網際網路上語音傳輸機制之研究

Delivering Voice over the Internet

郭經華、周家麟、許宗傑、陳維浩

電腦與網路 (CAN)實驗室
資訊工程學系
淡江大學
E-mail: chkuo@mail.tku.edu.tw

摘要

由於網際網路有封包延遲差異和封包丟失的問題，在這樣環境之上進行即時傳輸是複雜且具挑戰性的。許多研究人員已致力於這個研究領域。本論文中我們發展一個語音傳送方案以解決上述問題。爲了克服延遲差異和封包丟失，可調適語音同步方案採用回饋式。此方案由以下機制所組成：(1) 延遲與丟失測量機制，(2) 服務品質調整機制。眾所皆知，對網路型多媒體系統而言，語音是溝通表達上非常重要的媒體。因此，本論文得到的結果對許多網路型多媒體系統如電腦協同作業、網路電話或多媒體隨選教材等均有直接貢獻。

1. 簡介

在網路型多媒體系統中，語音是非常重要的媒體。然而目前網際網路只提供點對點盡力傳輸的服務品質，如此會造成封包延遲、封包延遲差異和封包丟失。這些現象對即時多媒體應用而言是不利的，而在這樣的環境之上進行語音傳輸是個複雜且具挑戰性的議題。簡單來說，解決方法可分爲兩種：網路層類型和應用層類型。網路層類型中，如ATM公會和IETF的點對點研究小組，專注於提供多媒體網路環境以支援即時連續媒體傳輸。應用層類型中，專注於在應用層發展可調適方案以彌補網路缺陷。本論文採用第二種方法。

近幾年來有許多進展。第一代語音會議系統，如 Nevot [25]，和 vat [26]，在網路負荷較輕的情況下能傳送好的語音品質。第二代語音工具有更進一步的改良，包括：(1) 低位元率重複資訊傳送機制以克服封包丟失 [10][11]，(2) 可調適播放機制以降低網路差異 [5][16]，(3) 媒體間同步機制 [6]。目前已有許多能在網際網路上進行語音傳輸的商業產品，如 I-phone，Cool Talk，和 Net Meeting 等等。然而這些產品，到目前爲止，在中等網路負荷情況之下進行語音播放，其表現並不理想。

本論文中，我們設計了一個網際網路語音工具，所提出的機制包括重複資訊發送機制，網路監控機制和可調適播放機制。重複資訊發送機制從一壓縮封包格式集合中決定發送封包的格式以補償封包丟失。網路監控機制收集目前網路的傳輸情況如延遲、延遲差異和封包丟失，所得到的資訊將用於播放時間的調整。可調適播放機制根據網路情況調整播放時間。我們強化Moon et al. [1]所提的可調適播放機制，也就是在播放時間決定上考慮封包丟失。於是，語音播放的品質可更進一步地改善。

本論文其他部分組成如下：第2 節包含可調適語音播放系統概觀和相關成果，第3 節包含所提出的語音傳送方案的設計與分析，第 4 節爲本機制的實驗結果，最後是結論。

2. 系統概觀

如圖 1 所示，在封包交換網路上語音傳輸包含六個階段：在發送端進行數位化，壓縮和傳輸；在接收端進行接收，解壓縮和播放。如前所述，在網路上進行語音傳輸可能遭遇封包延遲，封包延遲差異和封包丟失。於是，爲了確保能在接收端連續地進行語音的播放，需要能彌補網路延遲差異和能復原已丟失封包的機制。

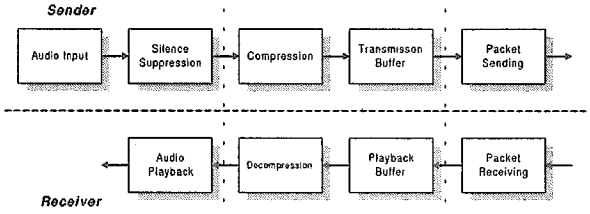

圖 1 語音傳輸系統架構圖

延遲差異補償機制的設計上有三種方法，包括(1) 緩衝區式，(2)預測式，和 (3)統計式

(1) 緩衝區式

此法在接收端使用佇列長度當做選擇播放時間的主要因素。當佇列長度大於某一預先定義的門檻值，則簡單地以捨棄之後到達的封包以減少整體播放延遲 [11][12][13]。這種方法適用於延遲差異小的情況。它減少因緩衝區饑餓而造成的播放不連續。然而，在面對延遲差異大的情況下，此法無法得到令人滿意的結果。

(2) 預測式

此法藉由測量網路延遲或網路延遲差異動態地調整播放時間。其使用線性遞迴過濾器去估計平均網路延遲和延遲差異。而且，在 [5] 所發展出的尖峰時期偵測演算法可處理突然湧現的延遲問題。

