.02	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.01
4	0.00	0.00	0.00	0.00	0.00	0.00	0.06	0.00	0.09	0.08	0.08	0.03	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.02
5	0.00	0.00	0.00	0.00	0.00	0.00	0.07	0.07	0.10	0.09	0.07	0.03	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.02
6	0.00	0.00	0.00	0.00	0.00	0.00	0.07	0.07	0.11	0.13	0.08	0.04	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.03
7	0.00	0.00	0.00	0.00	0.00	0.08	0.12	0.08	0.11	0.13	0.09	0.06	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.03
8	0.00	0.00	0.00	0.00	0.00	0.09	0.13	0.10	0.12	0.15	0.09	0.07	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.04
9	0.00	0.00	0.00	0.00	0.00	0.10	0.17	0.12	0.14	0.17	0.11	0.09	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.05
10	0.00	0.00	0.00	0.00	0.00	0.11	0.20	0.12	0.16	0.16	0.12	0.09	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.05
11		0.00	0.00	0.00	0.00	0.00	0.08	0.05	0.12	0.12	0.07	0.05	0.01	0.00	0.00	0.00	0.00	0.00		0.03
12		0.00	0.00	0.00	0.00	0.06	0.04	0.08	0.09	0.15	0.03	0.04	0.00	0.00	0.00	0.00	0.00	0.00		0.03
13		0.00	0.00	0.00	0.00	0.00	0.00	0.07	0.07	0.10	0.02	0.02	0.00	0.00	0.00	0.00	0.00	0.00		0.02
14		0.00	0.00	0.00	0.00	0.00	0.00	0.09	0.07	0.11	0.03	0.02	0.00	0.00	0.00	0.00	0.00	0.00		0.02
15		0.00	0.00	0.00	0.00	0.00	0.00	0.11	0.09	0.15	0.04	0.03	0.00	0.00	0.00	0.00	0.00	0.00		0.02
16		0.00	0.00	0.00	0.00	0.00	0.05	0.10	0.09	0.18	0.05	0.04	0.00	0.00	0.00	0.00	0.00	0.00		0.03
17		0.00	0.00	0.00	0.00	0.07	0.06	0.14	0.12	0.20	0.04	0.06	0.00	0.00	0.00	0.00	0.00	0.00		0.04
18		0.00	0.00	0.00	0.00	0.09	0.08	0.19	0.15	0.22	0.07	0.08	0.00	0.00	0.00	0.00	0.00	0.00		0.05
19		0.00	0.00	0.00	0.00	0.09	0.11	0.19	0.14	0.23	0.07	0.10	0.03	0.00	0.00	0.00	0.00	0.00		0.06
20		0.00	0.00	0.00	0.00	0.11	0.16	0.19	0.17	0.23	0.09	0.15	0.03	0.00	0.00	0.00	0.00	0.00		0.07
21			0.00	0.00	0.00	0.01	0.04	0.08	0.08	0.12	0.09	0.07	0.01	0.00	0.00	0.00	0.00			0.03
22			0.00	0.00	0.00	0.00	0.05	0.07	0.07	0.09	0.05	0.04	0.00	0.00	0.00	0.00	0.00			0.02
23			0.00	0.00	0.00	0.00	0.00	0.00	0.04	0.09	0.05	0.02	0.00	0.00	0.00	0.00	0.00			0.01
24			0.00	0.00	0.00	0.00	0.04	0.05	0.07	0.09	0.05	0.02	0.00	0.00	0.00	0.00	0.00			0.02
25			0.00	0.00	0.00	0.00	0.04	0.07	0.08	0.10	0.05	0.02	0.00	0.00	0.00	0.00	0.00			0.02
26			0.00	0.00	0.00	0.00	0.05	0.07	0.09	0.11	0.06	0.02	0.04	0.00	0.00	0.00	0.00			0.03
27			0.00	0.00	0.00	0.00	0.09	0.10	0.12	0.12	0.08	0.04	0.04	0.00	0.00	0.00	0.00			0.04
28			0.00	0.00	0.00	0.00	0.11	0.15	0.14	0.14	0.09	0.07	0.04	0.00	0.00	0.00	0.00			0.05
29			0.00	0.00	0.00	0.07	0.15	0.17	0.18	0.15	0.09	0.09	0.06	0.00	0.00	0.00	0.00			0.06
30			0.00	0.00	0.00	0.09	0.15	0.17	0.18	0.15	0.11	0.09	0.06	0.00	0.00	0.00	0.00			0.07
31				0.00	0.00	0.03	0.06	0.09	0.12	0.08	0.04	0.07	0.03	0.00	0.00	0.00				0.04
32				0.00	0.00	0.00	0.04	0.06	0.07	0.09	0.07	0.03	0.00	0.00	0.00	0.00				0.03
33				0.00	0.00	0.00	0.00	0.05	0.05	0.09	0.05	0.00	0.00	0.00	0.00	0.00				0.02
34				0.00	0.00	0.00	0.00	0.06	0.05	0.10	0.05	0.02	0.00	0.00	0.00	0.00				0.02
35				0.00	0.00	0.00	0.05	0.07	0.08	0.11	0.06	0.02	0.00	0.00	0.00	0.00				0.03

表 17 以 Flexible Bit-Rate 傳輸 100 通 VoIP 封包遺失率之變化-PART2

Calls	0~10	10~20	20~30	30~40	40~50	50~60	60~70	70~80	80~90	90~100	100~110	110~120	120~130	130~140	140~150	150~160	160~170	170~180	180~190	Loss
36				0.00	0.00	0.00	0.05	0.07	0.10	0.12	0.08	0.02	0.00	0.00	0.00	0.00				0.03
37				0.00	0.00	0.00	0.07	0.08	0.14	0.13	0.09	0.02	0.00	0.00	0.00	0.00				0.04
38				0.00	0.00	0.00	0.11	0.12	0.17	0.17	0.10	0.05	0.00	0.00	0.00	0.00				0.05
39				0.00	0.00	0.07	0.14	0.18	0.17	0.17	0.13	0.07	0.04	0.00	0.00	0.00				0.07
40				0.00	0.00	0.07	0.20	0.19	0.17	0.20	0.11	0.06	0.03	0.00	0.00	0.00				0.08
41					0.00	0.04	0.05	0.05	0.07	0.10	0.06	0.07	0.00	0.00	0.00					0.04
42					0.00	0.00	0.04	0.06	0.09	0.11	0.04	0.02	0.00	0.00	0.00					0.03
43					0.00	0.00	0.00	0.05	0.06	0.06	0.03	0.00	0.00	0.00	0.00					0.02
44					0.00	0.00	0.00	0.06	0.07	0.08	0.04	0.01	0.00	0.00	0.00					0.02
45					0.00	0.00	0.00	0.07	0.08	0.09	0.06	0.04	0.03	0.00	0.00					0.03
46					0.00	0.00	0.00	0.10	0.09	0.10	0.07	0.07	0.03	0.00	0.00					0.04
47					0.00	0.08	0.04	0.11	0.11	0.14	0.09	0.09	0.03	0.00	0.00					0.06
48					0.00	0.08	0.05	0.14	0.14	0.15	0.10	0.10	0.03	0.00	0.00					0.07
49					0.00	0.08	0.07	0.13	0.18	0.16	0.12	0.15	0.04	0.00	0.00					0.08
50					0.00	0.07	0.07	0.16	0.18	0.18	0.12	0.16	0.05	0.00	0.00					0.09
51						0.02	0.08	0.08	0.10	0.06	0.05	0.09	0.02	0.00						0.06
52						0.04	0.08	0.06	0.06	0.10	0.04	0.04	0.03	0.00						0.05
53						0.00	0.00	0.04	0.05	0.08	0.02	0.02	0.02	0.00						0.03
54						0.00	0.04	0.05	0.05	0.09	0.03	0.02	0.02	0.00						0.03
55						0.00	0.05	0.05	0.07	0.10	0.03	0.00	0.04	0.00						0.04
56						0.00	0.07	0.07	0.07	0.13	0.05	0.04	0.06	0.00						0.05
57						0.04	0.11	0.07	0.11	0.12	0.07	0.04	0.06	0.00						0.07
58						0.05	0.14	0.11	0.11	0.16	0.09	0.06	0.07	0.00						0.09
59						0.05	0.17	0.11	0.11	0.17	0.09	0.10	0.06	0.00						0.09
60						0.06	0.19	0.14	0.13	0.16	0.09	0.07	0.07	0.00						0.10
61							0.04	0.07	0.12	0.12	0.04	0.04	0.00							0.06
62							0.04	0.06	0.09	0.12	0.03	0.04	0.00							0.05
63							0.00	0.05	0.04	0.07	0.02	0.00	0.00							0.03
64							0.00	0.05	0.07	0.08	0.03	0.00	0.00							0.03
65							0.00	0.05	0.10	0.13	0.05	0.03	0.00							0.05
66							0.00	0.05	0.11	0.13	0.06	0.04	0.00							0.06
67							0.04	0.07	0.18	0.18	0.07	0.06	0.00							0.08
68							0.05	0.07	0.21	0.19	0.07	0.06	0.00							0.10
69							0.06	0.07	0.24	0.22	0.11	0.07	0.00							0.11
70							0.07	0.12	0.26	0.20	0.10	0.09	0.00							0.12

表 18 以 Flexible Bit-Rate 傳輸 100 通 VoIP 封包遺失率之變化-PART3

Calls	0~10	10~20	20~30	30~40	40~50	50~60	60~70	70~80	80~90	90~100	100~110	110~120	120~130	130~140	140~150	150~160	160~170	170~180	180~190	Loss
71								0.07	0.11	0.05	0.03	0.06								0.06
72								0.04	0.08	0.09	0.06	0.01								0.06
73								0.00	0.06	0.06	0.04	0.00								0.03
74								0.04	0.06	0.09	0.07	0.00								0.05
75								0.04	0.09	0.12	0.10	0.01								0.07
76								0.04	0.09	0.14	0.15	0.03								0.09
77								0.06	0.11	0.16	0.16	0.04								0.11
78								0.09	0.14	0.19	0.21	0.06								0.14
79								0.10	0.14	0.26	0.22	0.06								0.16
80								0.14	0.17	0.23	0.22	0.10								0.17
81									0.05	0.07	0.07									0.06
82									0.04	0.12	0.00									0.06
83									0.03	0.05	0.01									0.03
84									0.03	0.07	0.00									0.03
85									0.04	0.11	0.01									0.05
86									0.04	0.10	0.01									0.05
87									0.07	0.19	0.02									0.10
88									0.08	0.20	0.03									0.10
89									0.10	0.20	0.03									0.11
90									0.09	0.24	0.04									0.12
91										0.08										0.08
92										0.05										0.05
93										0.02										0.02
94										0.01										0.01
95										0.02										0.02
96										0.04										0.04
97										0.07										0.07
98										0.07										0.07
99										0.09										0.09
100										0.11										0.11
AVG	0.00	0.00	0.00	0.00	0.00	0.03	0.06	0.08	0.10	0.12	0.07	0.05	0.02	0.00	0.00	0.00	0.00	0.00	0.00	

4.3.3 Scalable Codec VoIP 實驗結果與分析

圖 26中記錄著以Scalable Codec 傳送 VoIP時的，封包遺失率與通話品質MOS的變化，圖 27為傳輸bit-rate與MOS的變化。實驗中每隔10秒增加10通VoIP語音通話進入網路，直到網路中有著100通VoIP同時進行傳輸為止。

實驗初始到50秒之間封包遺失率為零，可看出網路頻寬相當充裕；當實驗時間來到50秒到60秒時，網路開始出現不穩定的狀況，封包遺失率到達2.94%，此段時間在封包遺失開始之前封包來回時間已開始拉長，當開始偵測到這樣的狀況後，立即選擇了語音編碼層次中的Base_rate，網路壅塞狀況立刻得到舒緩。當通話數來到70通時，網路壅塞況狀持續加劇，又再度選擇更低的語音層次，封包遺失率依舊控制在約在5%左右或以下。由於能準確的利用DCCP偵測網路的RTT以及封包遺失率來偵測網路狀況，並且選擇合適的語音層次，在實驗70秒到100秒之間，封包遺失率都控制在5%左右或以下，而且在網路壅塞狀況最嚴重的實驗時間90秒到100秒，語音品質也能維持在MOS 3分左右的通話品質。

