台灣學術網路上因果次序之群組通訊協定實作與測試

溫照成
國立中正大學電機工程研究所
Email: d8742005@ccu.edu.tw

郭文亮
國立中正大學通訊工程研究所
Email: m8914@cn.ee.ccu.edu.tw

摘要

近來由於網路技術與電腦科技的發展，電腦系統的環境逐漸地由單機單工的系統架構朝向多機多工的分散式系統架構發展；藉由分散式系統的架構，能夠整合多方資源，甚至改變人們的生活習性。分散式的系統雖帶來了許多的好處，但仍伴隨著一些技術上的問題，其中以同步各分散點的工作為最主要的課題。端點同步問題的產生，主要是因為各端點間的訊息傳送延遲不同，而造成網路非同步現象。為了克服上述因網路非同步的現象，許多相關的多端點（群體）通訊同步協定被提出[3][4][5][6][14]；其中，最著名的是*因果次序協定*（*causal order protocol* [2][7]）。之後所提Δ—causal order protocol [10]同時考慮因果次序及即時性，符合新一代網際網路應用服務的需求，但之前的研究並未分析即時性與因果次序效能上的平衡（trade-off），本文實作Δ—causal order protocol，並在台灣學術網路上測試。實驗中我們發現Δ值的取捨必須在分散式即時應用對端點的最低處理訊息的通透率（Throughput）要求下力求盡可能的小。另一方面，我們也指出當Δ大時，適當的可靠傳輸機制將可以減輕某些端點的同步負擔。我們的成果可供設計群組軟體效能上的參考。

Abstract

The revolutions of networking and computing technology motivate the deployment of widely distributed system that integrates resources and make people convenient. People take advantage of distributed system but there still have many technical issues to be addressed such as synchronization or coordination of the distributed processes. The distributed processes suffer from asynchronous phenomenon because of the delay variance of messages through the heterogeneous networks. This situation becomes worse when the end-to-end delays of links in the distributed system are asynchronous due to different bandwidth, different routing or switching technology, physical location of stations, etc.. To overcome the above problem, many coordination protocols for group-ware applications are proposed. In these protocols, delta-causal order protocol considers both requirements of causal order, guarantees cause and effect between messages, and real-time, guarantees messages are significant for the receiver. Nevertheless, previous research does not study the performance trade-off between real-time and causal order. In this paper, we implement the protocol and analyze its performance on Internet. From our experiments, we find that the value must be as possible as small, subject to the lowest throughput requirement of the distributed real-time system. If the value is large, we find that retransmission may help some sites to decrease the overhead of synchronization. We believe our research will be beneficial to the improvement of the group communication in the future and can provide some helpful reference for the designer.

Keywords: Causal Order, Synchronization, Group communication, network delay, TANET.

1. 背景

近來由於網路技術與電腦科技的發展，電腦系統的環境逐漸地由單機單工的系統架構朝向多機多工的分散式系統架構發展；分散式系統架構能夠整合多方資源，甚至改變人們的生活習性，其相關的應用，如視訊會議、遠距教學、線上遊戲、線上交易等等也正日漸盛行；然而，分散式系統仍存在著一些技術上的問題，其中以同步各分散端點的工作為最主要的課題。本文將針對分散式系統之各端點的非同步問題進行探討，更進一步，特別針對即時（Real-time）應用在分散式系統架構下之訊息時效性的問題，將提出相關實驗測試及效能分析。

一般所謂的分散式系統，是由許多獨立的端點（或是稱為元件）透過通訊網路連接所組成的；因此，各端點間如何相互溝通來達到彼此合作運行於整個系統，將成為系統是否完善運作的大課題，這牽涉到各端點如何依特定的訊息次序來維持訊息間（Message）的關連影響（Cause and effect），此相關的課題便是廣為人知的端點非同步問題。