(3) 統計式

統計式在 [1][6] 被提出。此法計算最近封包延遲的統計量。當一個新的封包抵達接收端時便更新網路

延遲和延遲差異統計資訊。然後根據事先定義好了的封包丟失率以選擇播放時間。

封包丟失復原設計上，簡單的解決方法是以複製封包插入或用靜音替代那些已丟失的封包。除此之外，重複封包發送機制和錯誤復原機制出現在 [8][9][10][11]，其想法是使用高壓縮比但低品質壓縮方法以包裝重複資訊以補償那些已丟失的封包。

在我們的設計上，我們整合可調適統計式播放法和重複資訊發送機制。我們設計所包含的機制可參看圖 2。在發送端方面，在語音輸入裝置之後有 (1) 語音訊號處理，(2) 壓縮和重複資訊的產生，和 (3) 即時封包傳輸機制。在接收端方面，有 (1) 即時封包接收，(2) 延遲與丟失測量，(3) 封包重建，(4) 可調適播放緩衝區，(5) 語音解壓縮，然後是語音輸出裝置。

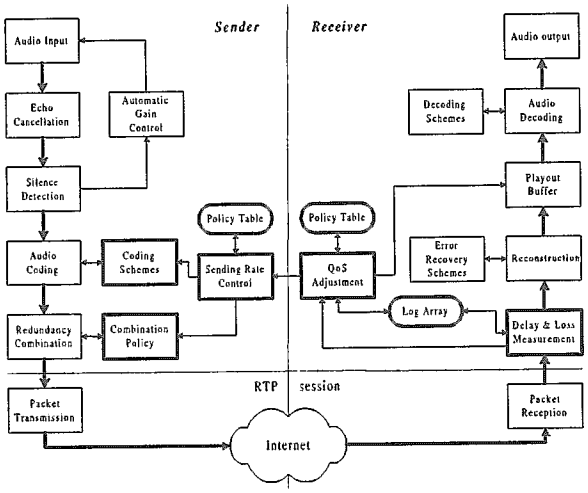

圖 2 可調適即時語音傳輸系統架構圖

以下簡述我們設計上相關的部分：

(1) 語音訊號處理

語音訊號處理是數位化語音傳輸的前處理過程。這部分包括靜音偵測，迴音消除和自動增益控制。這部分的實作能明顯地改善網路上語音品質。

(2) 壓縮和重複資訊產生

爲了有效使用網路頻寬，壓縮方法的選擇是非常重要的。許多有著不同計算複雜度和壓縮比的壓縮標準已被提出。爲了克服傳輸時封包丟失，如 [8] 所提，可用不同的壓縮方法構成一封包格式集合。然後，根據不同的網路情況發送不同的封包格式。

(3) 即時封包傳輸

我們使用 RTP 機制當成傳送即時多媒體服務的主要工具。使用 RFC1889 [23] 將不同的封包格式整合進一個封包，然後再送出。在接收端接收到封包後，其時間戳記可被取出以更進一步地分析網路情況。

(4) 延遲與丟失測量機制

延遲與丟失測量機制的主要工作包含判斷是否爲新的談話時期開始封包，判斷目前網路延遲狀態，判斷是否需要使用重複資訊重建已丟失的主要資訊，更新網路延遲時間的分佈並且將重複資訊的使用量、封包序號、封包網路延遲時間一併記錄到記錄陣列。

(5) 服務品質調整機制

根據延遲與丟失測量機制傳來的資訊決定頻寬使用量和重複資訊使用量，向決策表格查詢以得到符合條件的組合方式，接著再決定出下一個談話時期的播放時間，最後將相對應的決策編號送給發送端以告知封包發送機制。假如封包無法在播放時間前抵達接收端就使用複製封包以重建那些丟失封包。

