乙太網路上適用於多媒體傳輸的時間校正協定

Clock Synchronization Protocol for Multimedia Transmission on Ethernet

顧金福[+] 李美雯[+] 李曜琮[+] 何健明[+] 劉如生[*]
Ku Chin-Fu Li Mei-Wen Li Yau-Tsung Ho Jan-Ming Liu Ru-Sheng

[+]中央研究院資訊科學研究所電腦系統與通訊實驗室 [*]元智大學電機與資訊工程研究所

Abstract

A simple but precise clock synchronization protocol is proposed to provide more exact time information for distributed multimedia and real-time applications to control data transmission over Ethernet. Basic ideas behind our design is that if we make the variance of packet transfer delay constant then we could correct every host's local clock more precise. As shown in preliminary results of our experiments, the clock difference between any two hosts in a LAN could be small than 1 millisecond.

摘要

共同的時間基準是分散式處理的一個基本要求，對多媒體與即時應用程式也是一項必需的功能。本文中提出一套以純軟體方式實作的時間校正協定，其基本概念是將網路上封包傳輸的時間控制在幾近常數的情況，藉由周期性的校正，以提高時間校正所能達到的精確度。目前在實作本文中所提時間校正協定原型後所做實驗結果顯示，在單段乙太網路上可將各主機時間校正誤差在一毫秒內。

Key Word:Ethernet, time synchronization protocol, multimedia.
關鍵字：乙太網路，時間校正協定，多媒體。

1 緒論

電腦科技的快速發展使得許多現在使用的協定、產品等受到挑戰與質疑："它們是否仍適用於愈來愈快速的現代？"。時間校正協定（Clock Synchronization Protocol）也是如此。當一般電腦的執行速度愈來愈快，網路傳輸速率也在不斷提高的情況下，許多應用軟體對系統提供的時間資訊，要求的精確度也愈來愈高。

在目前使用網路架構下，區域網路仍是主要的角色；而乙太網路是目前使用最廣的區域網路協定。然而乙太網路對多媒體或即時資料的傳輸，也有其不足之處；例如沒有提供優先權模式、可能發生不公平現象等；如何利用其特性，如何取其長而補其短，使時間校正能更精確、簡單，是發展此協定的動機之一。

需要時間校正協定的原因，是因爲各個主機因製程、品質、使用環境、溫度等因素影響，彼此的時間在一段時間後會有所誤差，而且如果沒有校正，誤差會越來越大。而目前網路上普遍使用的校正協定，有NTP（Network Time Protocol）[8, 9, 7. 11]、SNTP（Simple Network Time Protocol）[10]或TP（Time Protocol）[14]等；這些協定不是太過複雜，就是精確度太低。一個純軟體的時間校正協定或許可以提供合適的解決方案。

1.1 相關研究

目前網路上存在的校正協定，有 NTP（Network Time Protocol）[8, 9, 11]、SNTP（Simple Network Time Protocol）[10] 或 TP（Time Protocol）[14]等；然而即使是其中最精確的 NTP，還有一點在沒有特殊硬體輔助下，僅能精確到 10 至 100 毫秒（millisecond）；而在有特殊硬體如 GPS、PPS 的輔助下，則可以達到 45 微秒（microsecond）的精確度 [11]。在這些時間校正協定中，以 NTP [8]較爲著名，精確度也較高。雖然 NTP 在有特殊硬體的輔助下，可以達到小於毫秒的精確度，但是這種需要特殊硬體的方式不僅太過昂貴，協定也太過複雜。

NTP 的基本原理是將來自多個來源的時間資訊，利用過濾（Filtering）、選擇（Peer selection）、合併（Combining）等演算法，取得正確的時間資訊，在藉由鎖相迴路控制"電壓控制振盪器"（VCO），用來校正本身的時鐘（振盪器）[8]。

而其他的時間校正協定，如 UNIX 4.3BSD 中的 *timed* [3]、Daytime protocol [13]、Time protocol [14]、ICMP 時間標籤（time-stamp）[12] 等，雖然不像 NTP 那麼複雜，但是卻無法提供高精確度的時間校正。

1.2 問題簡述

時間校正協定的目的，是要讓一定區域範圍內的主機有一共同的時間標準，以作爲彼此互動的依據。而一般主機上的時鐘，都是使用一個計數器、一個預除器（prescaler）及一個石英振盪器，請參考圖 1。由石英振盪器提供穩定頻率的脈衝給預除器，驅使預除器在一定間隔時間發出訊號（中

