國 立 交 通 大 學

電信工程學系

碩 士 論 文

正弦音長調整在網路語音封包播放時序之應用

Adaptive Playout Scheduling for VoIP with Sinusoidal Time-Scale Modification

研 究 生：蔡淑羚

指導教授：張文輝 博士

中 華 民 國 九 十 四 年 六 月

正弦音長調整再網路語音封包播放時序之應用

Adaptive Playout Scheduling for VoIP with Sinusoidal Time-Scale Modification

研 究 生：蔡淑羚　　**Student：Shu-Lin Tsai**

指導教授：張文輝　　**Advisor：Wen-Whei Chang**

國立交通大學

電信工程學系碩士班

碩士論文

A Thesis

Submitted to Institute of Communication Engineering

College of Electrical Engineering and Computer Science

National Chiao Tung University

In Partial Fulfillment of Requirements

for the Degree of

Master of Science

In Electrical Engineering

June 2004

Hsinchu, Taiwan, Republic of China

中華民國　九十四年　六月

正弦音長調整在網路語音封包播放時序之應用

學生：蔡淑羚　　指導教授：張文輝 博士

國立交通大學電信工程學系碩士班

中文摘要

由於網際網路語音通訊的即時性，接收端會將語音封包暫存在一緩衝器中並延遲其播放時間，以補償網路延遲顫動的影響。針對三種適應性播放時序演算法，包含一個以話務為基礎及兩個以封包為基礎的演算法，在平均緩衝延遲和晚到漏失率之間做取捨衡量。對於以封包為基礎的延遲調整來說，配合使用一種基於諧波正弦表示的語音分析合成機制，對個別的語音封包做適當的音長重建以達到連續播放語音的效果。每個正弦波組成的貢獻包括描述振福、激發相位和聲道系統的參數，控制這些參數以完成音長比例調整。我們的網路模擬是基於單一伺服器的排隊模型，一連串定期的語音封包受網際網路串流的影響，則以一批伯努利串流近似。模擬結果顯示配合正弦音長調整，NLMS時序演算法能使緩衝延遲和晚到漏失率之間有最好的取捨衡量，同時明顯改善其播放語音的品質。

Adaptive Playout Scheduling for VoIP with Sinusoidal Time-Scale Modification

Student：Shu-Lin Tsai　　Advisor：Dr. Wen-Whei Chang

Institute of Communication Engineering

National Chiao Tung University

Abstract

For real-time voice communication over internet networks, voice packets are buffered at a receiver and their playout delayed in order to compensate for the network delay jitter. Three adaptive playout scheduling algorithms, one per-talkspurt based and two per-packet based, are examined for a trade-off between average buffering delay and late loss rate. For per-packet based delay adjustment, proper reconstruction of continuous playout speech is achieved by time-scaling individual voice packets using a new technique based on a sinusoidal representation of the speech production mechanism. The proposed time-scale modification involves the manipulation of functions which describes the amplitude and phase of the excitation and vocal tract system contributions to each sine-wave component. Our network simulation is based on a single-server queueing model in which the impact of the internet traffic on a periodic stream of audio packets is approximated by a batch Bernoulli traffic. Simulation results indicate that with the aid of sinusoidal time-scale modification, NLMS-based scheduling algorithm improves the playout speech intelligibility with the best trade-off between buffering delay and late loss rate.

誌謝

本篇論文的完成，首先要感謝指導教授張文輝老師的悉心指導，在研究所兩年的諄諄教誨，讓我從一個對研究懵懂的新生，進而了解做研究的態度與方法，更使我無論在學業或生活上皆有深刻的體認與成長。另外也要感謝實驗室的學長、同學及學弟們，協助我解決課業或研究及生活上的問題，同時也要感謝一路上陪伴我的好友們，讓我緊繃的心情，找到宣洩的出口。最後我要感謝的父母及家人，給我最大的支持與鼓勵，使我順利完成研究所學業。僅此以本論文獻給我最親愛的家人及所有關心我的人。

目錄

圖目錄

第一章 緒論

網路的時代已經來臨，網際網路的快速發展，將資訊交流推向另一個高峰。以往網路的應用只限於網路資料的查詢，以及一些網路資料傳送，而傳送的資料所重視的是可否能完整傳送到接收端，而不需考慮資料何時傳送到接收端。但是隨著網路快速的發展與研究，在網路上傳送即時(Real-time)的多媒體資料將是一個發展重點，其中更以語音為發展重點。語音(Voice)資料大小遠比影像(Video)資料來的小，應用到網際網路上更能減低資料在網路上的延遲(Delay)，相較之下，在網路上傳送語音比影像更能達到即時的效果。

多媒體資料在有即時性的考慮下，就不能單單考慮資料能否完整的傳送到接收端，還要考慮資料是否能在限定的時間內到達接收端。如果資料在預定播放的時間之後才到達的話，則視為無用的資料。VoIP(Voice over IP)就是在網際網路上利用這樣的即時觀念而發展出來的。

VoIP 顧名思義就是利用網際網路來即時的傳送語音資料，由於目前長途電話或國際電話的費用仍然非常的昂貴，所以一些業者就利用免費的網際網路來提供長途或國際電話服務，因為它只需要付給當地電信業者(ISP)一些上網線路費用，與長途或國際電話無關。所以費

用比傳統電話(Public Switched Telephone Network, PSTN)費用便宜很多，並提供傳統電話所不能提供的加值服務(如傳送文件、顯示通話端的影像等等)。因此近年來 VoIP 以成為許多公司及學校努力發展的重點技術，而其成效也日益成熟。

1.1 研究動機與方向

在網路網路上，通常在語音即時通訊上最主要關心的課題就是服務品質(Quality of service, QoS)的問題。然而現今網際網路的不可靠及不定性，導致封包網路傳輸的延遲(delay)不固定甚至會有漏失(loss)的情形，嚴重影響服務品質。傳輸延遲、延遲顫動(delay jitter)，及漏失這三項因素是現今即時語音通訊中所面臨最主要的課題。在通訊架構中各不同的層級皆嘗試做大量的改進以期能使延遲降低、顫動更平穩且回復漏失的封包。

一個具體可行的方式為在接收端的應用層(application layer)中控制每個語音封包的播放時間(playout time)。在語音封包播放前，先暫存在一播放緩衝器(playout buffer)中以吸收延遲顫動的影響。當使用這個技巧，語音封包在接收後並不會立即被播放出去，而是會被暫存在緩衝器中直到排定的播放時間為止。雖然這種方式會導致早到封包的額外延遲，但也使得晚到封包被順利播放的機率提高了。因此，語

音封包在緩衝器暫存的時間(buffering delay)及因為晚到而被視為漏失的比率(late loss rate)之間，面臨了一個權衡取捨的問題。若排定一個較晚的播放時間，將會提高更多封包播放的機率而得到較低的封包漏失率，但這就會導致有較高的緩衝延遲；反之亦然，要減少緩衝延遲，很難在不增加漏失率下達到。通常在一般的語音對話中，語音封包從傳送端產生至接收端播放出來的延遲在 400ms 內及漏失率最高到 5%之內是可被容忍的。

在本論文中，我們介紹三種不同的播放時序演算法(playout scheduling algorithm)，其中兩種是對每個獨立的封包都做播放時間的調整，以適應網路在話務(talkspurt)中間變動的情形。為了保持語音播放的連續性，必須配合使用一種音長比例調整(time-scale modification)的技術，可以明顯的降低延遲及漏失率進而增加整體的效能。

1.2 章節概要

第二章介紹三種不同的播放緩衝時序演算法，第三章則介紹利用諧波正弦(sinusoidal)語音轉換機制來做音長比例的調整，第四章則先介紹一種網路延遲模型，並進而客觀比較三種演算法的效能分析和主觀比較聽覺品質上的差異。

第二章　播放時序演算法

網路電話(VoIP)是近年來吸引企業界在網際網路上應用的一種服務，因為它能節省大筆的長途或國際電話費。但 VoIP 仍有些問題尚待解決，大致上會遇到的問題為 end-to-end delay、delay jitter、packet loss、echo 等。聲音在網路上傳送通常是被切割成一個一個封包，所以封包到達時的延遲(Delay)和封包漏失數(Losses)，視為評估網路電話品質好壞的準則。

在傳送端，語音信號會以固定的間隔來產生封包並透過網際網路傳送到接收端。每個封包的網路延遲會取決於所走的路徑及該路徑上路由器的擁塞程度而有所不同，而這些網路延遲的差異即為延遲顫動(delay jitter)。為降低顫動在接收端的影響，封包在播放前會先被暫存在一緩衝器中一小段時間；嚴重晚到的封包，即封包在排定的播放時間後才到達，則被視為漏失(Lost)。藉由增加緩衝器延遲(buffer delay)，漏失的封包將會減少；然而封包的整體延遲(end-to-end delay)將會增加。有鑑於此，本論文的研究重點是期望在封包的漏失率及平均整體延遲之間找一個權衡點。

這一章將說明藉由播放緩衝器來降低延遲顫動的影響，首先介紹在接收端播放緩衝器的角色；接下來探討目前相關研究中三種主要的

播放時序演算法(playout scheduling algorithm)，同時針對網路延遲的一特殊現象－Spike 作調整。

2.1 播放緩衝器簡介

圖 2.1 說明了顫動所造成的問題，及在接收端使用播放緩衝器的必要性。語音信號以固定的時間間隔 L 產生封包並經由網路傳送，因為網路本身的特性，每個封包延遲並不會固定，導致有些封包會在它們的播放時間之後才到達。

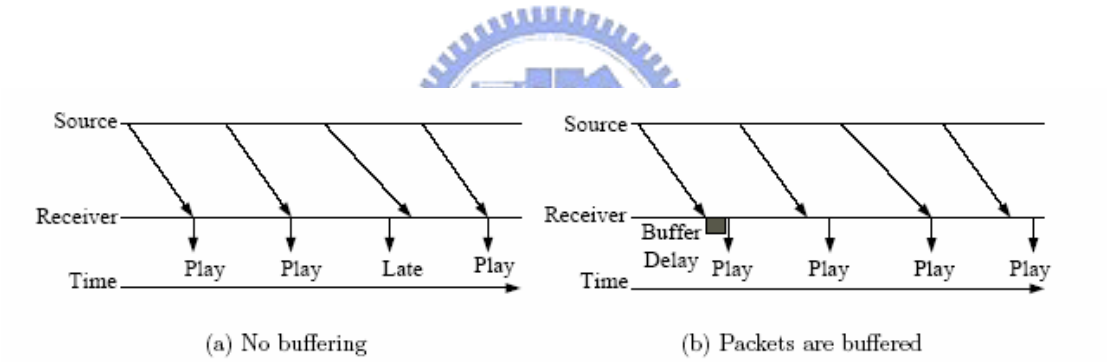


圖 2.1 播放緩衝器的影響

在缺乏播放緩衝器的情形下，封包會在被接收到的同時即被播放出去，如圖 2.1(a)所示。第一個封包抵達時間即為其開始播放時間，接下來的第 i 個封包將以和第一個封包的播放時間間隔 $(i\text{-}1)L$ 作為播放時間。然而，較大的網路延遲就會造成晚到的封包無法播出，這樣的情形下會導致大部分的封包漏失而降低通話品質。一具體可行的解

決方案是加入播放緩衝器，如圖 2.1(b)，將封包暫存一小段時間再播放。此方法可大幅減少封包因晚到而漏失的機率，但整體延遲將擴大為網路延遲與緩衝延遲的總合。

圖 2.2 說明三種基本的播放時序設計，其中語音封包的網路延遲以黑點表示，整體延遲以實線表示。當排定的播放時間越晚，整體的延遲就越大。封包在播放時間之後到達，例如黑點在實線之上，即判定為漏失。這些播放時序設計的理想目標是盡可能使實線降低(降低整體的延遲)，同時使黑點在實線以上的數目越少越好(晚到漏失最小化)。

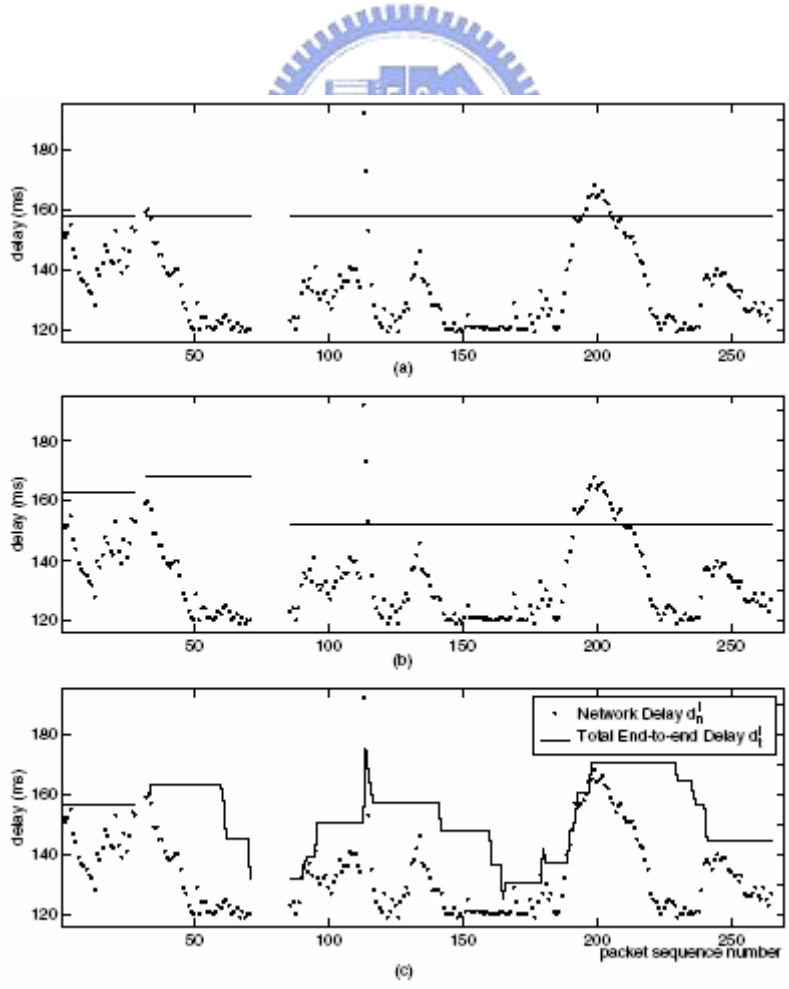


圖 2.2 播放時序演算法的影響

最簡單的方法，如圖 2.2(a)所示，對每一個語音封包皆使用固定的整體延遲，但這方法並不能有效的使延遲和漏失同時降低。主要是因為網路的特性隨時間而改變，而固定的播放時間並不能反映這些變化。有鑑於此，較合理的播放時序演算法是參考網路延遲的變化，適應性地調整不同區間的整體延遲時間。因為基於對話的特性，聲音會被靜音(silence)區間區分為數個話務(talkspurt)區間。靠著延長或壓縮靜音區間的長度來調整個別話務的播放時間，圖 2.2(b)可看出比圖 2.2(a)的結果好。這種方法是以話務為單位作調整，會因為話務區間太長及在一個話務中網路延遲變化太大而使其效力受限。舉例來說，在第三個話務內的第 113、114、115 個封包有瞬間很大的網路延遲，也是一般常見的 Spike 發生。此種方法並不能適應這種情形，導致許多封包漏失造成聽覺上聆聽品質的降低。