實驗時間100秒後，每隔10秒結束10通VoIP語音通話，直到所有語音通話結束後終止實驗。在這個過程中網路狀況逐漸舒緩，但是在網路中通話數仍然存在超過網路能夠負荷前，網路壅塞以及封包遺失的狀況仍然可能存在，因此，依舊沒有選擇較高的語音層次，到實驗時間140秒到150秒確認網路狀況較充裕時，才逐步恢復語音品質的層次。

實驗詳細結果記錄於表 19、表 20與表 21，表中白色網底是封包遺失率介於0%到10%之間，在VoIP中可能還聽不太出來差異，淺色網底是封包遺失率介於10%到20%之間，這是VoIP勉強還能容忍的範圍，深色網底是封包遺失率大於20%，已經嚴重影響到通話品質。可看出網路整理狀況相當穩定，沒有出現深色網底。

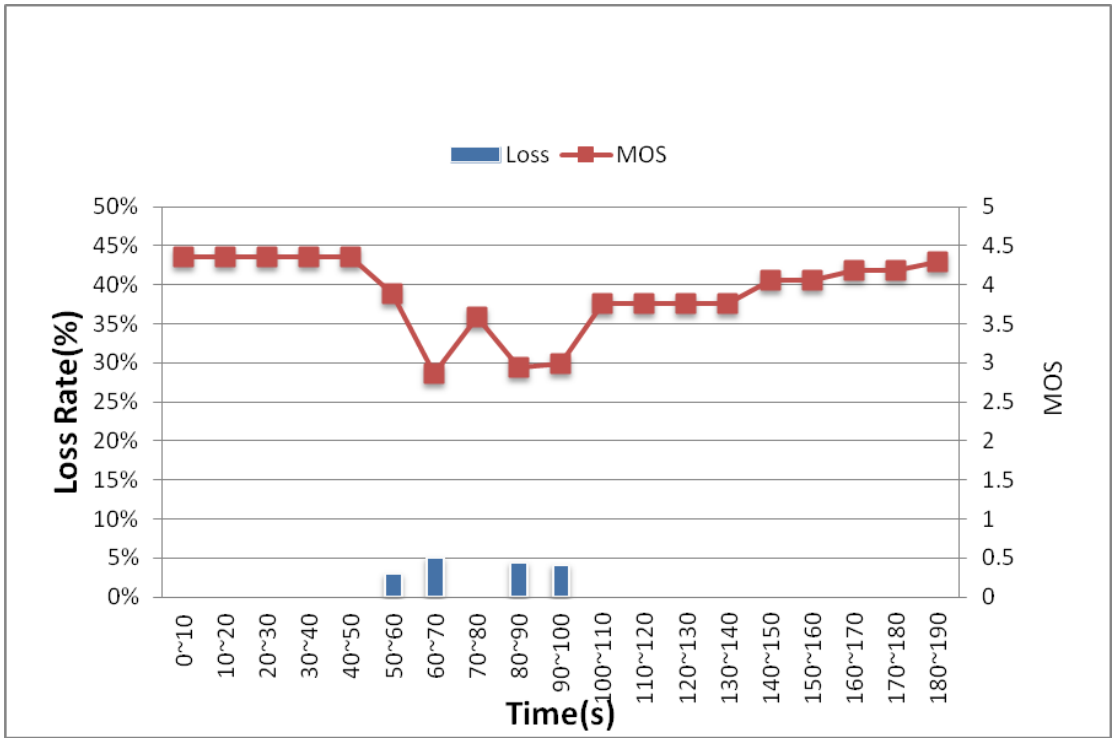

圖 26　Packet Loss Rate 與 MOS 之變化–Scalable Codec

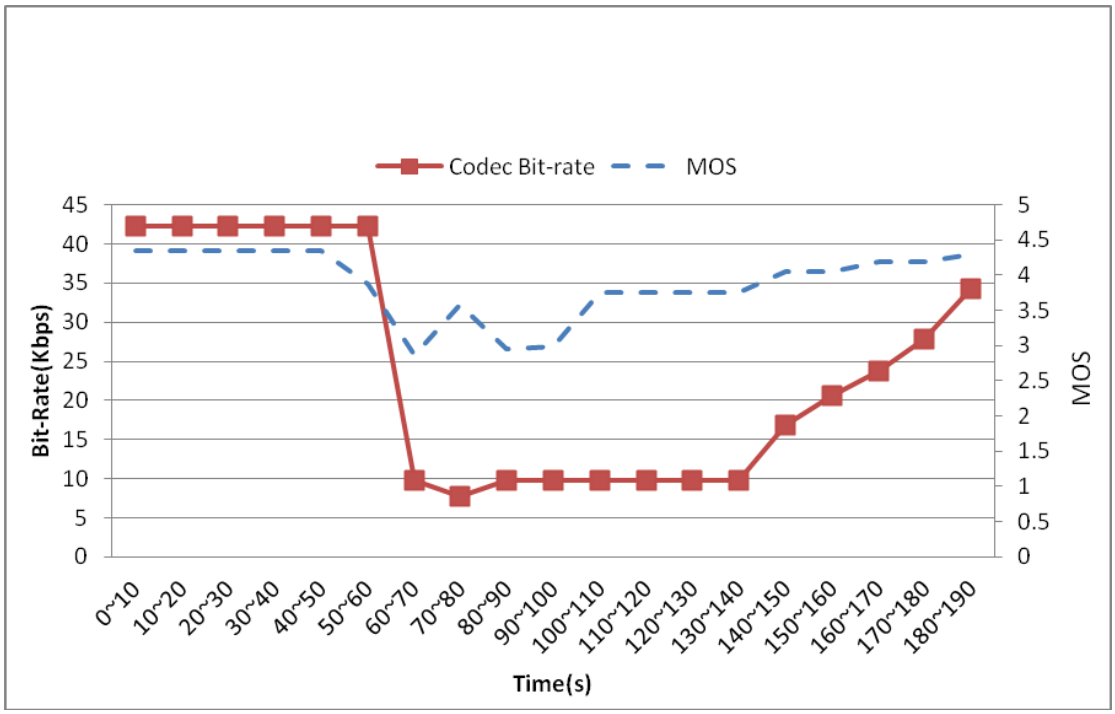

圖 27　Bit–rate 與 MOS 之變化– Scalable Codec

表 19 以 Scalabe Codec 傳輸 100 通 VoIP 封包遺失率之變化-PART1

Calls	0~10	10~20	20~30	30~40	40~50	50~60	60~70	70~80	80~90	90~100	100~110	110~120	120~130	130~140	140~150	150~160	160~170	170~180	180~190	Loss
1	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
2	0.00	0.00	0.00	0.00	0.00	0.07	0.09	0.00	0.06	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.01
3	0.00	0.00	0.00	0.00	0.00	0.05	0.00	0.00	0.06	0.07	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.01
4	0.00	0.00	0.00	0.00	0.00	0.06	0.00	0.00	0.06	0.07	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.01
5	0.00	0.00	0.00	0.00	0.00	0.07	0.09	0.00	0.00	0.07	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.01
6	0.00	0.00	0.00	0.00	0.00	0.08	0.10	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.01
7	0.00	0.00	0.00	0.00	0.00	0.12	0.10	0.00	0.06	0.07	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.02
8	0.00	0.00	0.00	0.00	0.00	0.13	0.11	0.00	0.06	0.07	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.02
9	0.00	0.00	0.00	0.00	0.00	0.15	0.11	0.00	0.06	0.07	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.02
10	0.00	0.00	0.00	0.00	0.00	0.19	0.10	0.00	0.06	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.02
11		0.00	0.00	0.00	0.00	0.02	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00		0.00
12		0.00	0.00	0.00	0.00	0.00	0.09	0.00	0.06	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00		0.01
13		0.00	0.00	0.00	0.00	0.00	0.09	0.00	0.06	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00		0.01
14		0.00	0.00	0.00	0.00	0.00	0.09	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00		0.01
15		0.00	0.00	0.00	0.00	0.00	0.09	0.00	0.06	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00		0.01
16		0.00	0.00	0.00	0.00	0.03	0.09	0.00	0.07	0.09	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00		0.02
17		0.00	0.00	0.00	0.00	0.04	0.09	0.00	0.06	0.09	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00		0.02
18		0.00	0.00	0.00	0.00	0.06	0.09	0.00	0.07	0.09	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00		0.02
19		0.00	0.00	0.00	0.00	0.09	0.09	0.00	0.06	0.09	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00		0.02
20		0.00	0.00	0.00	0.00	0.12	0.09	0.00	0.07	0.11	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00		0.02
21			0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00			0.00
22			0.00	0.00	0.00	0.02	0.08	0.00	0.07	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00			0.01
23			0.00	0.00	0.00	0.00	0.08	0.00	0.07	0.09	0.00	0.00	0.00	0.00	0.00	0.00	0.00			0.02
24			0.00	0.00	0.00	0.00	0.08	0.00	0.08	0.09	0.00	0.00	0.00	0.00	0.00	0.00	0.00			0.02
25			0.00	0.00	0.00	0.00	0.08	0.00	0.09	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00			0.01
26			0.00	0.00	0.00	0.00	0.08	0.00	0.09	0.09	0.00	0.00	0.00	0.00	0.00	0.00	0.00			0.02
27			0.00	0.00	0.00	0.00	0.00	0.00	0.09	0.09	0.00	0.00	0.00	0.00	0.00	0.00	0.00			0.01
28			0.00	0.00	0.00	0.03	0.08	0.00	0.09	0.09	0.00	0.00	0.00	0.00	0.00	0.00	0.00			0.02
29			0.00	0.00	0.00	0.04	0.08	0.00	0.09	0.09	0.00	0.00	0.00	0.00	0.00	0.00	0.00			0.02
30			0.00	0.00	0.00	0.06	0.00	0.00	0.09	0.10	0.00	0.00	0.00	0.00	0.00	0.00	0.00			0.02
31				0.00	0.00	0.04	0.00	0.00	0.01	0.01	0.00	0.00	0.00	0.00	0.00	0.00				0.00
32				0.00	0.00	0.00	0.08	0.00	0.07	0.09	0.00	0.00	0.00	0.00	0.00	0.00				0.02
33				0.00	0.00	0.00	0.08	0.00	0.07	0.09	0.00	0.00	0.00	0.00	0.00	0.00				0.02
34				0.00	0.00	0.00	0.08	0.00	0.07	0.09	0.00	0.00	0.00	0.00	0.00	0.00				0.02
35				0.00	0.00	0.00	0.08	0.00	0.07	0.10	0.00	0.00	0.00	0.00	0.00	0.00				0.02

表 20 以 Scalabe Codec 傳輸 100 通 VoIP 封包遺失率之變化-PART2

Calls	0~10	10~20	20~30	30~40	40~50	50~60	60~70	70~80	80~90	90~100	100~110	110~120	120~130	130~140	140~150	150~160	160~170	170~180	180~190	Loss
36				0.00	0.00	0.03	0.08	0.00	0.08	0.10	0.00	0.00	0.00	0.00	0.00	0.00				0.02
37				0.00	0.00	0.03	0.08	0.00	0.07	0.10	0.00	0.00	0.00	0.00	0.00	0.00				0.02
38				0.00	0.00	0.03	0.08	0.00	0.07	0.10	0.00	0.00	0.00	0.00	0.00	0.00				0.02
39				0.00	0.00	0.00	0.08	0.00	0.08	0.10	0.00	0.00	0.00	0.00	0.00	0.00				0.02
40				0.00	0.00	0.03	0.08	0.00	0.08	0.10	0.00	0.00	0.00	0.00	0.00	0.00				0.02
41					0.00	0.06	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00					0.01
42					0.00	0.00	0.00	0.00	0.00	0.10	0.00	0.00	0.00	0.00	0.00					0.01
43					0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00					0.00
44					0.00	0.00	0.00	0.00	0.00	0.10	0.00	0.00	0.00	0.00	0.00					0.01
45					0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00					0.00
46					0.00	0.00	0.00	0.00	0.00	0.10	0.00	0.00	0.00	0.00	0.00					0.01
47					0.00	0.00	0.00	0.00	0.00	0.10	0.00	0.00	0.00	0.00	0.00					0.01
48					0.00	0.00	0.00	0.00	0.00	0.10	0.00	0.00	0.00	0.00	0.00					0.01
49					0.00	0.00	0.00	0.00	0.00	0.11	0.00	0.00	0.00	0.00	0.00					0.01
50					0.00	0.03	0.00	0.00	0.00	0.11	0.00	0.00	0.00	0.00	0.00					0.01
51						0.01	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.00						0.00
52						0.00	0.08	0.00	0.00	0.07	0.00	0.00	0.00	0.00						0.02
53						0.00	0.00	0.00	0.00	0.07	0.00	0.00	0.00	0.00						0.01
54						0.00	0.08	0.00	0.00	0.00	0.00	0.00	0.00	0.00						0.01
55						0.02	0.08	0.00	0.00	0.07	0.00	0.00	0.00	0.00						0.02
56						0.02	0.09	0.00	0.04	0.07	0.00	0.00	0.00	0.00						0.02
57						0.00	0.09	0.00	0.04	0.00	0.00	0.00	0.00	0.00						0.01
58						0.02	0.09	0.00	0.04	0.07	0.00	0.00	0.00	0.00						0.02
59						0.02	0.08	0.00	0.04	0.07	0.00	0.00	0.00	0.00						0.02
60						0.02	0.09	0.00	0.04	0.08	0.00	0.00	0.00	0.00						0.03
61							0.00	0.00	0.00	0.00	0.00	0.00	0.00							0.00
62							0.00	0.00	0.06	0.10	0.00	0.00	0.00							0.02
63							0.00	0.00	0.06	0.00	0.00	0.00	0.00							0.01
64							0.00	0.00	0.06	0.00	0.00	0.00	0.00							0.01
65							0.00	0.00	0.06	0.00	0.00	0.00	0.00							0.01
66							0.00	0.00	0.06	0.00	0.00	0.00	0.00							0.01
67							0.00	0.00	0.06	0.00	0.00	0.00	0.00							0.01
68							0.00	0.00	0.06	0.00	0.00	0.00	0.00							0.01
69							0.00	0.00	0.06	0.00	0.00	0.00	0.00							0.01
70							0.00	0.00	0.06	0.00	0.00	0.00	0.00							0.01