斷），供計數器計數（增加）之用。

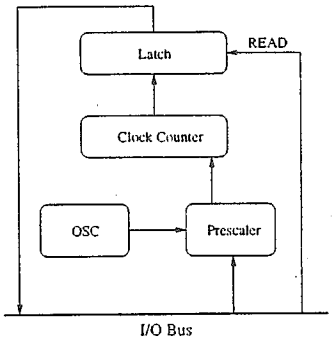

圖 1: 主機時鐘架構圖

就一般時間校正協定所使用的方法而言，幾乎都是由一標準主機將時間標籤裝在封包中送出，由接收端取出並校正其時間，亦即校正其計數器之值。這種方式最主要的問題就在於封包的傳輸。以圖 2 來解說，圖中 d_1 與 d_2 爲校正封包傳輸的延遲時間。主機 B 爲了將時間校正成與主機 A 的相同，則必須在收到第一個校正封包時（t_{b1}），將其時間校正成 $t_1 + d_1$，在收第二個校正封包時（t_{b2}），將時間校正爲 $t_2 + d_2$，以此類推下去。但問題在於每次的封包傳輸延遲時間不是一個常數（如圖中的 d_1、d_2 ...），而且無法得知其眞正值。這個問題在乙太網路上會更爲嚴重，因爲它多重存取的特性會使得封包傳輸時間更爲難以預測。不像其他的區域網路協定（例如記號環（Token Ring）、記號匯流排（Token Bus）、FDDI等），乙太網路會發生碰撞、沒有優先權模式、會發生不公平現象等，都使得在乙太網路上發展時間校正協定，需要注意更多事項。

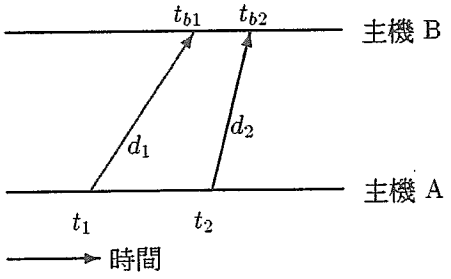

圖 2: 校正封包的傳輸延遲

傳輸延遲值的變化大，主因來自乙太網路上封包傳輸成功所需的等待時間無法預測。換句話說，主要問題在於乙太網路中等待時間的標準差（Standard Deviation）太大。任一塊乙太網路卡在成功傳輸一個封包前所要等待的時間從零到數十毫秒都有可能。就時間校正協定而言，標準差大卻易以使得校正後的時間精確度很難預估。爲了解決這樣的問題，必須將造成標準差大的所有原因消除：網路忙碌及碰撞。因爲網路忙碌，網路卡在測知網路有資料在傳輸的時候會退回等待，而這個等待時間會以幾何級數成長，這是造成等待時間不定的原因之一。

另一個原因則是碰撞，因爲發生碰撞後，發生碰撞的兩方都會退回（Back-off）等待，而這等待時間也是會依據發生碰撞的次數而成幾何級數增加（時間長度可能從 50 微秒到 50 毫秒之間 [5]）。但若碰撞次數超過十五次則會放棄傳輸該封包，所以這個等待時間並非沒有上限的。

1.3 章節概要

前面已將問題及相關研究作了簡明的介紹，下面則將就提出的時間校正協定作進一步說明。章節 2 將就此協定的基本構想與作法，作一介紹與探討。而章節 3 則就原型實作與實驗方面，予以分析與探討。最後在章節 4 中，則爲結論及未來工作。

2 基本原理

請參見圖 3 ，主機 B 的計時器相較於主機 A，速度較快；因爲主機 B 上計時器之值爲橫座標軸上 x 時，標準時間僅爲 x'（$x > x'$）。若設 A 之斜率爲一，從主機 B 那條線的斜率小於一可以得知其計時器快於 A 的。