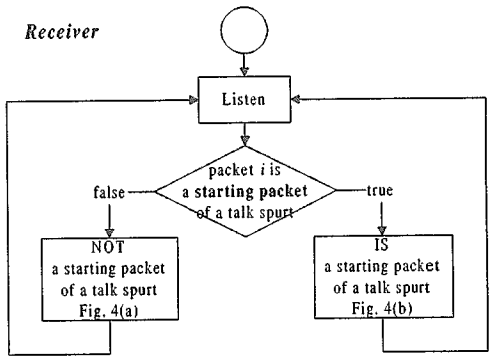

圖 3 接收端封包處理流程圖

如圖 3 所示，接收端監聽到有封包進來時，首先判斷是否爲新的談話時期開始封包，接下來步驟如圖 4(a)(b) 所示。

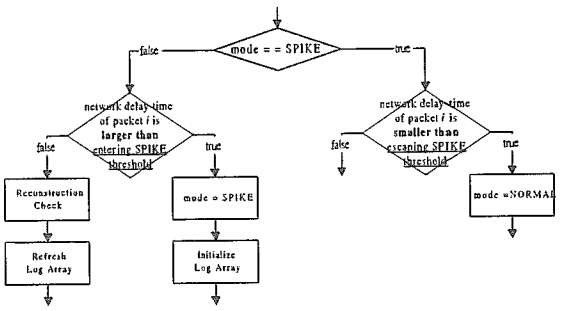

圖 4 (a) 封包 *i* 非新的談話時期開始封包之流程圖

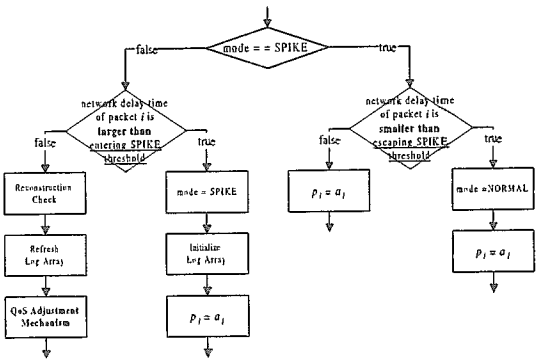

圖 4 (b) 封包 *i* 爲新的談話時期開始封包之流程圖

3. 語音工具的設計

在介紹我們所提出的方法前，先定義下述名詞：(1) 整體延遲時間和 (2) 封包丟失率。

定義：整體延遲時間 (Total End-to-End Delay; TED)

封包 i 的整體延遲時間定義如下：

$d_{Ti} = d_{Ni} + d_{Ri} + d_{Pi}$

其中 d_{Ni} 代表封包 i 的網路延遲時間，
d_{Ri} 代表封包 i 的重建延遲時間，
d_{Pi} 代表封包 i 的播放延遲時間。

定義系統可容忍的最大整體延遲時間爲 D_{max}，封包 i 的抵達時間必須滿足 $d_{Ti} \leq D_{max}$。

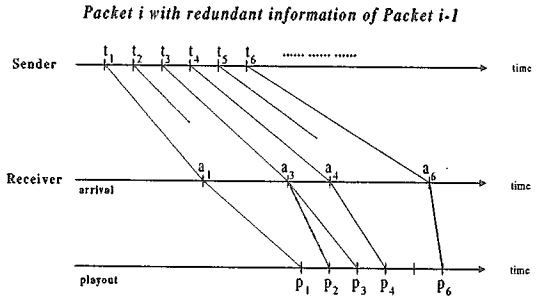

圖 5 封包整體延遲時間

整體延遲時間的描述如圖 5 所示。對每一個抵達封包 i 而言，藉由計算發送端和接收端之間時間戳記的差值得到封包 i 的網路延遲時間。藉由使用線性遞迴過濾器或線上分配統計的方式可估計出播放延遲時間 d_{Pi}。定義在連續 w 個封包中，百分之 q 的封包可以被播放的延遲時間爲 D_q。封包 i 可以被播放的條件爲 $d_{Ni} \leq D_q$。封包 i 的播放延遲時間爲 $d_{Pi} = D_q - d_{Ni}$。q 愈大則可以被播放的封包百分比愈高。封包 i 的重建延遲時間爲 d_{Ri}，此延遲時間爲的是進行以重複資訊爲主的錯誤復原機制重建丟失資訊之用。這額外的延遲對滿足 $p_i - a_{i+1} > 0$ [4] 的丟失封包復原是有幫助的。最差的情況下，封包重建延遲時間 d_{Ri} 可選擇重複封包的最大相偏差。可調適方法使用重建係數 λ 和封包內重複資訊個數 n 和封包發送間隔 Δ 三者的乘積，也就是 $d_{Ri} = \lambda * n * \Delta$。$\lambda$ 愈大，封包重建的百分比愈高，但會造成較高的封包重建延遲時間 d_{Ri}。

定義：整體封包丟失率 (Total Packet Loss Rate; TLR)

最近 w 個封包的整體封包丟失率定義如下：

$l_{Tw} = l_{Nw} + (1 - l_{Nw}) * l_{Pw}$ ..... 不需重建
$l_{Tw} = l_{Rw} + (1 - l_{Rw}) * l_{Pw}$ ..... 需要重建

其中 l_{Nw} 代表最近 w 個封包於傳輸期間的封包丟失率，也就是網路丟失率，

l_{Rw} 代表最近 w 個封包於重建後的封包丟失率，也就是重建後丟失率，

l_{Pw} 代表最近 w 個封包因來不及被播放而被捨棄的封包遺棄率，也就是播放丟失率。

定義系統可容忍的最大整體封包丟失率爲 L_{max}，最近抵達的 w 個封包必須滿足 $l_{Tw} \leq L_{max}$。

[1] 並沒有重複封包發送機制，所以在分析過程中並沒有考慮到重建過程。在我們的設計中，因爲重複封包發送機制可以有效處理封包丟失的問題，因此我們會考量到重建過程。整體封包丟失率 l_{Tw} 等於封包網路丟失率加上封包接收率和播放丟失率的乘積。TLR 的定義和傳統上封包丟失率是不同的。在圖 6 我們用一個例子說明如何計算整體封包丟失率 (TLR)。

