基於 rwnd 的 CMT-SCTP 傳輸機制研究

張林煌
臺中教育大學
資訊科學學系
lchang@mail.ntcu.edu.tw

黃柏勛
朝陽科技大學
資訊與通訊系
s9630605@cyut.edu.tw

顏丞均
朝陽科技大學
資訊與通訊系
s9730618@cyut.edu.tw

廖俊鑑
朝陽科技大學
資訊與通訊系
jjliaw@cyut.edu.tw

***摘要*—由 IETF 所提出的 Stream Control Transmission Protocol (SCTP) 是一種新的傳輸協定，改善了原有 Transmission Control Protocol (TCP) 協定在資料傳輸上效率以及安全性等問題。然而儘管 SCTP 傳輸協定擁有原生的多路徑傳輸能力，卻同時僅能使用單一路徑進行傳輸，其餘路徑供備援使用。目前有著相當多的學者進行著使 SCTP 能利用多路徑傳輸機制的研究，利用各種網路因素如頻寬、封包遺失率、延遲等偵測機制來作為封包排程的依據，以期達到負載均衡的目的，進而提升整體傳輸效能。本文中我們藉由修改 Linux Kernel 上的 SCTP Module 部份，將原先單一路徑傳輸修改成支援多重路徑的同步傳輸的架構，並且結合了我們所提出的 ARD-CMT 傳輸機制，利用接收端的 Receiver Window(RWND)及發送端的平均速率來計算權重，進行資料封包的排程。最後並以實際環境下的測量來證明我們提出的 ARD-CMT 機制不僅僅可以提升傳輸效率，且因有良好的封包排程機制，進而實現負載均衡的目的。**

***關鍵詞*—SCTP、CMT、Linux、效能、封包排程**

一、前言

隨著網際網路與資訊科技的蓬勃發展，現今可利用於傳輸的媒介更是多樣，為了追求更快、更可靠、更有效率的網路傳輸技術，有不少研究學者們在如何增進最大傳輸量的多路徑傳輸與無線網路的行動管理等方面進行研究。由於早期網路環境較為單純，當時發展的 Transmission Control Protocol (TCP)[6] 和 User Datagram Protocol (UDP)[7] 等通訊協定有著諸如：(1) Head-of-Line (HOL) blocking (2)訊息串流導向(stream-oriented) (3)不支援路徑多宿(multi-homing) (4) TCP 無法避免主機資源耗盡的阻斷式攻擊(Denial of Service, DoS) 等問題。也因此在以往的研究當中，都需要額外付出成本才能夠改善其問題點。

由於 SCTP 具有 multi-homing 的特性，multi-homing 允許建立關聯(association)的端點(endpoint)間同時擁有多組網路介面位址，並且不會因為單一路徑斷線而中斷關聯，此特性讓 SCTP 十分適合延伸發展同步多路傳輸(Concurrent Multipath Transfer, CMT)的功能。但由於 SCTP 的傳輸機制是以 TCP 傳輸協定為基礎架構，故截至目前為止其傳輸機制中仍然只會選擇單一路徑作為主要傳輸的路徑(Primary Path)，而其餘路徑則定義成備援路徑(backup path)，這些備援路徑將負責重傳和容錯功能，僅於主要路徑無法作用時才會由備援路徑取代主要路徑繼續傳輸資料，這種特性雖然提昇了網路的可靠性，但因為其他備援路徑將會長時間處於閒置(idle)狀態，無法善用整個頻寬來提昇網路傳輸效能，也因此本論文將設計並提出 ARD-CMT 的系統架構，藉此善用所有可以傳輸資料的網路資源，以提昇資料傳輸效能。

本文之架構與內容如下所述，第二部份介紹 SCTP 通訊協定的特性與相關的研究，第三部份介紹本論文所提出的系統架構與運作方式，第四部份為對我們所實作的實驗環境系統介紹及效能分析，最後是本篇論文結論及未來展望。

二、文獻探討

由於 TCP/IP 本身設計結構之缺陷導致安全性與及效能等問題漸漸出現，因此國內外有不少學者針對可以彌補 TCP 缺點的 SCTP 做研究，在近年來 SCTP 如何使用 CMT 技術增進傳輸效能跟可靠度是熱門研究之一，如 R. Stewart[9] 等學者預期使用 SCTP 的 multi-homing 透過所有網路介面傳輸資料來達到提升傳輸效能之目的，這些學者們也針對他們提出的多路徑傳輸架構中因為所有路徑共享資源的問題，他們使用了 Cwnd Update for CMT(CUC)演算法來改善在 CMT 的架構中，CWND 會因為 SACK 並無法辨別封包遺失的路徑，但是由於他們只使用 CWND 做為封包排程的基礎，當接收端發生佇列擁塞時，將會在接收端造成更大的擁塞問題。