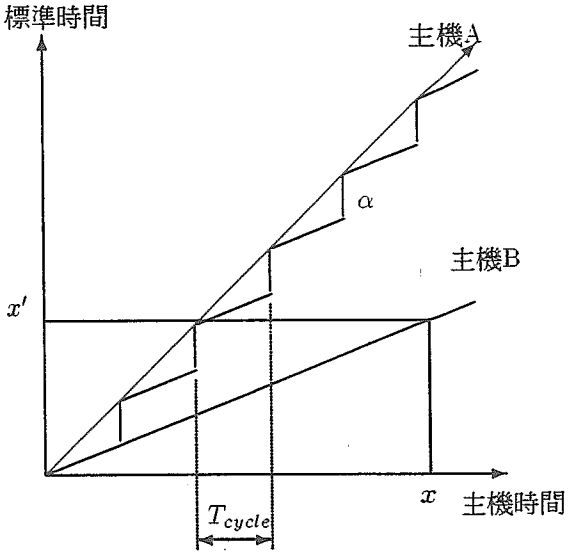

圖 3: 主機間的計時器差異及校正

此協定的基本假設則是任一台主機計時器的計時速率爲一常數。換句話說，若假設主機 A 之計時器計時速率爲 γ_A，在標準時間 t_0 時，主機 X 的時間被校正成 t_0 （即 $X(t_0) = t_0$），當過了 t 時間後，主機 X 的時間爲

$$X(t) = t_0 + t + \gamma_A(t - t_0)$$

就實際而言這個假設是合理，因爲會影響計時器計時速率的最主要因素就是「溫度」（[1]文中有溫度對振盪器的影響說明及其公式），而現今一般電腦主機所放置的位置，溫度變化都相當小，所造成的計時器計時速率的變化非常微小。在先期實驗中也顯示計時器的變化速率爲常數，證實此項假設。

在此假設之下，如果周期性地校正主機 B 的計時器，則兩台主機間的誤差便可以限制在某一常數

內。例如以一固定周期（圖中的 T_{cycle}）校正主機 B 的計時器，則主機 B 的計時器與主機 A 之間的誤差就會被限制在一常數內（圖中的 α），而達到時間校正的目地（此處假設主機 A 之時間爲基準）。

2.1 基本方法

此協定的基本構想是將連續的時間切割爲一個一個連續的期間，稱爲時間片段（Time Slot），以 T_{slot} 表示。而校正周期 T_{cycle} 是由三個子期間所組成；分別是共用期（$T_{csma/cd}$）、靜肅期（T_{silent}）及校正期（T_{resync}），它們各別由一定數量的 T_{slot} 組成；請參考圖 4。

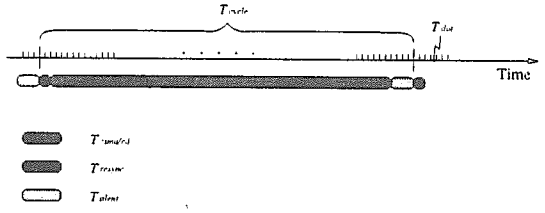

圖 4: 連續時間分割

網路上的主機則區分爲基準主機（Clock Master）及一般主機（Clock Client），基準主機的時間作爲各個一般主機時間校正的基準。基準主機會定期地發送時間校正封包，而各個一般主機則在收到校正封包後更新本身計時器的計數值，如此周而復始。

在共用期，所有主機均可發送封包；而在靜肅期及校正期，一般主機不可以發送封包。基準主機會在校正期時送出校正封包，而爲了確保在基準主機傳輸校正封包時，不會發生碰撞的情況（這是前提之一），因此在校正期前加入靜肅期，作爲保護校正封包之用。網路上任一主機啓動後必須等候由基準主機發出的校正封包，校正其本身封包後在等待由基準主機發出的時間表，以便知道何時可以傳送封包。

2.2 模式描述

爲了維持主機間計時器的計數值相同，必須周期性的校正各主機的計時器值。而此校正周期究竟須爲多少時間？這牽涉到兩個因素：一、各主機計時器所用振盪器振盪頻率的差異，二、所能容許的的時間誤差。

令 T_{cycle} 爲校正周期，α 爲任兩主機間最大可容許的時間誤差，而 $\gamma_{A/B}$ 爲主機 A 振盪器的振盪頻率相對於主機 B 的差值。此三者間的關係可以下列式子表示：