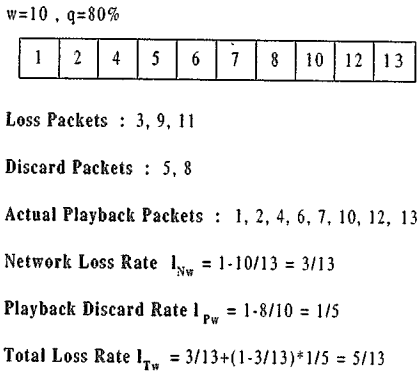

圖 6 整體封包丟失率

一個典型的網路延遲機率分佈如圖 7 所示。假設沒有網路丟失封包的情況，只要調整播放時間即可減少因爲網路延遲所造成的播放不連續。舉例而言，假如百分之 95 的封包在播放時間 p 之前抵達，則被捨棄的部分占百分之 5。假設此時相對應的最大傳輸延遲爲 d，滿足 $d \leq p$。假如想要降低播放遺棄率到百分之 3，則百分之 97 的封包必須在播放時間之前抵達，最大傳輸延遲的範圍將擴大到 d'。播放時間將從 p 移到 p' 使得 $p' \geq d'$。於是播放時間的調整同 影響封包丟失率和播放互動性。在沒有網路丟失封包的情況下，播放時間和封包丟失率這兩個因素的關係是互爲消長的。

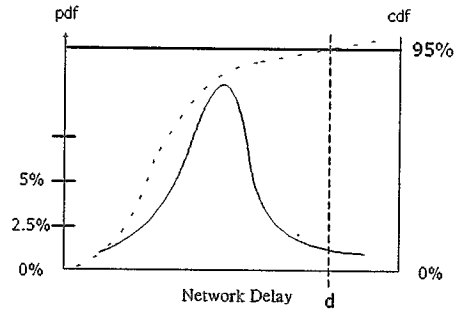

圖 7 網路延遲分佈圖

接著我們簡述 Moon et al. [1] 的結果。如圖 8 所示，演算法可被用來線上決定播放延遲時間。它從抵達接收端的封包收集資訊，然後所收集到的資訊被用來估計播放延遲時間。演算法記錄每一個抵達封包網路延遲時間然後更新相對應的統計量。每一個談話時期從最近 w 個封包所得到的統計量決定可被播放的比例。然後該值被用來決定下一個談話時期的播放延遲時間。當一個封包到達接收端，演算法 1~7 列負責尖峰時期的偵測。談話時期偵測的靈敏度由演算法中參數 *head* 和 *tail* 所決定。

假如目前封包網路延遲時間大於 *head* 和 curr_delay 的乘積則切換爲尖峰狀態。同時，演算法將停止計算網路延遲統計量而將目前網路延遲時間當做播放延遲時間的估計值。假如目前網路延遲時間小於 *tail* 和 curr_delay 的乘積則切換爲正常狀態。

```
IF ( mode == SPIKE )
  IF ( d_i ≤ tail * old_d )   /* the end of spike */
    mode = NORMAL;
ELSE
  IF ( d_i > head * p_i )     /* the beginning of a spike */
    mode = SPIKE;
    old_d = p_i               /* save p_i to detect the end of a spike later */
  ELSE
    IF ( delays[curr_pos] ≤ curr_delay)
      count --1;
      distr_fcn[delays[curr_pos]] --1;
      delays[curr_pos] = d_i ;
      curr_pos = (curr_pos + 1) % w;
      distr_fcn[d_i] ++1 ;
    IF ( delays[curr_pos] < curr_delay)
      count ++1 ;
    while ( count < w * q )
      curr_delay ++ unit;
      count ++ distr_fcn[curr_pos};
    while ( count > w * q )
      curr_delay -- unit;
      count -- distr_fcn[curr_pos};

IF ( mode == SPIKE )
  p_i = d_i;
ELSE ( mode == NORMAL )
  p_i = curr_delay;
```

