利用同步訊息回覆的方法將移動式交易型代理人模式傳遞路徑快速化

陳忠信 鍾佩霖 洪振偉
朝陽科技大學 朝陽科技大學 亞洲大學
資訊與通訊研究所 資訊與通訊研究所 資訊工程系
助理教授 研究生 助理教授
jschen26@cyut.edu.tw s9630616@cyut.edu.tw stewart@asia.edu.tw

摘要(Chinese Abstract)

移動式交易型代理人模式會由一個任務代理人在不同的機器上依序執行不同階段的工作，當全部的階段執行完畢，才算完成此任務；否則任務就算失敗。傳統的方式，會因網路的不穩定(例如：擁塞和延遲)，都會使任務代理人無法準時到達，導致任務無法完成。本論文中提出一個可提高完成率和降低溝通時間的方法，利用同步訊息回覆的方法，找出數個可以繼續執行的下個階段，最快回應的一個階段(沒有產生網路擁塞的問題)，繼續執行。利用此方法，可以避免網路擁塞而沒有辦法繼續執行的問題，也可縮短階段間跟階段間，訊息溝通的時間。模擬的結果也驗證，我們的方法非常有效，可以大幅提高任務代理人的成功率和縮短階段間跟階段間訊息傳遞的時間。

關鍵詞：行動交易型、代理人、移動式、網路擁塞和容錯。

Abstract

A mobile transactional agent model divided a task into several strategies and sequentially performed by a task agent at different machines. The task is finished only when all the strategies have been performed. In the conventional method, the unstable conditions, such network congestion or delay will cause the task can not be finished. Accordingly, in this paper, we propose a method that can speed up the finish and decrease the communication among agents of a task. When a strategy is finish, the method will chose some strategies, which can be performed at next strategy, and then, send coordination messages to all of the targeted strategies. The strategy that first sends an acknowledgment back will be assigned as the next strategy to be performed. Notably, any unstable network conditions will block the response time among agents. Therefore, our method can speed up the success rate of a task and reduce the communication time among agents. Simulation results also demonstrated our method is efficient.

Keywords: mobile transactional model, agent, mobility, network congestion, and fault tolerance.

1. 序論

移動式交易型代理人模式(Mobile Transactional Agent)是一種特殊而嚴謹的程式，能夠同時一邊在各電腦終端機間移動，一邊能自主地運算[11]。它具有提高分散計算系統效率性及便利性，同時又可降低分散式系統上的傳遞延遲與電腦間傳送次數，因此代理人在分散處理系統中相當受到矚目[1][4][5]。移動式交易型代理人模式是由不同機器利用網路連結而成的，因此彼此互相溝通是非常重要地，在一筆交易中，我們不允許任何機器或網路產生錯誤，所以我們需要充分控制代理人，使代理人正確計算[8]。由於移動式交易型代理人模式相當嚴謹，只要在任何的環節出現錯誤，即容易造成工作失敗，而無法充分發揮代理人自主性的特色[6]，因此在交易代理人中使錯誤率與花費成本降低是相當重要的議題[3]。

在移動式交易型代理人模式架構中，一個使用者的任務被分配成數階段，依階段完成後再回傳結果給來源端的使用者，才算完成整個任務。就移動式交易型代理人模式而言，他會依序到各階段代理使用者執行任務，在此架構下若移動式交易型代理人模式再某一階段遇到問題(例如：網路短暫擁塞)時，將讓整個任務無法完成，必須重頭開始，如此會容易造成成本浪費[2]。為了改進此缺點，一、我們發現每一個機器中不僅僅只有單一關係，有時會發生兩個或兩個以上的相互關係，在執行順序上可以做更多選擇；二、當軟硬體或網路錯誤時，代理人即會判斷錯誤，因而中斷造成工作失敗，但是當網路擁塞或者機器忙碌時，無法在時間內回應，並非錯誤，而是短時間內無法處理[9]；結合上述兩點我們可以利用代理人自