而由 A. Abd El[2] 等學者提出的 load sharing-SCTP(LS-SCTP)機制中，藉著修改 SCTP chunks 來使用封包資料平均分配到所有可傳輸路徑來達到多路徑傳輸，並且每個路徑都有獨立的 congestion control 跟接收端緩衝區(buffer)來保證路徑之間不會互相影響其效能；另外 Chung-Ming Huang 等學者[3] 提出延伸 SCTP 協定的 Wireless Multi-Path SCTP (WiMP-SCTP)來同時到達到多路傳輸跟換手功能，而 WiMP-SCTP 的架構中每個路徑都有個虛擬的 buffer 來傳輸資料，並且使用資料平均方式來送到每條路徑上來達到多路徑傳輸。而這些多路徑傳輸都是使用資料平均法，沒有加入封包排程作為基礎，因此當路徑產生封包遺失等環境干擾因素時，其傳輸效能將會持續下降。

F. Perotto 等學者[5] 文章中提到的基於 sender-based packet pair SCTP(SBPP-SCTP)[11] 及 Westwood SCTP[4] 所延伸出的 Westwood SCTP-PR 等方法，而 C. Casetti 等學者[11] 也修改 partial reliability stream control transport protocol(PR-SCTP) [12] 來達到多路徑即時資料傳輸的研究，他們提出 Bandwidth-aware Scheduling，讓封包依據各路徑的傳輸速度和 CWND 值恢復的速度等參數計算出的權重值排列及分配，提昇了整體傳輸的效率和網路使用率；而在 Jianxin Liao 等學者在[8] 提出的 cmpSCTP 改善標準 SCTP 的 multi-homing 機制達到多路徑傳輸的功能，cmpSCTP 延伸 SCTP multi-homing 的特性，透過路徑頻寬偵測技術跟封包排程機制選擇最好的路徑來傳輸資料，並達成多路徑傳輸的功能來改善標準傳輸速率。在這些研究中由於皆有頻寬偵測的方法，因此可以避免當路徑擁塞所造成傳輸品質下降，但是當發生封包延遲回應或者封包遺失時，由於沒有封包排程的基礎，因此並沒辦法在易產生延遲或者封包遺失的環境下進行傳輸。

在以往的相關文獻中，頻寬或者資料平均的封包分配方法並無法及時反映資料傳輸品質，當路徑品質不佳時，將會對傳輸過程產生影響，而接收端 RWND 會隨著資料傳輸的情況隨之變動，當路徑品質不佳或者資料傳輸失敗時，接收端 RWND 都會受到影響，而目前的相關研究中並沒有一個針對接收端 RWND 做為封包傳輸依據的演算法；因此，本論文將設計與實作出 ARD-CMT 系統達到多路徑傳輸，並且基於接收端的 RWND 和發送端的平均速率提出分配封包順序之演算法，藉此提升 ARD-CMT 的傳輸效能之外，更達到負載均衡之目的。

三、系統設計

由於在 SCTP 的傳輸機制標準中並沒有多路徑傳輸的能力，為實作 SCTP 所有路徑皆可傳輸資料，我們將修改系統核心程式碼和部份封包欄位格式，以達到提升 SCTP 傳輸效能之目的。另外在眾多對於 CMT 提出許多封包排程的

演算法來改善其傳輸品質的文獻中，我們在於 SCTP 封包到各個路徑界面的順序中使用延伸 adjusted arrival rate dependent (A-ARD)之基於 RWND 的封包排程方法，除了提升傳輸效能外，也避免網路干擾所造成的服務品質下降。

3.1 基於 RWND 的 A-ARD 演算法

本演算法是將 Chang 等人[1] [10] 所提出的數學演算法，應用到 CMT 的架構上，A-ARD 演算法主要是藉由分析抵達率與服務率之間的關係，動態計算權重的分配。由於在現行的 CMT 機制中，雖然也有基於頻寬等封包排程等方法，但普遍都還是使用 Round Robin(RR)的封包排程方式，當路徑擁有干擾因素時，使用頻寬偵測機制的排程法則即會產生誤判的情形，因此我們提出了基於 RWND 值的 CMT 傳輸機制，藉由發送端的平均發送率和接收端的 RWND 來動態計算各路徑的權重比例，並加以分配至各個輸出佇列(output queue)。

在 RFC 4960[13] 中，RWND 是 SCTP 的發送端用來計算接收端還可接收多少 window 的變數，單位為 byes，而發送端可以從 SACK chunks 傳回來 advertised receiver window credit(a_rwnd) 欄位取得接收端 buffer 的資訊，當端點要建立關聯時，會在初始關聯時互相告訴對方雙方的 a_rwnd 值；而接收端在每次接收新的資料時都會減去其 a_rwnd，這樣在接收端 buffer 不足時能夠有效限制發送端的傳輸資料，當接收端成功從接收端 buffer 讀取資料到 ULP 時，會補足送到 ULP 的資料量，這樣的效果可以表示此資料已成功的從這路徑傳輸到對方的應用層，並且反映封包遺失率等告訴發送端還可傳輸更多的資料量，而如果資料在其接收佇列中還未送到 ULP 的話則不增加其 a_rwnd，此時 a_rwnd 也反映出其佇列處理能力。