端點非同步問題的產生，主要是因為各端點間之訊息的通訊網路傳送延遲（Network transmit delay）不同，以及傳送延遲之變異性（Network transmit delay variance）無法預測等等因素，造成網路非同步現象（Network asynchronous

phenomenon），此現象將導致各端點在接收（Receive）訊息的次序上發生無法一致的現象，舉例來說，圖 1 中 P_i、P_j、P_k 為三個不同行程（Process），假設其行為模式為 P_i 送尋問訊息 $m(q)$ 給 P_k 和 P_j，而 P_j 收到 $m(q)$ 之後，將送出回答訊息 $m(s)$ 給 P_i 與 P_k；對 P_k 而言，應先收到 $m(q)$ 再收到 $m(s)$，然而，假設 $m(q)$ 因通訊網路發生擁塞而導致遺失或傳輸延遲很長的時間，導致 P_k 可能收到 $m(s)$ 時仍未收到 $m(q)$，因此，P_k 如果採先收到先處理（FIFO）的訊息處理模式而先處理 $m(s)$ 時，P_k 將會發生困惑，並導致與其它端點處理訊息步調不一的情形發生，如圖 1 左邊所示。圖 1 右邊則顯示出因 $m(s)$ 受到網路擁塞而產生 P_k 等待 $m(s)$ 很長時間的問題，在此情形下，P_k 可能再收到 P_i 送出第二筆資料 $m(q')$，如此 $m(s)$ 與 $m(q)$ 的相依性便被破壞以致系統協調失誤，倘若 P_k 在 $m(s)$ 到達前發生愈時（Time out）而未將之後遲到的 $m(s)$ 丟棄，也會造成系統功能錯誤。因此，如何解決因網路非同步現象所產生的上述問題，便成為在分散系統應用中亟需解決的課題。

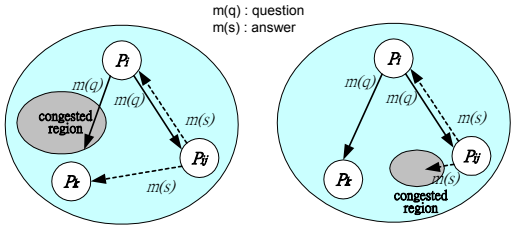

圖 1 三角不等延遲現象

1978 年 L.Lamport 提出了"*happened before*"定義（→）來定義訊息的次序，其定義出與訊息相關之事件（Event）的前後關係，相對地與事件相關之訊息的前後關係也跟著確立了，因此各分散端點在處理訊息的次序上便有了依據，而此次序稱之為*因果次序*（*Causal Order*）；一旦各分散端點皆能以因果次序來處理接收到的訊息時，各分散端點便消除了網路非同步現象的影響並達到同步，許多群體同步通訊協定便此概念的而相繼被提出。

在這些群體同步通訊協定中，最著名的是*因果次序協定*（*Causal ordering protocol* [12]），此協定要求各分散端點在處理（Deliver）訊息的次序上不能夠破壞"happened before"的關係，更清楚的意思為「假使有兩筆訊息在傳送的事件（Send event）上有因果關係且有共同的接收端點時，則此接收端點在處理這兩筆訊息的次序上必須遵照這兩筆訊息在傳送的事件上因果關係，即傳送事件先的訊息被先處理」，因此，訊息能否被接收端點*處理的條件*（*Deliver condition*）在於「先前與該筆訊息有因果關係之全部訊息皆被接收端點處理完」，該訊息才能被接收端點處理，因此，一旦有訊息遺失，將使得其後與此訊息因果相關的訊息為了等待這筆訊息而在接收端點被暫存，此將產生被暫存訊息逐漸累積的現象，最終產生暫存區滿溢且端點無法運作的問題，也因此因果次序協定需運作在訊息不可遺失的通訊網路傳輸環境。

隨著分散式即時性應用對端點同步與訊息時效的雙重要求 [8][9][11]，因此有人提出了改良自因果次序協定的*Δ—因果次序協定*（*Δ—causal ordering protocol* [10]）；Δ—因果次序協定與因果次序協定最大的不同在於「Δ—因果次序協定給與每筆訊息有著相同Δ時間的訊息*生存期間*（*Life time*）」，生存期間就是即時性應用對訊息對時效的要求，應用程式在此前提下進行端點同步的處理；換言之，為了滿足即時應用對時效上的要求，Δ—因果次序協定對失去時效性的訊息與予丟棄，含示可將遺失的訊息視為傳輸延遲久並失去時效性，端點對訊息是否全部依因果次序被處理的要求不再嚴苛，所以Δ—因果次序協定可運作在底層的通訊網路不可靠的環境下。各分散端點僅對所有未失去時效性的訊息進行因果次序處理，並以此達到某種程度上的同步運作，滿足對分散系統對端點同步的要求。