主性，在階段任務中自行選擇最佳執行路線，可以避免非實質上錯誤的錯誤，如此整個任務能可以按部就班的執行。

基於上述的介紹，在這篇文章中，我們提出利用三方交握的方法將移動式交易型代理人模式傳遞路徑最佳化。在此機制中目前任務代理人必須向後監控代理人實施三方交握，只要任何工作站跟前工作站有關係，都可以作為後監控代理人，當後監控代理人回報訊息時，可依照回覆訊息的快慢，判斷工作站或是網路的正常與否，也可以大大減少因網路擁塞和機器忙碌所造成任務失敗，讓整個系統的失敗率降低。

文章的其他內容安排如下，首先第二節介紹採用的策略，接下來第三節將介紹模擬結果，最後第四節做一總結。

2. 採用的策略

2.1 系統架構

行動代理人從來源端至目的端在一連串的機器上執行不同階段(Stage)的任務，稱為旅遊路程，全部的階段可視為一個使用者所交付的任務 (Task)。本文中用 $T=\{S_1,S_2,.....,S_n\}$，代表一件使用者所交付的任務 T，其中 $S_1, S_2,....., S_n$ 分別代表 T 的階段 1，階段 2，…，階段 n 等必須完成的工作。

一個交易型的代理人模式，指的是由一個代理人來完成這 n 個階段的任務，當 n 個階段被完成時代表這個任務完成，否則代表這個任務失敗。對任一兩個階段 S_i 和 S_j，其中 $i<j$，則 S_i 和 S_j 的關係性有依序關係(>>)、不為依序關係(><)和平行關係(//)三種，如定義一所示。

定義一： 一任務 $T=\{S_1, S_2,..., S_n\}$ 中的任兩個階段 S_i 和 S_j 必滿足下列三種關係的一種關係[13]：

- $S_i >> S_j$ ：代表 S_i 和 S_j 為依序關係，其中 S_j 可被執行的必要條件為 S_i 已經被完成。
- $S_i >< S_j$：代表 S_i 和 S_j 不為依序關係，其中 S_j 可被執行條件不需要 S_i 已經被完成。
- $S_i // S_j$ ：代表 S_i 和 S_j 為平行關係，其中必須滿足 $S_i >< S_j$ 且 $S_j >< S_i$。

以圖 1 為例，使用者交付的任務(T)共有 $S_1, S_2, \cdots, S_5$，其中 S_1, S_3 和 S_5 具備依序關係($S_1>>S_3>>S_5$)，S_1, S_4 和 S_5 具備依序關係($S_1>>S_4>>S_5$)和 S_2, S_4 和 S_5 具備依序關係($S_2>>S_4>>S_5$)。因為 S_1 和 S_2 不具備依序關係($S_1 >< S_2$)而且 S_2 和 S_1 不具備依序關係($S_2 >< S_1$)，所以 S_1 和 S_2 具備平行關係($S_1//S_2$)。相同的，因為 $S_3><S_4$ 且 $S_4><S_3$，所以 $S_3//S_4$。

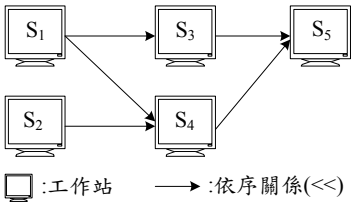

圖 1.交易任務的階段關係

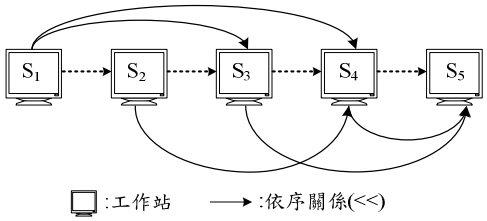

圖 2.階段排程

使用者所交付的任務 $T=\{S_1, S_2, S_3, S_4, S_5\}$ 中，代理人會依指定的任務執行每一個階段，以圖 1 為例，$\{S_1, S_2, S_3, S_4, S_5\}$ 和 $\{S_1, S_2, S_4, S_3, S_5\}$ 即為兩個不同的階段排程。當代理人執行完 S_1 和 S_2 的工作後，可以將 T 分成 T_N 和 T_F 兩個子集合，其中 $T_N = \{S_3, S_4, S_5\}$ 為 T 中未完成的階段，$T_F = \{S_1, S_2\}$ 為 T 中已經完成的階段。代理人接下來會由未完成的階段 $T_N = \{S_3, S_4, S_5\}$ 找出可以執行的階段 $Next(T, T_N, T_F)=\{S_3, S_4\}$ 來繼續執行，即 S_3 和 S_4 皆為可以繼續執行的階段。一個任務執行的階段如定義二所示。