針對 A-ADR 所做之延伸修改，主要是基於 CMT 的傳輸架構下應用 A-ARD 的方法，藉由發送端的平均速率跟接收端的 RWND 來動態計算其每個子系統的權重。原先 A-ARD 的架構只負責 traffic direct 單一端點，當應用在 CMT 傳輸時，只計算發送端的服務率和抵達率有可能會因為路徑情況的異動而導致權重分配判斷錯誤；故此處我們將 A-ARD 的服務率改成接收端的 RWND，因為 RWND 可接確認對方是否接收資料成功與否，且計算單位時間內的傳輸速率和接收端過往的 RWND 紀錄即可反映其傳輸過程是否有延遲現象，所以我們應用 A-ARD 的演算法在 CMT 的傳輸機制下，實作出基於 RWND 的 CMT 傳輸機制，此方式將可以讓 CMT 的傳輸機制真正有效的因應傳輸情況而動態調整權重，達到最好之負載均衡，其程式虛擬碼如圖 1：

On receipt of a SACK [sender side behavior]

1) If($T_{now} - T_{last} > 200$ ms) then
2) for each path P_i to process a_rwnd do:
   (i) if path P_i are receive INIT_ACK chunk (include SACK chunk) then
       set $P_i.avg\_rwnd = P_i.now\_rwnd$;
   (ii) else
       set $P_i.avg\_rwnd = (1-\alpha) * P_i.avg\_rwnd + \alpha * P_i.now\_rwnd$;
3) ARD-CMT used A-ARD formula then
   (i) set $\beta = \dfrac{(\frac{1}{\lambda}) * (\sum_{i=1}^{N} \sqrt{P_i.avg\_rwnd})^2}{\frac{1}{\lambda^2}(\sum_{i=1}^{N} \sqrt{P_i.avg\_rwnd})^2 - \frac{2}{\lambda}(\sum_{i=1}^{N} \sqrt{P_i.avg\_rwnd}) + 1}$
   (ii) set $W_i = (P_i.avg\_rwnd / \lambda) * (1 - \sqrt{\dfrac{\lambda}{P_i.avg\_rwnd * \beta}})$;
   (iii) set $P_i.weight = W_i$;

圖 1、ARD-CMT 演算法

T_{now}：目前程式執行時間。

T_{last}：上次計算權重的時間。

α：預設為 $\frac{1}{8}$ 的常數，讓以往的 RWND 權重擁有較重的比例。

W_i：發送端第 i 條路徑的權重。

P_i：發送端第 i 條路徑，i 為路徑編號，avg_rwnd 是以往的 RWND 平均值，now_rwnd 是目前接收端可用的 RWND。

當端點開始傳輸後，SCTP 預設在每 200ms 會發送一個 SACK chunk，當封包延遲超過接收端接收時間時，接收端會丟棄其 DATA chunks，而單位時間內 a_rwnd 越少的話，代表其路徑品質越差。另 SCTP 在單位時間內所有路徑會從 SACK 的 a_rwnd 取得目前接收端可用的 RWND 資訊，當第一次收到 SACK chunk 時不需計算過往的 RWND，之後則會利用 α 常數增加其過往記錄的權重，常數的定義參考 RFC4960[13]，各路徑記錄 RWND 之後，ARD-CMT 會使用 A-ARD 公式計算出路徑 i 的權重 W_i，然後再帶入路徑 P_i 結構中。

應用程式的資料會經過切割接著送入發送端 buffer 再由傳輸層封裝成 chunks 往下層封裝後送出，而 CMT 的架構主要是由發送端 buffer 接收應用程式的資料時，計算接收端的 RWND 資訊後依此選擇路徑將資料轉送出去；不僅考慮發送端的服務能力，也需將雙方傳輸的狀況納入考量，因此我們提出跟路徑狀況和端點傳輸能力息息相關的 RWND，如圖 2 所示，λ 指的是平均傳輸速率，單位時間內發送到發送端緩衝區塊的傳輸量，當需要被服務的資料量進入到緩衝區裡，會依據 A-ARD 演算法推導的每個子系統所分配到的權重 ϕ_i 比例分配封包到每個輸出佇列裡，而權重 ϕ_i 比例的分配是依據 λ 和 μ_i 來決定，而 μ_i 則是指接收端每個子系統 i 可接收的資料量，在 CMT 的架構下，我們利用 SCTP 機制中，當接收端接收 DATA chunk 後回傳的 SACK chunk，SACK chunk 的 a_rwnd 欄位將會把可接收的資料量大小告訴發送端。