最理想的播放時序設計，應該是不只調整靜音區間也可針對話務內個別封包做調整。每個獨立的封包都可根據變動的延遲統計來決定其不同的排定播放時間，其結果可由圖 2.2(c)表示。這較新的演算法可以透過動態性及反應性較佳的方法來有效的調整播放時間同時降低漏失率及平均延遲。但特別強調的是，這種以封包為單位作調整的演算法，必須配合音長調整(time-scaling)以保持連續的播放效果。

長期觀察網路延遲的實地量測數據，可發現許多網路延遲皆會有

spike 現象的發生。若一封包突然有網路延遲大幅度增加的現象，視為 spike 的開始點，雖然接下來的封包通常網路延遲都會遞減，但它們延遲仍然很大。當網路延遲回到一個穩定狀態值時，則視為 spike 的結束。網路延遲的 spike 是由於網路路由器突然發生大量壅塞的排隊現象造成，典型的延遲 spike 如圖 2.3 所示。有鑑於此，有效的播放時序演算法必須配合加入 spike 偵測機制作合理規劃。

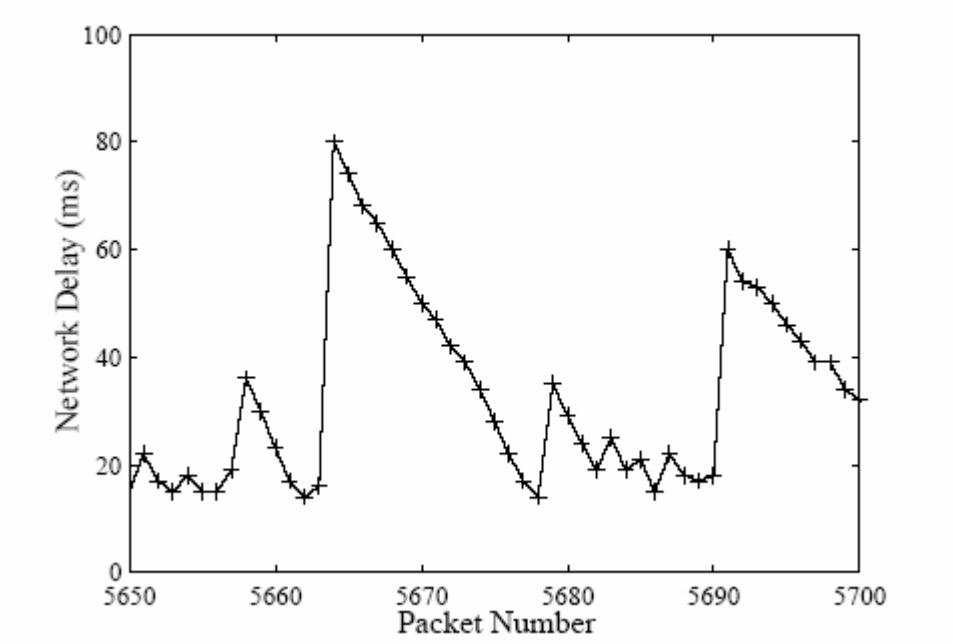


圖 2.3 典型網路 spike 現象

2.2 客觀效能評估分析

在前一節中，關於播放時序設計的目的及其適應性調整的好處已經有直覺性的概念，我們還需要定義一個客觀的效能分析方法。接下來先介紹本論文會用到的一些基本標號，然後再定義兩項效能評估參

數：平均緩衝延遲(average buffering delay)及晚到漏失率(late loss rate)。

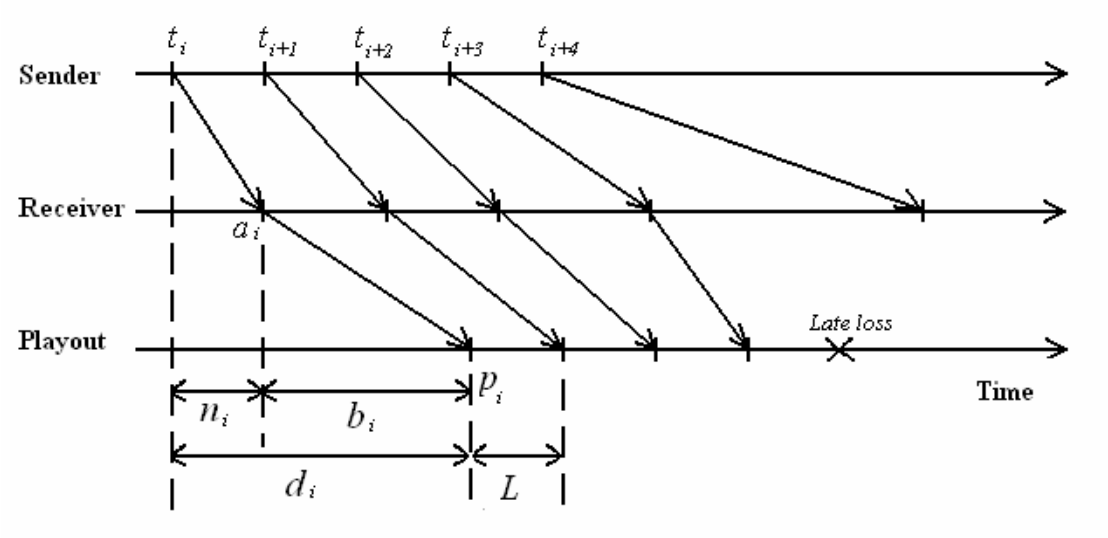


圖 2.4 第 i 個封包的相關時間參數

語音以固定長度編碼後並封裝成固定大小的封包傳送出去，每個封包的間隔為 L。如圖 2.4 所示，下標 $i=1,2,...,N$ 表示封包的次序號碼，假設總共有 N 個封包被傳送。圖 2.4 所示的為單一封包長度不做調整的播放演算法的圖例，接下來就介紹一些基本標號：

t_i：第 i 個封包的傳送時間。

a_i：第 i 個封包的接收時間。

p_i：第 i 個封包的播放時間。

b_i：第 i 個封包播放前在緩衝器內的等待時間，$b_i = p_i - a_i$。

d_i：第 i 個封包從傳送直到播放所經過的時間差，$d_i = p_i - t_i$，即為第

i 個封包的播放延遲(playout delay)。

n_i：第 i 個封包因傳輸所造成的網路延遲，$n_i = a_i - t_i$，主要包含兩部分：一為傳送端到接收端的固定傳播延遲(propagation delay)，另一則為傳送過程的排隊延遲(queueing delay)。

當我們評斷不同的播放時序設計時，主要針對兩個數值作效能評估。第一個是平均緩衝延遲，如下所示：

$$d_b = \frac{1}{|\mathrm{P}|}\sum_{i\in \mathrm{P}}(d_i - n_i)\,, \qquad (2.1)$$

其中 $\mathrm{P} = \{i \mid p_i > a_i\}$，為可被播放出來封包的集合，$|\mathrm{P}|$ 為此集合的個數。第二個數值為晚到漏失率，定義如下：

$$\varepsilon = \left(N - |\mathrm{P}|\right)/N\,, \qquad (2.2)$$

其中我們假設所有的封包都能順利抵達接收端，沒有任何傳輸漏失。這兩個數值可反映出上面提及到的漏失及延遲的取捨衡量，也可用來比較不同播放時序演算法的效能分析。

2.3 適應性播放時序演算法

適應性播放時間的調整方式主要分成兩大類，第一類為每一個話務作一次調整(per-talkspurt)，第二類則為每一個封包都做調整(per-packet)。第一類的調整方式主要著重在各個話務的第一個封包：

- 如果第 i 個封包為第 k 個話務的第一個封包，它的播放時間 p_i 可以

下式計算：

$$p_i = t_i + D^k \text{，} \qquad (2.3)$$

其中 D^k 為第 k 個話務所屬個別封包的固定整體延遲。

- 在第 k 個話務接下來的封包，其整體延遲皆和第一個封包相同，所以若第 j 個封包存在於第 k 個話務內，其播放時間可定義為：

$$p_j = t_j + D^k = p_i + t_j - t_i \text{，} \qquad (2.4)$$

第二類的調整方式則是針對話務內每個封包都做調整，所以每個語音封包又必須作音長調整的動作，使它們可以剛好在下一個封包的預測到達時間內播出。動態的調整播放時間在封包漏失的容忍範圍下，可降低整體延遲進而有效的改善系統通話品質。音長調整方式將於下一章節說明。接下來說明本論文主要用到的三種播放延遲估計方法，其中自迴歸[1]估計屬於第一類(per-talkspurt)的演算法，而統計分析[2][3]及適應性濾波器[4][5]則屬於第二類(per-packet)的演算法。

2.3.1 基於自迴歸估計的演算法

這個演算法採用第一類 per-talkspurt 的調整方式，其整體延遲 D^k 只有在一個新的話務開始時才更新，但每個封包的網路延遲 n_i 仍用於其平均延遲 $\hat{d}_i$ 及變異數 $\hat{v}_i$ 的估計。主要是利用自迴歸(autoregressive, AR)的方式，來估計網路延遲的平均值及其變異數。在正常模式下，

對第 i 個封包而言，估計其平均網路延遲 $\hat{d}_i$ 為：

$$\hat{d}_i = \alpha \hat{d}_{i-1} + (1-\alpha) n_i \text{，} \qquad (2.5)$$

其中 n_i 為第 i 個封包的網路延遲，α 則為加權參數用以控制演算法的收斂速度。在前人研究[1]中，α 值設為 0.875。網路延遲的變異數也以自迴歸方式估計：

$$\hat{v}_i = 0.875\hat{v}_{i-1} + 0.125\left|\hat{d}_i - n_i\right| \text{。} \qquad (2.6)$$

至於整體延遲 d_i，則以下列公式計算：

$$d_i = \hat{d}_i + \beta \hat{v}_i \text{，} \qquad (2.7)$$

其中 $\beta\hat{v}_i$ 為一安全緩衝項次用來確保整體延遲足夠大到減少封包因晚到而漏失的機率。β 參數用來調節整體延遲和晚到封包漏失之間的取捨衡量，一個較大的 β 值會導致一個較低的漏失率，也就是更多的封包可在預定播放時間前到達，然而整體延遲就會增加。

在這演算法中，有關平均延遲 $\hat{d}_i$ 的估計又可區分為正常及 spike 兩種模式，在正常模式下 $\hat{d}_i$ 是依據(2.5)式子估計。若偵測到有 spike 發生，平均延遲 $\hat{d}_i$ 的估計只會和最近的網路延遲有關。更明確地說：

$$\hat{d}_i = \hat{d}_{i-1} + (n_i - n_{i-1}) \text{，} \qquad (2.8)$$

當 spike 結束後，演算法就回到原來正常的模式估計其平均延遲。

一個延遲 spike 的開始是有一個急速增加的延遲特徵，可據以設計 spike 的偵測機制。若兩個相鄰封包的網路延遲相差值超過一個臨

界值，$|n_i - n_{i-1}| > 2|v_{i-1}| + 20$ ，就判斷為 spike 的開始。其後持續量測網路延遲的斜率，當延遲較平坦而斜率小於一個臨界值，此時演算法就會判斷 spike 結束並回到正常的模式。自迴歸演算法的詳細虛擬程式碼如下所示。

// AR-based scheduling Algorithm

1. n_i *=Receiver_timestamp – Sender_timestamp;*

2. if (mode==NORMAL){

if($|n_i - n_{i-1}| > 2|\hat{v}_{i-1}| + 20$*){*

var=0; / Detected beginning of spike */*

mode=IMPULSE;

}

else{

var=var/2+$|2n_i - n_{i-1} - n_{i-2}|/8$*;*

if(var<=8){

mode=NORMAL; / End of spike */*

$n_{i-2} = n_{i-1}$*;*

$n_{i-1} = n_i$*;*

return;

}

}

3. if (mode==NORMAL)

$\hat{d}_i = 0.125n_i + 0.875\hat{d}_{i-1}$*;*

else

$\hat{d}_i = \hat{d}_{i-1} + (n_i - n_{i-1})$*;*

$\hat{v}_i = 0.125|n_i - \hat{d}_i| + 0.875\hat{v}_{i-1}$

4. $n_{i-2} = n_{i-1}$*;*

$n_{i-1} = n_i$*;*

return;

2.3.2 基於統計分析的演算法

此方法屬於第二類(per-packet)的演算法，主要是統計過去封包的網路延遲，再據以計算下一個封包的播放延遲，以期晚到封包漏失的比率在可控制在人耳容忍範圍內。在前人研究[3]中，先紀錄過去 w 個封包的延遲，下一個封包的整體延遲 d_{i+1} 則根據一個使用者事先指定的漏失率 $\hat{\varepsilon}$ 來決定，而下一個封包必須在最後期限前到達才可順利被播放出來。下面將詳述 d_{i+1} 的計算方法。

過去 w 個封包的網路延遲，$n_{i-w+1}, n_{i-w+2}, \ldots, n_i$，依其大小作排序可得到：

$$D^1 \le D^2 \le \ldots \le D^w, \tag{2.9}$$

則網路延遲 n 不大於第 r 個次序統計 D^r 的機率分佈為：

$$F\left(D^r\right) = P\left(n \le D^r\right),\ r = 1,2,\ldots,w \text{。} \tag{2.10}$$

及其期望值為：

$$\mathrm{E}\left(F\left(D^r\right)\right) = \frac{r}{w+1},\ r = 1,2,\ldots,w\ , \tag{2.11}$$

此值代表封包可在 D^r 之前抵達接收端的期望機率。

在此應用中，我們將原本的次序統計擴展，加入最小的可能延遲：

$$D^0 = \max\left(D^1 - 2s_n, 0\right)\ , \tag{2.12}$$

及最大的可能延遲：

$$D^{w+1} = D^w + 2s_n\ , \tag{2.13}$$

其中 s_n 為 $n_{i-w+1}, n_{i-w+2}, \ldots, n_i$ 的標準差(standard deviation)。依此得到擴展的次序統計及期望播放機率分別為：

$$D^0 \le D^1 \le \ldots \le D^w \le D^{w+1} \text{ ，}$$

$$\mathrm{E}\left(F\left(D^r\right)\right) = \frac{r}{w+1},\ r = 0,1,\ldots,w+1 \text{。} \qquad (2.14)$$

這個擴展的用意，是為了解決原本的期望播放機率無法低於 $1/(w+1)$ 或高於 $w/(w+1)$ 的問題。

我們的任務是要達到使用者指定漏失率 $\hat{\varepsilon}$ 的要求下，找出一個最適當的可能延遲 $D^{\hat{r}}$，及其相對應的索引值 $\hat{r}$。以另一個角度來看，我們要找出滿足 $\mathrm{E}\left(F\left(D^{\hat{r}}\right)\right) \le 1-\hat{\varepsilon}$ 的條件下最大的 $D^{\hat{r}}$。由(2.14)，可得到相對應的索引 $\hat{r} = \left\lfloor (w+1)(1-\hat{\varepsilon}) \right\rfloor$，而播放延遲 d_{i+1} 可用 $D^{\hat{r}}$ 及 $D^{\hat{r}+1}$ 的內插來近似：

$$d_{i+1} = D^{\hat{r}} + \left(D^{\hat{r}+1} - D^{\hat{r}}\right)\left[(w+1)(1-\hat{\varepsilon}) - \hat{r}\right] \text{，} \qquad (2.15)$$

此種以統計分析為基礎的估算法，有一個重要的特色就是允許使用者自訂可接受的漏失率 $\hat{\varepsilon}$，而演算法就可據以自動調整相對的延遲。因此，緩衝器延遲和漏失率之間的取捨也可明確的被控制。一個更精確的延遲分布可以利用一個較大的 w 而得，然而缺點是較不適用於具高度變動性的網路延遲。因此，w 的選擇決定了演算法適應變動量的速度，必須及在準確度和反應度之間作取捨考量。