在Δ—因果次序協定中，Δ的值是值得探討的重點，Δ值的物理意義是的訊息在時效上要求，設想當即時應用要求所有的運作都能很迅速完成時，Δ的值的必定設小，訊息如未能在Δ生存期間之內到達接收端點時，便會在接收端點被*丟棄*（*Discard*），假使端點處理的訊息量多寡能反應出端點在同步應用上的效能，那當訊息被丟棄的數量過多時，該同步應用的效能當然也就降低，可見Δ值大小對整個群體通訊同步協定的運作會產生何種的影響，這也是本論文專注的重點。

本論文之後的組織如下：第二章介紹相關研究，第三章介紹實作系統架構、實驗環境與測試結果，第四章是結論。

2. 相關研究

為了描述分散式系統所發生的事件關係，L.Lamport 提出"happened –before relationship" 來定義訊息的次序關係。他首先定義三種與訊息相關的事件（Event）：*傳送事件*（*Send*）、*接收事件*（*Arrival*）、*處理事件*（*Deliver*）。當一個端點送出一個訊息 m 到通訊網路時，此事件為傳送事件（Send(m)）；當一個訊息 m 由通訊網路到達接收端

點時，定義為接收事件（Arrival(m)）；而接收端點將此訊息 m 往上層遞送給應用層時稱作是處理事件（Deliver(m)），(請參考圖 2)。

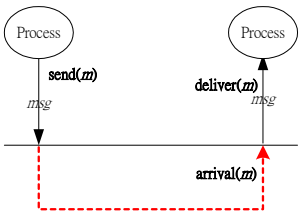

圖 2 三種事件圖式

在群組中若是兩事件 e1、e2 具有"happened before"關係的話（以"→"來表示）必滿足下列之一的條件：

1. e_1 和 e_2 兩事件發生於同一行程，並且 e_1 較 e_2 發生得早。
2. 對於訊息 m 而言，傳送的事件記為 e_1，到達接收端的事件記為 e_2，則 $send(m)$ 一定發生在 $arrival(m)$ 之前。
3. 假設存在一事件 e_* 使得 $e_1 \rightarrow e_*$ 而且 $e_* \rightarrow e_2$，則 $e_1 \rightarrow e_2$.

若是兩事件並未符合上述條件，則此兩事件同時進行（*Concurrent*），因果次序協定（*Causal ordering protocol*）則是完全保證所有群組內事件都滿足 happened-before 關係以提供應用程式因果次序服務。換句話說：

若在一分散式系統內兩訊息 m_1 和 m_2 滿足 $send(m_1) \rightarrow send(m_2)$ 而且目的地一樣（m_1 和 m_2 可能來自不同行程，）則目的端對於兩訊息的處理順序應該滿足 $deliver(m_1) \rightarrow deliver(m_2)$。

為了滿足上述關係，當端點收到訊息時，先判別該筆訊息是否符合端點本身因果次序的要求，如果符合，端點立即處理該訊息，如果不符合，端點便將此訊息先存放到*因果次序錯誤暫存區*（*Out-of-causal-order queue*）內，直到先前與該訊息因果相關之訊息皆到來依因果次序處理後，再處理暫存在因果次序錯誤暫存區內的該筆訊息。Birman 及 Joseph 提出了實現因果次序協定的簡單演算法—*RST 演算法* [2][12]，在此機制中，各端點皆擁有一個二維矩陣記載著任兩行程間已經傳送多少訊息的因果次序資訊，並藉由訊息攜帶（Piggyback）傳送端點之因果次序的資訊，當接收端點收到訊息時，便比對接收端點自身因果次序資訊與接收訊息所攜帶之因果次序資訊是否符合*處理條件*（*Deliver condition*），以決定該筆訊息是否可立即被處理，如果可以，立即處理該筆訊息並且更新端點自身之因果次序資訊，維持各端點因果次序資訊上的一致性。如果不可以，則先暫存此訊息於因果次序錯誤暫存區，直到該筆訊息輪為正確處理次序時再被處理。

進一步考量一般即時應用中，允許訊息少量的遺失以避免過多的訊息因等待遺失訊息而失去時效性，故針對分散式的即時應用，Δ-因果次序協定被提出來。Δ-因果次序協定與因果次序協定最大的不同在於「Δ-因果次序協定給與訊息多加上一個Δ*生存期間*（*Life-time*）」，以Δ生存期間代表即時性應用對時效上的要求，並且各端點在訊息Δ生存期間內維持訊息間因果次序的關係，換言之，一旦訊息自傳送到接收的時間超過Δ生存期間，該訊息將被視為失去時效性，可視作它已遺失，而此訊息的遺失將不會影響各端點同步的進行；在生存期間到達接收端的訊息，則根據根據原始的因果次序協定處理方式，等待該訊息前的所有其他訊息直到Δ時間，下圖 3 是完整的演算法。