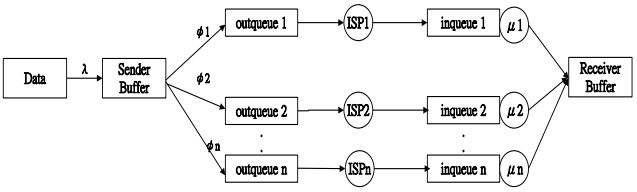

圖 2、ARD-CMT 佇列模式架構圖

3.2 系統架構

CMT SCTP 實作架構如圖 3 所示：(1)傳輸資料時會將 payload 經由不同的 stream 送入到緩衝區，(2)當緩衝區收集到一定程度的 payload 後，會將資料集中送到傳送端的 Queue Management 模組，並且(3)傳送端的 Location Info Management 會利用 SACK chunk 蒐集到的雙方傳輸品質和路徑品質記錄成資訊，而(4)傳送端的 Queue Management 會將緩衝區送來的資料跟傳送端的 Location Info Management 的資訊做媒合，把此 Data chunk 放入適合傳輸的傳輸序列(PID)，並讓(5)傳送端的 Packet Dispatcher 依照 DATA chunk 中的 PID 組成封包，並且依照 PID 丟到每一條輸出佇列(output queue)中，在由下層傳輸至接收端的輸入佇列(input queue)，(6)接收端的 Packet Dispatcher 模組收到每一條輸入佇列的封包後，將其解開並送到(7)Queue Management 將收到的封包資訊與 payload 分開來，由接收端的 Location Info Management 記錄路徑品質與及對方傳輸品質，之後再將所有輸入佇列的 payload 送到緩衝區塊蒐集後，(8)由上層組成完整的資料。

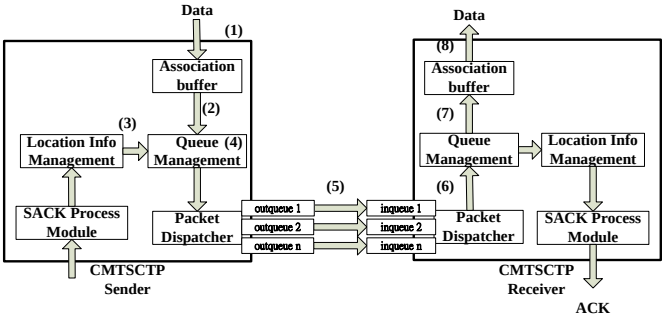

圖 3、ARD-CMT 系統架構圖

3.3 資料區塊格式

在傳輸資料方面，原本 TSN 用來當作資料排序的用途，為了避免各路徑上 TSN 不連號的情況，我們將 TSN 由 32 bit 減少為 16 bit，前 16 bit 則是新增 PSN 欄位，PSN 是每一條路徑的封包連續獨立編號，在各路徑使用 Congestion Control 時，用以判別是否有封包遺失的情況。另外為了識別 DATA chunk 的路徑所屬，我們修

改 Stream Identifier 原本 16 bit 切割成 8 bit，前 8 bit 作為路徑的識別碼，用以做為每一條路徑的獨立編號，以此辨別 DATA chunk 的路徑，由於此兩欄位在傳輸資料過程中不會用到超過一半的資料長度，因此我們修改標準 DATA chunk 也不會造成傳輸過程發生問題，修改後的 DATA chunk 格式如圖 4。

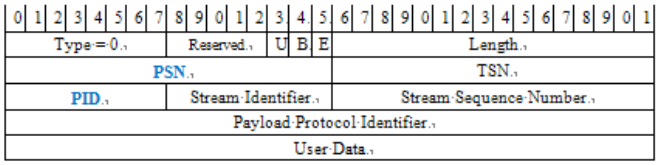

圖 4、修改後的 DATA chunk

3.4 程式架構

在本研究中，我們藉由實作出 CMT 多路徑傳輸機制之研究，透過導入 A-ARD 之概念，進而實作基於 RWND 之 ARD-CMT 系統，提供 ARD-CMT SCTP 使用者在擁有多張網路介面卡的狀況下，將能夠基於接收端 RWND 來分配路徑封包的順序，並且達到負載均衡之目的。

圖 5 為 ARD-CMT 系統流程，當應用程式啟動時，建立連線後開始需要讀取緩衝區大小的資料，並存到緩衝區中，在傳輸資料前，程式會使用 pthread_mutex()建立多線程，讓 SCTP 的傳輸程式達到多路徑傳輸，當路徑可以進行傳輸時，將會計算目前所有路徑的平均傳輸速率，並判斷此路徑上次計算權重時間是否已超過 200 ms，在 RFC 4960[13]所描述，當收到任何 DATA chunks 後，在 200 ms 時間之中會回傳 SACK chunk，因為我們這邊設定計算權重也是用 200 ms 為一個週期，去執行 SACK Process Module 取得 a_rwnd 值，並記錄到 Location Info Management 的結構中，之後開始使用 A-ARD 演算法計算其權重，並由 Queue Management 將其設定每各位址的權重比例，讓每一條路徑的傳輸資料程式依這比例來傳輸資料；在 user space 開始將資料送到 kernel space 後，kernel 會讀取其路徑位址，在標頭的程式會讀取我們強制指定的目標位址去傳輸資料，並將資料封裝到 DATA chunk 的 payload 中，設定 PSN 和 TSN 的流水序號，然後在比對目的地位址的 ID 後代入 PID 欄位，最後封裝成 DATA chunk 使用 Packet Dispatcher 往下層發送。