圖 8 播放時間決定演算法

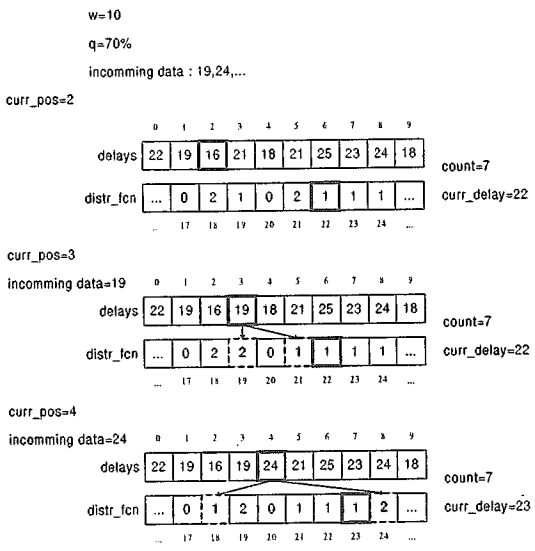

圖 9 線上延遲統計量計算方法

基於線上延遲統計播放延遲調整的簡略說明如圖 9 所示。演算法從抵達封包收集資訊以估計播放時間。它記錄每一個抵達封包的網路延遲時間和更新相對應的統計量。對每一個談話時期而言，它計算最近 w 個封包的統計量以決定可以被播放的百分比 q 的播放延遲時間。然後這個值被用來當做是下一個談話時期的播放延遲時間。以下用一個例子來說明演算法的運作情形。假設 w = 10 且 q =0.7，接下來連續進來的封包網路延遲時間分別爲 19 和 24ms。當網路延遲時間爲19ms的封包抵達接收端，演算法將它記錄在記錄陣列中且更新記錄陣列中其他相對應的值。計數器 *count* 計算 $w*q$ =7 接著平移 curr_delay 指標爲滿足 $w*q$ 的目前播放延遲時間。下一個網路延遲時間爲 24ms 的封包也是以相同的運作方式進行。每當一個封包抵達接收端就更新一個新的估計值。於是，在遇到一個談話時期開始封包時的計算結果是下一個談話時期的播放延遲時間。

3.1 延遲與丟失測量機制

延遲與丟失測量機制主要工作包含判斷是否爲新的談話時期開始封包，判斷目前網路延遲狀態，判斷是否需要使用重複資訊重建已丟失的主要資訊，更新網路延遲時間分佈並且將重複資訊的使用量、封包序號、封包網路延遲時間一併記錄到記錄陣列。

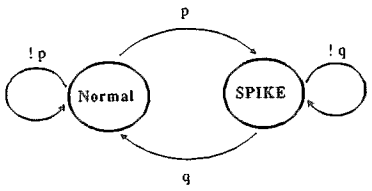

圖 10 網路延遲狀態

接收端收到封包之後，先根據該封包標示位元值判斷是否爲新的談話時期開始封包，然後根據網路延遲狀態值（mode）判斷是否屬於尖峰狀態，然後計算該封包的網路延遲時間判斷目前是屬於正常狀態（Normal mode）或是尖峰狀態（SPIKE mode），如圖 10 所示。其中 p 代表該封包網路延遲時間大於進入尖峰門檻值，!p 代表該封包網路延遲時間不大於進入尖峰門檻值；q 代表該封包網路延遲時間小於離開尖峰門檻值，!q 代表代表該封包網路延遲時間不小於 離開尖峰門檻值。

```
01  IF (mode == SPIKE)
02    IF (d_Ni < tail * old_d)
03      mode = NORMAL;
04  ELSE
05    IF (d_Ni > head * d_Ti)
06      mode = SPIKE;
07      old_d = d_Ti;
08      LogArray_init( );
09    ELSE
10      ReconstructionCheck( );
11      distr_func[delays[curr_pos]] - = 1;
12      distr_func[d_Ni] + = 1;
13      use_redundancy[curr_pos] = c_i
14      sequence_num[curr_pos] = s_i
15      delays[curr_pos] = d_Ni;
16      curr_pos = (curr_pos + 1)%w;
```

圖 11 尖峰狀態偵測與網路延遲時間線上統計演算法

3.1.1 封包 i 非新的談話時期開始封包

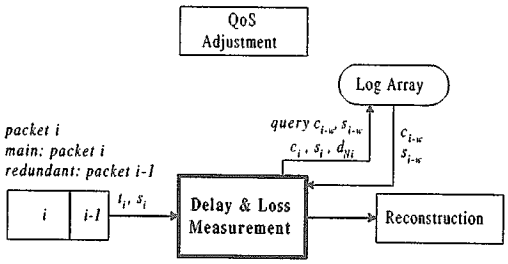

圖 12 延遲與丟失測量機制（封包 i 非新的談話時期開始封包）

接收端收到封包後取出該封包RTP標頭內的時間戳記以計算出該封包網路延遲時間 d_{Ni}。

$$d_{Ni} = a_i - t_i$$

接著如圖 12 所示，延遲與丟失測量機制根據該封包的網路延遲時間 d_{Ni} 判斷目前是屬於正常狀態（mode == NORMAL）還是尖峰狀態（mode == SPIKE），過程如圖 11 列號 01~08 所示。

然後進行是否重建之前已丟失封包的檢查。先確認封包 i 是否有之前封包的重複資訊。如果封包 i 內沒有重複資訊，則設定封包 i 的重複資訊使用量 c_i 爲 0。如果封包 i 內有重複資訊，則延遲與丟失測量機制向記錄陣列查詢之前封包丟失情形，然後根據目前談話時期每個封包的播放時間點來判斷倘若封包 i 內重複資訊相對應的主要資訊已丟失，封包 i 內相對應的重複資訊是否來得及拿來重建而被播放，藉此決定封包 i 的重複資訊的使用量 c_i。