我們可以很輕易地察覺，在Δ-因果次序協定中，Δ的大小決定了即時要求的程度，當Δ很大時，Δ-因果次序協定與因果次序協定相似，但當Δ取很小時，比如小於網路平均延遲，訊息會因嚴苛的即時要求而造成接收端大量丟棄(discard)訊息。已過研究多屬於協定的驗證，並沒有實作上的輔證，本論文以下將就運用在分散式即時應用最根本的Δ—因果次序協定在台灣學術網路上測試效能及分析其行為。

3. 實作與實驗

我們以 Linux 及 FreeBSD 作業系統平台，於 UDP 協定上建構Δ-因果次序協定，為了呈現及評估協定的運作情形，我們定義下述三向量化值：

(1) Ratio of out-out-causal-ordering messages at process i (R_i^{ooco})：一般認定該訊息是否因果次序錯誤的依據在於該筆訊息是否可以被端點接收後馬上送到應用層，即是否符合接收端之處理條件（Deliver condition），如果不可以，代表該筆訊息必須在接收端點等待先前較早傳送並與之因果次序相關的訊息，因此接收端會將該筆訊息放入因果次序錯誤暫存區中，直到符合上傳至應用層之處理條件（Deliver condition）成立後；因此本論文以放入因果次序錯誤暫存區的訊息個數表示因果次序錯誤的嚴重程度；此外，我們予之與全部收到訊息進行比率得此測量數值。在本論文定義下，接收訊

Global variable $current\_time$;

/* Process i sends the message m to process j */
Procedure $SEND_i(m, j)$
Begin
$m.send\_time := current\_time;$
$m.sent := SENT_i;$
$send(m);$
$SENT_i[i, j] := m.send\_time;$
End

/* Message m arrives process j from process i */
Procedure $RECEIVE(m, i, j)$
Begin
If ($m.send\_time + \Delta > current\_time$)
discard m and return;
$wait(DC(m));$
$deliver(m);$
$DELIVER_j[i] := m.send\_time;$
$SENT_j[j, i] := m.send\_time;$
$\forall x, y \quad SENT_i[x, y] :=$
$\max(SENT_i[x, y], m.SENT[x, y]);$
End

/* Delivery condition at process i */
$DC(m)$:
True: $\forall x$
($DELIVER_i[x] \geqq m.SENT[x, i]$ or
$m.SENT[x, i] < current\_time + \Delta$)
False:
Put to out of causal order queue and wait for $DC(m)$ become true.

圖 3 Delta-causal ordering protocol

息因果次序錯誤率越高，代表發生因果次序錯誤的情形越頻繁。

(2) *Queuing delay of out-of-causal-order messages at process i (QD_i^{ooco})*：端點接收到因果次序錯誤之訊息時，會將該訊息放入因果次序錯誤暫存區中，因此訊息在因果次序錯誤暫存區中的延遲將導致端點在處理訊息的效率上降低，並間接拖累整個端點的運作效能；我們記錄因果次序錯誤暫存區之總延遲來表示端點在處理效率上的影響，並予之正規化（Normalize）得此數據。根據這定義，因果次序錯誤暫存區之總延遲越久，代表端點在處理訊息的效率上越差，也代表著端點必需在執行端點同步的運作上增加更多處理負載（Overhead）。

(3) *Ratio of out-of-date messages at process i (R_i^d)*：Δ-因果次序協定中，Δ參數代表訊息之時效性，故訊息必須在此時效時間內被處理（Deliver），否則端點將視該訊息失去時效性而丟棄（Discard），因此Δ參數的選取將會影響端點接收訊息的通透率（Throughput）並間接影響該應用的品質，特別是對端點同步上的影響，但為了增加訊息被處理的總量而放大Δ參數，則可能導致訊息不符合該應用對時效上的要求；訊息過時丟棄率越高，代表端點接收訊息的通透率（Throughput）越低，意謂端點要維持訊息間因果同步的要求將越難達成。在此定義訊息過時丟棄率為訊息被端點因過時而丟棄的量除以端點收到的訊息總量。