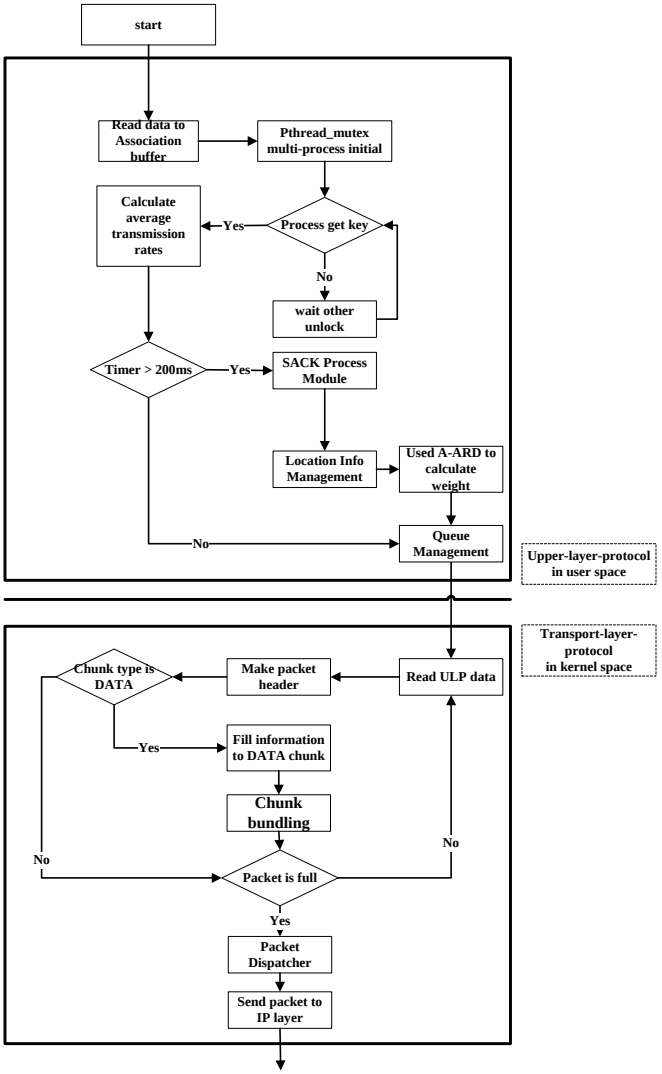

圖 5、ARD-CMT 系統流程圖

四、實驗方法與結果

本實驗的目的是觀察我們提出的系統是否能提升其傳輸效能，基於 RWND 封包分配演算法是否能在路徑情況惡劣時，能夠利用演算法來減低傳輸效能的下降，進而達到負載平衡之目的。圖 6 為本實驗的基本網路環境架構圖，

在兩端的 Host 都安裝了我們所實作的 CMT 模組，並且都各自配備三張網路介面，當 Linux 作業系統掛載 CMT 模組後，將可以使用所有網路介面進行傳輸。本實驗在兩台主機之間分別使用三條路徑作為資料的傳輸，各路徑的 buffer 都是設定 1024K，中間有三台 Network Emulator 主機模擬成路由器，可以產生實際環境中因為封包擁塞的封包遺失率跟封包延遲等情境，另外當沒有使用 CMT 模組時，標準 SCTP 程式將會使用 Interface 1 當作主要路徑，其餘的路徑做為備援路徑。

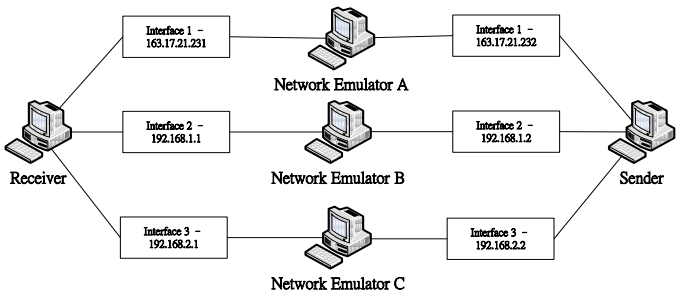

圖 6、網路環境架構圖

表 1、實驗設定規格表

伺服器(Endpoint)：	
CPU	Intel(R)Pentium(R)4CPU3.00GHz
RAM	1GB
NIC1	Intel Corporation 82546GB Gigabit Ethernet Controller
NIC2	Intel Corporation 82546GB Gigabit Ethernet Controller
NIC3	Intel Corporation 82573E Gigabit Ethernet Controller
作業系統版本	Ubuntu 8.10
核心版本	2.6.27
函式庫	LKSCTP 1.0.8
程式語言	C 語言
伺服器(Router)：	
CPU	Intel(R)Pentium(R)4CPU3.00GHz
RAM	1GB
NIC(s)	Intel 82573E Gigabit Ethernet *2
網路模擬軟體	NIST NET 2.0.12
實驗參數：	
傳輸速率	2.5MB
檔案大小	100MB
payload 大小	1040 bytes
實驗次數	30 次