定義二： 一任務 $T=\{S_1, S_2,..., S_n\}$，其中 $T_N \subseteq T$ 且 $T_F \subseteq T$，分別為 T 中未完成的階段和已經完成的階段。可繼續執行的階段 $Next(T, T_N, T_F)$ 定義如下:

$Next(T, T_N, T_F) = \{S_i \mid S_i \in T_N$ 且對任意的 $S_j >> S_i, S_j \in T_F\}$。

對於任意階段 S_i 包含了任務代理人 α_i 和監控代理人 ω_i 兩種來執行工作[7][12]:

- 任務代理人(α_i)：實際在第 i 階段中執行使用者所交付的任務之代理人。
- 監控代理人(ω_i)：被動的等待、接收 α_i 以外階段任務所傳來的訊息，利用訊息傳遞的速度選擇下一個階段任務。

現階段(S_i)跟下階段(S_j)會傳遞下列訊息：

- $SYN_m(S_i)$：由 S_i 傳送 S_j 要求同步的訊息。
- $ACK\_SYN_m(S_j)$：由 S_j 傳送 S_i 的回應，用

來回應此 S_i 所傳過去的 $SYN_m(S_i)$訊息。

– $ACK\_ACK_m(S_i)$：由 S_j 傳送 S_i 的回應，挑選優先到達訊息的工作站，並傳遞 $ACK_SYN_m(S_i)$訊息告知接續任務。

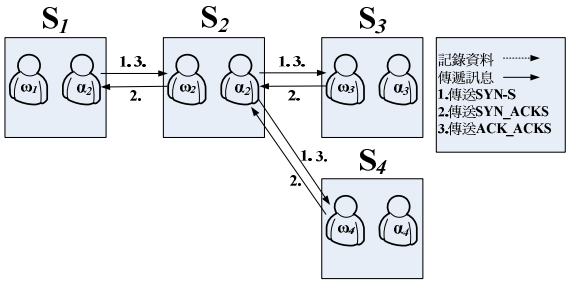

圖 3.流程圖

圖 3 流程圖顯示執行代理人需在各個階段 $T=\{S_1,S_2,.....,S_n\}$ 中執行使用者的局部溝通任務。首先，以圖 3 為例在部分階段 S_1 到 S_2 的任務流程為例：

當 S_1 階段任務完成後，則 S_1 利用 $Next(T, T_N, T_F)$找尋下一階段任務，即要求與下一階段同步，任務代理人(α_1)傳送 $SYN_m(S_1)$訊息給等待工作站 S_2，詢問等待工作站是否可以服務。等待工作站 S_1 回應訊息 $ACK_SYN_m(S_2)$給任務階段工作站 S_1，告知任務階段工作站已經待命。優先選擇訊息先到的工作站，傳遞 $ACK_ACK_m(S_1)$給等待工作站，則完成同步動作，並重覆依序上列執行步驟，直到完成任務。

2.2 系統策略

初始: T_N= T 和 T_F= ∅

步驟1： 初始階段(S_1):由任務代理人(α_1)，實際在第 1 階段中執行使用者所交付的工作。當 α_1 完成 S_1 工作後，則由監控代理人(ω_1)完成步驟 2 的動作。

步驟2： 監控代理人(ω_1)將 T_F 指定 $\{T_F\}\cup\{S_1\}$ ，T_N 指定 $\{T_N\}-\{S_1\}$則計算 $Next(T, T_N, T_F)$，接續執行步驟 3。