接著延遲與丟失測量機制向記錄陣列查詢前 w 個封包重複資訊使用量 c_{i-w}，得到 c_{i-w} 再向系統取得目前 c_{Tw} 值，如此即可更新最近到達接收端的 w 個封包重複資訊的總使用量 c_{Tw}。

$$c_{Tw} = c_{Tw} + c_i - c_{i-w}$$

然後更新網路延遲時間的分佈，將 c_i，s_i，d_{Ni} 記錄到記錄陣列之中，如圖 11 列號11~16所示。

接著延遲與丟失測量機制向記錄陣列查詢前 w 個封包的序號，得到 s_{i-w} 再取出該封包 RTP 標頭內的序號 s_i，即可求出接收端收到封包 i 時最近接收的 w 個封包於發送端實際傳送的封包個數 δ_i，如此即可求得收到封包 i 之後最近 w 個封包的網路丟失率 l_{Nw}。

$$\delta_i = s_i - s_{i-w}$$

$$l_{Nw} = 1 - \frac{w}{\delta_i}$$

如此即可更新代表長時間網路丟失率的平滑後丟失率 l_{smooth}。

$$l_{smooth} = (1-\alpha) * l_{smooth} + \alpha * l_{N_w}$$

然後拆解RTP標頭之後的資料將傳給重建機制處理，接著系統等待下一個封包的到來。

3.1.2 封包 i 爲新的談話時期開始封包

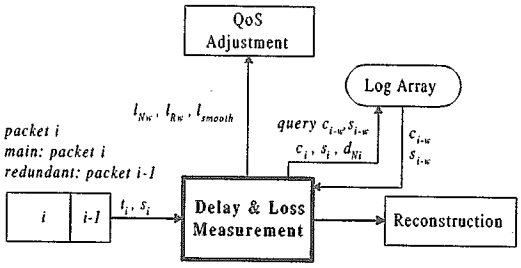

圖 13 延遲與丟失測量機制（封包 i 爲新的談話時期開始封包）

接收端收到封包後依序判斷目前網路延遲狀態，判斷是否需要使用重複資訊重建已丟失的主要資訊，更新網路延遲時間分佈並且將重複資訊的使用量、封包序號、封包網路延遲時間一併記錄到記錄陣列，這幾個部分如同3.1.1所闡述。

得到 c_{Tw} 和 δ_i 之後，即可求得最近到達接收端的 w 個封包的錯誤修正率 e_{Tw}，接著並可算出最近到達接收端的 w 個封包重建後的丟失率 l_{Rw}。

$$e_{Tw} = \frac{C_{Tw}}{\delta_i - w}$$

$$l_{Rw} = l_{Nw}(1 - e_{Tw})$$

然後將 l_{Nw}，l_{Rw} 和 l_{smooth} 傳給服務品質調整機制，而拆解 RTP 標頭之後的資料將傳給重建機制處理。

3.2 服務品質調整機制

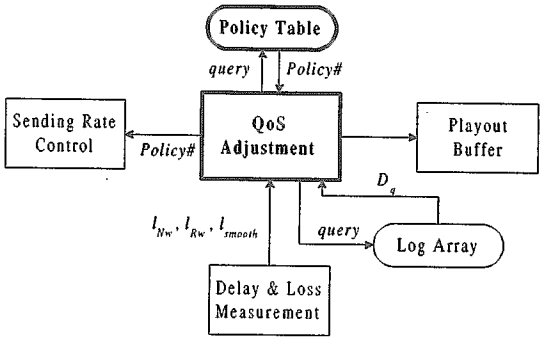

圖 14 服務品質調整機制

接收端收到新的談話時期開始封包才會去執行服務品質調整機制。如圖 14 所示，服務品質調整機制根據延遲與丟失測量機制傳來平滑後的網路丟失率 l_{smooth} 查詢相對應的網路擁塞狀態以決定頻寬的使用量，如表 1 所示。

然後根據延遲與丟失測量機制傳來的 l_{Nw} 和 l_{smooth}，取其中較大者以查詢重複資訊的使用量，如表 2 所示。

接著根據剛才所得的頻寬使用量和重複資訊的使用量向決策表格查詢符合條件的可能組合，選取其中主要資訊和重複資訊 MOS 值最高者，然後將決策編號傳給發送端(表 3)。

表 1 網路擁塞狀態 vs. 頻寬使用量

Loss Rate	State	Bandwidth	New Bandwidth
< L% L_{max} = a%	Unloaded	Increase	bandwidth = bandwidth+ μ
L% ~ U% L_{max} = b%	Loaded	Maintain	bandwidth = bandwidth
> U% L_{max} = c%	Congested	Reduce	bandwidth = bandwidth* ν

表 2 丟失率 vs. 重複資訊使用量

Loss Rate	Primary	1st Redundancy	2nd Redundanc
< L'%	○		
L'% ~ U'%	○	○	
> U'%	○	○	○