表 1 為設備規格表，其中伺服器(Endpoint)配備 3 張網卡來傳輸和接收資料，伺服器(Router)利用兩張網卡模擬成路由器轉送資料。本論文於 Host A 依照其 buffer 的大小來傳輸 100 MB 之影像檔案至 Host B，分為以下三種情況，第一種情況是使用標準 SCTP 進行傳輸，僅使用 Interface1 為主要路徑連線至 Host B 進行傳輸資料；第二種情況為在 Host A 端點使用 CMT 模組，並且使用資料平均法進行資料傳輸；第三種方法則是使用 ARD-CMT 封包分配演算法來看當在不同的實驗環境，我們所提出的演算法是否可以減少傳輸效能的降低。本論文分別實驗上述三種情況，我們在各種實驗環境中，大致上的實驗方法如下：

1. 分別實驗 Host A 各路徑擁有同樣以及不同的 RWND 分別傳輸 100 MB 之檔案至 Host B，使用封包分配與多條路徑的效能分析。
2. 分別實驗 Host A、B 之間的所有路徑擁有不同的封包延遲時間，由 Host A 傳輸 100MB 的檔案至 Host B，所有傳輸機制的效能分析。

為了觀察我們實作的系統是否達到多路徑傳輸機制，我們實驗環境參數設定如表 2 所示，分別在接收端所有路徑上設定同樣的接收端 Buffer Size，然而在實際環境中接收端的 RWND 不可能全都一樣，因此當接收端的 RWND 都有差距時，我們提出的基於 RWND 的 ARD-CMT 傳輸機制將預期提升其傳輸效能，並達到負載均衡之目的，我們另外實驗了當接收端 Buffer Size 相異的情境，觀察我們提出的 ARD-CMT 可以基於 RWND 達到路徑分配，最後並實驗各傳輸機制的 association throughput (association throughput 表示傳輸機制在關聯中所有路徑 throughput 的加總)，並分析其效能。

表 2、以 recv buffer 為主的實驗參數表

Scenario 1	Interface 1	Interface 2	Interface 3
same buffer size	1024Kb	1024Kb	1024Kb
diff buffer size	512Kb	768Kb	1024Kb

實驗結果如圖 7 所示，我們可以觀察到當接收端每一條路徑的 RWND 都一樣時，使用多路徑傳輸的 ARD-CMT 和 CMT 傳輸機制明顯優

於標準 SCTP 的傳輸機制，但是使用封包排程機制的 ARD-CMT 因為增加了判斷 RWND 的功能和 A-ARD 權重的運算，因此實驗過程中發現，ARD-CMT 傳輸機制和 CMT 傳輸機制的傳輸效能幾乎相近，因為接收端的 RWND 都是一樣，所以所有路徑所得到的權重也幾乎相同，如表 4-5 描述所有傳輸機制的平均速率，誤差約 1%。

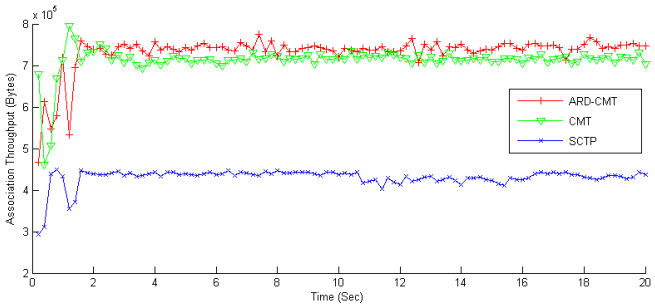

圖 7、同樣 RWND 的傳輸效能比較圖

圖 8 表示在接收端每一條路徑的 RWND 不一樣的情況下，各傳輸機制的傳輸效能比較圖，X 軸為傳輸的時間，Y 坐標軸為傳輸機制以每秒時間為單位的平均傳輸速率，但是因為標準 SCTP 並沒有提供多路徑傳輸，因此標準 SCTP 的傳輸效能比起其他兩個傳輸機制是較不佳的，而 CMT 的平均資料傳輸機制並沒有像我們所提出的演算法可以基於 RWND 來分配封包順序，如表 4-7 所示，我們提出的 ARD-CMT 進行傳輸時，在傳輸過程中，可以將路徑負擔較多的封包轉到路徑負擔較少的，因此比起 CMT 平均資料傳輸機制相對提升其傳輸效能 11.3%左右，並達到負載均衡之目的。