在以統計為基礎的演算法中，也同樣加入 spike 偵測的機制，偵

測方式和之前自迴歸演算法提到的方式相同。在 spike 模式下，第一個封包視為漏失，其餘的封包延遲則設為第一個封包的延遲值。延遲統計在 spike 模式時並不作更新的動作，直到回到正常模式後，就以之前所儲存的延遲來估計延遲。

2.3.3 基於適應性濾波器的演算法

最近提出的一種演算法[5]，直接用適應性濾波器演算法來預測網路延遲，而播放延遲就以網路延遲的預測值及其變異數來決定。若能精確地預測網路延遲，可以快速追蹤到網路流量的變化，自然能夠更有效率地調整延遲。

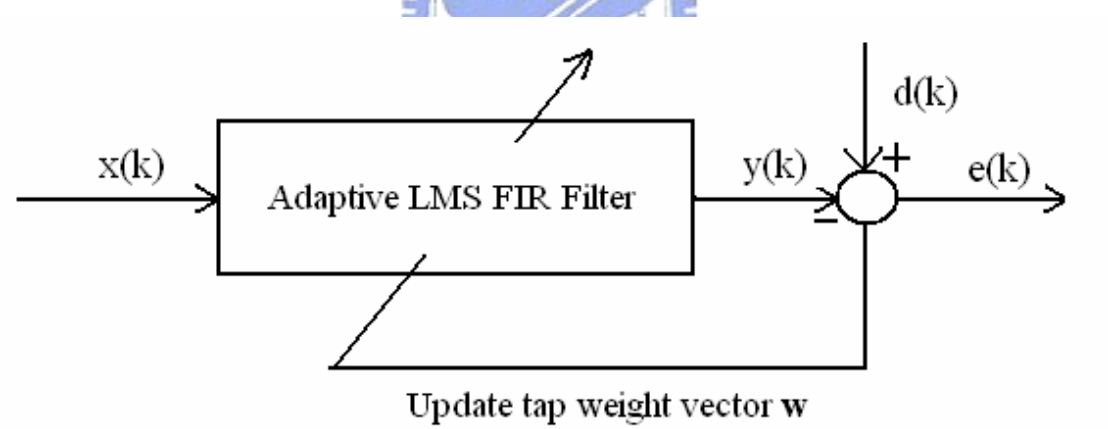


圖 2.5 適應性濾波器系統方塊圖

傳統上，適應性濾波器演算法通常是用在等化器及回音消除器上，主要目的為使實際的數據和估計值之間的均方差期望值最小化。過去的數據被暫存在一有限脈衝響應(FIR)濾波器中以用來計算現在

的估計值，均方差用來調整適應性濾波器的係數向量。適應性濾波器的系統方塊圖如圖 2.5 所示。

在本論文中，採用正規化最小均方(Normalized least mean square, NLMS)演算法來做適應性預測。第 i 個封包的網路延遲預測值 $\hat{d}_i$ 可用下式表示：

$$\hat{d}_i = \underline{w}_i^T \underline{n}_i \ , \qquad (2.16)$$

其中 $\underline{w}_i$ 為 $M\times 1$ 維度的適應性濾波器係數向量，$\underline{n}_i = \{n_{i-1}, n_{i-2}, \cdots, n_{i-M}\}$ 為包含過去 M 個網路延遲的向量。

濾波器的係數向量 $\underline{w}_i$ 根據 NLMS 演算法作更新的動作：

$$\underline{w}_{i+1} = \underline{w}_i + \frac{\mu}{\underline{n}_i^T \underline{n}_i + a} \underline{n}_i e_i \ , \qquad (2.17)$$

其中 μ 為階層(step size)大小，a 為一個很小的常數。而估計誤差 e_i 表示為：

$$e_i = n_i - \hat{d}_i \ , \qquad (2.18)$$

其中 n_i 為及 $\hat{d}_i$ 分別為第 i 個封包網路延遲的實際值與預測值。至於網路延遲的變異數 $\hat{v}_i$ 及整體延遲 d_i ，則是分別用自迴歸演算法的公式(2.6)和(2.7)求得。

圖 2.6 顯示了不考慮 spike 偵測功能的 NLMS 演算法預測的結果。在 spike 發生時，由於延遲突然變大，通常第一個封包都會漏失，而接下來的封包預測會很快的追上變大的值，而使接下來封包的網路

延遲預測值比 spike 模式的第一個封包的延遲還大一點。但在 spike 區間，其延遲特性是會越來越小，且全部的整體延遲 d_i 又會加上安全緩衝項 $\beta\hat{v}_i$，所以導致在 spike 區間的整體緩衝延遲過大。為了改善這種缺點，在 NLMS 演算法中加入 spike 偵測機制，降低 β 值使整體延遲降低。

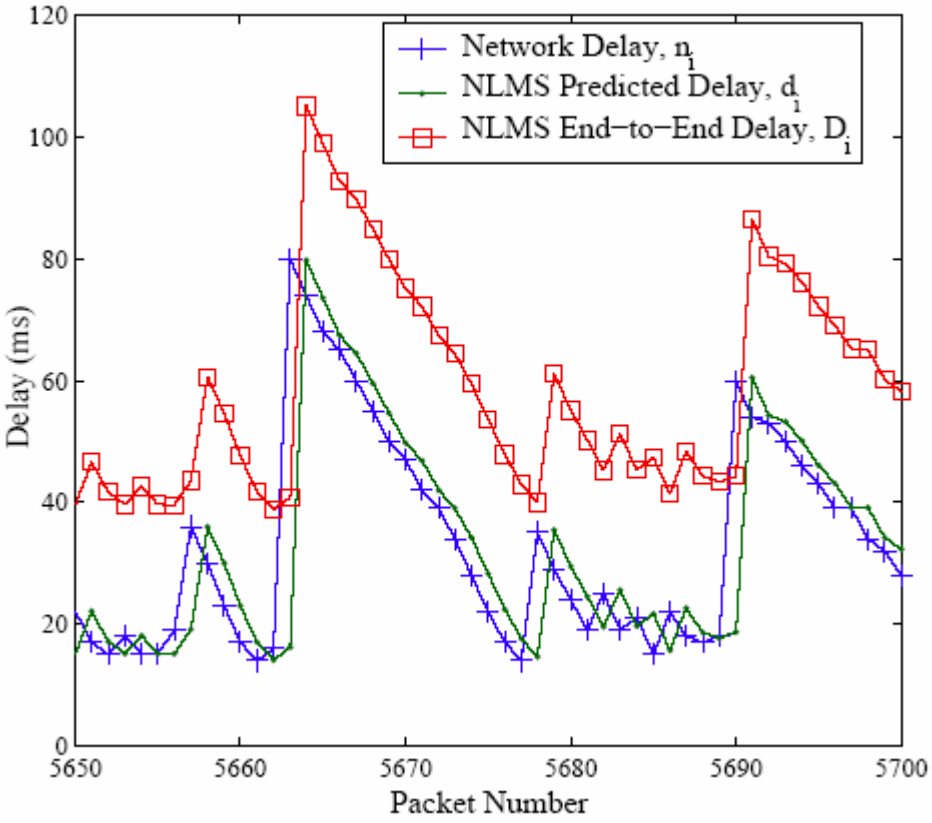


圖 2.6 NLMS 播放演算法

在正常模式下，用 NLMS 演算法配合安全緩衝係數 β 來預測延遲；當前一個封包漏失或實際的網路延遲比前一個延遲差超過一個臨界值，則判斷為 spike 模式。在 spike 模式下，仍用 NLMS 演算法來預測延遲，但安全緩衝係數降低為 $\beta/4$，這樣便可簡單且有效的降低

整體的延遲。此外又加入一個條件，為了不讓播放延遲下降的太快，以前面提到的自迴歸演算法所得到的的延遲當作延遲估計的下限。而當 NLMS 演算法預測出來的延遲$\hat{d}_i$不再大於實際的網路延遲n_i，即結束 spike 模式回到正常模式。圖 2.7 顯示加上 spike 偵測的 NLMS 演算法的結果，可以有效地降低整體延遲。

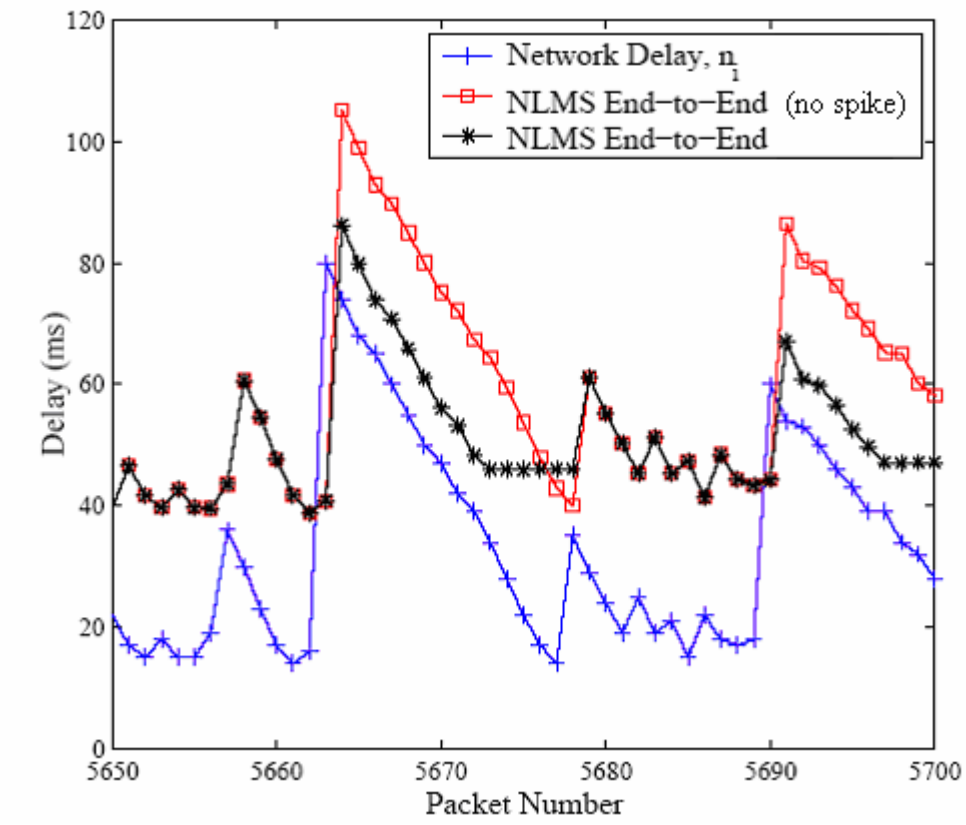


圖 2.7 spike 偵測對 NLMS 演算法的影響

最後，NLMS 演算法的虛擬程式碼如下所示：

// NLMS-based scheduling algorithm

$$\hat{d}_i = \underline{w}_i^T \underline{n}_i;$$
$$ARdelay_i = \alpha ARdelay_{i-1} + (1-\alpha) n_i;$$
$$\hat{v}_i = \alpha \hat{v}_{i-1} + (1-\alpha)\left|\hat{d}_i - n_i\right|;$$

if(mode==SPIKE)

$varfactori = \beta / 4\hat{v}_i$;

$D_i = \hat{d}_i + varfactori;$

if ($D_i < ARdelay_i + \beta\hat{v}_i$)

$D_i = ARdelay_i + \beta\hat{v}_i$;

end

else // Normal mode

$varfactori = \beta\hat{v}_i$;

$D_i = \hat{d}_i + varfactori;$

end

// *if end-to-end delay < network delay*

if ($D_i < n_i$)

packeti=LOST;

else

packeti=IN_TIME;

end

if ($n_i > \hat{d}_i$)

mode=NORMAL;

end

if ($n_i > \hat{d}_i + 5\hat{v}_i$)*or(packeti==LOST)*

mode=SPIKE;

end

// *Update Adaptive NLMS Filter Tap Weights*

$e_i = \hat{d}_i - n_i;$

$\underline{w}_{i+1} = \underline{w}_i + u/(\underline{n}_i^T \underline{n}_i + a)\underline{n}_i e_i;$

第三章　音長比例調整

語音封包在傳送端是以固定間隔產生，在接收端也必須以相同的速率連續播放出來。一般而言，語音封包會先被暫存在緩衝器中，使其能以規律的間隔在排定的時間播放。在第二章介紹的 AR 播放演算法是放大或縮小兩話務間的靜音區間來調整播放時間，因為在靜音區間調整播放延遲不易被使用者察覺。

至於 OS 及 NLMS 兩種演算法，是針對每個封包調整其播放時間，而非每個話務間調整，因此造成每個封包的播放間隔不固定。而為了語音播放的連續性，必須放大或縮短每個語音封包的音框長度。[6][7]在本論文中，我們用正弦編碼轉換(STC)系統來完成音長比例調整的動作[8][12]。利用諧波正弦模型來分析語音信號以擷取其特徵參數，接著透過參數對映函數進行特徵參數的轉換，最後利用轉換後之特徵參數合成聲音輸出。特徵參數轉換機制方面，主要分為頻譜轉換、音高及音長調整，在本論文中，只針對音長調整部分作調整，其餘兩項參數均不變。

在這一章一開始先介紹語音轉換系統，其中包括諧波正弦的分析及合成部份，之後再針對音長調整的部分做詳細說明，並配合前一章的演算法作適當的調整機制。

3.1 語音轉換系統

語音轉換最常見的方式首先是先分析語音信號，得到特徵參數再進行需要的轉換，例如音長或音高調整，最後再利用轉換後的特徵參數合成產生相對應的語音。

3.1.1 諧波正弦分析

諧波正弦分析可有效地模擬聲音的激發源與聲道共振特性。假設語音信號 *s(t)* 是由聲門之激發訊號 *e(t)* 經過一模擬聲道濾波器 $h(\tau,t)$ 產生。激發訊號模擬聲帶以下的發聲特性，例如聲帶振動的頻率、幅度、氣流長短，影響語音的聲調；而聲道濾波器則是模擬口腔以上的結構，包括唇、齒、舌、顎咽部，直接影響語音發出的構音內容。這兩部分由於發音位置不同，可在不受對方的影響下分別進行，將視為獨立分開探討。

其數學關係式可表示為

$$s(t)=\int_0^t h(t-\tau,t)e(t)d\tau \qquad (3.1)$$

而在清／濁音激發模型中，可假設激發訊號 *e(t)* 是由一組具有不同振幅、頻率及相位的正弦波相加而成，則 *e(t)* 可寫成

$$e(t)=\sum_{k=1}^{K(t)} a_k(t)\cdot\cos[\int_0^t w_k(\sigma)d\sigma+\Omega_k(t)] \qquad (3.2)$$

其中 $a_k(t)$，$w_k(t)$，$\Omega_k(t)$ 分別第 k 個諧波對應的激發振幅、頻率及相位，並且設定聲道濾波器之頻率響應為

$$H(w;t) = M(w;t) \cdot \exp[j\psi(w;t)] \tag{3.3}$$

由線性系統的定義可得到

$$\begin{aligned} s(t) &= \sum_{k=1}^{K(t)} a_k(t) M(w_k;t) \cdot \cos[\psi(w_k(t),t) + \Omega_k(t) + \int_0^t w_k(\sigma) d\sigma] \\ &= \sum_{k=1}^{K(t)} A_k(t) \cdot \cos[\theta_k(t) + \int_0^t w_k(\sigma) d\sigma] \end{aligned} \tag{3.4}$$