本文的實際網路環境測試平台如圖 4 所示，分別在台大計中、中正計中及中正電機實驗室各架設一台 PC 為測試端點，並假設三個端點之通訊行為能代表整個分散式系統之群體通訊模式。本論文分別將以 CCUEE 代表中正電機實驗室，CCUCC 代表中正計中，而 NTUEE 代表台大計中。

實際網路之傳輸路徑為 CCUCC 到 NTUCC 雙向透過 TANET，而 NTUCC 到 CCUEE 則為 TANET，但 CCUEE 到 NTUCC 則為 TANET2，CCUCC 到 CCUEE 雙向則為中正校園網路，基本上整個網路環境涵蓋距離為兩百三十公里的廣域網路及中正內部校園網路。台大計中與中正計中分別架設 GPS Receiver 校正系統時間，以及中正電機實驗室定期的以 NTP [13]方式對中正計中校時來達到時間上的同步。

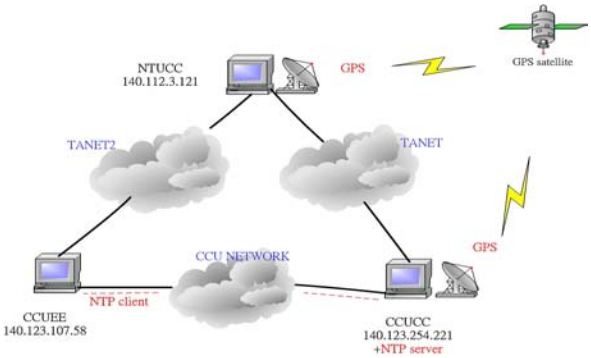

圖 4 實驗平台

為了瞭解網路延遲對實驗的影響，我們在台灣學術網路使用量大的時段，同時收集三點間的單向延遲統計數據並整理在表格 I。我們就兩種不同傳送訊息之帕送分怖平均速率，分別為 lambda= 200

與 lambda = 400（單位皆為 messages / sec），在不同的訊息時效要求下（Δ=3000 us、4000 us、5000 us、6000 us、7000 us、8000 us、9000 us）進行測量，並觀察各端點在觀察項目中之行為。在下列各圖示中，以三角型符號代表 CCUEE，菱形符號代表 CCUCC，而 NTUCC 則以方形符號表示各點所觀察的結果，另外，實線與空點為 lambda = 200 之實驗數據，虛線與實點則為 lambda = 400 之數據。

表格 I 單向延遲統計數據

連結 \ 統計	Mean (ms)	Variance	Loss ratio (%)
CCUEE->CCUCC	1.514	0.078	1.9
CCUEE->NTUCC	6.14	0.481	1.6
CCUCC->CCUEE	1.50	0.012	2.4
CCUCC->NTUCC	3.15	0.060	3.9
NTUCC->CCUEE	3.04	0.249	3.2
NTUCC->CCUCC	3.24	0.215	4.1

圖 5 中，橫軸表示不同Δ，縱軸則表示在所有接收訊息當中，被放入因果次序錯誤暫存區之訊息比率。由此圖可知當Δ值變大，因果次序錯誤率也跟著變大。此外，本圖也可發現，當 lambda 越大時，發生次序錯誤的情況也越嚴重。不同的送出速率也會影響發生因果次序錯亂的情形，越高的送出率使得因果次序混亂得越嚴重。在圖 6 中，橫軸表示不同Δ，縱軸則表示所有因果次序錯誤訊息在暫存區中的總延遲。圖 5 與圖 6 有相對應之關係，當放入因果次序錯誤暫存區的訊息次數多時，因果次序錯誤暫存區的總延遲相對的也增多，此外，我們也發現當Δ越大，三端點之因果次序錯誤暫存區的總延遲皆跟著變大，而相近的兩端點增大的幅度大於網路傳送延遲較久的端點，且逐漸擴大的區勢，換言之，若為了遷就網路傳送延遲較久的端點時，Δ必要設大些，但此導致其它端點在因果次序錯誤暫存區的總延遲增加，而增加處理訊息同步上的負擔，整個分散系統的應用效能也可能跟著降低，圖 7 中，橫軸表示不同Δ，縱軸則表示訊息因傳送延遲超出Δ而被丟棄率。訊息被接收端丟棄的關鍵仍在於傳輸網路的延遲，從先前的各端點傳輸延遲表可知，救 NTUCC 來看 CCUEE 到 NTUCC 的傳輸延遲很接近 6ms，所以在Δ = 6 ms 時，NTUCC 約 25%的訊息會超出Δ而被丟棄，當Δ = 5 ms 時，CCUEE 傳給 NTUCC 的大部份訊息都超出Δ而被丟棄，故總共約會有 50%的訊息被丟棄。