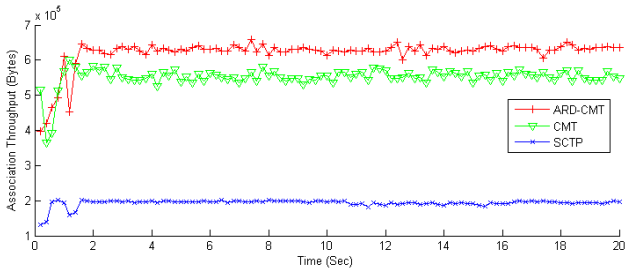

圖 8、接收端不同 RWND 的傳輸效能比較圖

在實際網路環境中，除了路徑擁有不同的 Receive buffer 情況會發生之外，通常也會因為網路擁塞造成的 delay，而導致接收端的計時器超出時間，而判斷 DATA chunk 遺失等情況。在 RFC 4960[13]的 6.2 節中描述標準 SCTP 的每一個 DATA chunks 在送出後，如果在 1 秒時間內沒有回傳 SACK chunks，則要重送尚未回應的 DATA chunks，當路徑 delay 越嚴重，單位時間內的 RWND 值將會越少，而我們也將預期基於 RWND 的 ARD-CMT 能夠在路徑之間達到負載均衡之目的。如表 3 的環境參數，我們分別假定在通常的延遲狀況以及在高延遲的狀況下，由於當 delay 的差距到達標準制定的 RTO 門檻值時，因為有些路徑都面臨重傳，所以我們所提出基於 RWND 的 ARD-CMT 傳輸機制可否如預期的減少 delay 所造成的 Throughput 下降。此實驗的主要目的是觀察基於 RWND 的 ARD-CMT 傳輸機制可以降低 delay 所造成的傳輸效能的下降。

表 3、以 Delay 為主的實驗參數表

Scenario 2	Path 1	Path 2	Path 3
similar Delay	40ms	20ms~60ms	20ms
dissimilar Delay	400ms	200ms~600ms	200ms

在此實驗中，我們為了比較在每一條路徑擁有不同delay且RWND同樣大小的情況下，因為封包延遲回應而造成RWND延遲成長，因此，當我們的ARD-CMT系統每200 ms計算一次權重時，將能夠基於RWND使封包傳輸到延遲較少的路徑，進而提升傳輸效能；如圖9表示所有傳輸機制在所有路徑擁有不同delay的傳輸效能比較圖，X為傳輸時間，Y軸是每一個association的平均傳輸速率，從圖中可以發現，每一台Network Emulator設定不同的delay時，標準SCTP 使用單一路徑透過Network Emulator A的路徑進行傳輸，且路徑的delay為40ms，因此傳輸速率明顯低於其他兩種傳輸機制，而使用

資料平均方法的CMT傳輸機制無法像ARD-CMT傳輸機制可以藉著RWND偵測delay，因此無法將封包流向delay較少的路徑，相較之下ARD-CMT傳輸機制更能有效的達到負載均衡，且ARD-CMT傳輸機制跟CMT傳輸機制相對可以有效的增加21.5%平均傳輸速率，而避免了因為delay所造成的Throughput下降。

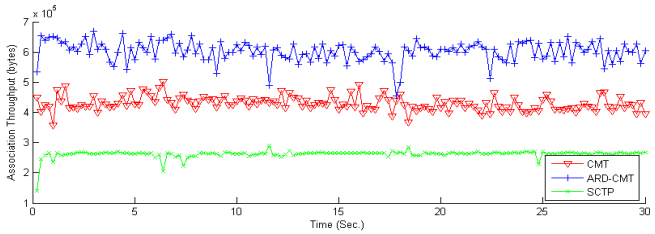

圖 9、不同 delay 的傳輸效能比較圖

接著我們為了觀察當路徑的延遲達到接近重送的條件下，我們提出的基於 RWND 的 ARD-CMT 是否可以達到負載均衡，並顯示比其他人有更好的 throughput，圖 10 表示所有傳輸機制在所有路徑擁有嚴重 delay 的傳輸效能比較圖，X 為傳輸時間，Y 軸是協定的 Throughput，因為標準制定 RTO.mini=1 秒，第一條路徑已經趨近於幾乎都在重傳狀態，而使用我們基於 RWND 的 ARD-CMT 演算法整體會選擇 RWND 較大的，所以平均速率都會趨近於穩定。而 CMT 因為沒有封包排程的機制，無法判斷哪一條路徑的 delay 較大，所以可以發現整體來說他的傳輸效能是相對較小的。