其中

$$A_k = a_k M(w_k;t) \tag{3.5}$$

$$\theta_k(t) = \psi(w_k(t),t) + \Omega_k(t) \tag{3.6}$$

由此可知，合成的訊號 *s(t)*確實可以用一組不同的振幅、頻率及相位之正弦波元來組成。

3.1.2 語音分析模型

在了解諧波正弦模型的可行性後，進一步找出模型中所需要的各項參數。語音信號一般假設為短期穩定(short-term stationary)，而利用此特性則可以適當的音框(frame)作為處理的單位，並假設音框內的訊號是穩定的(stationary)，亦即特徵參數是固定非時變的。在本論文中，我們的語音信號以 11kHz 的頻率取樣，以 46.4ms 的漢明視窗(Hamming windows)來分析，每 13.6ms 後移一次，因此分析的音框長度為 13.6ms[11]。因此第 *i* 個音框中之語音訊號，可表示如下：

$$s(n)=\sum_{k=1}^{K(i)}A_k(i)\cdot\cos[nw_k(i)+\theta_k(i)]\text{，}\qquad(3.7)$$

輸出參數為一組振幅$\{A_k\}$、頻率$\{w_k\}$、相位$\{\theta_k\}$，但此組參數並不適合做為語音轉換的特徵參數。因此接下來進一步說明擷取語音轉換參數的分析流程。

假設語音信號*s(n)*為完美濁音(perfectly voiced speech)，則各個正弦頻率可視為基頻w_0的諧波，即$w_k=k\times w_0$。依據最小均方誤差(minimum mean-square error, MMSE)原則，來找出最佳合成訊號所需的一組正弦波。所以我們首先預估語音信號的濁音機率P_v及基頻w_0，再將原始信號*s(n)*以漢明視窗作前置處理，接著作短時傅立葉轉換(Short-Term Fourier Transform, SFFT)，得到語音頻譜*A(w)*，再以基頻的整數倍作取樣求得正弦分量的振幅$\{A_k\}$。為配合執行構音轉換所採用的特徵參數以及最小相位模型，故選擇以倒頻譜係數$\{c_l\}$取代$\{A_k\}$作為輸出的特徵參數。如圖 3.1 所示，為音長調整的系統方塊圖，語音經過正弦分析後可得到倒頻譜係數$\{c_l\}$、濁音機率P_v及基期P_0。

3.1.3 諧波正弦合成

在正弦波元的振幅、頻率、相位這三項參數中，人耳聽覺對相位的變化較不靈敏，因此我們可以利用基週、濁音機率以及最小相位的

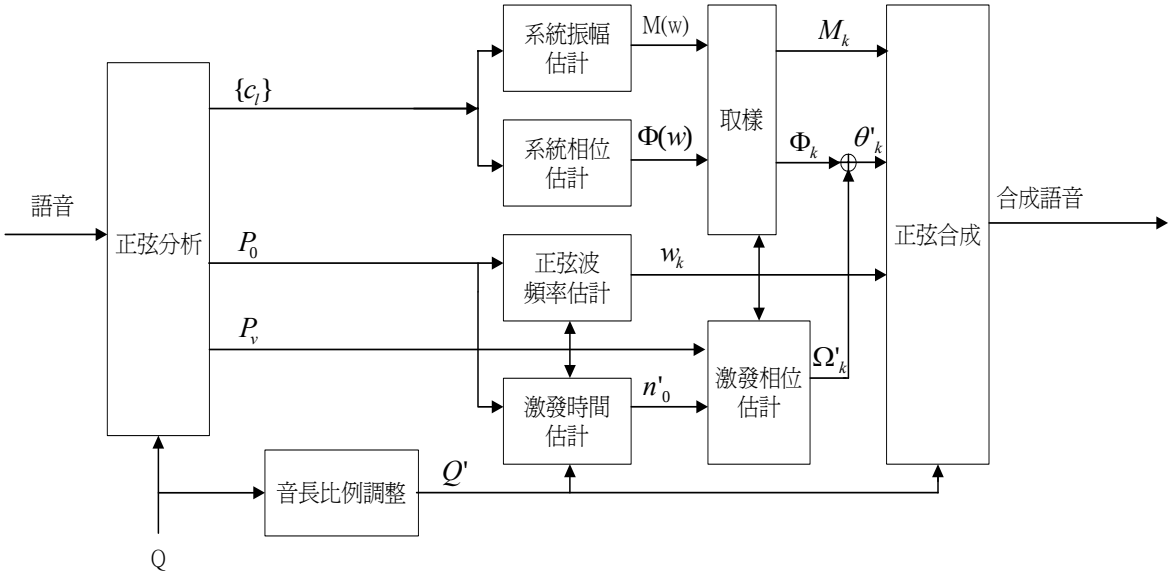


圖 3.1　音長調整系統方塊圖

聲道特性來估測第 k 個諧波所對應的相位 θ_k，由(3.6)式及圖 3.1 可將其分為激發相位 Ω_k 以及聲道濾波器相位 ψ_k 來得到。若假設聲道濾波器之頻譜響應為最小相位系統 $H(w)=M(w)\exp[j\Phi(w)]$，則可由倒頻譜係數推得其振幅及相位，關係如下：

$$\log\left|M(w)\right| = c_0 + 2\sum_{l=1}^{M-1} c_l \cos(lw) \tag{3.8}$$

$$\Phi(w) = -2\sum_{l=1}^{M-1} c_l \sin(lw) \tag{3.9}$$

但由實驗發現，最小相位系統的假設與真實的相位仍有些許差異，欲消去這些差異必須在最小相位系統後串接一全通濾波器加以補償。如圖 3.2 所示。而激發相位 Ω_k 可由基週、濁音機率轉換求得，如圖 3.1 所示，由基週及濁音機率估計出正弦波頻率，而激發時間可由基週及

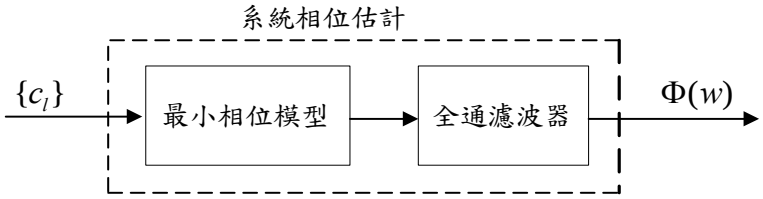


圖 3.2 系統相位估計

濁音機率加上語音長度求得。整體的激發相位如下所示：

$$\Omega_k = -n_0 w_k + \hat{\varepsilon}(w) \tag{3.10}$$

其中 $\hat{\varepsilon}(w)$ 為殘餘相位，和濁音機率相關。所以完整的諧波合成相位可由下式表示：

$$\theta_k = \Phi_k + \Omega_k \tag{3.11}$$

所以諧波正弦合成語音所需的三項要素，振幅、頻率及相位，可分別由倒頻譜係數 c_l、基週 P_0 及濁音機率 P_v 三項特徵參數加以還原。若整個合成訊號是由基頻及其整數倍諧波的正弦波形累加而得，這樣的訊號將會有明顯人工合成之跡象，這是由於諧波合成時忽略了語音訊號中的清音部分，並以不足量的諧波個數來合成語音所導致。要解決前項問題，需找到截止頻率，並劃分出清音部分的區間，在增加其於頻率軸上的取樣，其圖 3.3 所示。加入清音部分之合成訊號會使合成語音之品質提昇且聽起來的效果也較為自然。諧波頻率的選取表示如下：

$$\hat{w}_k = \begin{cases} k\hat{w}_0 & \text{for } k\hat{w}_0 \le w_c(P_v) \\ k^*\hat{w}_0 + (k-k^*)w_u & \text{for } k\hat{w}_0 > w_c(P_v) \end{cases} \tag{3.12}$$

其中 w_u=100Hz，k^* 是在 $k^*\hat{w}_0 \le w_c(P_v)$ 條件下之最大整數值。

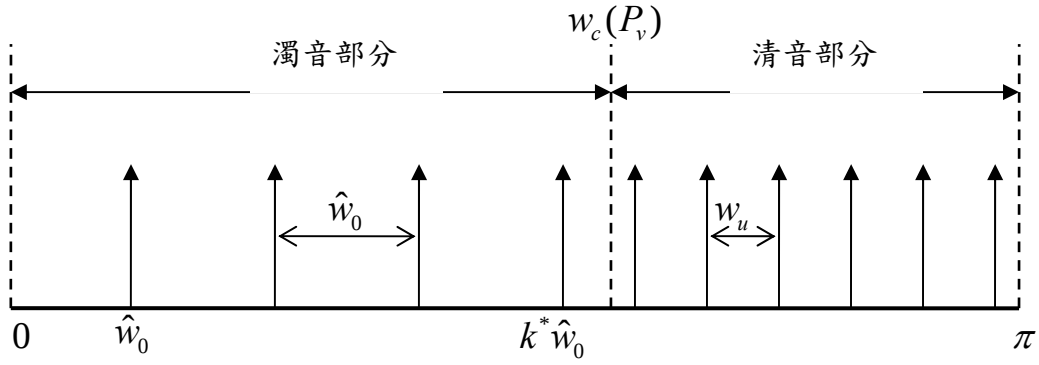


圖 3.3 諧波合成法之示意圖

截止頻率(cutoff frequency)為：

$$w_c(P_v) = \begin{cases} 0.375\pi & \pi P_v \le 0.375\pi \\ \pi \cdot P_v & \text{otherwisw} \end{cases} \qquad (3.13)$$

在取得諧波頻率 $\hat{w}_k$ 後，便可在聲道濾波器的振福及相位作取樣得到諧波分量 $\{\hat{M}_k\}$ 與 $\{\hat{\Phi}_k\}$，在依據(3.11)式取得諧波相位 $\hat{\theta}_k$ 即可合成訊號。

3.2 音長比例調整

語音轉換的核心技術為語音特徵參數轉換機制的設計，其中聲韻轉換與頻譜轉換分別為兩大獨立的語音轉換機制，而聲韻轉換部分包含語音音長與音高兩部分的調整。本論文的重心將放在實現音長比例調整，這是為了使第二章提到的後兩種播放演算法的播放具有連續性，必須作延長或縮短原來的發音內容。

音長比例調整機制的主要目標是改變語者的發聲速率而不影響原語者發聲的特性，在以音框為單元之分析合成系統中，可利用調整合成音框的長度來實現。如圖 3.1 所示，欲將時間長度固定為 Q 的音框依時變的比例 $\rho(i)$ 作調整，調整後的音長為 $Q'(i)=\rho(i)Q$ 。若 $\rho>1$ 表示將時間軸拉長，即減緩發聲速率；若 $\rho<1$ 則表示將時間軸縮短，即加快發聲速率。為維持原語音的發聲特性，聲道濾波器之頻率響應及基頻調整前後均保持不變：

$$\begin{aligned} M'(w) &= M(w) \\ \Phi'(w) &= \Phi(w) \\ w'_0 &= w_0 \end{aligned} \tag{3.14}$$

而合成時仍以相同基頻 w_0 及其倍數作取樣，因此聲道濾波器第 k 個諧波的振福及相位參數 $\hat{M}_k, \hat{\Phi}_k$ 皆保持不變，分別表示如下：

$$\begin{aligned} M'_k &= M_k \\ \Phi'_k &= \Phi_k \end{aligned} \tag{3.15}$$

當激發訊號時間軸拉長或縮短，脈衝發生位置必會隨之改變，而激發起始時間定義為最靠近音框中點的脈衝發生位置，亦同時受到影響。正確的估計與激發相位相關的激發起始時間，即為音長比例調整的關鍵。假設原音框長度 Q，調整後的音框長度為 $Q'(i)=\rho(i)Q$ ，可依新的音框長度求得調整後的激發起始時間 n'_0 ，及其激發相位為：

$$\Omega'_k(i) = -n'_0(i)w_k(i) + \varepsilon_k(i) \tag{3.16}$$

最後相位為：

$$\theta'_k(i) = \Omega'_k(i) + \Phi_k(i) \qquad (3.17)$$

利用調整後參數所合成的訊號表示為：

$$s(n) = \sum_{k=1}^{K(i)} M_k(i)\cos[nw_k(i) + \theta'_k(i)],\ \ t_i \le n \le t_{i+1} - 1 \qquad (3.18)$$

其中 $t_i = \sum_{l=1}^{i-1} Q'(l)$ 為此合成音框的起始時間。

音長調整比例 ρ 可由播放時序演算法中求得，第二章提到的演算法可用來預估下一個封包的播放時間 $\hat{p}_{i+1}$，因此第 i 個封包的音長就需調整為 $\hat{L}_i = \hat{p}_{i+1} - \hat{p}_i$，而原始的音長為 $L_0 = Q$，因此，$\rho(i) = \hat{L}_i / L_0$。為了避免過度的調整音長大小，使合成語音有人工且不自然的現象，造成聲音品質下降。我們定義最大及最小可容忍的封包長度，分別為 $L_{\max}$ 及 $L_{\min}$。在音長調整機制中，限定 $L_{\max} = 2L_0$ 及 $L_{\min} = 0.5L_0$。而這些限制必須加在我們的播放時序演算法當中，適應地調整最後的播放時間 $\hat{p}_{i+1}$。調整的虛擬碼如下所示，其中第二行的部分即為各播放演算法原本估計出的播放時間 $\hat{p}_{i+1}$。

// Algorithm for adaptive playout time adjustment with packet scaling

1 Receive packet i;
2 Estimate and set the playout time for packet i+1, $\hat{p}_{i+1}$
3 Calculate the desired length of packet i, $\hat{L}_i = \hat{p}_{i+1} - \hat{p}_i$
4 **if** $\hat{L}_i > 2L_0$
5 Scale packet i with target length $\min(\hat{L}_i, L_{\max})$;

6 **elseif** $\hat{L}_i < 0.5L_0$

7 Scale packet i with target length $\max(\hat{L}_i, L_{\min})$;

8 **else**

9 Keep packet i without modification

10 **endif**

11 Output packet i with actual length $\hat{L}_i$;

12 Update the playout time of packet i+1, $\hat{p}_{i+1} = \hat{p}_i + \hat{L}_i$;

第四章　實驗結果

在這一章中，模擬三種播放時序演算法，比較不同環境下的客觀效能分析與主觀聽覺測試。網路延遲是由一個網路延遲模型去模擬產生。比較每個演算法的晚到封包造成的漏失率及平均的緩衝延遲來判斷客觀效能的好壞。至於聽覺測試的部分，以封包為單位作調整的演算法，每個單獨封包的伸展或壓縮是透過 STC 的音長調整機制來完成。

4.1 節先介紹一個適用於網際網路的網路延遲模型，在 4.2 節中比較不同 spike 偵測機制的差異，最後在 4.3 節中用短詞和完整長句搭配不同的網路延遲來分析三種演算法的效能，最後並比較利用 STC 作音長調整後的音質效果。

4.1　網路單向延遲模型

在這一節中，我們介紹一種數學模型，用來模擬不同環境網路單向延遲，並據以比較各演算法的效能。

4.1.1 模型簡介

第 n 個語音封包的整體延遲以 d_n 表示，圖 4.1(a)表示其量測值的時間連續圖，即 d_n 為 n 的函數。但在這裡，我們發現用相位圖(phase

plot)來觀察整體延遲會更適當，如圖 4.1(b)。在相位圖中，若 $x = d_n$ 且 $y = d_{n+1}$，則在座標點(x, y)作標記，這樣對每個封包的整體延遲皆作標記的動作，即可得到相位圖。圖 4.1 中，n 的範圍為 0 至 800，且兩個封包傳送間的時間差 δ ＝50ms。