步驟3： 當 T_N=∅時，則表示任務已完成，否則，監控代理人(ω_1) 則傳送 $SYN_m(S_1)$訊息給 $Next(T, T_N, T_F)$每一個候選監控代理人(ω_j)，其中(ω_j)所在在的階段 S_i 屬於 $Next(T, T_N, T_F)$，當 ω_j 接收到 $SYN_m(S_1)$ 時，則會回應 $ACK_SYN_m(S_j)$給 ω_1，當 ω_1 收到第一個回應訊息 $ACK_SYN_m(S_k)$時，將傳遞 $ACK_ACK_m(S_1)$給 S_k，由 S_k 接續未完成任務。

S_i 的執行步驟如下(i≠1)：

步驟1： 階段(S_i):由任務代理人(α_i)，實際在第 i 階段中執行使用者所交付的工作。當 α_i 完成 S_i 工作後，則由監控代理人(ω_i)完成步驟 2 的動作。

步驟2： 監控代理人(ω_i)將 T_F 指定 $\{T_F\}\cup\{S_i\}$ ，$\{T_N\}$指定 $\{T_N\}-\{S_i\}$則計算 $Next(T, T_N, T_F)$，接續執行步驟 3。

步驟3： 當 T_N=∅時，則表示任務已完成，否則，監控代理人(ω_i) 則傳送 $SYN_m(S_i)$訊息給 $Next(T, T_N, T_F)$每一個候選監控代理人(ω_j)，其中(ω_j)所在在的階段 S_i 屬於 $Next(T, T_N, T_F)$，當 ω_j 接收到 $SYN_m(S_i)$ 時，則會回應 $ACK_SYN_m(S_j)$給 ω_i，當 ω_i 收到第一個回應訊息 $ACK_SYN_m(S_k)$時，將傳遞 $ACK_ACK_m(S_i)$給 S_k，由 S_k 接續未完成任務。

以圖 1 交易任務的階段關係為例：

初始階段(S_1):$\{T_N\}$= S_1 S_2 S_3 S_4 S_5 和 $\{T_F\}$= ∅由任務代理人(α_1)，實際在第 1 階段中執行使用者所交付的工作。當 α_1 完成 S_1 工作後，則由監控代理人(ω_1)將$\{T_F\}$指定$\{T_F\}\cup\{S_1\}$ ，$\{T_N\}$指定$\{T_N\}-\{S_1\}$則計算 $Next(T, T_N, T_F)$= S_2 和 S_3，則監控代理人(ω_1)則傳送 $SYN_m(S_1)$訊息給 $Next(T, T_N, T_F)$ = S_2 和 S_3，當 ω_2 和 ω_3 接收到 $SYN_m(S_1)$時，則會回應 ACK_SYN_m 給 ω_1，當 ω_1 收到第一個回應訊息 $ACK\_SYN_m(S_2)$ 時，將傳遞 $ACK_ACK_m(S_2)$給 S_2。

階段(S_2):由任務代理人(α_2)，實際在第 2 階段中執行使用者所交付的工作。當 α_2 完成 S_2 工作後，則由監控代理人(ω_2) 將 T_F 指定$\{T_F\}\cup\{S_2\}$，$\{T_N\}$指定$\{T_N\}-\{S_2\}$則計算 $Next(T, T_N, T_F)$= S_3 和 S_4，則監控代理人(ω_2) 則傳送 $SYN_m(S_2)$訊息給 $Next(T, T_N, T_F)$ = S_3 和 S_4，當 ω_3 和 ω_4 接收到 $SYN_m(S_2)$時，則會回應 ACK_SYN_m 給 ω_2，當 ω_2 收到第一個回應訊息 $ACK_SYN_m(S_4)$時，將傳遞 $ACK_ACK_m(S_4)$給 S_4。

階段(S_3):由任務代理人(α_4)，實際在第 3 階段中執行使用者所交付的工作。當 α_4 完成 S_4 工作後，則由監控代理人(ω_4) 將 T_F 指定$\{T_F\}\cup\{S_4\}$，$\{T_N\}$指定$\{T_N\}-\{S_4\}$則計算 $Next(T, T_N, T_F)$= S_3，則監控代理人(ω_4)則傳送 $SYN_m(S_4)$訊息給 $Next(T, T_N, T_F)$ = S_3，當 ω_3 接收到 $SYN_m(S_4)$時，則會回應