表 3 決策表格

Redun-dancy	Primary	1st redundant	2nd redundant	Band-width	Polic
0	PCM	-	-	64	1
0	ADPCM	-	-	32	2
0	GSM	-	-	13	3
1	PCM	ADPCM	-	96	4
1	PCM	GSM	-	77	5
1	PCM	LPC	-	68.8	6
1	ADPCM	GSM	-	45	7
1	ADPCM	LPC	-	36.8	8
1	GSM	LPC	-	17.8	9
2	PCM	ADPCM	GSM	109	10
2	PCM	ADPCM	LPC	100.8	11
2	PCM	GSM	GSM	90	12
2	PCM	GSM	LPC	81.8	13
2	PCM	LPC	LPC	73.6	14
2	ADPCM	GSM	GSM	58	15
2	ADPCM	GSM	LPC	49.8	16
2	ADPCM	LPC	LPC	41.6	17
2	GSM	GSM	GSM	39	18
2	GSM	GSM	LPC	30.8	19
2	GSM	LPC	LPC	22.6	20

然後根據從延遲與丟失測量機制傳來的重建後的丟失率 l_{Rw} 和向系統查詢該網路擁塞狀態所能容忍的最大整體丟失率 L_{max}，即可求得到達接收端的 w 個封包的網路延遲分佈中可被播放的百分比 q。

$$L_{\max} \geq l_{Tw}$$
$$L_{\max} \geq l_{Rw} + (1 - l_{Rw}) * l_{Pw}$$
$$L_{\max} \geq l_{Rw} + l_{Pw} - l_{Rw} * l_{Pw}$$
$$L_{\max} \geq l_{Rw} + (1-q) - l_{Rw} * (1-q)$$
$$q(1 - l_{Rw}) \geq 1 - L_{\max}$$
$$q \geq \frac{1 - L_{\max}}{1 - l_{Rw}}$$

接著向記錄陣列反覆查詢，可決定出最近到達接收端的 w 個封包的網路延遲分佈中，百分之 q 的封包可以被播放的延遲時間 D_q，如圖 15所示。

```
count = 0;
curr_delay = 0;
WHILE (count < w * q)
    curr_delay + = unit;
    count + = distr_func[curr_delay];
```

圖 15 線上統計式播放延遲調整機制演算法

得到 D_q 之後，服務品質調整機制再根據重複資訊的使用量 n，計算出下一個談話時期的整體延遲時間。

$$d_{Ti} = D_q + d_{R1}$$
$$d_{Ti} = D_q + \lambda * n * \Delta;$$

圖 16 決定下一個談話時期整體延遲時間

封包播放方式採用絕對時間法，也就是在每一個談話時期的第一個封包到達接收端之後就決定了往後屬於同一個談話時期的其他到達封包的播放時間。如網路延遲爲尖峰狀態則下一個談話時期的播放時間爲下一個談話時期第一個封包到達接收端的時間；如網路延遲爲正常狀態，若無錯誤復原機制則每個 D_q 的計算結果作爲下一個談話時期的播放時間，若有錯誤復原機制則除了 D_q 之外還必須增加重建時間 d_{Ri}，而 $d_{Ri} = \lambda * n * \Delta$ 。

因此根據絕對時間法的播放模式：

每個談話時期第一個封包的播放時間改變爲：

$$\textbf{IF (mode} == \text{SPIKE)}$$
$$p_1 = a_1$$
$$\textbf{ELSE (mode} == \text{NORMAL)}$$
$$p_1 = t_1 + D_q + d_{R1}$$

每個談話時期第一個封包以外其他封包的播放時間改變爲：

$$p_i = p_1 + t_j - t_1$$

4. 實驗結果

我們的實驗是由台北的淡江大學至高雄的中山大學之間來進行，網路的丟失率大約爲 20% 至 30%，實驗參數的選擇爲：α = 0.998002，L_t = L' = 5，U = U' = 25，μ = 2.4kbps and ν = 0.875. 語音封包是以每 20 ms 當作發送單位，實驗中一共測量了 38,000 個語音封包，結果如圖17所示。

圖中的實驗是經由此調適機制的執行結果，虛線表示網路的丟失率(l_{Nw})，實線則代表重建後的丟失率(l_{Rw})。實驗結果顯示出，網路丟失率大約縮小至 10% 左右，而重建後的封包丟失率則降到 1% 至 2% 之間。

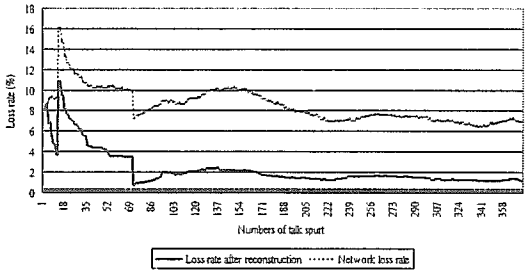

圖17 實驗結果

5. 結論

眾所皆知，網際網路有著封包延遲，封包延遲差異，封包丟失等問題。在這樣的環境下要進行連續媒體傳輸是個複雜且具挑戰性的事。本論文中，我們整合了三個重要的機制：(1)根據所收集到的資訊以決定發送封包格式和組合策略的服務品質調整機制，(2) 考慮封包丟失的可調適語音播放機制　處理網際網路上語音封包傳輸，和(3)為了解決封包經由網路傳輸而可能造成丟失的重複資訊發送機制。如此提供的整合性方案可提供網路型多媒體系統當做語音工具之用。