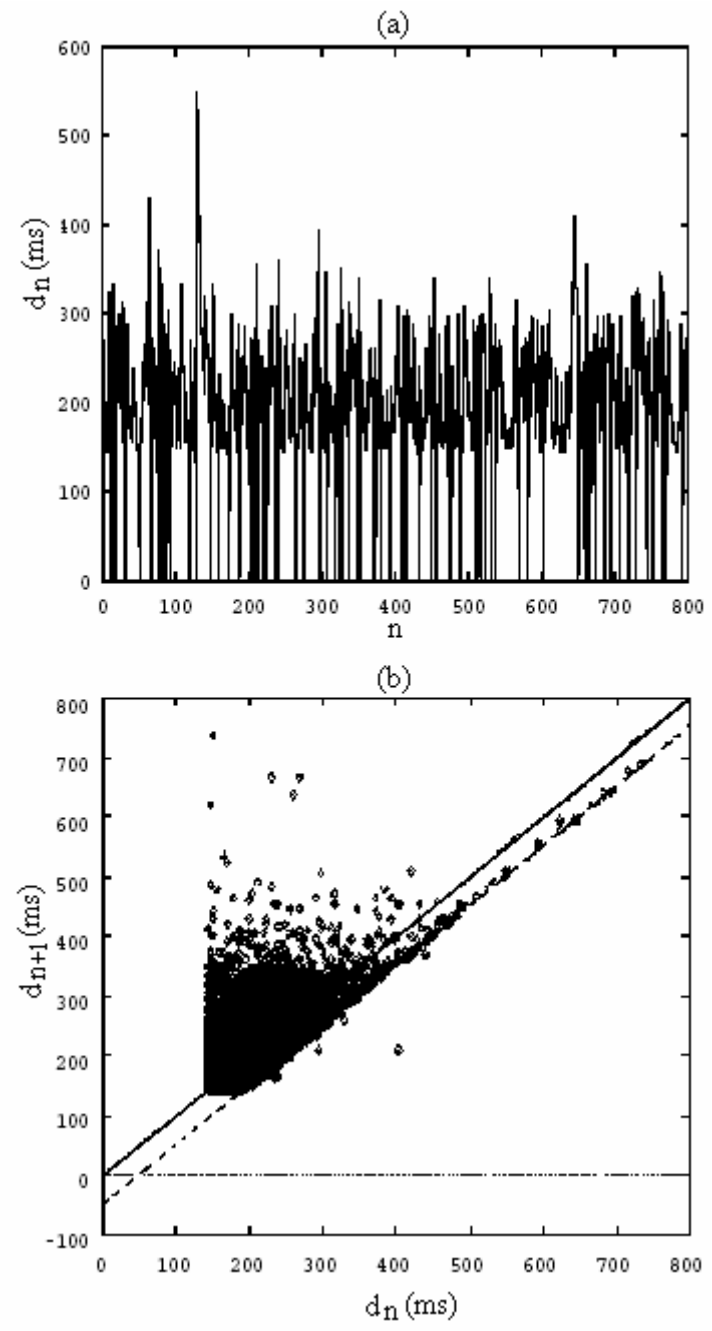


圖 4.1 整體延遲的時間連續圖及相位圖

基於前人研究[9][10]的結果，一連串定期傳送的語音封包受網際網路串流(internet traffic)的影響，可被一批伯努利隨機程序(batched Bernoulli process)近似。因此，在我們的實驗中，用單一伺服器排隊模型來模擬，此伺服器具一 FIFO 緩衝器，有兩個輸入端，一個代表我們的語音串流(audio traffic)而另一個則為網際網路串流。模型如圖 4.2 示，其中 D 表示語音封包整體延遲中的固定部分，例如傳輸延遲。而伺服器的服務速度(service rate)以 μ 表示，單位為 bits/s。聲音訊號為依固定大小的音框(frame)且週期性地傳送，以 P 表示其長度，單位為 bits；δ 表示兩個封包傳送間的時間差。而網路串流會以許多串流依序疊加而成，而和聲音訊號競爭分享一般的網路資源。

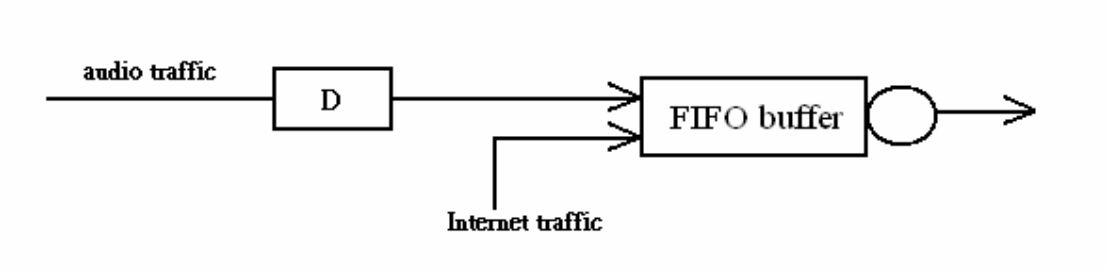


圖 4.2 網路延遲的模型

4.1.2 網路延遲分析

首先考慮第一種情形，網路串流很小的情形，例如緩衝器內的網路封包很少且這些封包都很小，如 Telnet 封包。在這個情況下，連續

語音封包的等待時間(waiting time)都接近為一常數。第 n 個語音封包的等待時間(不包括服務時間)以 w_n 表示，可得：

$$w_{n+1} = w_n + \varepsilon_n \text{ ，} \qquad (4.1)$$

其中 ε_n 為一平均值為 0 且變異數很小的隨機程序。所以可得到整體延遲 $d_{n+1} = D + w_{n+1} + P/\mu$ 及 $d_n = D + w_n + P/\mu$ ，因此：

$$d_{n+1} - d_n = w_{n+1} - w_n = \varepsilon_n \text{ 。} \qquad (4.2)$$

在這種情形下，相位圖上的點幾乎以對角線 $d_{n+1} = d_n$ 為中心(如圖 4.1(b)所示)，且離最小延遲點(*D,D*)很近，在圖 4.1(b)中，$D \approx 140$ ms。

接下來討論另一種在封包時間差 δ 很小的情形下，且兩個連續語音封包中間收到一個很大的網路封包時(一個或多個 FTP 封包)。假設 B 為此網路封包的大小，單位為 bits，在第 *n+1* 個語音封包前收到。因此第 *n+1* 個語音封包的排隊延遲為：

$$w_{n+1} = w_n + B/\mu \text{ ，} \qquad (4.3)$$

因此，

$$d_{n+1} - d_n = B/\mu \text{ 。} \qquad (4.4)$$

如果 $w_{n+1} > \delta$ ，就會有一個以上的語音封包累積在第 *n+1* 個語音封包後，因為要等待伺服器處理這個較大的網路封包。假設有 k 個語音封包累積，且在第 *n+1* 個語音封包及第 *n+k* 個語音封包間沒有其他網

路封包到達此伺服器。則第 $n+1$ 個語音封包至第 $n+k$ 個語音封包會以固定間隔離開此排隊伺服器，此間隔為 P/μ。因此，可得到：

$$\begin{aligned} d_{n+2} - d_{n+1} &= (a_{n+2} - t_{n+2}) - (a_{n+1} - t_{n+1}) \\ &= (a_{n+2} - a_{n+1}) - (t_{n+2} - t_{n+1}) \text{，} \\ &= P/\mu - \delta \end{aligned} \tag{4.5}$$

其中 a_n 及 t_n 分別為第 n 個封包的接收及傳送時間。同樣地：

$$d_{n+3} - d_{n+2} = \ldots = d_{n+k} - d_{n+k-1} = P/\mu - \delta \text{，} \tag{4.6}$$

因此，滿足上式的相位圖上的點會位於 $d_{n+1} = d_n + P/\mu - \delta$ 線上，這直線在圖 4.1(b)以虛線表示。若 $\delta < P/\mu$，語音封包會導致排隊伺服器完全飽和，因此，必須保持 $\delta > P/\mu$。

接下來根據圖 4.2 的排隊模型來分析網路延遲，這分析是根據 Lindley 遞迴方程式的兩個連續應用，而 Lindley 遞迴方程式說明了單一通道排隊中的兩個連續封包的等待時間關係。令第 n 個封包的等待時間為 w_n，第 n 個封包的服務時間為 y_n，而第 n 個封包與第 $n+1$ 個封包的抵達時間差為 x_n，因此：

$$w_{n+1} = \begin{cases} w_n + y_n - x_n & \text{if } w_n + y_n - x_n > 0 \\ 0 & \text{otherwise} \end{cases} \tag{4.7}$$

上式可以圖形證明之，如圖 4.3 所示，本論文接下來部分皆用 x^+ 表示 $\max(x,0)$，利用這個標記法，我們可將式子(4.7)改寫為：

$$w_{n+1} = (w_n + y_n - x_n)^+ \text{。} \tag{4.8}$$

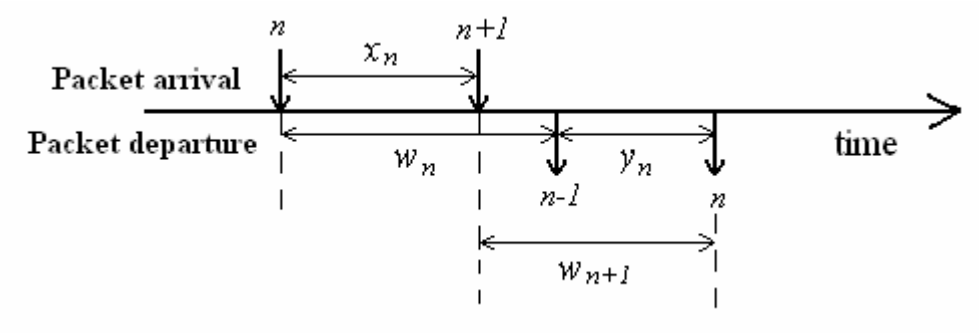


圖 4.3 Lindley 遞迴方程式證明

我們假設第一個語音封包到達此排隊緩衝器的時間為 δ，因此第 n 個語音封包到達的時間為 $n\delta$。我們假設網路串流 b_n，單位為 bits，在時間 $n\delta$ 及 $(n+1)\delta$ 中間到達，b_n 為一個表示網路串流的隨機變數，再假設所有的 b_n 皆在相同的時間 t_n 到達排隊緩衝器。讓 wb_n 表示此網路封包的等待時間，將 wb_n 及 w_n 套用在 Lindley 遞迴方程式上，可得：

$$wb_n = (w_n + P/\mu - t_n)^+ \text{，} \qquad (4.9)$$

將 w_{n+1} 及 wb_n 再套用在 Lindley 遞迴方程式上，可得：

$$w_{n+1} = (wb_n + b_n/\mu - (\delta - t_n))^+ \text{，} \qquad (4.10)$$

將(4.9)式帶入(4.10)，可得：

$$w_{n+1} = ((w_n + P/\mu - t_n)^+ + b_n/\mu - (\delta - t_n))^+ \text{，} \qquad (4.11)$$

只要在區間 $[n\delta, t_n + n\delta]$ 內緩衝器不是空的，$w_n + P/\mu - t_n$ 項就會是正值。因此，上式就可簡化為：

$$\begin{aligned} w_{n+1} &= (w_n + P/\mu - t_n + b_n/\mu - (\delta - t_n))^+ \\ &= (w_n + (P + b_n)/\mu - \delta)^+ \end{aligned} \text{，} \qquad (4.12)$$

只要在區間$[n\delta, t_n + n\delta]$內緩衝器不是空的，$w_n + (P + b_n)/\mu - \delta$項就會是正值。因此，上式就可簡化為：

$$w_{n+1} = w_n + (P + b_n)/\mu - \delta \text{ ，} \qquad (4.13)$$

因此，

$$b_n = \mu(w_{n+1} - w_n + \delta) - P \text{ ，} \qquad (4.14)$$

因此，b_n的機率分佈(在區間δ網路串流量的機率分佈)，可依$w_{n+1} - w_n$的分佈來估計。然而注意上式成立的條件，必須在區間$[n\delta, t_n + n\delta]$內緩衝器不是空的，若在區間內緩衝器是空的，則(4.14)式並不成立。因此在δ足夠小的情況下，如$\delta\mu$的乘積小於b_n的平均值時，以(4.14)式來估計b_n是合理的。

基於上述的分析，可隨機產生M個網路封包b_n，依使用者設定的網路狀態來分配 Telnet(light)及 FTP(heavy)封包個數的比率。表示第n個語音封包進入緩衝器後與下一個語音封包進入緩衝器前有一個網路封包b_n進入緩衝器，影響每個語音封包的等待時間。所以利用式子(4.14)，可得：

$$w_{n+1} - w_n = (P + b_n)/\mu - \delta \text{ ，} \qquad (4.15)$$

若在兩語音封包間沒有網路封包進入，即$b_n = 0$，因此：

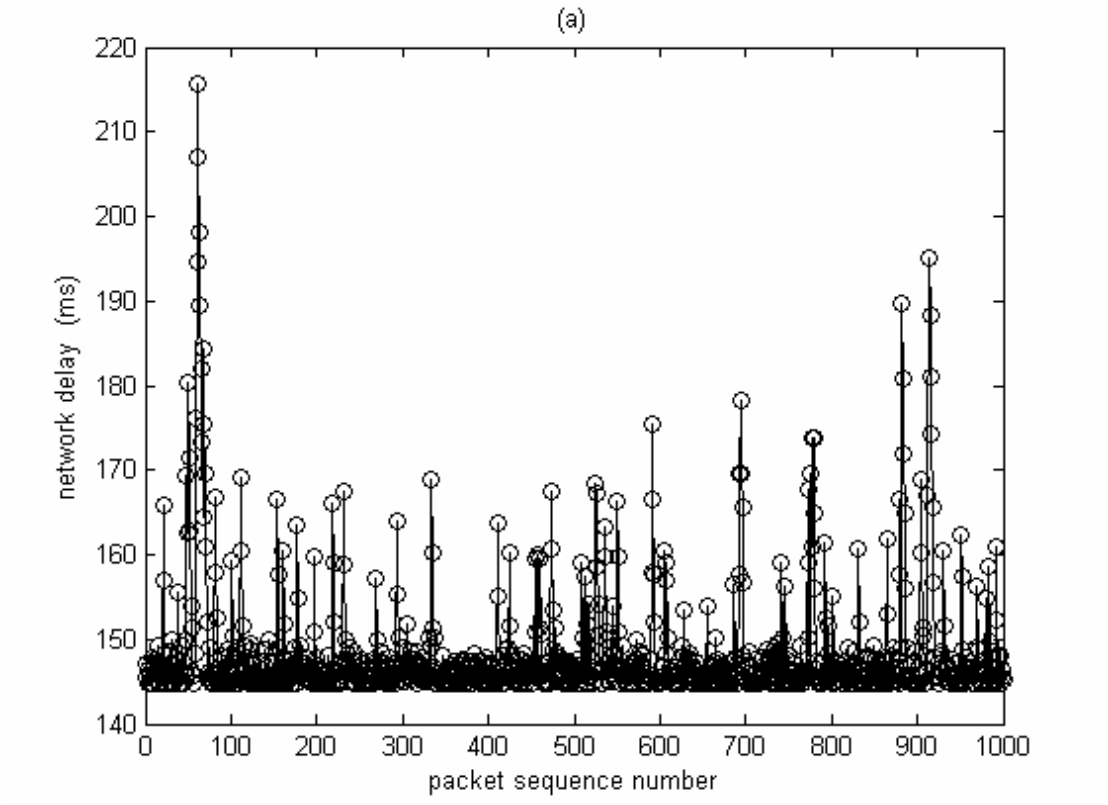
$$w_{n+1} - w_n = P/\mu - \delta \text{ 。} \qquad (4.16)$$