ACK_SYN_m 給 ω_4，當 ω_4 收到第一個回應訊息 $ACK_SYN_m(S_3)$時，將傳遞 $ACK\_ACK_m(S_3)$ 給 S_3。

階段(S_4):由任務代理人(α_3)，實際在第 4 階段中執行使用者所交付的工作。當 α_3 完成 S_3 工作後，則由監控代理人(ω_3) 將 T_F 指定$\{T_F\}\cup\{S_3\}$，$\{T_N\}$指定$\{T_N\}-\{S_3\}$則計算 $Next(T, T_N, T_F)= S_5$，則監控代理人(ω_3)則傳送 $SYN_m(S_3)$訊息給 $Next(T, T_N, T_F) = S_5$，當 ω_5 接收到 $SYN_m(S_3)$時，則會回應 ACK_SYN_m 給 ω_5，當 ω_3 收到第一個回應訊息 $ACK_SYN_m(S_5)$時，將傳遞 $ACK\_ACK_m(S_5)$ 給 S_5。

階段(S_5):由任務代理人(α_5)，實際在第 5 階段中執行使用者所交付的工作。當 α_5 完成 S_5 工作後，則由監控代理人(ω_5) 將 $\{T_F\}$指定$\{T_F\}\cup\{S_5\}$，$\{T_N\}$指定 $\{T_N\}-\{S_5\}$則計算 $Next(T, T_N, T_F)= \varnothing$，當 $T_N=\varnothing$時，則表示任務已完成。

表 1.模擬環境的參數定義

變數名稱	數值	說明
n	10,20,40, 80	一項任務所需的階段數量
c	0, 0.1, 0.2,...,1	階段與階段依序關係率
$SYN_m(S_i)$	100	傳遞 SYN-S 的時間
$ACK\_SYN_m(S_j)$	100	傳遞 SYN-ACKS 的時間
$ACK\_ACK_m(S_i)$	100	傳遞 ACK-ACKS 的時間
T_{clock}	210	監控代理人時間計時器的時間

3. 模擬結果

驗證中針對階段任務和階段任務在不同的平行關係和依序關係的環境下，分析探討訊息傳遞的時間與使用者任務的成功率做相對性的比較。

在模擬當中，假設一個工作有 n 的階段(n=10、20、40 和 80)，在每一個階段本來就需執行的任務，這段時間不列入計數，傳送分別為 SYN_m、ACK_SYN_m 和 ACK_ACK_m 三種訊息的時間平均為 100 單位，由 poisson 程序產生，當 SYN_m 和 ACK_SYN_m 產生的時間超過 210 單位時，則認定網路擁塞或是網路斷線產生。任意兩個階段 S_i跟 S_j(i<j)具備依序關係(即 S_i>>S_j)機率值為 c，其中 c=0, 0.1, 0.2,...,1。當 c =0 時，全部的階段都沒有依序關係，相反的，當 c =1 時，表示 S_i 可以被執行的條件為 S_{i-1} 已被完成，所以整個階段排程，只有 S_1 先被執行然後 S_2 接著 S_3，依次類推直到 S_n 一種而已。具備全部的模擬參數如表一所示，圖 4 到圖 8 顯示模擬結果。

我們模擬三個方法：1、sequence，2、ours，3、boundary。第一種方法為 sequence，為一般移動式交易型代理人模式，一般移動式交易型代理人模式都是依序執行任務，只有 S_1 先被執行然後 S_2 接著 S_3，依次類推直到 S_n 一種而已；第二種方法為 ours，此為我們提出的方法，當 $Next(T, T_N, T_F)$>1 時，優先選擇最快傳遞 SYN_m 和 ACK_SYN_m 訊息，當下一個階段候選人；第三種方法為 boundary，此為我們的方法最差的情況，當 $Next(T, T_N, T_F)$>1 時，選擇傳遞 SYN_m 和 ACK_SYN_m 訊息最慢的工作站，作為下一個階段候選人。