4. 結論

在互動式分散系統架構中，因為網路非同步因素的影響，導致群體之間傳遞的訊息在接收時會發生次序錯亂的情形，而此次序錯亂的情形，將可能會導致整個系統發生非預期性的執行結果，尤其對於一些控制與管理的訊息，這些訊息執行次序必須正確無誤，而因果次序協定解決了這類的問題；然而，針對於即時性的分散系統應用，因果次序協定並不適用，其原因在於此應用不僅要求訊息次序之正確性，也要求訊息的時效性，而Δ-因果次序協定則解決了因果次序協定在這一方面的不足。

本文透過實際撰寫Δ-因果次序協定之程式，並在實際的網際網路上實驗，了解在Δ-因果次序協定架構下多端點之間互動的影響，有助於提供群組「服務保證機制」設計時的參考，我們由實驗數據得知幾項結論：

1. 因各端點傳送訊息越頻繁，發生因果次序錯亂的情況越嚴重。換句話說，互動頻繁的群組應用必須更嚴格的維護群組事件間的因果次序，否則非常容易導致服務中斷的情形。

2. Δ參數代表訊息之時效性，而訊息因超出Δ而被接收端丟棄的多寡程度主要決定於傳輸網路之延遲狀況。當Δ參數越小時，訊息因超出Δ而被接收端丟棄機率增加，對端點而言，端點能否在僅收到的訊息中進行同步運作將是一大問題。當Δ參數越大，端點把接收訊息暫存的機會就越大，此將造成端點在處理訊息同步上承受更多的負擔，換言之端點在處理訊息的效率上將降低，特別是在異質性網路環境（WAN 與 LAN 共存的環境下）且訊息會因網路傳輸而遺失的狀況下，在 LAN 中端點在處理訊息同步上承受的負擔更重。

3. 在上述第（2.）中，我們能藉由在 LAN 中建立可靠的傳輸機制來降低訊息遺失率所造成的影響，特別在上述異質性網路環境而且 LAN 中的各端點互動很頻繁的環境下，有可靠傳輸的機制可降低因訊息遺失所造成端點處理因果次序同步運作上的問題，使得 LAN 內部端點的同步機制將運作更完善，因為端點在處理訊息量上增加，而使得端點能維持更完整的訊息因果次序。

此外從測試結果與分析中，本論文也針對Δ值的取捨上有詳細的探討並提出取捨的原則，其原則如下(1)先依該應用對訊息的時效要求找出Δ值的最大上限；(2)再依網路傳輸延遲狀況最差之端點的傳輸延遲分佈與應用對端點的最低處理訊息的通透率（Throughput）要求決定Δ值的較小下限，並以此下限為Δ的值；(3)如果Δ值的上限無法滿足應用對端點的最低處理訊息的通透率要求時，則此即時應用無法運作在此端點上。希望此原則能提供日後在設計分散式即時性應用上Δ值參考的依據。