所以由隨機產生的b_n值可反推得到$w_{n+1} - w_n$，也就可以得到

$d_{n+1}-d_n$，利用疊代的方式，就可以將每個語音封包的網路延遲估計出來。

4.1.3 模擬結果

這裡我們呈現兩種不同的網路狀態的模擬結果，一種屬於 light 的情形，其中 telnet 封包占 75%而 FTP 封包占 25%；而第二種屬於 heavy 的情形，其中 telnet 封包占 25%而 FTP 封包占 75%。基本參數設定為 D＝140ms，μ＝180 kbit/s，P＝108 byte，δ＝13.6ms。圖 4.4(a)為 light 的網路延遲結果，圖 4.4(b)為 heavy 的結果。



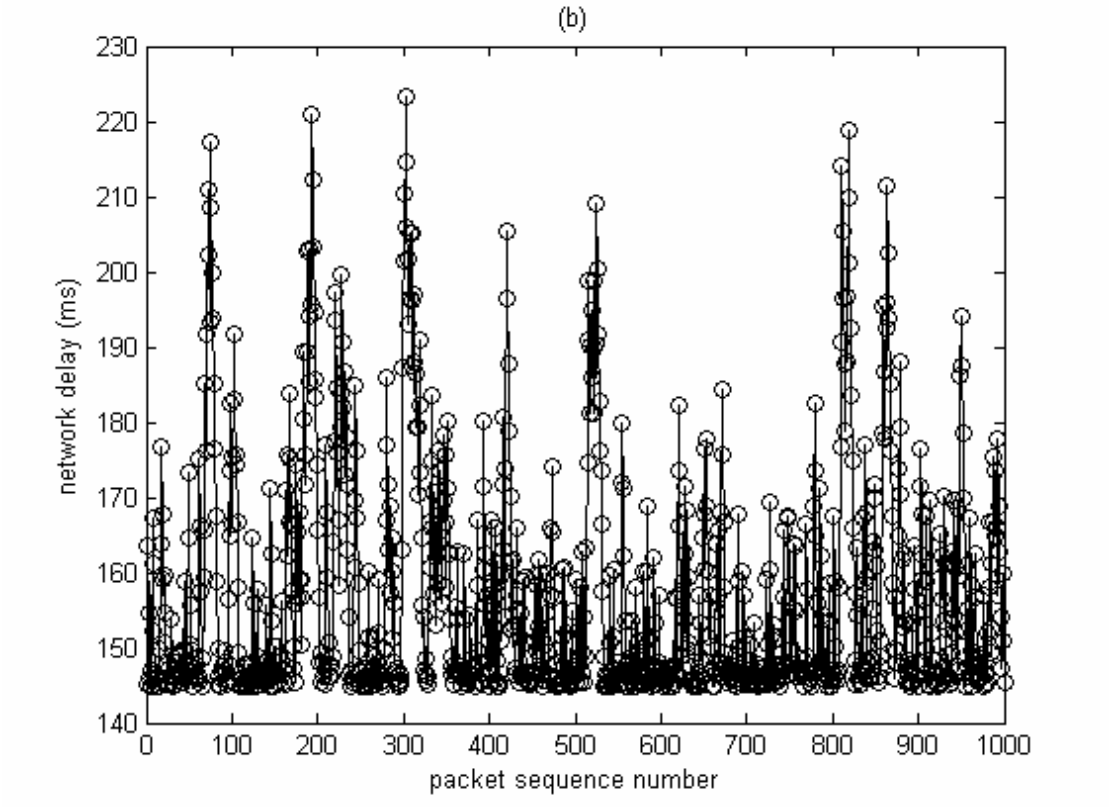


圖 4.4 網路延遲模擬結果

4.2 Spike 偵測機制的比較

在第二章中所提到的三種不同演算法，AR(autoregressive)表示第一種演算法，OS(order statistics)表示為第二種演算法，NLMS表示為第三種演算法。其中第二種演算法(OS)的 spike 偵測方式和第一種演算法(AR)是相同的，所以我們就比較第一種演算法的 spike 偵測機制及第三種演算法(NLMS)的 spike 偵測機制。

我們用上一節中的模型，產生一筆網路延遲，再分別用這兩種 spike 偵測機制來偵測所有 spike 的開始與結束點。第一種 spike 偵

測結果如圖 4.5 所示，第二種 spike 偵測結果如圖 4.6 所示。

每一個方塊為一個 spike 區間，很明顯的兩種偵測機制皆能有效的偵測出 spike 的起始點，而 NLMS 演算法的偵測機制又可偵測出較長的 spike 區間，所以 NLMS 演算法可隨 spike 區間調整緩衝延遲的大小效果就較為顯著。但值得注意的是，在每個 spike 偵測機制中，受環境不同的影響，其參數大小就必須隨之改變，所以並無法完全客觀的單獨比較每個 spike 偵測的好壞。

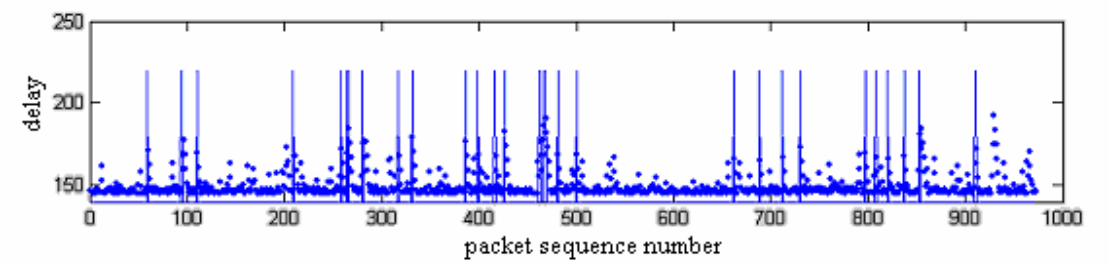


圖 4.5 AR 演算法的 spike 偵測

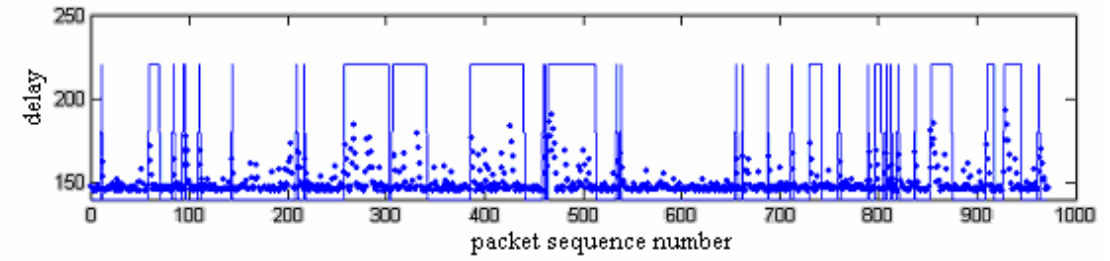


圖 4.6 NLMS 演算法的 spike 偵測

4.3 演算法效能比較

在這一節中，主要針對第二章的三種播放時序演算法進行客觀的效能分析，並進行主觀的聽覺測試以驗證 STC 調整音長大小後

的聲音品質好壞。

4.3.1 短詞比較

首先，利用前人研究[3]中的網路延遲數據，當中共具有三個 talkspurt，整體長度包含兩個靜音與三個話務區間，共有 270 個封包，每個封包的長度為 13.6ms。配合整個封包個數，錄製適當長度的句子：“你_我_他”，來比較各演算法的效能。圖 4.7 為客觀的平均緩衝延遲及漏失率的關係圖，橫軸為平均緩衝延遲，而縱軸為相對的漏失率。所以兩個參數皆越小越好，即曲線越靠近原點表示整體效能越好。由圖 4.7 可得第三種播放時序演算法(NLMS)效能明顯優於其他兩者。

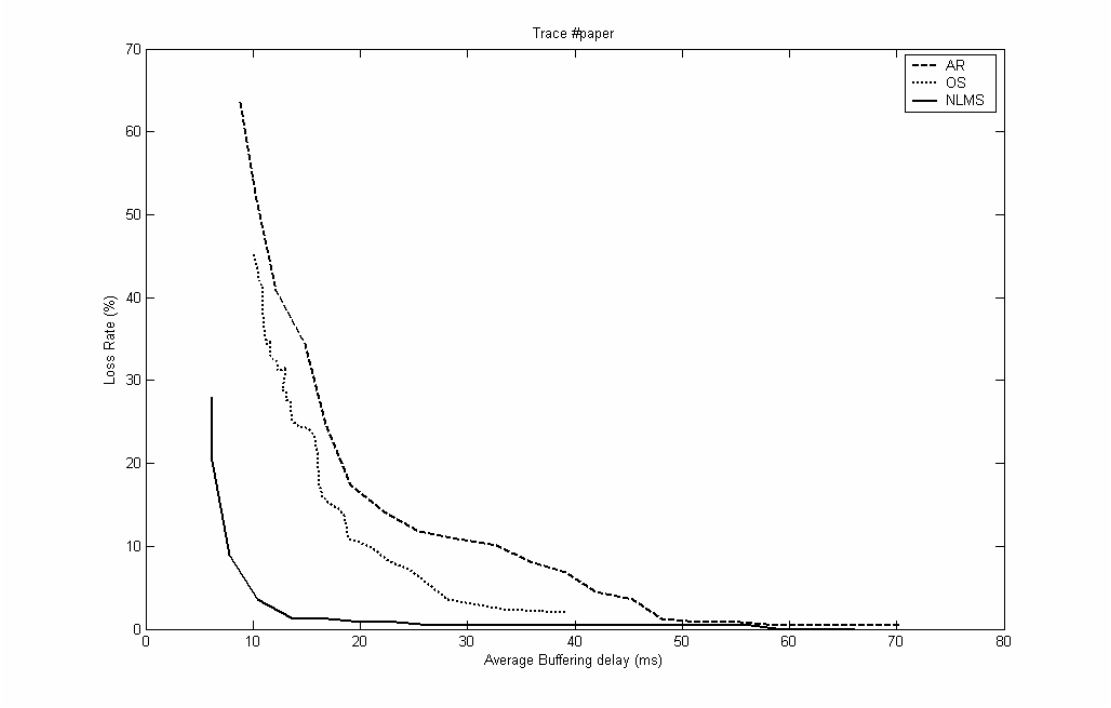


圖 4.7 短詞的各演算法效能分析

接下來，我們選擇在一個固定的緩衝延遲下，來比較三種演算法的漏失率高低程度。首先選擇緩衝延遲約為 18ms 下，我們控制各演算法的參數來達到緩衝延遲約為 18ms 的效果：

	調整參數	平均緩衝延遲	漏失率
AR 演算法	$\beta=5$	19.044025ms	17.408907%
OS 演算法	$\hat{\varepsilon}=9\%$	17.947872ms	14.574899%
AF 演算法	$\beta=5$	17.039563ms	1.214575%

上面各演算法的結果，可用圖 4.8 來表示，每張圖中的橫軸表示封包的號碼，縱軸為相對應的延遲，而黑點表示實際的網路延遲，實線為各演算法所計算出的整體延遲。由圖 4.8 可看出 NLMS 演算法在網路延遲的預估方面有很好的效果，且在相同的平均延遲下，漏失率也可大幅降低。接下來再觀察短詞播放波形，如圖 4.9 所示，由上至下依序為第一至第三種演算法的結果，其中 OS 及 NLMS 兩種演算法必須配合 STC 音長調整。結果顯示，雖然 NLMS 演算法每個封包幾乎都作了音長的調整，但不失其連續性，更重要的是在聽覺測試上也不會因音長調整而有所不同。

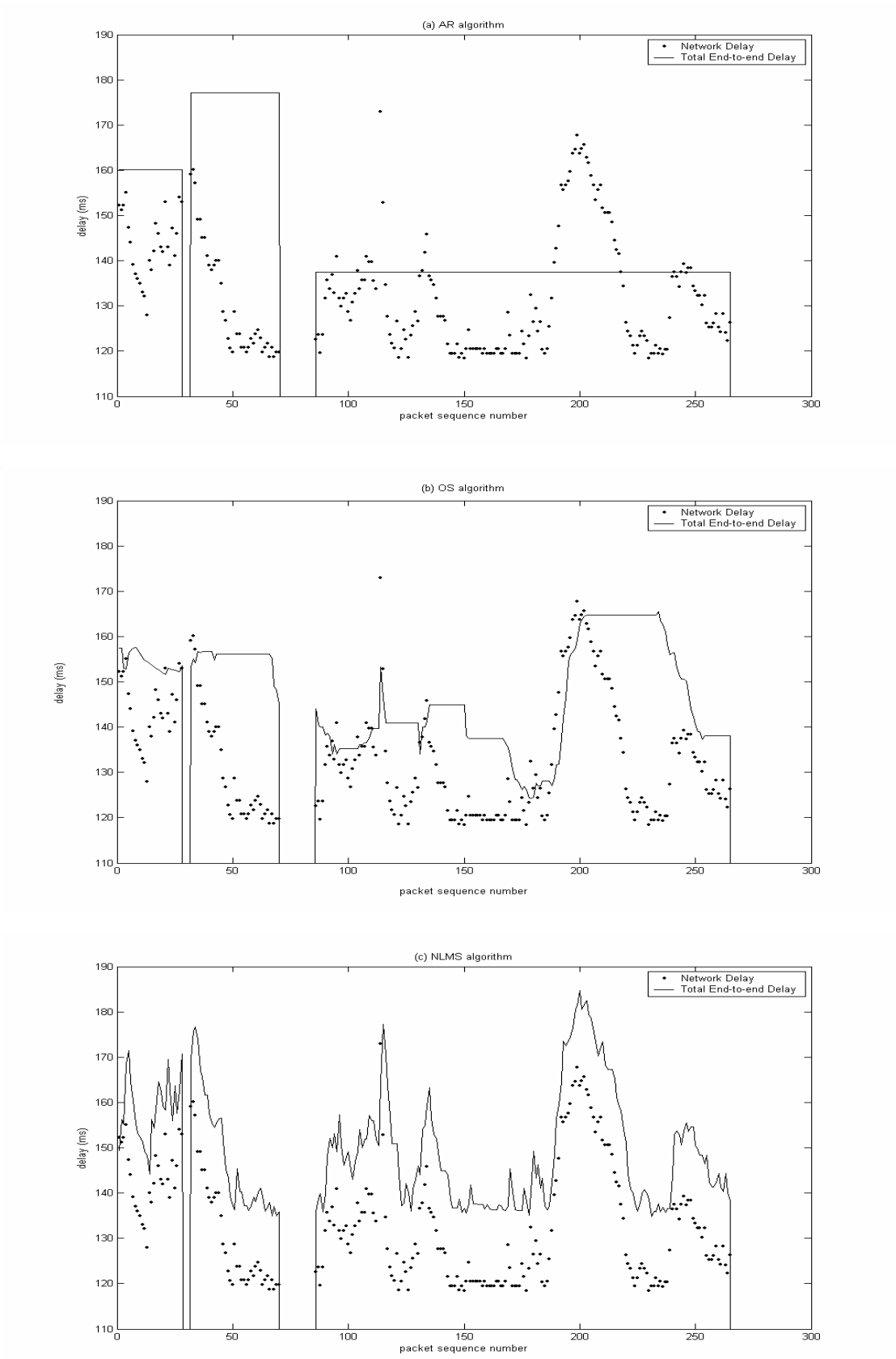
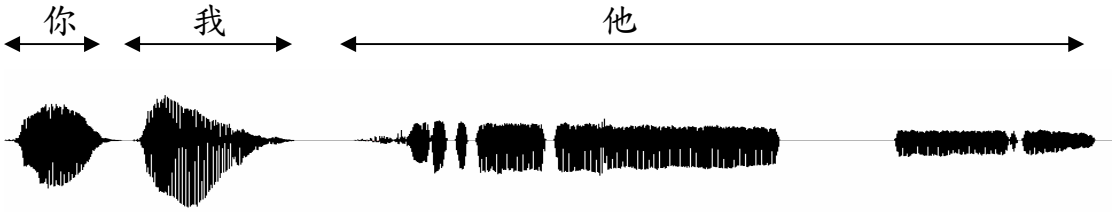