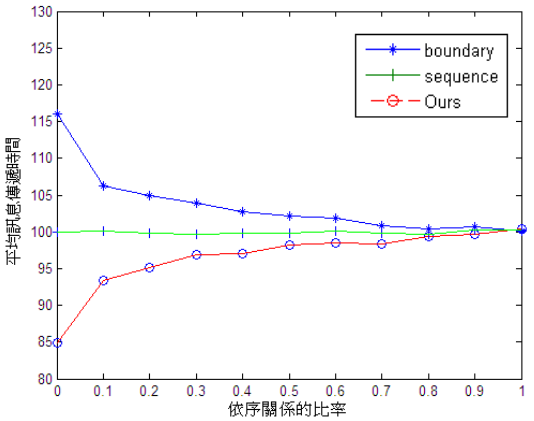

圖 4. 訊息傳遞平均時間

傳遞時間平均所花費的時間。根據圖 4 的模擬結果，橫軸為階段依序關係的比例，縱軸為三種訊息平均所花費的時間，訊息傳遞是利用 poisson 分配的情況下，可以發現 sequence，無論是在 c=0 或是 c=1，都是依序逐一執行任務，平均傳遞時間都維持在 100 單位，；而我們提出的方法 ours，可以優先選擇最快傳遞 SYN_m 和 ACK_SYN_m 訊息，因此在 c=0 時，全部的階段都沒有依序關係，我們的方法可以達到最佳成效，相反的當 c=1 時，表示 S_i 可以被執行的條件為 S_{i-1} 已被完成，所以 $Next(T, T_N, T_F)$=1，即使可以優先選擇，也只能挑選一個，就同等於一般移動式交易型代理人模式，因此模擬結果會趨近於 sequence；我們也提出 boundary，假設當所有的訊息都處於擁塞的情況下，在 c=0 時，$Next(T, T_N, T_F)$>1，選擇最慢

傳遞 SYN_m 和 ACK_SYN_m 訊息，為下一個工作站，會導致成最差情形，所消耗的訊息時間會比較長，相反的在 c=1 時，$Next(T, T_N, T_F)$=1，只能逐一執行無法挑選優先到達的訊息，因此在 c=1 時會趨近 100，在數線上表示為 boundary。

以圖 4、圖 5、圖 6、圖 7 和圖 8 為例，當依序關係比率為 0 時，階段任務會與多個階段任務有關係，相反的，當依序關係比率為 1 時，階段任務只會與一個階段任務有關係，則無論是我們提出的 Ours 或 boundary，只要在關係比率為 1 的情況下，都變成 sequenc，則 Ours 或 boundary 在關係比率為 1 的情況下，都會趨近於 sequenc 的模擬數值。