國科會計劃編號：NSC88-2213-E032-002

參考資料

1. Sue B. Moon, Jim Kurose, Don Towsley, "Packet Audio Playout Delay Adjustment: Performance Bounds and Algorithms", *ACM Multimedia*, Vol. 6, Jan 1998, pp. 40-48.
2. K. V. Chin, S. C. Hui and S. Foo, "Enhancing the quality of Internet voice communication for Internet telephony systems", *Journal of Network and Computer Applications*, Vol. 21, July 1998, pp. 203-218.
3. Po. L. Tien, Maria C. Yuang, "Intelligent Voice Smoother for VBR Voice over ATM Network", *Proc. IEEE Infocom '98.*
4. Bert J. Dempsey Yangkun Zhang, "Destination Buffering for Low-Bandwidth Audio Transmission using Redundancy-Based Error Control", *21st IEEE Local Computer Networks Conference, Minneapolis, MN*, pp. 345 – 355, October 1996.
5. R. Ramjee et al, "Adaptive Playout Mechanisms for Packetized Audio Applications in Wide-Area Network", *Proc. IEEE Infocom'94*, pp.680-688, Jun 1994.
6. N. Shivakumar, C. J. Sreenan, B. Narendran,P. Agrawal, "The Concord Algorithm for Synchronization of Networked Multimedia Streams", *Proc.ICMCS'95*, pp.31-40.
7. M. Podolsky , C. Romer , S. McCanne , "Simulation of FEC-Based Error Control for Packet Audio on the Internet", *Proc. IEEE Infocom'98.*
8. Jean-C. Bolot, Andres V-Garcia, "Control Mechanism for Packet Audio in the Internet", *Proc. IEEE Infocom '96*, pp.232-239
9. C. Perkins, J-C Bolot, "RTP Payload for redundant Audio Data", *RFC 2198*, Sep 1997.
10. Jean-C. Bolot, Andres V-Garcia, "The Case for FEC-based Error Control for Packet Audio in the Internet", *ACM Multimedia Systems.*
11. Vicky Hardman, Mark Handley, "Reliable Audio for Use over the Internet", *Proc. INET'95, Honolulu, HI*, pp.171-178, Jun 1995.
12. Donald L. Stone, Kevin Jeffay, "An Empirical Study of Delay Jitter Management Policies", *ACM Multimedia Systems*, pp.267-279, 1995.
13. Sanjay Jha, Micheal Fry, "Continuous Media Playback and Jitter Control", *Proc ICMCS'96*, pp.245-252.
14. Jones A., Hopper A, "Handling Audio and Video Streams in a Distributed Environment", *Proc. ACM symposium on Operating Systems Principles, Asheville, NC, Operating Systems Review Vol.27 No.5*, pp.231-243, 1993.
15. Felipe Alvarez-Cuevas, Miquel Bertran, Francesc Oller, Josep M. Selga, "Voice Synchronization in Packet Switching Networks", *IEEE Network*, pp.20-25, Sep 1993.
16. W. Montgomery, "Techniques for Packet Voice Synchronization", *IEEE J. on Selected Area in Communications, Sol. SAC-6, No.1*, pp.1022-1028, Dec 1983.
17. Jean-C. Bolot, Andres V-Garcia, "End-to-End Packet Delay and Loss Behavior in the Internet", *Proc. ACM Sigcomm '92*, pp.248-257
18. Jan Gecsei, "Adaptation in Distributed Multimedia Systems", *IEEE Multimedia*, pp.58-66, Apr-Jun 1997.
19. Van Jocobson, "Congestion Avoidance and Control", *ACM Sigcomm*, pp.314-329, *London, England*, Aug.1988.
20. Jean-C. Bolot, Andres V-Garcia, "A rate Control Scheme for Packet Video in the Internet", *Proc. IEEE Infocom '94*, pp.1216-1223, Jun 1996.
21. Thierry Turletti, Sacha Fosse Parisis, J-C Bolot, "Experiments with a Layered Transmission Scheme over the Internet", *to appear in Infocom '98.*
22. S. Casner, "First IETF Internet audiocast", *Computer communication Review, vol.22, no.3*, pp.92-97, July 1992.
23. H. Schulzrinne, S. Casner, R. Fredreck, V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", *RFC 1889, Internet Engineering Task Force*, Feb. 1. 1996.
24. Sue B. Moon, Jim Kurose, Paul Skelly, Don Towsley, "Correlation of Packet Delay and Loss in the Internet", *Technical Report 98-11, Department of Computer Science, University of Massachusetts, Amherst, MA 01003.*
25. H. Schulzrinnne, "Voice Communication across the Internet: A Network Voice Terminal", *Research report, Dept. Electrical and Computer Engineering, University of Massachusetts.*
26. vat - LBNL Audio Conferencing Tool, http://www-nrg.ee.lbl.gov/vat/