圖 4.8 三種演算法短詞的延遲比較(平均緩衝延遲為 18ms)



(a) AR algorithm

(b) OS algorithm

(c) NLMS algorithm

圖 4.9 短詞播放波形(平均緩衝延遲為 18ms)

接下來我們改變期望的平均緩衝延遲為 12ms,此時平均緩衝延遲降低,表示漏失率也會隨之增高。我們希望觀察在較高的漏失率情況下,各演算法的效能分析。藉由控制各演算法的參數來達到緩衝延遲約為 12ms 的效果:

	調整參數	平均緩衝延遲	漏失率
AR 演算法	$\beta=2$	12.128777ms	40.890188%
OS 演算法	$\hat{\varepsilon}=36\%$	11.914852ms	32.388664%
AF 演算法	$\beta=3.5$	12.098009ms	1.619433%

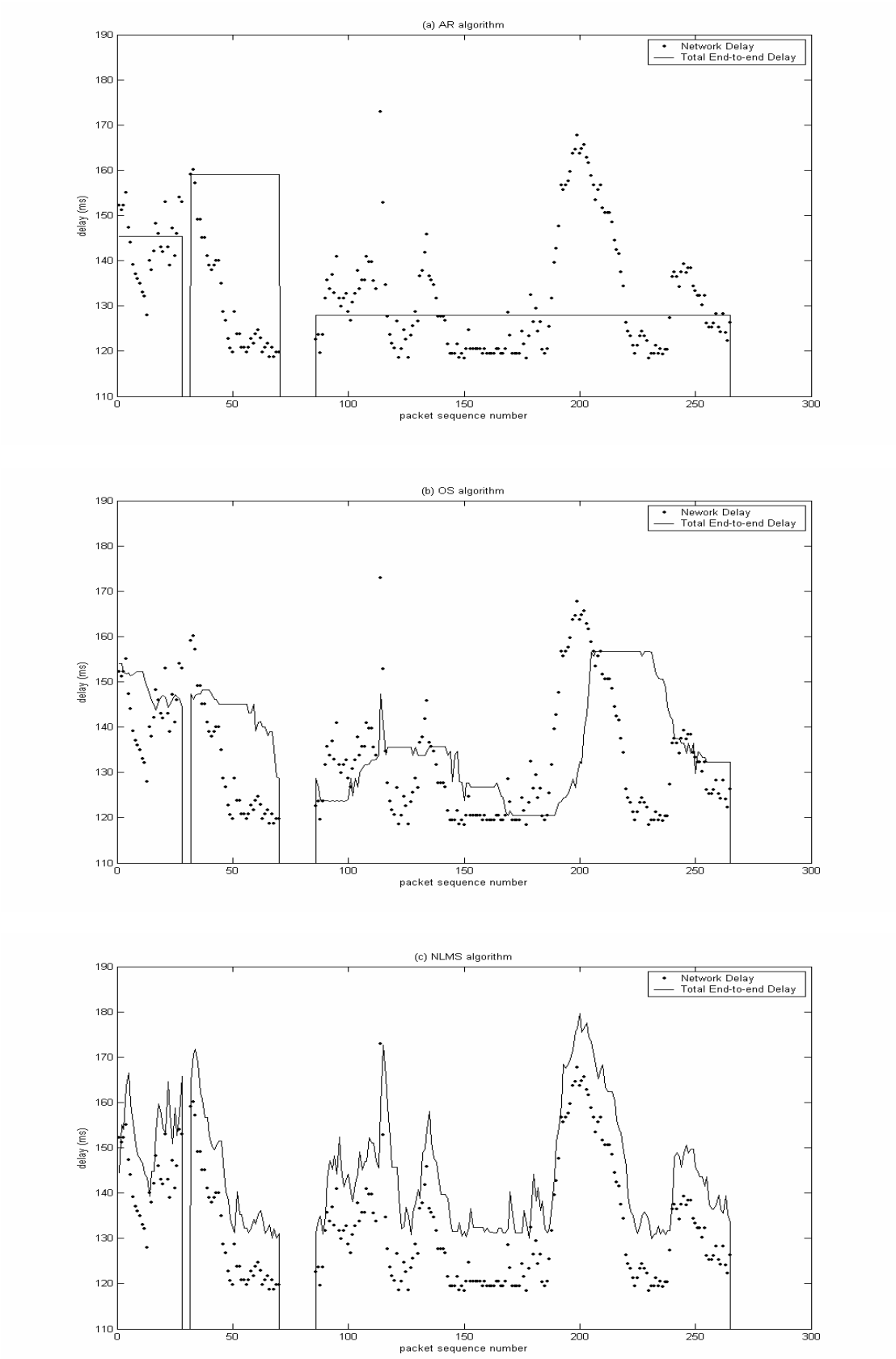
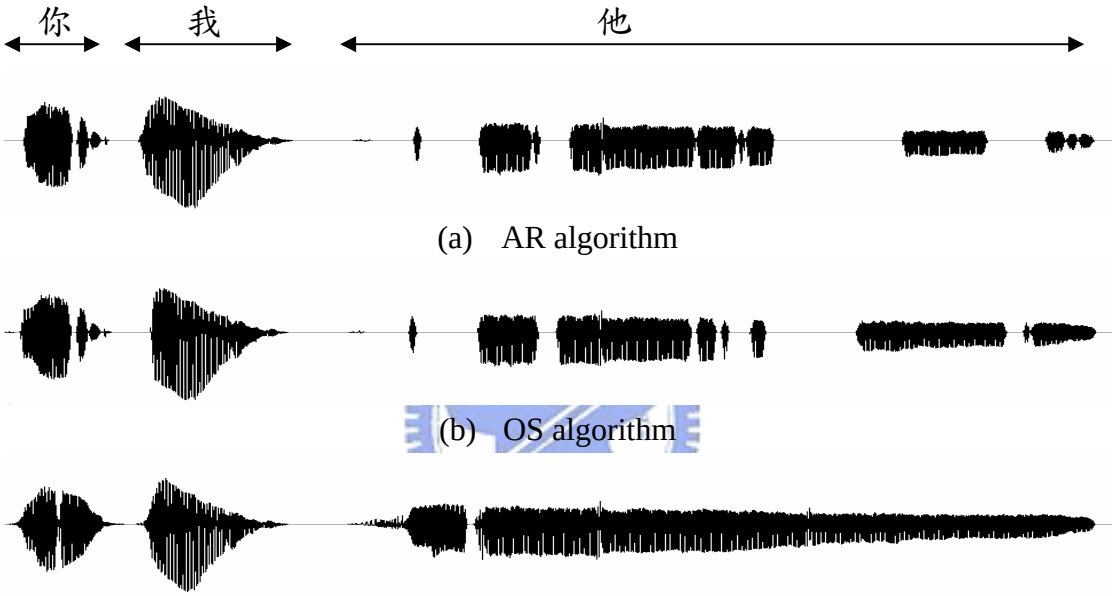


圖 4.10　三種演算法短詞的延遲比較(平均緩衝延遲為 12ms)

同樣地，由圖 4.10 可看出 NLMS 演算法在網路延遲的預估方面有很好的效果，且在相同的平均延遲下，漏失率也大幅降低。而觀察其調整音長後與其他演算法的比較，如圖 4.11 所示，NLMS 演算法在低平均延遲下仍能保持良好的播放波形與聲音品質。



(a) AR algorithm

(b) OS algorithm

(c) NLMS algorithm

圖 4.11 短詞播放波形(平均緩衝延遲為 12ms)

4.3.2 長句比較

接下來我們找一個較長的句子，共有 678 個封包，利用 4.1 節的模型來產生相同個數的封包網路延遲，每一封包長度同樣為 13.6ms。令 μ ＝128 kbit/s，競爭的網路封包有 200 個，並將靜音封包的網路延遲設為 0，可得到各個封包相對應的網路延遲。同樣分

別套入三種演算法中，比較效能差異。如圖 4.12 所示，客觀的比較三種演算法的平均緩衝延遲及漏失率的差異。第三種播放時序演算法(NLMS)的曲線較靠近原點，也就是效能最好。

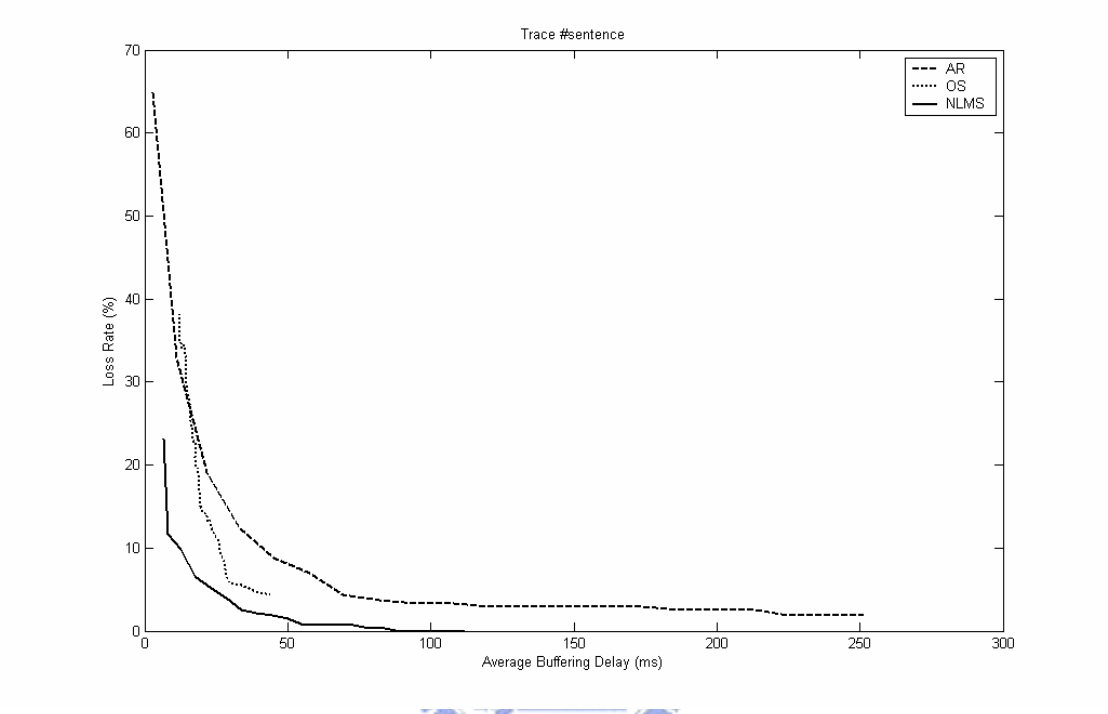


圖 4.12　長句的各演算法效能分析

同樣選擇在一個固定的緩衝延遲下，比較三種演算法漏失率及播放波形的不同。首先選擇緩衝延遲約為 20ms 下，我們控制各演算法的參數來達到緩衝延遲約為 20ms 的效果：

	調整參數	平均緩衝延遲	漏失率
AR 演算法	$\beta=2$	21.587351ms	19.047619%
OS 演算法	$\hat{\varepsilon}=16\%$	20.184547ms	14.285714%
AF 演算法	$\beta=3.5$	20.261569ms	5.494505%

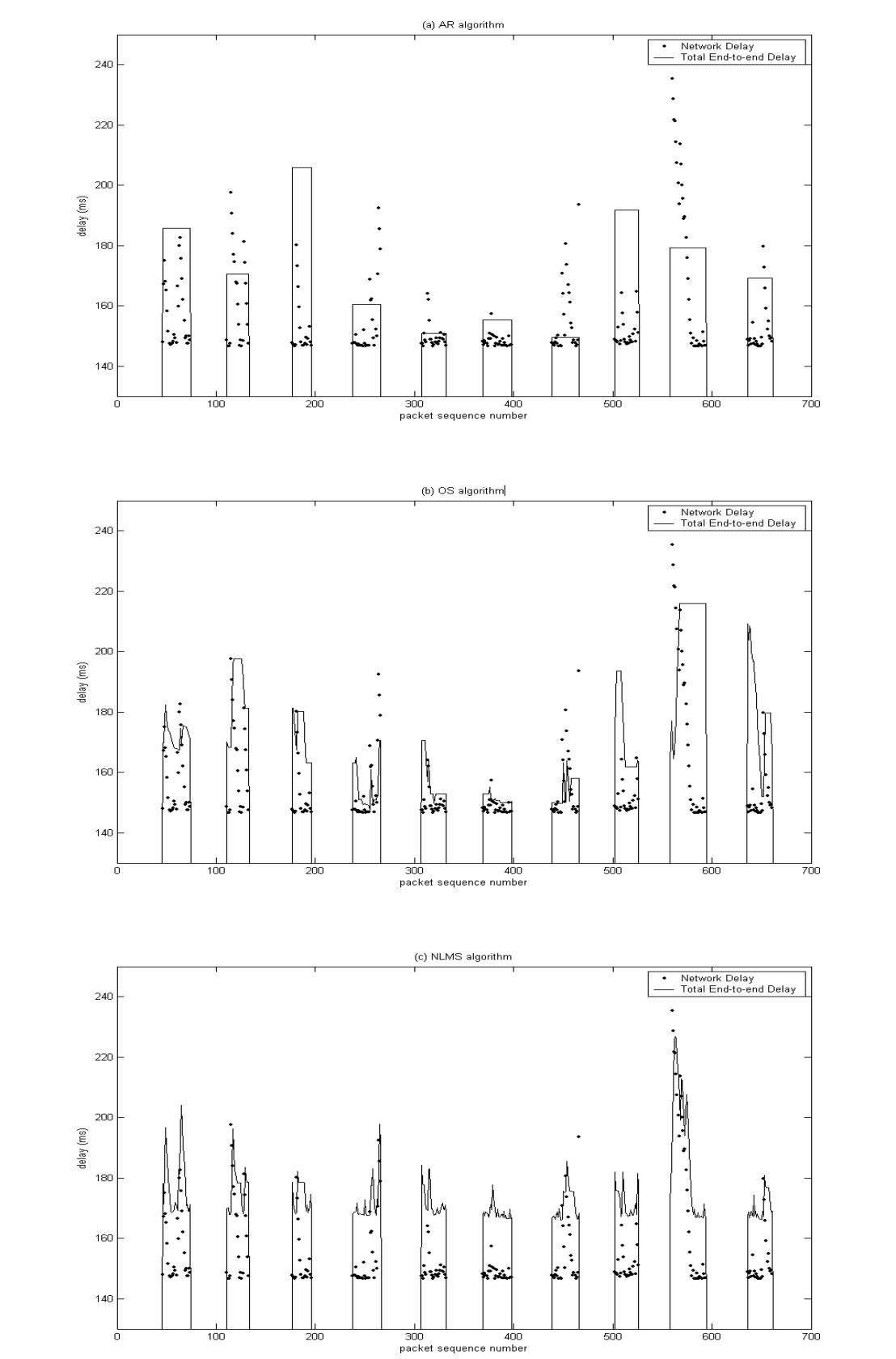
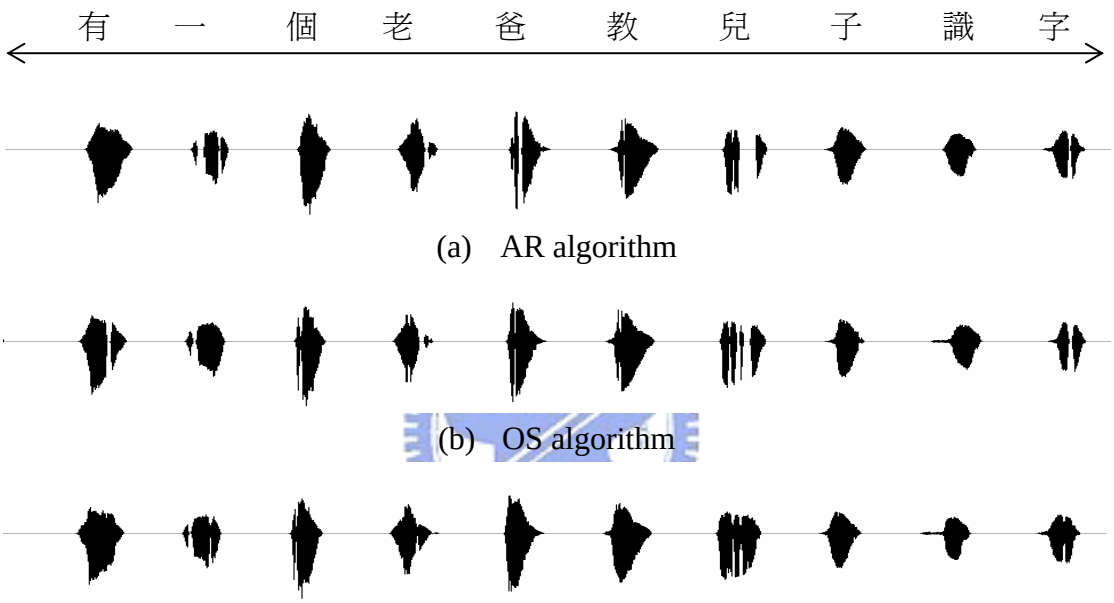