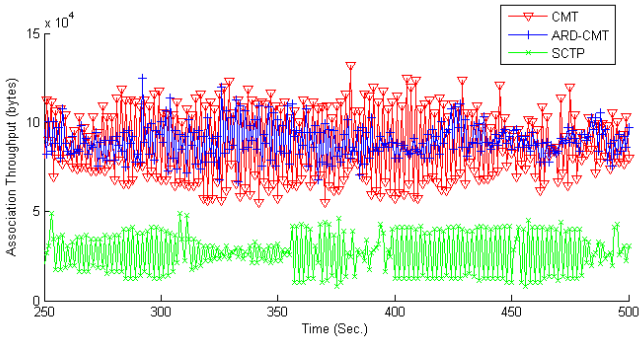

圖 10、嚴重 delay 的傳輸效能比較圖

五、結論與未來展望

隨著網路的負載與日俱增，如何充分且合理使用頻寬資源一直是一門重要的議題。在本論文中，我們修改了 SCTP 標準的架構，設計與實作多路徑傳輸機制，使得 SCTP 能夠有效利用頻寬資源，達到提升傳輸效能之目的；在權重分配的部分，我們在 CMT 架構下應用本實驗室提出的 A-ARD 演算法，提出了基於 RWND 的 ARD-CMT 傳輸機制，因應各路徑接收端的接收能力和發送端的平均速率來計算並給予不同的權重，由於 ARD-CMT 能利用 RWND 值有效偵測路徑的傳輸情況，因此可以避免路徑環境的干擾因素導致傳輸效能下降。藉由在有線及無線網路上實驗的結果，也驗證了使用我們提出的多路徑傳輸機制可以比標準的 SCTP 擁有更佳的傳輸效能。而在各種實驗干擾下，我們所提出的 ARD-CMT 也比一般 CMT 使用的 RR 分配，擁有明顯的改善效果。

在未來我們將朝著以下幾點繼續研究探討：(1)在傳輸能力有限的情況下，各路徑的壅塞控制分配機制，以改善傳輸效能 (2)當產生封包遺失的狀況時，ARD-CMT 依然會依照各路徑的權重值讓封包重新進入序列，此時依然會有機率會進入到發生遺失的路徑，因此我們勢必要將包遺失率也納入排程時候的重要考量 (3)在無線的網路環境中，並未考慮基地台的最佳化選擇，因此如果能夠考慮加進基地台的競爭機率，將可以有效的讓 ARD-CMT 在無線網路環境中有更好的傳輸效能。

六、致謝

本計劃之執行承蒙國科會計畫(編號：NSC 97-2221-E-142-004 及 NSC 98-2221-E-142-003)之支持，特此致謝。

參考文獻

[1] 戴丞峰，“Adaptation Load Balancing Scheme Using SCTP”，碩士論文，朝陽科

技大學資訊管理研究所，台中，2005.

[2] A. Abd El Al, T. Saadawi, Myung Lee, "Bandwidth aggregation in stream control transmission protocol", Ninth International Symposium on Computers and Communications, pp. 975 - 980, 2004.

[3] Chung-Ming Huang, Ching-Hsien Tsai, "WiMP-SCTP: multi-path Transmission using Stream Control Transmission Protocol (SCTP) in Wireless Networks", IEEE International Symposium on Frontiers in Networking with Applications, pp. 209-214, 2007.

[4] C. Casetti, W. Gaiotto, "Westwood SCTP: load balancing over multipaths using bandwidth-aware source scheduling", IEEE Vehicular Technology Conference, pp.3025 - 3029, 2004.

[5] F. Perotto, C. Casetti, G. Galante, “SCTP-based Transport Protocols for Concurrent multipath Transfer”, IEEE Wireless Communications and Networking Conference, pp.2969-2974, Mar. 2007.

[6] J. Postel, “Transmission Control Protocol,” IETF RFC 793, September 1981.

[7] J. Postel, “User Datagram Protocol,” IETF RFC 768, August 1980.

[8] Jianxin Liao, Jingyu Wang, Xiaomin Zhu, "cmpSCTP: An Extension of SCTP to Support Concurrent multi- Path Transfer", IEEE International Conference on Communications, pp.5762 - 5766, May 2008.

[9] J. R. Iyengar, P. D. Amer, R. Stewart, “Concurrent multipath Transfer Using SCTP multihoming Over Independent End-to-End Paths”, IEEE/ACM Transactions on Networking, Volume 14, Issue 5, Oct. 2006.

[10] L.H. Chang, D.J. Wang and K.C. Lai, "Network Gateway Design and Implementation Using Dynamic Load Balancing", Journal of Internet Technology, v5, No.1, pp. 19-25, 2004.

[11] M. Fiore, C. Casetti, "An adaptive transport protocol for balanced multihoming of real-time traffic", IEEE Global Telecommunications Conference, 2005.

[12] R. Stewart, M. Ramalho, Q. Xie, M. Tuexen, P. Conrad, "Stream Control Transmission Protocol (SCTP) Partial Reliability Extension", RFC3758, May 2004.

[13] R. Stewart and Q. Xie et. al., “Stream Control Transmission Protocol,” IETF RFC 4960, Sep. 2007.