$$T_{cycle} \leq \frac{\alpha}{\gamma_{A/B}} \qquad (1)$$

因爲希望各主機之時間誤差在一毫秒之內，所以將 α 值設爲 150 微秒。而一般主機的 γ 之值，依 [1] 文中所提，石英振盪器在一般正確使用下，精確度大約爲每個月 ±30 秒；然而如果使用不當（例如電容器值過大或過小），則可能達到每個月 ±4 ~ ±8 分鐘的誤差。而在網際網路上的常見問題集（FAQ）中，則有提到一般個人電腦使用的 Real-Time Clock IC（如 Dallas DS1285、DS1287、Motorola MC146818 等），誤差約爲每個月 ±1 分鐘；但是 RTC 僅能達到"秒"的精確度，因此系統中還有一個計時器（系統計時器），以提供精確度更高的時間控制；這通常是由石英振盪器與計數器所組成。因此，石英振盪器的良莠、設計與使用得當與否都會影響主機間時間的差距。在此以設計、使用上不當的情況考慮，取誤差爲每月 ±4 分鐘的值計算，亦即 $\gamma_{A/B}$ 爲 8 分鐘/月(185.2 ppm)，則 T_{cycle} 最大值約爲 0.8 秒：$T_{cycle} \leq \frac{\alpha}{\gamma_{A/B}} = \frac{150us}{8min/month} \approx 0.8sec$。

2.2 校正誤差

下面分析討論中會用到的一些符號，其用法及意義在此先說明。函數 $f^+(t)$ 代表 $f(t+\epsilon)$，而 $f^-(t)$ 則代表 $f(t-\epsilon)$，其中 $\epsilon > 0$ 是一個任意小的常數。若 $\gamma_{A/B}$ 表示主機 A 相對於主機 B 的計時器變化率，則依據定義 $\gamma_{A/B} = \gamma_A - \gamma_B$，及 $\gamma_{A/B} = -\gamma_{B/A}$。

假設在校正周期時，機準主機 A 的將時間裝入校正封包送給主機 B 作爲校正之用。設函數 $\phi_A(t)$ 爲一連續函數，表示在標準時間 t 時主機 A 的時間。d'_n 爲主機 A 第 n 次時間裝入封包到送出所花費的時間，d_n 則是第 n 個校正封包傳輸的時間。γ_A 與 γ_B 爲主機 A 與主機 B 相對於標準時間的計時速率，$\gamma_{B/A}$ 則爲主機 B 相對於主機 A 的計時速率。假設 $\hat{d}$ 表示封包單向傳輸延遲的平均值。τ 代表校正後與標準主機實際值的誤差，即校正的精確度；而以 T 代表校正周期 T_{cycle}。以上列符號表示，可以列出：$\phi_A(nT + d_n + d'_n) = \phi_A(nT) + \gamma_A(d_n + d'_n)$，$\phi_B^+(nT + d'_n + d_n) = \phi_A(nT)$，$\phi_B^-(nT + d'_n + d_n) = \phi_B^+((n-1)T + d'_{n-1} + d_{n-1}) + \gamma_B(T + d'_n + d_n - d'_{n-1} - d_{n-1})$。

令 δ_n 爲第 n 次校正前與校正後的差值，可得

$$\begin{aligned}
\delta_n &= \phi_B^+(nT + d'_n + d_n) - \phi_B^-(nT + d'_n + d_n) \\
&= \phi_A(nT) - [\phi_B^+((n-1)T + d_{n-1} + d'_{n-1}) + \\
&\quad \gamma_B(T - d_{n-1} + d'_{n-1} + d_n + d'_n)] \\
&= \phi_A(nT) - \phi_A((n-1)T) \\
&\quad -\gamma_B(T - d_{n-1} + d'_{n-1} + d_n + d'_n) \\
&= \gamma_A T - \gamma_B(T - d_{n-1} - d'_{n-1} + d_n + d'_n) \\
&= \gamma_{A/B} T - \gamma_B(T - d_{n-1} - d'_{n-1} + d_n + d'_n)
\end{aligned}$$

$$\begin{aligned}
\sum_{i=1}^{n} \delta_i &= \delta_1 + \delta_2 + \cdots + \delta_n \\
&= [\gamma_{A/B} - \gamma_B(T - d_0 - d'_0 + d_1 + d'_1)] \\
&\quad +[\gamma_{A/B} - \gamma_B(T - d_1 - d'_1 + d_2 + d'_2)] \\
&\quad + \cdots + [\gamma_{A/B} - \gamma_B(T - d_{n-1} - d'_{n-1} \\
&\quad +d_n + d'_n)] \\
&= nT\gamma_{A/B} - \frac{\gamma_B}{n}(d_n + d'_n - d_0 - d'_0)
\end{aligned}$$