表 21 以 Scalabe Codec 傳輸 100 通 VoIP 封包遺失率之變化-PART3

Calls	0~10	10~20	20~30	30~40	40~50	50~60	60~70	70~80	80~90	90~100	100~110	110~120	120~130	130~140	140~150	150~160	160~170	170~180	180~190	Loss
71								0.00	0.01	0.00	0.00	0.00								0.00
72								0.00	0.06	0.00	0.00	0.00								0.01
73								0.00	0.06	0.00	0.00	0.00								0.01
74								0.00	0.06	0.00	0.00	0.00								0.01
75								0.00	0.06	0.05	0.00	0.00								0.02
76								0.00	0.00	0.05	0.00	0.00								0.01
77								0.00	0.06	0.00	0.00	0.00								0.01
78								0.00	0.06	0.05	0.00	0.00								0.02
79								0.00	0.06	0.05	0.00	0.00								0.02
80								0.00	0.06	0.05	0.00	0.00								0.02
81									0.00	0.00	0.00									0.00
82									0.05	0.00	0.00									0.02
83									0.04	0.00	0.00									0.01
84									0.04	0.00	0.00									0.01
85									0.05	0.00	0.00									0.02
86									0.04	0.00	0.00									0.01
87									0.05	0.00	0.00									0.02
88									0.04	0.00	0.00									0.01
89									0.05	0.05	0.00									0.03
90									0.04	0.00	0.00									0.01
91										0.00										0.00
92										0.00										0.00
93										0.00										0.00
94										0.00										0.00
95										0.00										0.00
96										0.00										0.00
97										0.00										0.00
98										0.00										0.00
99										0.00										0.00
100										0.00										0.00
AVG	0	0	0	0	0	0.02	0.05	0	0.04	0.04	0	0	0	0	0	0	0	0	0	

4.3.4 三種傳輸方式實驗結果與分析

圖 28、圖 29顯示了CBR over UDP、Flexible Bit-Rate以及Scalable Codec三種方式傳輸VoIP的效能比較，分別為封包遺失率、通話品質(MOS)，以及頻寬效率。

結果顯示，如果是頻寬充裕的情況下，一般的CBR配合UDP的方式來傳輸封包的VoIP，的確能有最佳的通話品質，不過當頻寬不足時可以看得出來，封包遺失率大幅的增加，在實驗中的網路環境中能支持50通以CBR配合UDP傳輸的網路電話進行最佳品質的語音通話，可是當通話數量到達100通時，比起10通VoIP時，話品質指標MOS下降了60%左右，封包遺失率增加了40%以上。

使用Flexible Bit-rate以及Scalable Codec來傳輸VoIP通話時，可以看出相較於CBR配合UDP，他們都可以擁有比較好的MOS以及較低的封包遺失率。不過在比較壅塞況狀時，Scalable Codec因為利用DCCP偵測網路狀況的RTT以及封包遺失率來做網路偵測，而且有較細緻的語音層次，所以可以馬上得知網路的狀況，並且做最適當的反應，因此封包遺失率都來的比Flexible Bit-rate低，而且對於控制整體網路狀況的穩定性也比Flexible Bit-rate來的好。

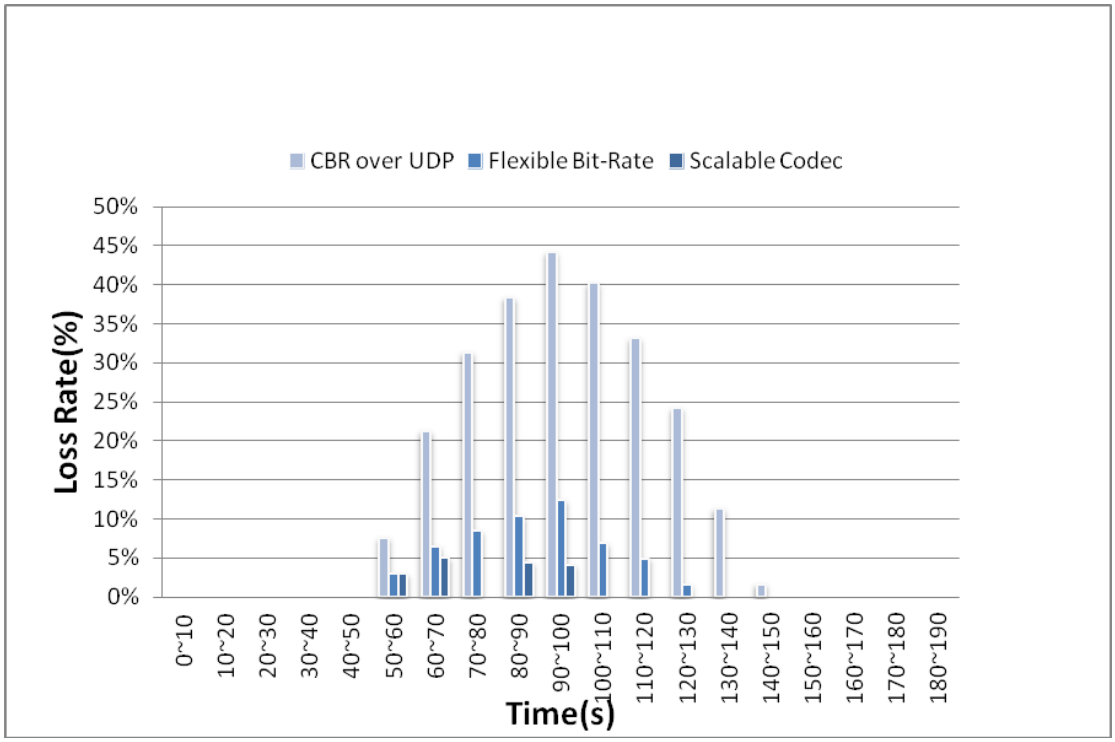

圖 28　三種方式傳輸 VoIP 的封包遺失率比較

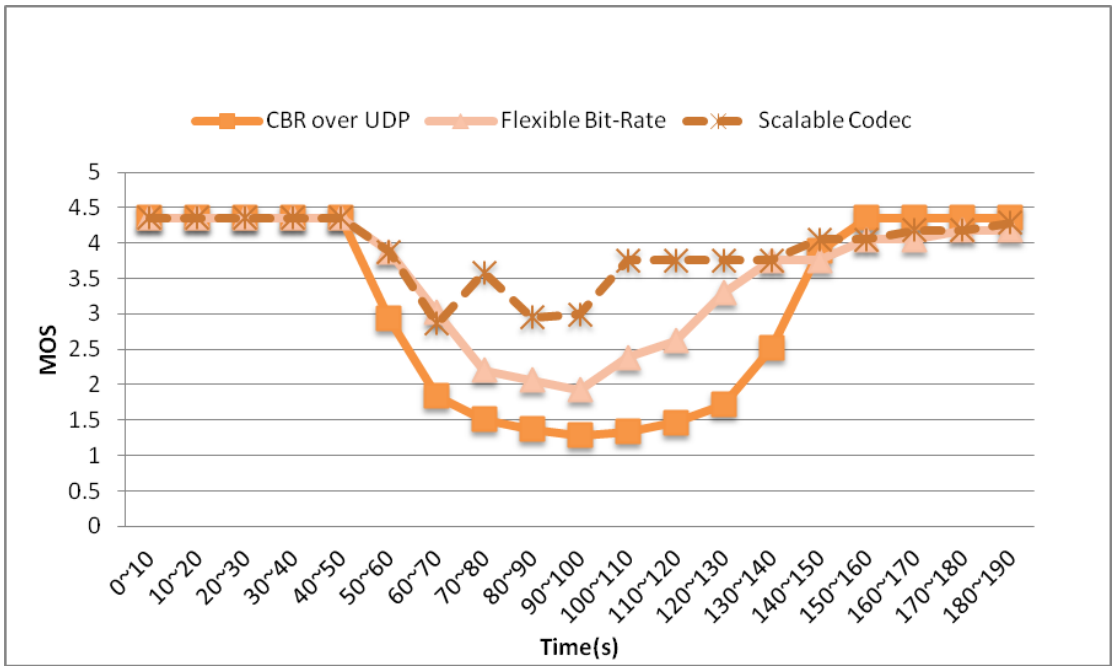

圖 29　三種方式傳輸 VoIP 的 MOS 比較

4.4 實驗二

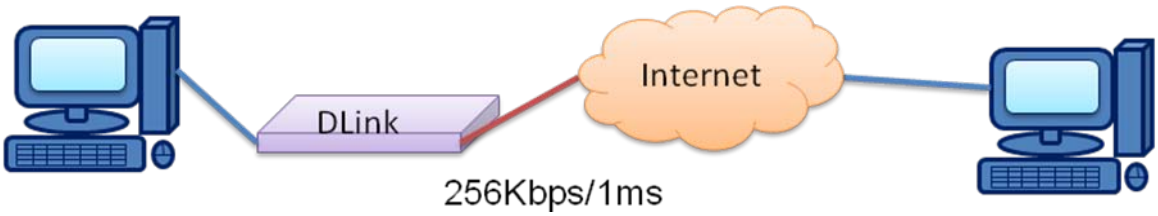

圖 30　實驗二實驗拓撲

在圖 30的網路情境下，本研究透過實際網路的實驗環境評估，分別以CBR over UDP以及Scalable Codec VoIP在頻寬不足時和TCP同時存在於網路中，比較對於頻寬的競爭能力的表現。以及比較在頻寬不足時，多個Scalable Codec VoIP(沒有其他TCP)的狀況下，Scalable Codec VoIP之間的頻寬競爭公平性。

4.4.1 CBR over UDP VoIP vs TCP實驗結果與分析

我們實驗觀察CBR over UDP VoIP先進入網路，之後TCP再進入網路中頻寬競爭的狀況。圖 31、表 22中記錄著CBR over UDP VoIP先進入網路以CBR over UDP VoIP在頻寬不足時和TCP同時存在於網路中所佔的網路Throughput百分比變化。在頻寬瓶頸為256Kbps的網路環境下，在實驗開始時傳送5通CBR over UDP 網路電話，在實驗第10秒時，在網路中再加入5條TCP傳送檔案資料流，在實驗第90秒時，結束所有TCP資料流，並在實驗第100秒時結束實驗。

在實驗開始時，因為網路中只存在CBR over UDP VoIP，所以CBR over UDP VoIP占據了所有的Throughput比例，到了實驗第10秒時，因為TCP資料流的加入，因為頻寬不足，CBR over UDP VoIP與TCP會競爭網路Throughput，在實驗第10到20秒時，可看出TCP因為緩啟動快速增長CWND，一開始的總體TCP可占據快接近40%的Throughput，但是隨著TCP偵測到網路壅塞的狀況，啟動壅塞控制機制，而UDP並沒有依照網路壅塞狀況而有壅塞控制反應，在實驗第20到30秒時TCP所佔的Throughput比例開始下降。在實驗第30到90秒，因為TCP是以AIMD