圖 4.13　三種演算法長句的延遲比較(平均緩衝延遲為 20ms)

上面各演算法的結果， 可用圖 4.13 來表示。亦看出 NLMS 演算法在網路延遲的預估方面有很好的效果，且在相同的平均延遲下，漏失率也大幅降低。如圖 4.14 所示，NLMS 演算法所產生的播放波形及其聲音品質也最好。



(a) AR algorithm

(b) OS algorithm

(c) NLMS algorithm

圖 4.14 長句播放波形(平均緩衝延遲為 20ms)

接下來改變平均緩衝延遲為較低的 12ms，觀察各演算法的效能分析。藉由控制各演算法的參數來達到緩衝延遲約為 12ms 的效果：

	調整參數	平均緩衝延遲	漏失率
AR 演算法	$\beta=1$	11.197133ms	32.967033%
OS 演算法	$\hat{\varepsilon}=48\%$	12.299993ms	34.798535%
AF 演算法	$\beta=2$	12.829377ms	9.890110%

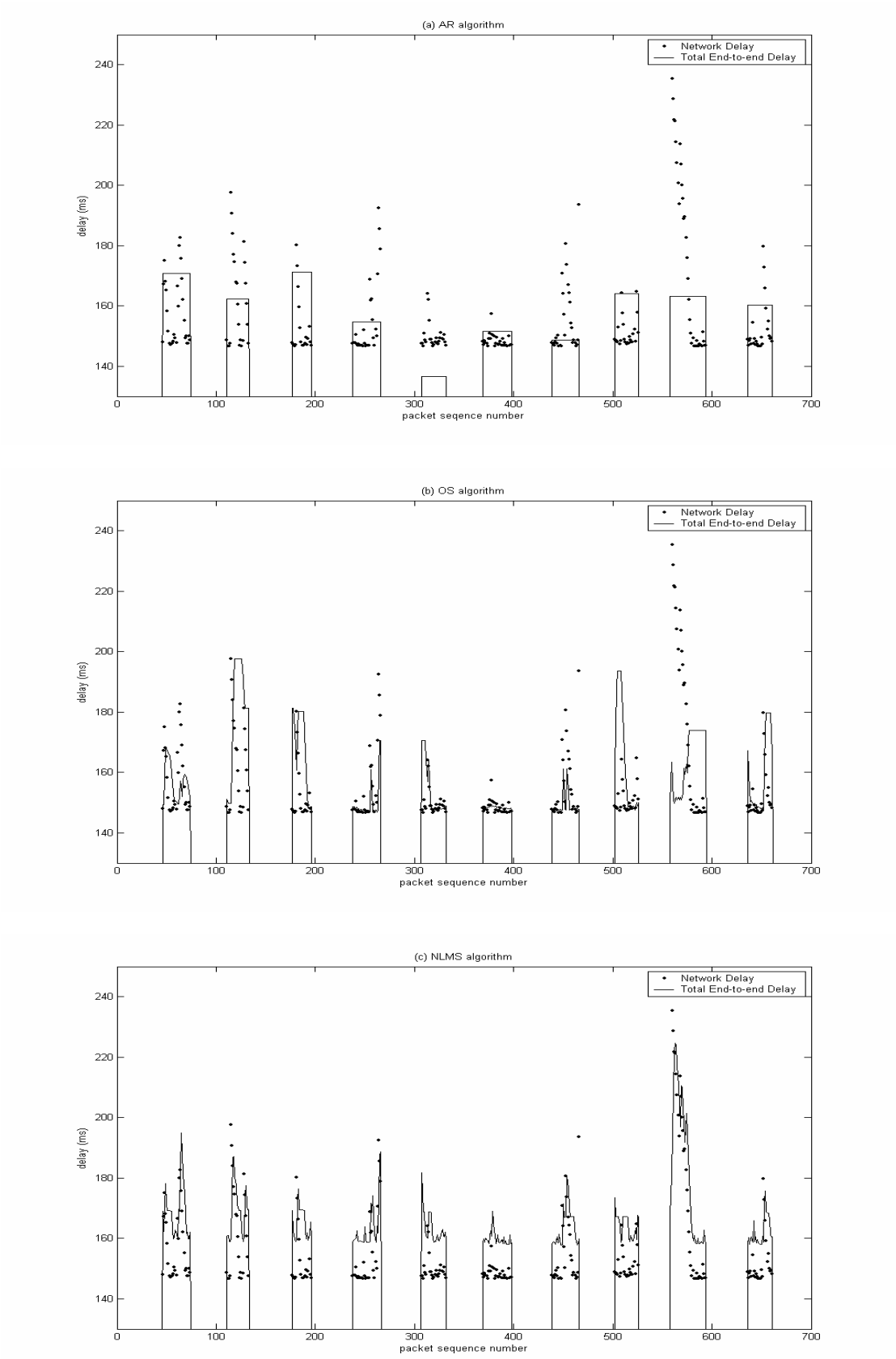
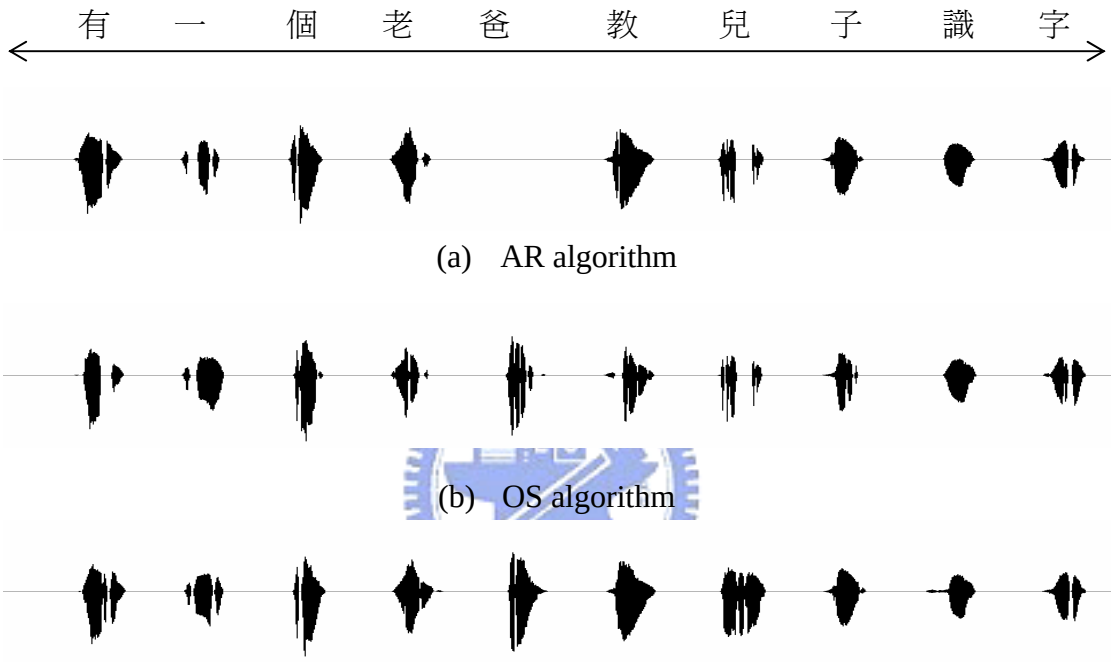


圖 4.15　三種演算法長句的延遲比較(平均緩衝延遲為 12ms)

同樣地，由圖 4.15 可看出 NLMS 演算法在網路延遲的預估方面有很好的效果。如圖 4.16 所示，前兩種演算法所產生的播放波形已嚴重失真，但 NLMS 演算法在低平均延遲下仍能保持很好的聽覺效果。



(a) AR algorithm

(b) OS algorithm

(c) NLMS algorithm

圖 4.16 長句播放波形(平均緩衝延遲為 12ms)

第五章 結論與未來展望

5.1 結論

本篇論文對現今 VoIP 環境中所面臨到的問題：網路延遲、延遲顫動及封包漏失率，做了一些初步的介紹，並針對網路延遲特性–Spike 造成的原因做了說明，進而針對這些現象來改進語音通訊的品質。當中提到的三種播放演算法皆為了改進網路顫動所造成通話品質的影響，由實驗結果中，可得到其中 NLMS 演算法最能適應網路的變動情形，並能最有效地使緩衝延遲及漏失率降低。

我們使用的語音封包音長比例調整方法，是採用正弦轉換編碼的方式來得到語音特徵參數，針對我們要的音長調整參數 ρ 來做適當的調整，使播放的語音能保持其連續性。也由於正弦轉換編碼的特性，每個封包間的獨立關係，即使前後封包漏失，也不影響其播放的品質。由實驗結果可觀察到以諧波正弦來做音長比例調整下，並不會因為調整個別封包的長度而改變原有語音的特性。讓使用者在不察覺語音品質有所變化的情形下，利用音長調整降低其平均緩衝延遲及封包漏失率，使整體效能提高。

5.2 未來展望

在本論文提到的演算法中，是假設在每個封包都能順利被接收到的情形下，沒有考慮到因網路傳輸而漏失的情形。也就是說若之前的封包沒有被接收到，也就無法得知其網路延遲的情形，因此如何適度的改善演算法以面對這種情形，是一項值得未來發展的課題。

現今演算法在偵測 spike 時，都是設定一個臨界值來判斷起始點，而此臨界值的設定又受網路環境的影響很大，因此在臨界值的設定上也是另一門學問。而在第四章中也又簡單比較 spike 偵測機制的不同，但由於網路環境的影響太大，要如何客觀的比較每個 spike 偵測機制的好壞也是值得探討的地方。

在前人相關的演算法研究中，我們發現在網路延遲上，有的研究是以實際傳送封包而得到延遲數據，有的則是如本論文中以網路模型來得到延遲數據。前者的優點是採用實際的延遲數據，後者則是模擬的結果；但相對來講，前者的缺點則是每得到一組數據就需要花費大量的時間，而後者則可無限制的得到想要的數據。但無論是哪一種方式，都是屬於 off-line 的實驗結果，因此如何使整個演算法系統，包含音長比例調整機制，能實現在整個即時語音通訊上，是最主要的方向。

以上我們做的研究皆是針對 VoIP 來討論，而現在無線網路的普

及，而在無線網路上面臨的延遲及封包漏失的情形也有所不同，如何面對不同的網路情形而適當調整類似的演算法，也是另一個有趣且值得研究的發展方向。

參考文獻

[1] R. Ramjee, J. Kurose, D. Towsley, and H. Schulzrinne, “Adaptive playout mechanisms for packetized audio applications in wide area networks,” in *Proc. IEEE Infocom Conf. Comp. Commun*., vol. 2, (Toronto, Canada), pp. 680-688, June 1994.

[2] S. B. Moon, J. Kurose, and D. Towsley, “Packet audio playout delay adjustment: Performance bounds and algorithm,” *ACM/Springer Multimedia Systems*, vol. 5, pp. 17-28, Jan. 1998.

[3] Y. J. Liang, N. Färber, and B. Girod, “Adaptive playout scheduling and loss concealment for voice communications over IP networks,” *IEEE Trans. Multimedia*, vol. 5, pp. 532-543, Dec.2003.

[4] P. DeLeon and C. Sreenan, “An adaptive predictor for media playout buffering,” in *Proc. IEEE Int. Conf. Acoustics, Speech, Signal Processing*, vol. 6, (Phoenix, AZ), pp. 3097-3100, Mar. 1999.

[5] A. Shallwani and P. Kabal, “An adaptive playout algorithm with delay spike detection for real-time VoIP,” in *Proc. IEEE Canadian Conf. Elec. Comp. Eng*., (Montreal, Canada), May 2003.

[6] Y. J. Liang, N. Färber, and B. Girod, ”Adaptive playout scheduling using time-scale modification in packet voice communications,” in *Proc. IEEE Int. Conf. Acoustics, Speech, and Signal Processing (ICASSP)*, vol. 3, Salt Lake City, UT, May 2001, pp. 1445-1448.

[7] F. Liang, J. Kim and C.-C. J. Kuo, “Adaptive delay concealment for Internet voice applications with packet-based time-scale modification,” in *Proc. IEEE Int. Conf. Acoustics, Speech, and Signal Processing (ICASSP)*, pp. 1461-1464, 2001.

[8] C. A. Rødbro and S. H. Jensen, "Time-scaling of Sinusoids for Intelligent Jitter Buffer in Packet Based Telephony", *2002 IEEE Speech Coding Workshop Proceedings*, October, 2002, pg. 71-73

[9] J.-C. Bolot, "Characterizing end-to-end packet delay and loss in the

Internet," *J. High-Speed Networks*, vol. 2, no. 3, pp. 289-298, Dec. 1993.

[10] J.-C. Bolot and A. Vega-Garcia," The case for FEC-based error control for packet in the internet," *ACM Multimedia Systems*, 1997.

[11] Lee, Chen-Long / Chang, Wen-Whei / Chiang, Yuan-Chuan (2004): "Application of voice conversion to hearing-impaired Mandarin speech enhancement", In *INTERSPEECH-2004*, 1829-1832.

[12] 楊雅茹, 「聲韻轉換及其在中文口語訓練之應用」, 國立交通大學碩士論文，民國九十二年。