若令 $\hat{\delta}_n$ 爲 δ_i 的算術平均值，則 $\hat{\delta}_n = \frac{\sum \delta_i}{n} =$

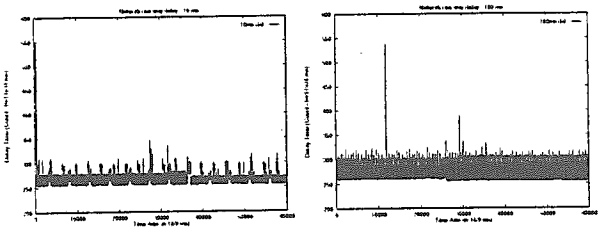

圖 5: 單段乙太網路上封包單向傳輸延遲圖[10ms及100ms]

$T\gamma_{A/B} - \frac{\gamma_B}{n}(d_n + d'_n - d_0 - d'_0)$；當 n 足夠大時，後項可以忽略而得到，$\hat{\delta}_n \approx T\gamma_{A/B}$。利用此項平均值，令 $\theta_n = \sum_{i=1}^{n}(\delta_n - T\gamma_{A/B}) = \gamma_B(d_0 + d'_0 - d_n - d'_n)$。至於 θ_n 所代表的意義稍後會討論到。

而校正誤差 τ 的值則可以下式表示：
$\tau = max_n\{|\phi_B^+(nT + d'_n + d_n) - \phi_A(nT + d'_n + d_n)|, |\phi_A(nT + d'_n + d_n) - \phi_B^-(nT + d'_n + d_n)|\}$，因爲 T 通常比 d_n 與 d'_n 大很多，所以上式可以簡化成

$$\tau = max_n\{|\hat{d} - d_n|, |T\gamma_{A/B} - \hat{d} + \gamma_B d_{n-1}|\} \quad (2)$$

在基準主機校正一般主機的過程中包括了：處理時間(processing time) 、傳輸時間(transmitting time)、傳導時間(propagation time)。處理時間是指封包在傳送前系統處理的時間，包括系統在記憶體中的搬移、程序切換、及在佇列中等待被傳送的時間等。傳輸時間則是將一個封包完整的從卡上傳到網路實體層所需的時間。傳導時間則是封包在網路上傳輸到目地端的時間，等於封包長度除以網路實體層傳輸速度；舉例而言，一個長度爲1K byte 的封包，在 10M bits/sec 的乙太網路上，傳導時間爲(1 * 8K)/10M = 0.8 毫秒。其中傳輸時間及傳導時間，在一般情況下均爲一常數，所以不列入考慮。其中會變化者只有處理時間，而這些時間會因系統負載高低、網路流量大小而變化；因爲處理時間基本上可以視爲系統處理時間加上等待時間。

圖 5 是在一段沒有其他資料傳輸的乙太網路上，分別以十及一百毫秒爲間隔，周期性地傳輸一個最小長度封包（64 byte），將所得到的來回時間除以二，所得到的單向傳輸延遲時間圖（One-way Delay Time）。因爲在安靜的網路上沒有任何其他的封包在傳輸，單向傳輸延遲可以視爲來回時間的一半；況且此實驗是在作業系統核心中計算而得，誤差應已降至最低。由圖中可知系統處理時間並不是常數，但是變化並不大。上限 350*10000/11931 = 293 us，下限 250 * 10000/11931 = 210 us，相差 83 微秒。因爲資料是在網路卡驅動程式中處理計算，造成此現象的原因，應該是系統在多工環境下處理工作切換及中斷所致。

就如前面提到的，周期性校正的前提之一是單向傳輸延遲是一常數（或說非常趨近常數）。然而，由上述實驗結果可以知道，作業系統所造成的延遲有可能破壞這個假設。雖然只有少數資料過大（例如圖 5 中，六萬筆資料中才只有兩三筆的值過大），但仍須將此納入考慮。換句話說，此協定必須有能力判別收到之校正封包中的資訊是否有因作業系統等所造成的誤差而不能使用。這就是前述中計算 θ_n 的用意：用來判別所收到的校正封包內資訊是否已經無法使用。而事實上，除了單向傳輸延遲的變化會影響校正封包資訊的正確外，靜肅期的長短（與網路上主機的數目、封包長度有關）也會有相當的影響，這在稍後會討論到。