的方式做壅塞控制，所以總體TCP所佔據的Throughput時高時低，但是都在40%以下，甚至有些時候所占據的Throughput只剩25%。而UDP因為沒有壅塞控制反應，而都占據60%以上的Throughput，甚至有時占據達70%以上的Thoughput比例。頻寬公平指標Jain's Index的表現平均只有85.26%。由此可看出TCP後來進入網路中的情況下，雖然剛開始因為TCP的緩啟動機制，頻寬競爭能力不致於太差，但是UDP持續沒有退讓，TCP感受到網路壅塞狀況後以AIMD的方式做壅塞控制，所以只能占據到較少且不公平的Throughput比例。

此外，我們也實驗觀察TCP先進入網路，之後CBR over UDP VoIP再進入網路中頻寬競爭的狀況。圖 32、表 23中記錄著TCP先進入網路以CBR over UDP VoIP在頻寬不足時和TCP同時存在於網路中所佔的網路Throughput百分比變化。在頻寬瓶頸為256Kbps的網路環境下，在實驗開始時傳送5條TCP傳送檔案資料流，在實驗第10秒時，在網路中再加入5通CBR over UDP 網路電話，在實驗第90秒時，結束所有TCP資料流，並在實驗第100秒時結束實驗。

在實驗開始時，因為網路中只存在TCP資料流，所以TCP占據了所有的Throughput比例，到了實驗第10秒時CBR over UDP VoIP加入，因為頻寬不足，CBR over UDP VoIP與TCP會競爭網路Throughput，在實驗第10到90秒，可以看出在CBR over UDP VoIP加入後，UDP馬上占據了60%以上的Throughput，而TCP因為以AIMD的壅塞控制機制方式持續退讓，Throughput雖有變動，但是在平均只能有30%左右的Throughput，沒有辦法搶佔公平的頻寬。頻寬公平指標Jain's Index的表現平均也只有81.54%。由此可看出TCP雖然是先進入網路中，但是一旦UDP進入網路中，頻寬不足時，頻寬就會被UDP搶走，而TCP則會因為壅塞控制機制的持續退讓而無法得到公平的頻寬。

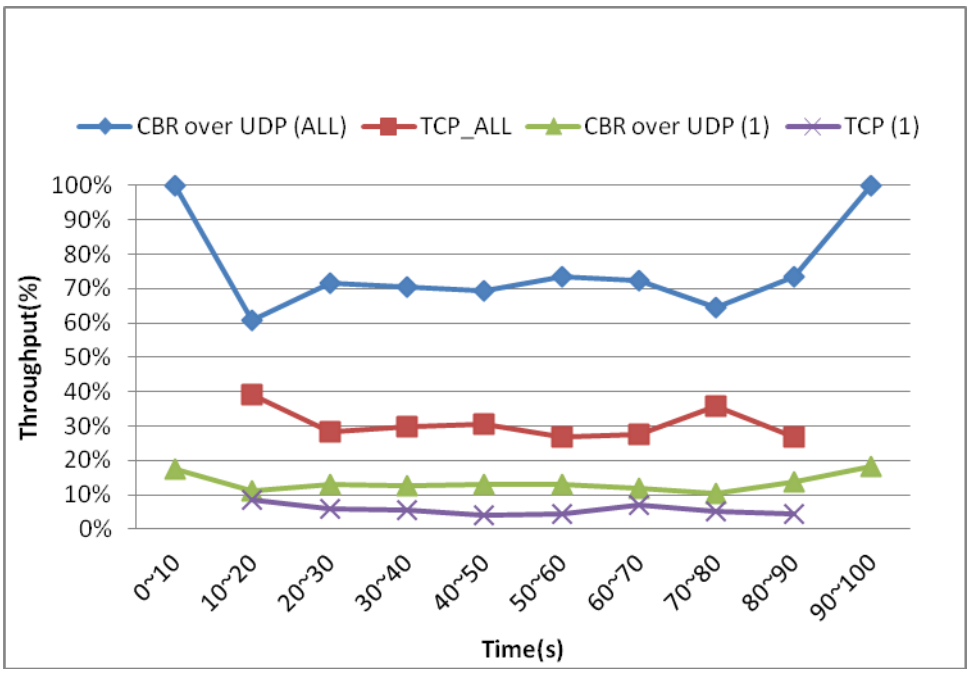

圖 31　CBR over UDP VoIP 與 TCP 吞吐率比較 – CBR over UDP VoIP 先進入

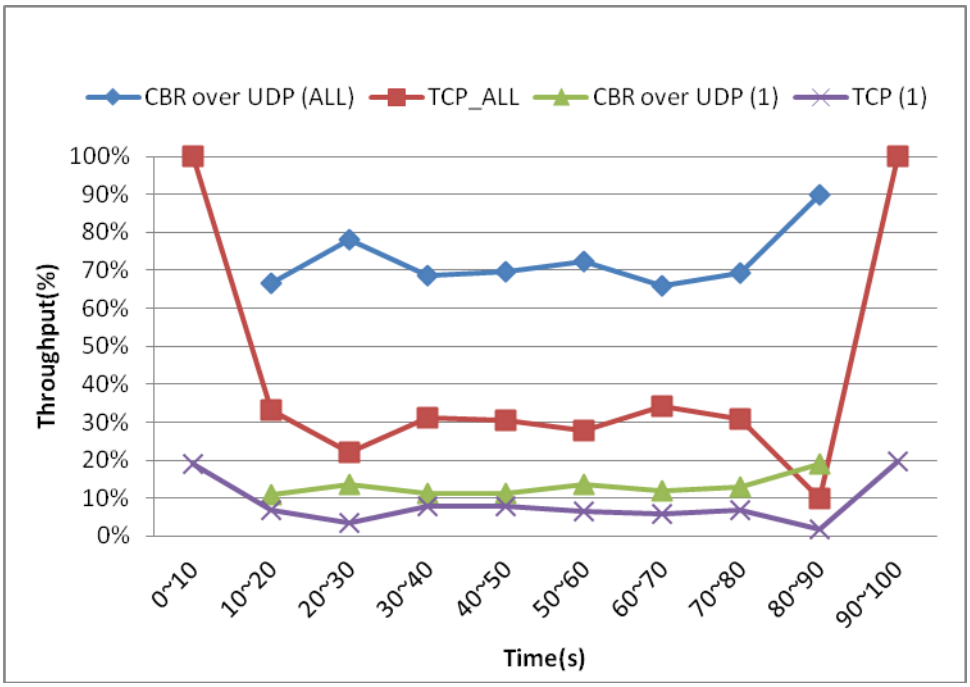

圖 32　CBR over UDP VoIP 與 TCP 吞吐率比較 – TCP 先進入

表 22 CBR over UDP VoIP 與 TCP 吞吐率比較 – CBR over UDP VoIP 先進入

Time(s)	0~10	10~20	20~30	30~40	40~50	50~60	60~70	70~80	80~90	90~100	Average
CBR over UDP VoIP 1	17.65%	11.21%	12.98%	12.50%	12.90%	12.98%	11.81%	10.43%	13.74%	18.40%	13.46%
CBR over UDP VoIP 2	20.59%	14.02%	16.03%	15.63%	16.13%	16.03%	15.75%	15.65%	15.27%	20.80%	16.59%
CBR over UDP VoIP 3	20.59%	13.08%	15.27%	14.84%	14.52%	15.27%	15.75%	14.78%	15.27%	20.80%	16.02%
CBR over UDP VoIP 4	20.59%	12.15%	13.74%	14.06%	13.71%	14.50%	14.96%	12.17%	14.50%	20.00%	15.04%
CBR over UDP VoIP 5	20.59%	10.28%	13.74%	13.28%	12.10%	14.50%	14.17%	11.30%	14.50%	20.00%	14.45%
TCP 1		8.41%	6.11%	5.47%	4.03%	4.58%	7.09%	5.22%	4.58%		5.69%
TCP 2		7.48%	6.11%	4.69%	7.26%	3.82%	6.30%	6.96%	3.82%		5.81%
TCP 3		6.54%	7.63%	5.47%	6.45%	5.34%	3.15%	6.09%	6.11%		5.85%
TCP 4		9.35%	3.82%	7.03%	7.26%	4.58%	4.72%	10.43%	5.34%		6.57%
TCP 5		7.48%	4.58%	7.03%	5.65%	8.40%	6.30%	6.96%	6.87%		6.66%
UDP_ALL	100.00%	60.75%	71.76%	70.31%	69.35%	73.28%	72.44%	64.35%	73.28%	100.00%	75.55%
TCP_ALL		39.25%	28.24%	29.69%	30.65%	26.72%	27.56%	35.65%	26.72%		30.56%
Jain's Index		94.39%	83.01%	85.07%	85.69%	80.99%	81.83%	89.45%	81.68%		85.26%

表 23 CBR over UDP VoIP 與 TCP 吞吐率比較 – TCP 先進入

Time(s)	0~10	10~20	20~30	30~40	40~50	50~60	60~70	70~80	80~90	90~100	Average
CBR over UDP VoIP 1		10.85%	13.48%	11.30%	11.35%	13.49%	11.97%	12.78%	19.09%		13.04%
CBR over UDP VoIP 2		15.50%	17.73%	14.78%	16.31%	16.67%	15.38%	15.79%	15.45%		15.95%
CBR over UDP VoIP 3		13.95%	16.31%	14.78%	14.89%	15.08%	14.53%	15.04%	19.09%		15.46%
CBR over UDP VoIP 4		13.18%	15.60%	13.91%	14.18%	14.29%	12.82%	13.53%	22.73%		15.03%
CBR over UDP VoIP 5		13.18%	14.89%	13.91%	12.77%	12.70%	11.11%	12.03%	13.64%		13.03%
TCP 1	19.09%	6.98%	3.55%	7.83%	7.80%	6.35%	5.98%	6.77%	1.82%	19.67%	8.58%
TCP 2	19.09%	6.98%	4.96%	6.09%	6.38%	5.56%	5.98%	6.77%	0.91%	19.67%	8.24%
TCP 3	20.91%	6.98%	4.26%	5.22%	6.38%	4.76%	6.84%	6.77%	0.91%	19.67%	8.27%
TCP 4	20.91%	4.65%	4.96%	6.09%	5.67%	7.14%	5.13%	4.51%	3.64%	21.31%	8.40%
TCP 5	20.00%	7.75%	4.26%	6.09%	4.26%	3.97%	10.26%	6.02%	2.73%	19.67%	8.50%
UDP_ALL		66.67%	78.01%	68.70%	69.50%	72.22%	65.81%	69.17%	90.00%		72.51%
TCP_ALL	100.00%	33.33%	21.99%	31.30%	30.50%	27.78%	34.19%	30.83%	10.00%	100.00%	41.99%
Jain's Index		88.66%	75.46%	86.85%	85.23%	82.41%	88.60%	86.15%	58.96%		81.54%

4.4.2 Scalable Codec VoIP vs TCP 實驗結果與分析

我們實驗觀察Scalable Codec VoIP先進入網路，之後TCP再進入網路中頻寬競爭的狀況。圖 33、表 24中記錄著Scalable Codec VoIP先進入網路以Scalable Codec VoIP在頻寬不足時和TCP同時存在於網路中所佔的網路Throughput百分比變化。在頻寬瓶頸為256Kbps的網路環境下，在實驗開始時傳送5通Scalable Codec網路電話，在實驗第10秒時，在網路中再加入5條TCP傳送檔案資料流，在實驗第90秒時，結束所有TCP資料流，並在實驗第100秒時結束實驗。

在實驗開始時，因為網路中只存在Scalable Codec VoIP，所以Scalable Codec VoIP占據了所有的Throughput比例，到了實驗第10秒時，因為TCP資料流的加入，因為頻寬不足，Scalable Codec VoIP與TCP會競爭網路Throughput，在實驗第10到20秒時，可看出TCP因為緩啟動快速增長CWND，一開始的總體TCP可占據快接近60%的Throughput，而Scalable Codec VoIP偵測到網路壅塞，啟動壅塞控制機制，降低傳送的語音編碼層次，傳送較小的封包，因此Throughput所佔據的比例降為40%左右。但是在實驗第20到30秒時，隨著TCP偵測到網路壅塞的狀況，啟動壅塞控制機制，TCP因為壅塞控制反應占據的Throughput比例不再上升，並且下降為50%左右。在實驗第30到90秒，因為TCP是以AIMD的方式做壅塞控制，所以總體Scalable Codec VoIP與TCP所佔據的Throughput有時TCP比較高，有時Scalable Codec VoIP比較高，但是 平均都維持在50%左右。頻寬公平指標Jain's Index的表現平均有到97.32%，可看出已非常接近最公平的狀況。由此可看出TCP後來進入網路中的情況下，雖然剛開始因為TCP的緩啟動機制以及Scalable Codec VoIP的壅塞控制機制，TCP占據較多的頻寬。但是隨著TCP感受到網路壅塞狀況後啟動壅塞控制機制，總體平均看來和Scalable Codec VoIP所占據的Throughput比例較為公平。