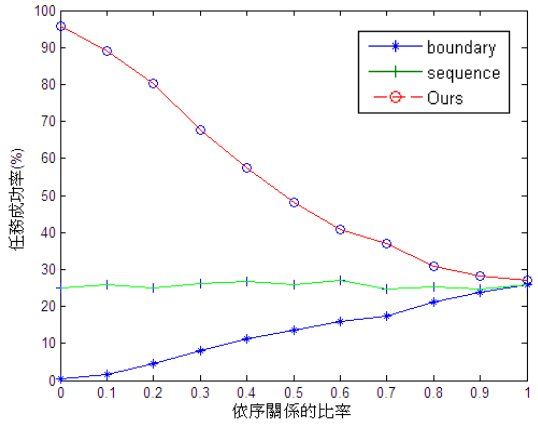

圖 5. n_s 為 10 的任務完成率

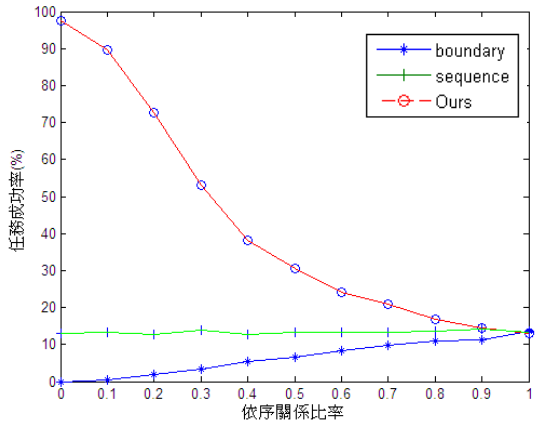

圖 6. n_s 為 20 的任務完成率

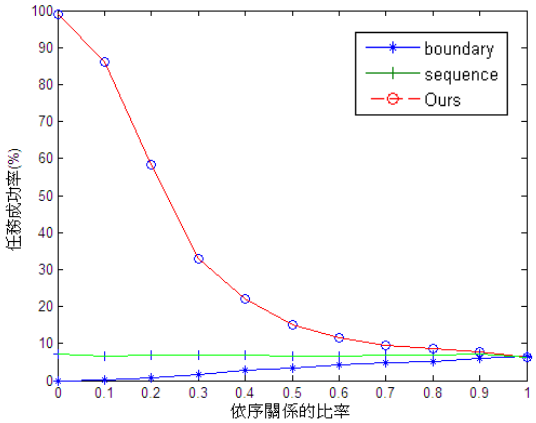

圖 7. n_s 為 40 的任務完成率

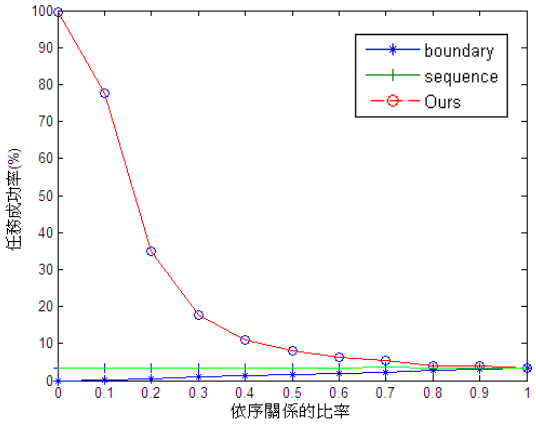

圖 8. n_s 為 80 的任務完成率

圖 5、圖 6、圖 7 和圖 8，顯示在使用者任務階段數分別為 10、20、40 和 80，分別判斷在不同階段數，使用者任務的成功率結果，每一個點模擬一千次的平均完成率。根據圖 5、圖 6、圖 7 和圖 8 的模擬結果，橫軸為階段依序關係的比例，縱軸為任務完成率，Tclock 為監控代理人時間計時器的時間，當傳遞訊息 SYNm 和 Next(T, TN, TF)傳遞 ACK_SYNm 訊息在超過 Tclock=210 時，我們則判斷任務失敗，訊息傳遞是利用 poisson 分配的情況下的模擬結果。

以 sequence 為例在圖 5、圖 6、圖 7 和圖 8 中，會因為 n 的階段數增加，成功機率會逐漸下降，當 n=80 時，成功機率僅維持在 4%；在 ours 的方法下，我們可以清楚得到圖 5、圖 6、圖 7 和圖 8 中，當 c=0 時 $Next(T, T_N, T_F)$>1，可選擇 $Next(T, T_N, T_F)$，可先篩選比較好的 Sk，讓成功率上升，因此成功率都維持在 95%以上，相反的，當 c=0 時 $Next(T, T_N, T_F)$=1，只能逐一執行無法挑選，則會漸漸趨近於 sequence；在 boundary 的方法下，我們可以在

圖 5、圖 6、圖 7 和圖 8 中發現到當當 c=1 時 $Next(T, T_N, T_F)$>1，成功率幾乎都是 0%的情況，會導致最差的情形，相反的當 c=0 時 $Next(T, T_N, T_F)$=1，只能逐一執行無法挑選，則會漸漸趨近於 sequence。