3 實作與實驗

3.1 實作

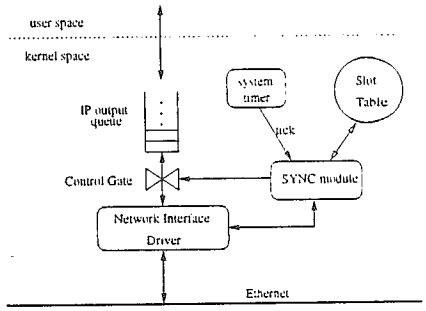

圖 6: 傳輸控制機制圖

由於對 Unix 相容系統熟悉度的關係，加上作業系統核心原始碼可否免費取得的考量，在選擇原型實作平台時，決定使用 Mach 2.6版。Mach 是由美國卡內基美濃大學(CMU)所發展出的一套微核心（micro kernel）作業系統。 Mach 2.6 作業系統核心內有一變數 tick ，而計時器（timer）中有一暫存器（count），從 11931 向下計數，計數到零時會發出中斷訊號，使 tick 值增加一。計時器受一外部振盪器驅動，該振盪器以 1.1931 MHz 使 count 值向下計數，所以 tick 值每 10ms 增加一；所以系統時間的精確度是 10 ms。由於這樣的時間精確度不夠，所以將 count 值改爲 1326 ；如此一來中斷便會每 10/9 ms 發出一次。但爲了不影響系統原來的操作模式，所以新增了一個系統變數 Rtick，每次中斷只會增加 Rtick 的值，而增加九次 Rtick 值後才增加一次 tick 值。 Rtick 的值也就是時間片段（T_{slot}）的長度，由前面提過校正周期設爲 0.8 秒，所以以得到一個校正周期長度共有 800(ms)*9/10(ms) = 720 個時間片段。

爲了能有效地依時間控制封包的傳輸，在作業系統核心中加上控制機制，這是最有效的方式，缺點則是不易移植到其他平台。時間校正基制的基本原理是在傳輸封包所使用的緩衝區後加上一個控制閘，只有在允許的時間才將緩衝區內的封包傳送到網路上。而當收到校正封包時，則將所需資訊取出用以校正本身的時間資訊。圖 6 中清楚表達了此種機制，修改網路驅動程式原始碼，使其能依據時間表開關輸出通道，並在收到校正封包取出所需資訊校正本身時間。系統計時器則會定時發出中斷，以驅使傳輸狀態（靜肅、校正、共用期）的改變。

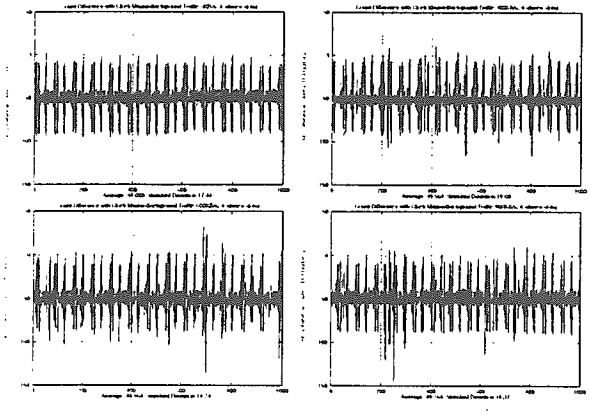

圖 7: 不同網路流量下的校正差值[0,300,600,900KB/s]

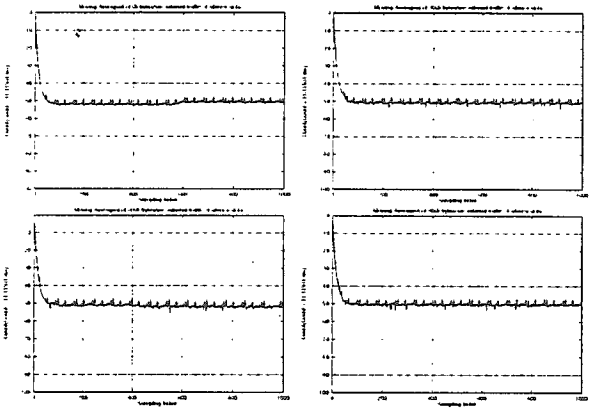

圖 8: 不同網路流量下校正差值的移動平均值[0K,300K,600K,900KB/s]