此外，我們也實驗觀察TCP先進入網路，之後Scalable Codec VoIP再進入網路中頻寬競爭的狀況。圖 34、表 25中記錄著TCP先進入網路以Scalable Codec VoIP在頻寬不足時和TCP同時存在於網路中所佔的網路Throughput百分比變化。在頻寬瓶頸為256Kbps的網路環境下，在實驗開始時傳送5條TCP傳送檔案資料流，在實驗第10秒時，在網路中再加入5通Scalable Codec 網路電話，在實驗第90秒時，結束所有TCP資料流，並在實驗第100秒時結束實驗。

在實驗開始時，因為網路中只存在TCP資料流，所以TCP占據了所有的Throughput比例，到了實驗第10秒時Scalable Codec VoIP加入，因為頻寬不足，Scalable Codec VoIP與TCP會競爭網路Throughput，在實驗第10到90秒，可以看出在Scalable Codec VoIP加入後，因為偵測到網路壅塞，啟動壅塞機制，而TCP也偵測到網路壅塞AIMD方式做壅塞控制機，雖然因為TCP的AIMD方式，讓總體Scalable Codec VoIP與TCP所佔據的Throughput有時TCP比較高，有時Scalable Codec VoIP比較高，但是平均都維持在50%左右。頻寬公平指標Jain's Index的表現平均有到96.60%，也非常接近最公平的狀況。由此可看出在TCP先進入網路的狀況下，Scalable Codec VoIP再進入網路中，在兩者壅塞機制的平衡下，總體平均看來TCP和Scalable Codec VoIP所占據的Throughput比例是公平的。

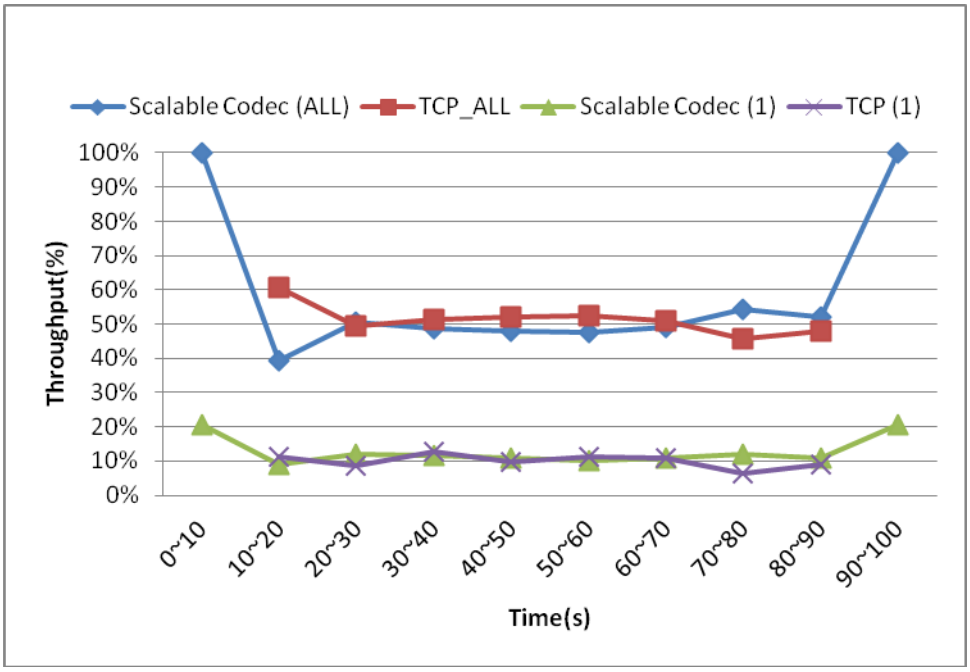

圖 33　Scalable Codec VoIP 與 TCP 吞吐率比較 – Scalable Codec VoIP 先進入

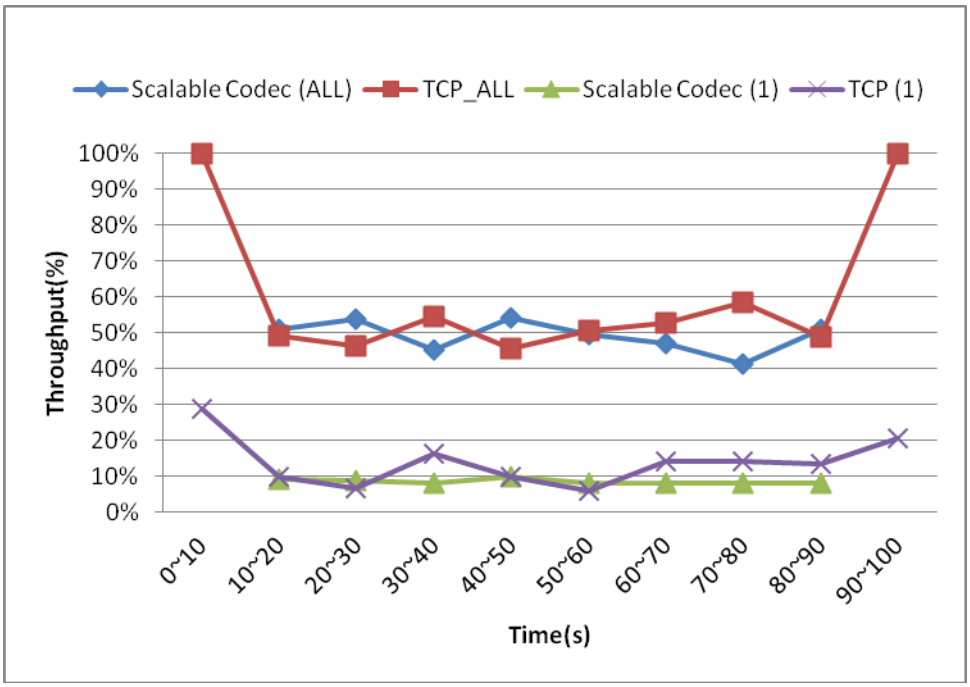

圖 34　Scalable Codec VoIP 與 TCP 吞吐率比較 – TCP 先進入

表 24 Scalable Codec VoIP 與 TCP 吞吐率比較 – Scalable Codec VoIP 先進入

Time(s)	0~10	10~20	20~30	30~40	40~50	50~60	60~70	70~80	80~90	90~100	Average
Scalable Codec VoIP 1	17.65%	6.74%	8.60%	8.14%	8.82%	8.41%	8.70%	8.70%	8.82%	17.81%	10.24%
Scalable Codec VoIP 2	20.59%	8.99%	11.83%	11.63%	10.78%	10.28%	10.87%	11.96%	10.78%	20.55%	12.83%
Scalable Codec VoIP 3	20.59%	8.99%	10.75%	10.47%	9.80%	10.28%	10.87%	11.96%	10.78%	20.55%	12.50%
Scalable Codec VoIP 4	20.59%	7.87%	9.68%	9.30%	9.80%	9.35%	9.78%	10.87%	10.78%	20.55%	11.86%
Scalable Codec VoIP 5	20.59%	6.74%	9.68%	9.30%	8.82%	9.35%	8.70%	10.87%	10.78%	20.55%	11.54%
TCP 1		11.24%	8.60%	12.79%	9.80%	11.21%	10.87%	6.52%	8.82%		9.98%
TCP 2		7.87%	11.83%	10.47%	10.78%	10.28%	9.78%	7.61%	10.78%		9.93%
TCP 3		12.36%	8.60%	9.30%	10.78%	10.28%	9.78%	9.78%	6.86%		9.72%
TCP 4		13.48%	9.68%	12.79%	8.82%	11.21%	11.96%	13.04%	9.80%		11.35%
TCP 5		15.73%	10.75%	5.81%	11.76%	9.35%	8.70%	8.70%	11.76%		10.32%
UDP_ALL	100.00%	39.33%	50.54%	48.84%	48.04%	47.66%	48.91%	54.35%	51.96%	100.00%	58.96%
TCP_ALL		60.67%	49.46%	51.16%	51.96%	52.34%	51.09%	45.65%	48.04%		51.30%
Jain's Index		92.20%	98.62%	96.05%	99.17%	99.30%	98.86%	96.16%	98.23%		97.32%

表 25 Scalable Codec VoIP 與 TCP 吞吐率比較 – TCP 先進入

Time(s)	0~10	10~20	20~30	30~40	40~50	50~60	60~70	70~80	80~90	90~100	Average
Scalable Codec VoIP 1		9.09%	8.79%	8.14%	9.78%	8.08%	8.24%	8.08%	8.16%		8.55%
Scalable Codec VoIP 2		10.91%	12.09%	10.47%	11.96%	11.11%	10.59%	9.09%	11.22%		10.93%
Scalable Codec VoIP 3		10.91%	10.99%	9.30%	10.87%	10.10%	9.41%	9.09%	11.22%		10.24%
Scalable Codec VoIP 4		10.00%	10.99%	9.30%	10.87%	10.10%	9.41%	8.08%	10.20%		9.87%
Scalable Codec VoIP 5		10.00%	10.99%	8.14%	10.87%	10.10%	9.41%	7.07%	10.20%		9.60%
TCP 1	28.85%	10.00%	6.59%	16.28%	9.78%	6.06%	14.12%	14.14%	13.27%	20.69%	13.98%
TCP 2	17.31%	10.91%	8.79%	6.98%	7.61%	13.13%	10.59%	14.14%	8.16%	19.54%	11.72%
TCP 3	13.46%	10.91%	10.99%	12.79%	7.61%	10.10%	10.59%	12.12%	7.14%	20.69%	11.64%
TCP 4	23.08%	10.00%	12.09%	10.47%	9.78%	11.11%	9.41%	7.07%	11.22%	18.39%	12.26%
TCP 5	17.31%	7.27%	7.69%	8.14%	10.87%	10.10%	8.24%	11.11%	9.18%	20.69%	11.06%
UDP_ALL		50.91%	53.85%	45.35%	54.35%	49.49%	47.06%	41.41%	51.02%		49.18%
TCP_ALL	100.00%	49.09%	46.15%	54.65%	45.65%	50.51%	52.94%	58.59%	48.98%	100.00%	60.66%
Jain's Index		98.85%	96.85%	93.60%	98.19%	96.96%	97.48%	93.81%	97.06%		96.60%

4.4.3 Scalable Codec VoIP 之間的頻寬競爭實驗結果與分析

我們實驗觀察在頻寬不足時，多個Scalable Codec VoIP(沒有其他TCP)的狀況下，Scalable Codec VoIP之間的頻寬競爭公平性。在頻寬瓶頸為256Kbps的網路環境下，在實驗開始時傳送1通Scalable Codec VoIP，之後實驗每隔10秒時，在網路中再加入1通Scalable Codec VoIP，直到實驗第90秒時，網路中同時存在10通Scalable Codec VoIP，在實驗第100秒時結束1通Scalable Codec VoIP，之後實驗每隔10秒時，再結束1通Scalable Codec VoIP，直到實驗第190秒時結束所有Scalable Codec VoIP。圖 35中顯示每通Scalable Codec VoIP進入網路時在於網路中所佔的網路Throughput除以所有使用頻寬的百分比變化。圖 36中顯示實驗時所有Scalable Codec VoIP之間的頻寬公平指標Jain's Index的表現。