4. 結論

本篇文章中我們提出利用三方交握的方法將移動式交易型代理人模式傳遞路徑最佳化的機制，用來解決在執行任務環境中，代理人因為網路擁塞、錯誤而造成任務無法順利執行。我們分別找出階段任務與階段任務間的關係，利用它們彼此之間的依序關係和平行關係，找出多種執行路徑，加上運用訊息的傳遞，優先選擇網路傳遞速度較快的工作站，也可以避免網路擁塞，造成訊息的遺失。根據模擬數據顯示，我們的方法在與一般移動式交易型代理人模式比較在 c=0 時，訊息的花費時間上比較短，在成功率的部份比較高，在 c=1 時，表現上也不會比一般移動式交易型代理人模式遜色，因此在數據上顯示，利用我們的方式，相較過去的機制都有較佳的表現，為移動式交易型代理人模式執行環境提供了一個有效的機制。在現實生活中，我們無法控制階段任務與階段任務之間的關係，因此我們提供以上數據，提供使用者使用上的依據。

本篇文章中我們針對移動式交易型代理人模式在選擇路徑上快速化，降低傳送後造成的失敗，未來在移動式交易型代理人模式環境中，我們會針對兩大問題做探討：一、當移動式交易型代理人模式在階段任務中發生資訊不一致，則會造成最後結果不一致；二、當移動式交易型代理人模式在階段任務中發生無法回應，則會造成代理人停留至同一階段，導致代理人無限期的等待。由上述的可知，移動式交易型代理人模式將會有複雜代理人錯誤情況，因此未來將對這些問題深入研究與探討。

參考文獻 (References)

[1] A. Holt, C.-Y. Huang, J. Monk, " Performance analysis of mobile agents," ***Communications, IET,*** vol. 1, 2007, pp. 532-538.

[2] Jin Yang, Jiannong Cao, and Weigang Wu, "CIC: An Integrated Approach to Checkpointing in Mobile Agent Systems," ***Knowledge and Grid,*** 2006, pp. 4-4.

[3] K. Mohammadi and H. Hamidi, "An Approach to Fault-Tolerant Mobile Agent Execution in Distributed Systems," ***The First IEEE and IFIP International Conference in Central Asia,*** 2005, pp. 26-29.

[4] L.M. Silva, V. Batista, and J.G. Silva, "Fault-Tolerant Execution of Mobile Agents," ***Proc. Int'l Conf. Dependable Systems and Networks, IEEE CS Press,*** 2000, pp. 135-143.

[5] M.A.M. Ibrahim, "Distributed Network Management with Secured Mobile Agent Support" ***Hybrid Information Technology,*** vol.1, 2006, pp.244-251.

[6] M. Dalmeijer, E. Rietjens, D. Hammer, A. Aerts, and M. Soede, "A Reliable Mobile Agents Architecture," ***Object-Oriented Real-Time Distributed Computing,*** 1998, pp. 64-72.

[7] M.R. Lyu, Xinyu Chen, and Tsz Yeung Wong, "Design and Evaluation of a Fault-Tolerant Mobile-Agent System," ***IEEE Digital Object Identifier,*** vol. 19, 2004, pp. 32-38.

[8] S. Pears, J. Xu, and C. Boldyreff, "Mobile Agent Fault Tolerance for Information Retrieval Applications: An Exception Handling Approach," ***Autonomous Decentralized Systems, IEEE CS Press,*** 2003, pp. 115-122.

[9] S. Pleisch and A. Schiper, "Fault-Tolerant Mobile Agent Execution," ***IEEE Trans. Computing,*** vol. 52, no. 2, 2003, pp. 209-222.

[10] T. Osman, W. Wagealla, and A. Bargiela, "An Approach to Rollback Recovery of Collaborating Mobile Agents," ***IEEE Trans. Systems, Man and Cybernetics,*** Part C, vol. 34, no. 1, 2003, pp. 48-57.

[11] Xuejun Meng, and Huanguo Zhang, "An Efficient Fault-Tolerant Scheme for Mobile Agent Execution," ***Systems and Control in Aerospace and Astronautics,*** 2006, pp. 19-21.

[12] Y. Tanaka, N. Hayashibara, T. Enokido, and M. Takizawa, "A Fault-Tolerant Transactional Agent Model on Distributed Object Systems," ***Advanced Information Networking and Applications,*** vol. 2, 2006, pp. 18-20.