3.2 實驗結果

實驗是以八部個人電腦，一部做爲基準主機（clock master）周期性的送出校正封包，其他所有的主機做爲一般主機，在收到封包後會校正自己的時間。其中一部一般主機在每次收到校正封包時，會將本身當時的時間值減去收到封包中的時間值並紀錄下來，而剩下的一般主機則會送出封包作爲控制網路流量大小之用。而實驗中的控制變數包括：網路流量大小、靜肅期的長短；觀察的變數則爲時間差距（漂移）。在網路流量控制方面，是由主機以 UDP 封包（traffic）送入網路，控制傳送封包的間隔時間（interval）以達到控制網路上總流量的目的。靜肅期的長短則是以控制時段表（Slot Table）的方式處理。

圖 7 爲不同網路流量下的校正差值；即每次校正所得 δ_i 值減去平均延遲 $\hat{d}$ 的值，分別在 0K 、300K 、600K 及900K bytes/sec 的網路流量下所得的圖；四個實驗的靜肅期均設爲四個時間片段，也就是大約 4.4 ms。從圖中可以發現以校正周期爲 800ms 的情形下，兩主機時間漂移約爲 50*10000/11931 = 42 us 左右。爲了驗證前面所做的分析，圖 8 呈現校正差值的移動平均值，此移動平均值的計算公式是

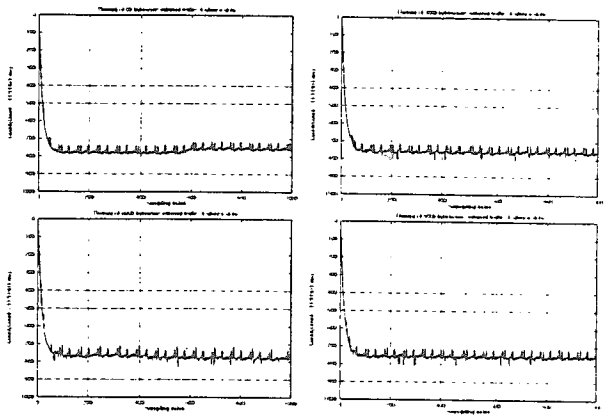

圖 9: 不同網路流量下 θ_n 移動平均值[0K,300K,600K,900KB/s]

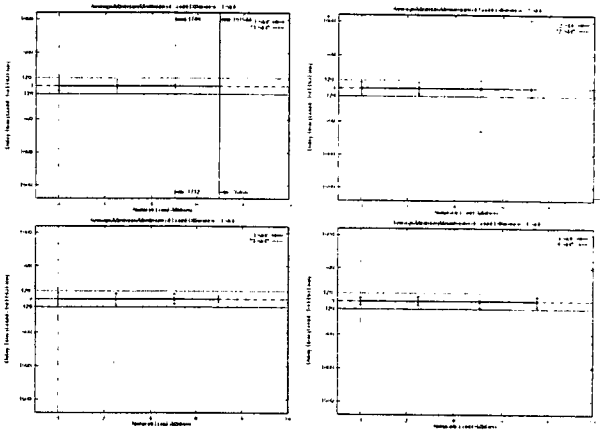

圖 10: 校正差值（最大值、最小值及平均值）與靜肅期之關係[1,2,3,4 slot]

從 [4] 中取用的。就如前面分析所得，校正差值會趨近一常數。而圖 9 則爲 θ_n 在不同網路流量下計算所得的值。實驗中將 $\hat{d}$ 設爲 151us，由實驗中可知網路傳輸所產生的誤差爲 50us。而因校正差值所產生的誤差平均爲 42us，而所產生的校正精確度誤差平均爲 92us。最差情況下，校正後的誤差則爲 150 + 100 = 250us。也就是說，即使在最差情況下，所得的結果仍符合所要求的一毫秒以下的誤差。

由於在實驗過程中經由網路分析儀的輔助下，發現在校正期時仍有封包傳輸而與校正封包碰撞，而當時的靜肅期由兩個時間片段組成，爲避免此種現象再發生，所以本文中實驗的靜肅期均爲四個時間片段。爲了證實我們的猜測，又另作了一組實驗確定靜肅期的長短與網路流量之間是否有關係存在。圖 10 中分別爲靜肅期長度爲一、二、三、四個時間片段，並在不同網路流量下，所得到校正差值的最大、最小及平均值。從其中可以發現當靜肅期小而網路流量大時，校正差值之特性會有大幅度的變化，而這會影響校正的精確度。