在實驗開始時，因為網路中只存在1通Scalable Codec VoIP，所以 該通Scalable Codec VoIP占據了所有的有使用的Throughput比例，到了實驗第0到60秒中間，每10秒加入1通Scalable Codec VoIP，因為頻寬足夠，每通Scalable Codec VoIP都可擁有足夠的頻寬使用較佳的語音品質，因此每通電話都可擁有固定的Throughput，所以就算隨著通數增加每通所佔的有使用的Throughput比例下降，但每通電話都是擁有差不多的比例，而實驗第0到60秒間的頻寬公平指標Jain's Index都維持在97%以上。實驗第60到90秒，陸續增加的通話數已經超出頻寬負荷，Scalable Codec VoIP偵測到網路的狀況後，將調整語音的層次，由於每通Scalable Codec VoIP可能偵測到的狀況有些微的差異，所以每通Scalable Codec VoIP的Throughput比例會有些微差距，頻寬公平指標Jain's Index也因此有些微的下降，但是每通電話之間並沒有過大的差距，Jain's Index還是幾乎維持在95%以上。實驗第100到190秒，隨著通話數慢慢的減少，瓶寬再度慢慢恢復足夠的狀況，每通Scalable Codec VoIP的Throughput比例之間的差距也慢慢減小。由此結果顯示Scalable Codec VoIP之間可以公平的競爭頻寬。

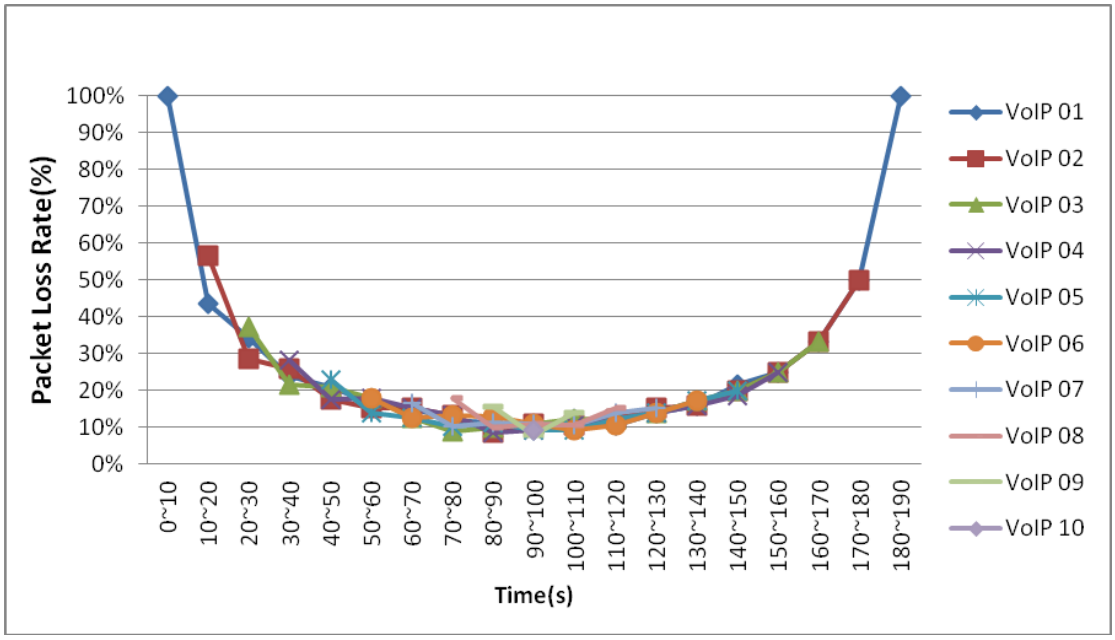

圖 35　多個 Scalable Codec VoIP(沒有其他 TCP)的吞吐率比較

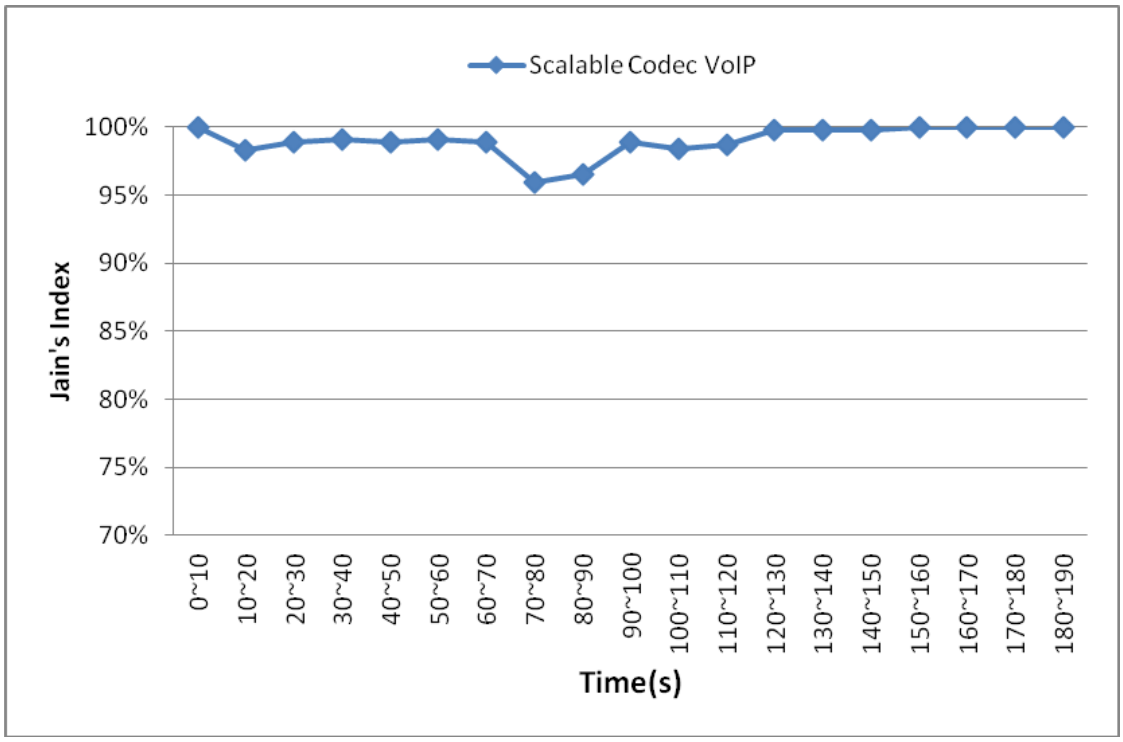

圖 36　多個 Scalable Codec VoIP(沒有其他 TCP)的 Jain's Index 比較

表 26 多個 Scalable Codec VoIP 的吞吐率以及 Jain's Index — 實驗 0 到 100 秒

Time(s)	0~10	10~20	20~30	30~40	40~50	50~60	60~70	70~80	80~90	90~100
Scalable Codec VoIP 1	100.00%	43.48%	34.29%	23.91%	21.05%	16.67%	15.19%	11.94%	11.43%	9.38%
Scalable Codec VoIP 2		56.52%	28.57%	26.09%	17.54%	15.28%	15.19%	13.43%	8.57%	10.94%
Scalable Codec VoIP 3			37.14%	21.74%	21.05%	18.06%	12.66%	8.96%	10.00%	10.94%
Scalable Codec VoIP 4				28.26%	17.54%	18.06%	15.19%	13.43%	8.57%	9.38%
Scalable Codec VoIP 5					22.81%	13.89%	12.66%	10.45%	11.43%	9.38%
Scalable Codec VoIP 6						18.06%	12.66%	13.43%	12.86%	10.94%
Scalable Codec VoIP 7							16.46%	10.45%	11.43%	10.94%
Scalable Codec VoIP 8								17.91%	10.00%	10.94%
Scalable Codec VoIP 9									15.71%	7.81%
Scalable Codec VoIP 10										9.38%
Jain's Index	100.00%	98.33%	98.87%	99.06%	98.90%	99.08%	98.95%	95.92%	96.53%	98.94%

表 27 多個 Scalable Codec VoIP 的吞吐率以及 Jain's Index — 實驗 100 到 190 秒

Time(s)	100~110	110~120	120~130	130~140	140~150	150~160	160~170	170~180	180~190
Scalable Codec VoIP 1	12.31%	10.77%	13.85%	15.94%	21.54%	25.00%	33.33%	50.00%	100.00%
Scalable Codec VoIP 2	10.77%	12.31%	15.38%	15.94%	20.00%	25.00%	33.33%	50.00%	
Scalable Codec VoIP 3	12.31%	12.31%	13.85%	17.39%	20.00%	25.00%	33.33%		
Scalable Codec VoIP 4	10.77%	12.31%	13.85%	15.94%	18.46%	25.00%			
Scalable Codec VoIP 5	9.23%	12.31%	13.85%	17.39%	20.00%				
Scalable Codec VoIP 6	9.23%	10.77%	13.85%	17.39%					
Scalable Codec VoIP 7	10.77%	13.85%	15.38%						
Scalable Codec VoIP 8	10.77%	15.38%							
Scalable Codec VoIP 9	13.85%								
Scalable Codec VoIP 10									
Jain's Index	98.42%	98.71%	99.76%	99.81%	99.76%	100.00%	100.00%	100.00%	100.00%

4.4.4 頻寬競爭能力實驗結果與分析

圖 37結果顯示，如果是CBR over UDP VoIP或Scalable Codec VoIP先進入的情況下，在TCP剛進入時會因為緩啟動而先搶得一些頻寬，但是如果在TCP和CBR over UDP VoIP競爭的狀況下，UDP並不會依照網路壅塞的狀況而有壅塞控制，而搶佔了60%以上的Throughput。TCP則除了剛開始的緩啟動搶得頻寬外，之後隨著偵測網路壅塞，而以AIMD的壅塞控制機制，雖搶得的頻寬有變化，但是持續因為網路壅塞而啟動壅塞控制機制，讓大部份的頻寬被UDP搶占，只能獲得較不公平的Throughput比例。可是相較於將CBR over UDP VoIP改為Scalable Codec VoIP的狀況下，因為Scalable Codec VoIP也偵測到網路壅塞的狀況，啟動壅塞控制機制，所以雖然TCP因為以AIMD的方式做壅塞控制而有變化，但是平均來說和Scalable Codec VoIP維持約50%左右的Throughput，圖 39中則可看出此情景下改為Scalable Codec VoIP的狀況下，頻寬競爭評比指標Jain's Index幾乎都高於使用CBR over UDP VoIP的狀況，大約高於12%左右，可知擁有較公平的Throughput分配比例。

圖 38結果顯示，在TCP先進入的狀況下，如果在頻寬不足下和後進入的CBR over UDP VoIP競爭頻寬，UDP沒有依照網路狀況作調整，持續傳送封包而搶占大部分頻寬，而TCP則會因為壅塞控制機制而退讓，只搶得平均30%左右不公平的Throughput比例。如果改以Scalable Codec VoIP和TCP競爭頻寬，因為Scalable Codec VoIP偵測到網路壅塞，啟動壅塞控制機制，和TCP競爭頻寬時雖然TCP因為AIMD的方式做壅塞控制而搶得的比例時高時低，但是平均來說和Scalable Codec VoIP搶占各約50%左右的Throughput比例，圖 40中也可看出由此可看出使用Scalable Codec VoIP的狀況Jain's Index也有15%左右的改善，由此可表示，此研究提出的方法可以在頻寬不足的情況下友善且公平的競爭頻寬。