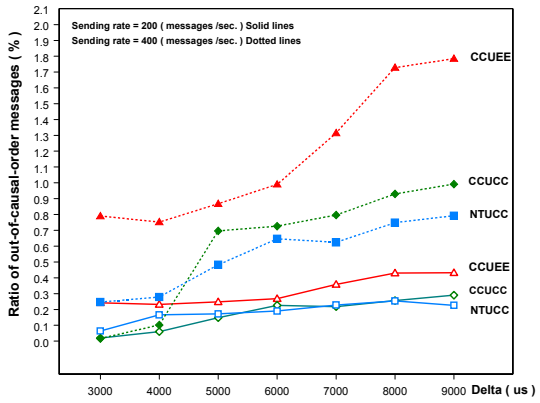

圖 5 Ratio of out-of-causal-order messages

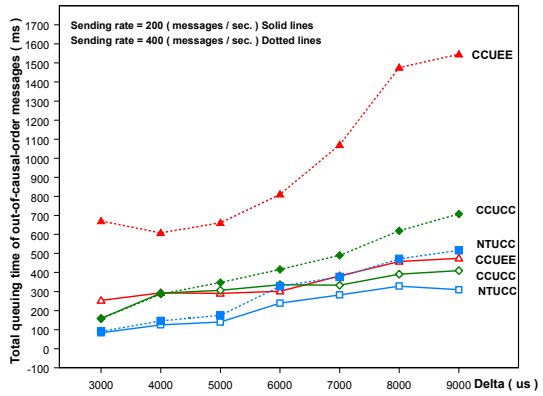

圖 6 Queuing time of our-of-causal-order messages

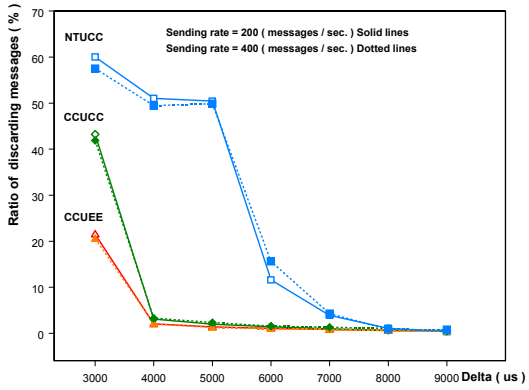

圖 7 Ratio of discarding messages

參考文獻

[1] L. Lamport, "*Time, Clocks and the Ordering of Events in a Distributed System,*" Communications of the ACM, 21(7):558-565, 1978.

[2] K. Birman and T. Joseph, "*Reliable communication in the presence of failure*," ACM Trns. on Comput. Syst., 5(1):47-76, 1987.

[3] Hrischuk, C.E. Woodside, C.M., "*Logical clock requirements for reverse engineering scenarios from a distributed system,*", IEEE Transactions on Software Engineering, pp.321-399, vol. 28 no. 4, April 2002.

[4] Abderahim Benslimane and Abdelhanfid Abouaissa, "*A Synchronization Protocol for Group Communication Systems*," Proc. of 7th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems, pp. 322 -329, Oct. 1999.

[5] Chengzheng Sun and Piyush Maheshwari, "*An Efficient Distributed Single-phase Protocol for Total and Causal Ordering of Group Operations*," Proceedings of 3rd International Conference on High Performance Computing, pp. 295 -300, Dec. 1996.

[6] F. Adelstein and M. Singhal, "*Real-Time Causal Message Ordering in Multimedia Systems,*" Proc. of IEEE ICDCS-15, pp.36-43, June 1995.

[7] A. D. Kshemkalyani and M. Singhal, "*Necessary and sufficient conditions on information for causal message ordering and their optimal implementation*," Tech. Repport TR33, Dept. of Comp. and Info. Science, the Ohio State University, Columbus, July 1995.

[8] Roberto BALDONI, Achour MOSTEFAOUI, Michel RAYNAL, "*Causal Multicast in Unreliable Networks for Multicast Applications*," IEEE Proceedings of International Conference on Computers and Communications, pp.51-57, March 1996.

[9] Roberto BALDONI, Achour MOSTEFAOUI, Michel RAYNAL, "*Broadcast with Time and Causality Constraints for Multimedia Applications*," IEEE Proceedings of EUROMICRO-22, pp.617-624, Sept. 1996.

[10] M. Raynal and M. Singhal, "*Logical time: capturing causality in distributed systems*," IEEE Computer Magazine, pp. 49-56 vol. 29, Feb. 1996.

[11] Takayuki Tachikawa, Hiroaki Higaki, and Makoto Takizawa, "*Group communication protocol for real-time applications*," Proc. of 18th International Conference on Distributed Computing Systems, pp. 40-47, May 1998.

[12] Birman, K. P., Schiper, A., and Stephenson, P., "*Lightweight Causal and Atomic Group Multicast*," ACM TOCS, vol.9, no.3, pp.272-314, 1991.

[13] D. L. Mills, "*Internet Time Synchronization: The Network Time Protocol*," IEEE Trans. on Comm. vol. 39, pp. 1482 -1493, Oct. 1991.

[14] Rodrigues, L.; Baldoni, R.; Anceaume, E.; Raynal, M., *"Deadline-constrained causal order*," IEEE Proc. of ISORC 2000, pp. 234 –241, 2000