就追蹤系統原始碼的結果，這現象的產生應來自於無法命令網路卡立即停止傳送封包。圖 6 中的控制閘只能在靜肅期及校正期不讓封包送達網路卡，

但是在共用期快結束前、靜肅期將到達之時送到網路卡上的封包，便有可能在靜肅期被送出，甚至可能因爲靜肅期太短、發生碰撞等因素，而在校正期中被送出，也因此與校正封包發生碰撞。實際上，命令網路卡立即停止傳輸封包也有困難之處：如果網路卡已將一個封包傳送了一半，該如何處理？而且就初步的資料收集，目前一般的網路卡，無法令其立即停止網路卡傳送。

4 結論

4.1 實作方式與成果檢討

不過如前面所述，靜肅期的長短與網路上的流量有關，這會影響此協定的穩定度。一個根本的解決方案是使網路卡可以立刻停止，如此這個現象就可以完全解決，更符合設計時的假設。但是目前一般網路卡驅動程式並不支援此種功能，因此有待克服。目前所做的是一個簡單而直覺的方法，是將靜肅期設爲最差狀況下的長度；也就是指主機數目、封包長度、網路流量等均最差情況下所需的靜肅期大小，但是卻會浪費頻寬。

此協定的基本觀念及設計相當簡單，雖純以軟體實作但對系統不致造成效能降低，是其優點之一。實驗結果顯示，此協定可以提供精確度小於一毫秒的時間校正。靜肅期與校正期的使用雖會佔用部份網路傳輸頻寬，但由於所佔比例甚小，也不致造成傳輸效能降低。舉例而言，以靜肅期四個時間片段、校正期一個時間片段計算，也不過佔用大約百分之零點七的頻寬；以 10Mbps 的乙太網路而言，僅大約 69.4Kbps 的頻寬被佔用而無法傳輸資料。

4.2 未來發展

首要工作是解決在路由器上多個網路段上同步控制傳輸的問題。如何將路由器（Router）、橋接器（Bridge）等裝置的影響納入考慮，以發展一個可供更大型網路使用的時間校正協定，是值得繼續探究的主題。

遠程的目標則是將此時間校正擴充作爲分時多重存取乙太網路 [6]的基礎。簡單的說，是將共用期的時間片段改爲自由時段，然後依各主機所要求傳輸資料量的多寡（所需頻寬），分配時間片段給各主機以滿足其需求。

參考文獻

[1] Dallas Semiconductor Corporation. "Application Note 58 - Crystal Considerations with Dallas Real Time Clocks". *Dallas Semiconductor data book.*, May 1996.

[2] Frank Van Gilluwe. *The Undocumented PC.* Addison-Wesley, Sep. 1994.

[3] R. Gusella and S. Zatti. "TEMPO– A network time controller for a distributed Berkeley UNIX system.". *Proc. Summer 1984 USENIX*, June 1984.

[4] R. Frederick H. Schulzrinne, S. Casner and V. Jacobson. "RTP: A Transport Protocol for Real-Time Applications". *RFC-1889*, Jan. 1996.

[5] Heinz-Gerd Hegering and et al. *Ethenet : Building a Communications Infrastructure.* Addison-Wesley, 1993.

[6] Chia-Hsiang Chang Kuo-Hui Tsai and et al. "Synchronous Ethernet - a network supporting real-time communication.". *Proc. 1st Workshop on Real-time and Media Systems*, July 1995.

[7] David L. Mills. "Network Time Protocol version 1 specification and implementation". *RFC-1059*, July 1988.

[8] David L. Mills. "Internet Time Synchronization : The Network Time Protocol". *IEEE Trans. on Commun.*, 39(10):1482–1493, October 1991.

[9] David L. Mills. "Network Time Protocol (Version 3) Specification, Implementation and Analysis". *RFC-1305*, March 1992.

[10] David L. Mills. "Simple Network Time Protocol (SNTP)". *RFC-1361*, August 1992.

[11] David L. Mills. "Improved Algorithms for Synchronizing Computer Network Clock.". *IEEE/ACM Trans. on Networks*, pages 245–254, June 1995.

[12] J. Postel. "Internet Control Message Protocol". *RFC-792*, Sept. 1981.

[13] J. Postel. "Daytime Protocol". *RFC-867*, May 1983.

[14] J. Postel and K. Harrenstien. "Time Protocol". *RFC-868*, May 1983.

[15] W. Richard Stevens. *UNIX Network Programming.* Prentice Hall, 1991.