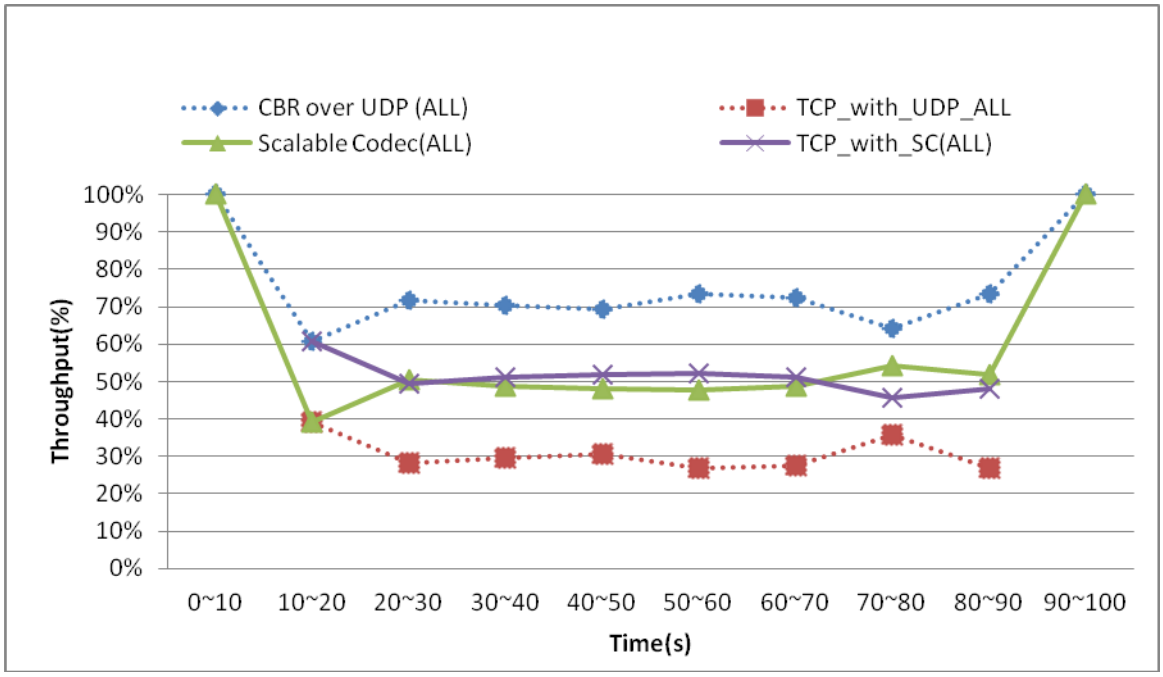

圖 37　頻寬競爭能力實驗結果比較

– CBR over UDP 或 Scalable Codec VoIP 先進入

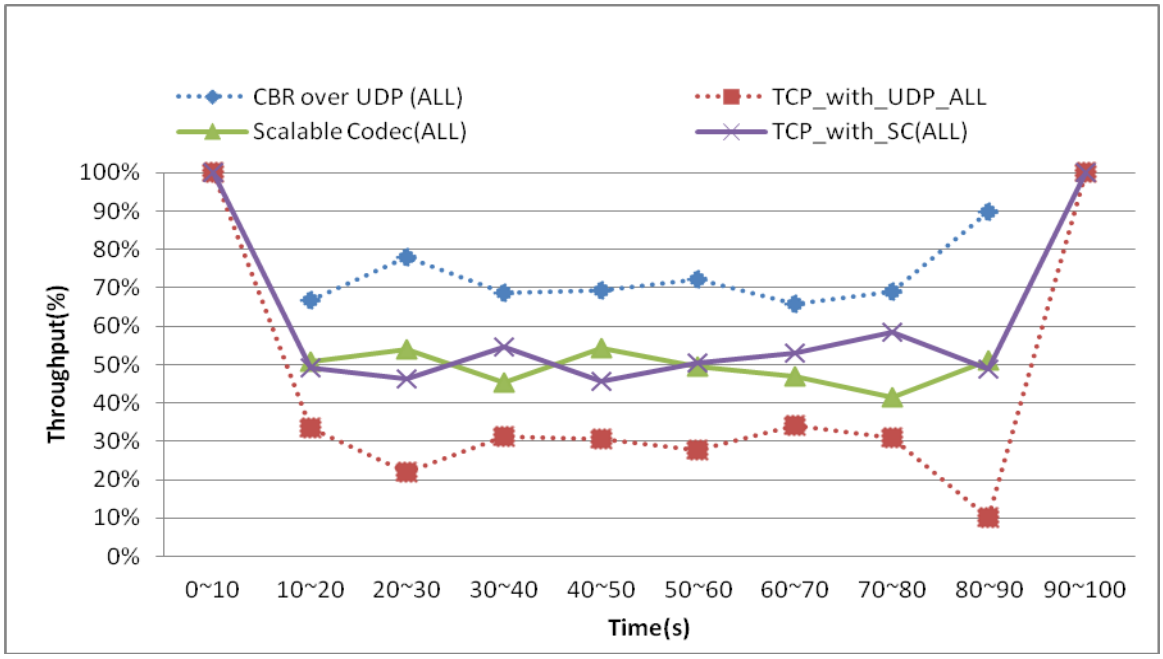

圖 38　頻寬競爭能力實驗結果比較 – TCP 先進入

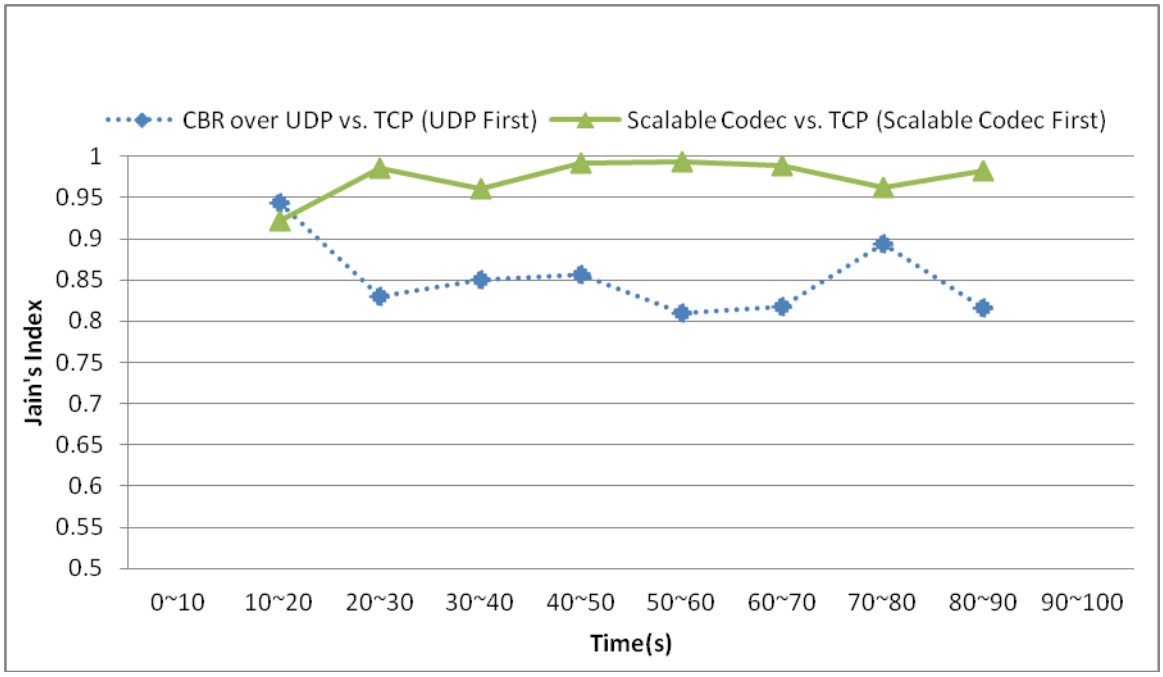

圖 39　頻寬競爭評比指標 Jain's Index 變化
– CBR over UDP 或 Scalable Codec VoIP 先進入

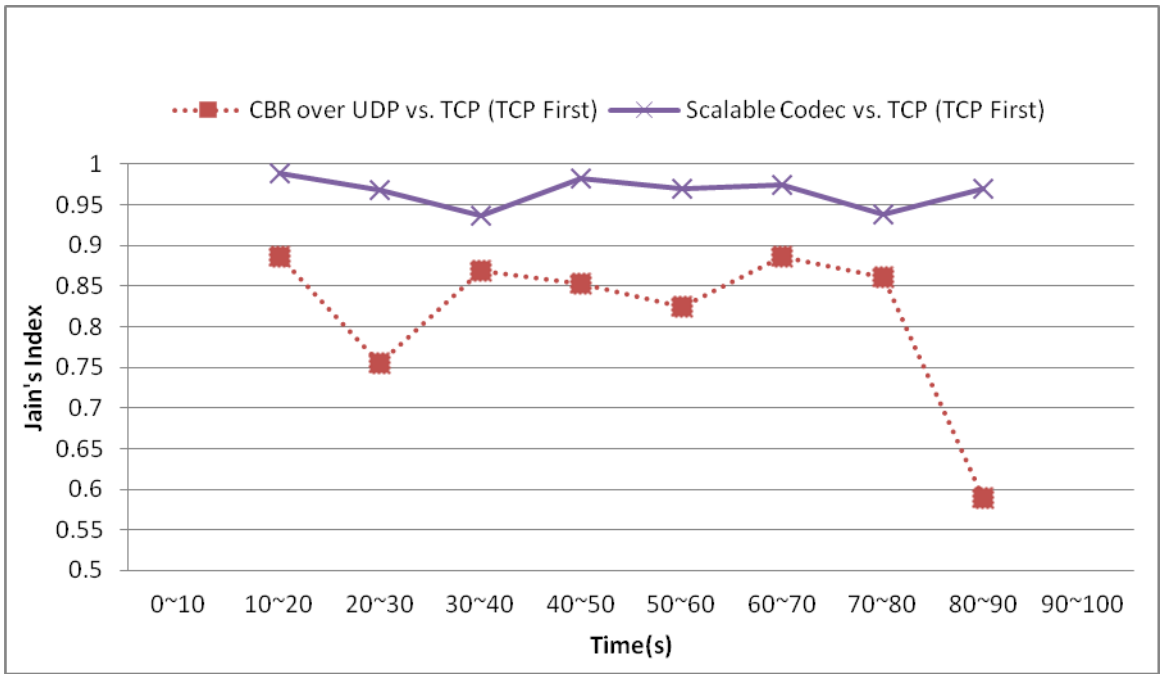

圖 40　頻寬競爭評比指標 Jain's Index 變化　– TCP 先進入

圖 41結果顯示，如果是CBR over UDP VoIP或Scalable Codec VoIP先進入的情況下，在實驗0到10秒時因為頻寬足夠，沒有任何的封包遺失。在實驗10到90秒時因為TCP的加入，網路頻寬不足，VoIP開始出現封包遺失的狀況，可看出因為TCP以AIMD的方式不斷的測試搶佔頻寬，在TCP的壅塞控制機制下會讓UDP維持大約封包遺失率大約在20%到40%之間，大約在UDP封包遺失率到達30%以上，TCP才會做速度上的退讓。如果改以Scalable Codec VoIP仍然無法改變TCP對於頻寬競爭的強勢，TCP依舊在造成在VoIP封包遺失率達30%以上時才做退讓，因此雖然Scalable Codec VoIP可以和TCP友善公平的競爭頻寬，但是對於在和TCP共存的情況下，還是無法避免和TCP競爭頻寬時，對於VoIP語音品質的傷害。

圖 42結果顯示，在TCP先進入的狀況下，如果在頻寬不足下和後進入的CBR over UDP VoIP競爭頻寬，VoIP因為頻寬不足在競爭頻寬時同樣也會出現封包遺失的狀況，封包遺失率平均在20%以上。如果改以Scalable Codec VoIP依舊會在和TCP競爭頻寬時發生封包遺失的狀況，封包遺失率平均同樣也在20%以上。顯示在TCP先進入的狀況下，Scalable Codec VoIP可以和TCP友善公平的競爭頻寬，但是對於在和TCP共存的情況下，還是無法避免和TCP競爭頻寬時，對於VoIP語音品質的傷害。

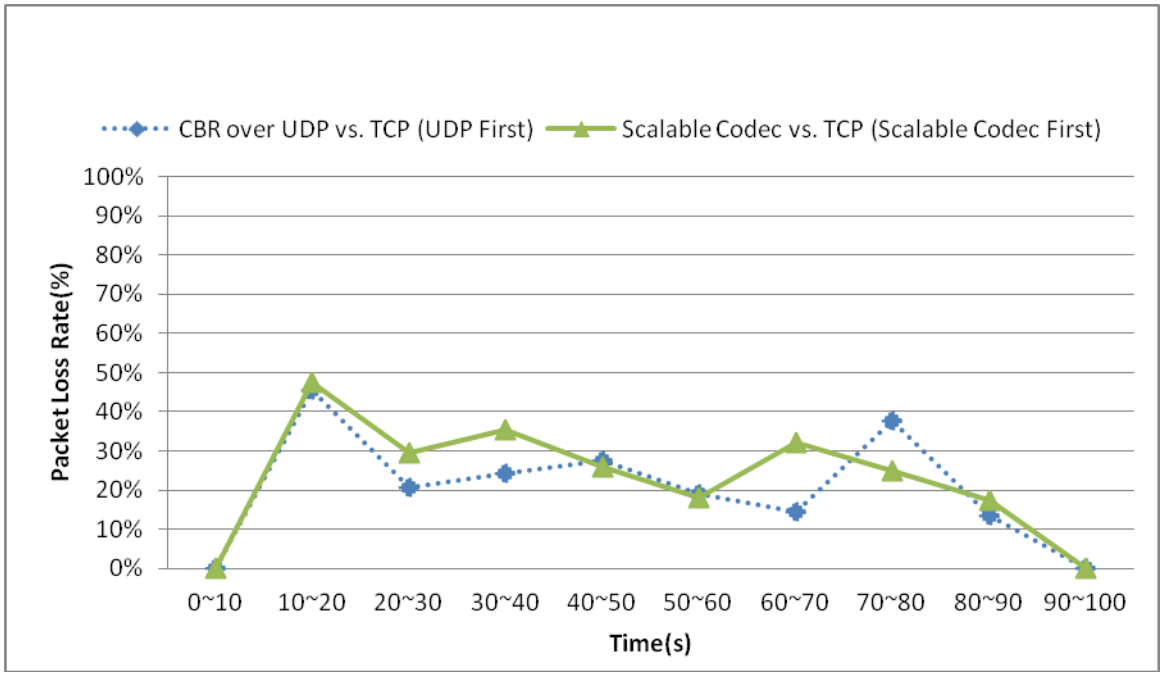

圖 41　封包遺失率變化

– CBR over UDP 或 Scalable Codec VoIP 先進入

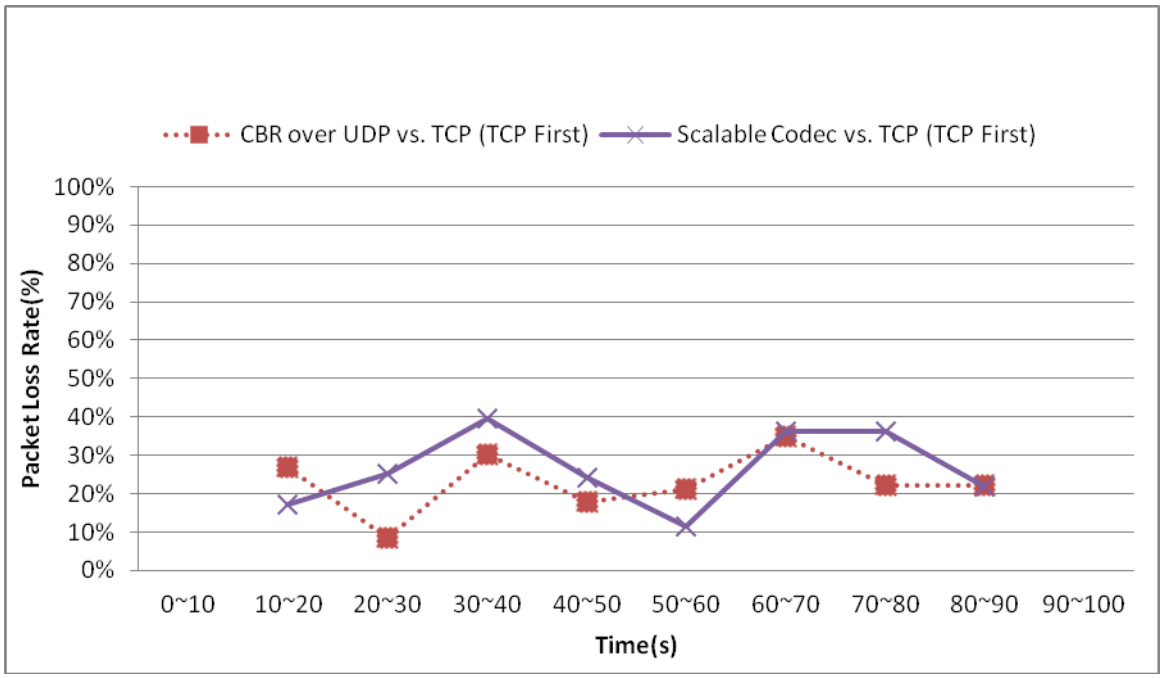

圖 42　封包遺失率變化 – TCP 先進入

表 28 CBR over UDP VoIP 與 TCP 封包遺失率比較 – CBR over UDP VoIP 先進入

Time(s)	0~10	10~20	20~30	30~40	40~50	50~60	60~70	70~80	80~90	90~100	Average
CBR over UDP VoIP 1	0.00%	45.58%	19.91%	25.22%	26.55%	20.80%	15.49%	43.81%	10.62%	0.00%	
CBR over UDP VoIP 2	0.00%	37.40%	13.78%	18.50%	16.93%	14.17%	11.42%	25.98%	11.02%	0.00%	
CBR over UDP VoIP 3	0.00%	42.52%	19.29%	21.65%	26.38%	16.93%	11.81%	31.50%	13.39%	0.00%	
CBR over UDP VoIP 4	0.00%	47.64%	24.80%	25.98%	31.10%	20.08%	15.35%	41.73%	15.75%	0.00%	
CBR over UDP VoIP 5	0.00%	53.94%	25.59%	30.31%	37.01%	23.23%	18.50%	44.88%	16.93%	0.00%	
Average	0.00%	45.41%	20.68%	24.34%	27.59%	19.04%	14.51%	37.58%	13.54%	0.00%	25.34%

表 29 CBR over UDP VoIP 與 TCP 封包遺失率比較 – TCP 先進入

Time(s)	0~10	10~20	20~30	30~40	40~50	50~60	60~70	70~80	80~90	90~100	Average
CBR over UDP VoIP 1		31.42%	11.95%	33.19%	27.43%	21.24%	35.84%	19.91%	12.75%		
CBR over UDP VoIP 2		18.90%	0.39%	25.98%	5.91%	10.24%	24.80%	12.20%	17.52%		
CBR over UDP VoIP 3		25.59%	5.91%	28.35%	11.81%	18.90%	30.31%	19.69%	27.45%		
CBR over UDP VoIP 4		28.35%	9.45%	30.71%	17.72%	25.20%	37.80%	25.98%	24.32%		
CBR over UDP VoIP 5		29.92%	13.78%	32.28%	26.77%	30.71%	45.28%	32.68%	29.67%		
Average		26.83%	8.29%	30.10%	17.93%	21.26%	34.81%	22.09%	22.34%		22.96%

表 30 Scalable Codec VoIP 與 TCP 封包遺失率比較 – Scalable Codec VoIP 先進入

Time(s)	0~10	10~20	20~30	30~40	40~50	50~60	60~70	70~80	80~90	90~100	Average
Scalable Codec VoIP 1	0.00%	51.77%	35.84%	38.94%	23.45%	16.81%	35.84%	30.97%	19.47%	0.00%	
Scalable Codec VoIP 2	0.00%	37.80%	20.47%	28.35%	21.26%	13.78%	25.59%	18.50%	15.75%	0.00%	
Scalable Codec VoIP 3	0.00%	43.31%	26.38%	31.89%	26.77%	16.54%	29.53%	21.65%	16.14%	0.00%	
Scalable Codec VoIP 4	0.00%	49.21%	30.71%	36.61%	28.74%	19.69%	32.28%	25.20%	17.32%	0.00%	
Scalable Codec VoIP 5	0.00%	55.12%	33.86%	40.94%	29.53%	23.23%	38.19%	27.95%	17.72%	0.00%	
Average	0.00%	47.44%	29.45%	35.35%	25.95%	18.01%	32.29%	24.86%	17.28%	0.00%	28.83%

表 31 Scalable Codec VoIP 與 TCP 封包遺失率比較 – TCP 先進入

Time(s)	0~10	10~20	20~30	30~40	40~50	50~60	60~70	70~80	80~90	90~100	Average
Scalable Codec VoIP 1		15.49%	29.65%	42.04%	26.11%	22.57%	37.17%	31.42%	22.06%		
Scalable Codec VoIP 2		13.78%	17.32%	34.25%	20.08%	5.12%	32.28%	29.92%	18.11%		
Scalable Codec VoIP 3		14.96%	23.23%	37.40%	23.23%	8.27%	35.04%	34.25%	20.87%		
Scalable Codec VoIP 4		20.08%	26.38%	40.94%	24.80%	9.45%	37.40%	40.55%	22.83%		
Scalable Codec VoIP 5		22.05%	29.13%	43.70%	26.77%	11.81%	38.98%	45.28%	25.98%		
Average		17.27%	25.14%	39.67%	24.20%	11.44%	36.17%	36.28%	21.97%		26.52%

第五章 結論與未來研究

壅塞控制是網路管理的重大問題，隨著網路網路普及，網路電話以及視訊實況轉播等即時網路多媒體應用興起，但是即時的網路應用程式大部分是採用UDP做為傳輸協定，UDP沒有壅塞控制機制。此研究主要探討的是：如何讓網路電話使用具有壅塞控制功能的傳輸協定？以及如何在網路壅塞狀況時可以維持較好的網路電話語音品質？此研究提出的TCP-friendly VoIP By Scalable Codec over DCCP，是利用DCCP偵測網路狀況的封包來回時間以及封包遺失率來判斷網路壅塞的程度，並設計出細緻的語音層次，讓網路的壅塞時可以依照壅塞程度有更適切的反應。

本研究透過實際網路的實驗環境中評估以CBR over UDP、Flexible Bit-Rate以及Scalable Codec三種方式傳輸網路電話封包的效能，結果顯示透過Scalable Codec方法，能有效降低網路電話的封包遺失率，維持通話品質，而且相較於Flexible Bit-Rate可以有更準確的壅塞偵測以及更合適的壅塞控制反應。研究結果顯示在網路壅塞最嚴重的情況下可以和一般的CBR配合UDP的封包遺失率達到40%左右的改善，與Flexible Bit-Rate方法也的達到8%以上的改善，並且整體網路狀況更為穩定。而在和TCP頻寬競爭實驗中可看出，在頻寬不足的情況下，此研究提出的方法可以和TCP友善且公平的競爭頻寬。

本研究中僅粗略的讓語音編碼配合DCCP，由於DCCP尚未常見於一般傳輸層協定，許多DCCP配合僅依照[10]中所定義的壅塞偵測方法做實做，未來將針對語音編碼器與DCCP做更細微的配合。

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]S. Andersen and A. Duric, “Internet Low Bit Rate Codec (iLBC), “ IETF RFC 3951, Dec. 2004.

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

[4]Ming Cao, "EVCCM: An Efficient VOIP Congestion Control Mechanism," IEEE GLOBECOM Workshops, 2008

[5]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, 1989, pp. 1-14.

[6]A. Falk, D. Katabi, Y. Pryadkin," Specification for the Explicit Control Protocol (XCP)," draft-falk-xcp-03.txt (work in progress), Jul. 2007.

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

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

[9]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.

[10] 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.

[11] Z. Fu, P. Zerfos, H. Luo, S. Lu, L. Zhang, and M. Gerla, "The impact of multihop wireless channel on TCP throughput and loss," IEEE INFOCOM, San Francisco, CA, Mar. 2003.

[12] M. Handley, "TCP Friendly Rate Control (TFRC):Protocol Specification," IETF RFC 3448, Jan. 2003.

[13] ITU-T Rec. P.800, "Methods for Subjective Determination of Transmission Quality," Aug. 1996.

[14] ITU-T Rec. G.107, "The E-Model, A Computational Model for Use in Transmission Planning, " May 2000.

[15] ITU-T Rec. G.723.1, "Dual Rate Speech Coder for Multimedia Communications Transmitting at 5.3 and 6.3 kbit /s," May 2006.

[16] ITU-T Rec. G.729, ”Coding of speech at 8 kbit/s using Conjugate-structure Algebraic-code Excited Linear-prediction (CS-ACELP),” Jun. 2012.

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

[18] E. Kohler, M. Handley and S. Floyd, "Datagram Congestion Control Protocol (DCCP)," IETF RFC 4340, Mar. 2006.

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

[20] Y.N. Lien, Y.C. Ding, "Congestion Control Enabled VoIP by Flexible Bit-rate" M.A. thesis, University of Chengchi, Taiwan, 2010

[21] De Cicco, L. and S. Mascolo, "A Mathematical Model of the Skype VoIP Congestion Control Algorithm," IEEE Transactions on Automatic Control, Mar. 2010

[22] J.Nagle, "Congestion Control in TCP/IP," RFC896, Jan. 1984.

[23] K. Nahm, A. Helmy, and C.-C J. Kuo, "TCP over Multihop 802.11 Networks: Issues and Performance Enhancement," ACM MobiHoc 05, Urbana-Champaign, Illinois, USA, May 2005.

[24] F. Sabrina and J.-M. Valin,"Adaptive Rate Control for Aggregated VoIP

Traffic," GLOBECOM 2008, pp. 1405-1410.

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

[26] Q.C. Wu, J.Y Wang, "Porting VoIP applications to DCCP", in Proc. of the International Conference on Mobile Technology, Applications, and Systems Article no. 8,2008

[27] Xiph.org Foundation, "The Speex Codec Manual Version 1.2 Beta 3", 2007.