TCP 擁塞控制演算法模擬器

連耀南　黃郁翔　黃冠傑

國立政治大學資訊科學系

Email: lien@cs.nccu.edu.tw

摘要

TCP/IP 協定是現在的網際網路中的主要命脈。其重要功能之一是控制傳送速率避免網路擁塞，每個傳送端依據它們各自所感受到的網路擁塞程度，限制自己將資料送入連線的速率。方法看似簡單，但因網路之擁塞控制極為不易，以致擁塞控制演算法的細節極為細緻複雜，初學者往往需要花上大量的時間去理解，不但耗時費力，更常導致錯誤的理解，複雜的控制機制更阻礙了更佳演算法的創新。

本研究使用了一個能自動產生具 GUI 互動能力的模擬器的軟體—FSMGEN 製作出一個 TCP 擁塞控制演算法的模擬器，讓使用者可以藉著操作模擬器深入瞭解 TCP 的擁塞控制機制。於設計新創擁塞控制演算法時，更可利用 FSMGEN 產生模擬器，實際操作進行初步驗證與除錯。使用者只需將 TCP 擁塞控制演算法的 State 和 Event 等資訊以簡單的文字格式描述作為輸入，即可自動模擬 TCP 擁塞控制演法。所產生的模擬器包含 HTML 和 Javascript 程式碼，使用者可利用一般的網頁瀏覽器操控並觀察其狀態變化。藉由互動式的模擬過程，使用者可以深切體驗狀態轉移的細節，因而更細緻地瞭解 TCP 擁塞控制演算法中有限狀態機的流程運作。

關鍵詞：FSM、TCP 擁塞控制。

1. 簡介

1.1 TCP擁塞控制

傳輸控制協定（Transmission Control Protocol, TCP），必須使用端點到端點的擁塞控制機制，而非網路協助的擁塞控制機制，因為IP層並不會提供終端系統有關網路擁塞狀況的明確回饋資訊。而TCP擁塞控制所採用的方式，是讓各傳送端依據它們所感受到的網路擁塞程度，限制自己將資料流送入連線的速率。如果TCP傳送端察覺到自己跟目的端之間的路徑上幾乎沒有擁塞情形，TCP傳送端就會增加其傳送速率；而TCP傳送端若察覺到路徑上有擁塞的情形，TCP傳送端會降低其傳送速率。TCP傳送端要如何限制自己將資料流送入連線的速率？TCP連線的兩端各有一份接收緩衝區、一份傳送緩衝區、以及數個變數例如LastByteRead、rwnd等(receive window，TCP會藉由讓傳送端維護這個變數，來提供流量控制)。傳送端運作的TCP擁塞控制機制會追蹤一個額外的變數，擁塞窗格(congestion window)，以cwnd表示，並根據cwnd限制TCP傳送端將資料區塊(例如：封包)送入網路的速率。明確的說，傳送端所送出的未經確認資料量，不得超過cwnd與rwnd兩者中較小值：

$$\text{LastByteSent} - \text{LastByteAcked} \le \min\{\text{cwnd}, \text{rwnd}\}$$

為了簡單說明擁塞控制機制，我們假設TCP的接收緩衝區都大到讓接收窗格的限制可以被忽略；因此，傳送端未經確認的資料量只會受到cwnd的限制。此外我們也假設傳送端永遠有資料需要傳送，也就是說，所有擁塞窗格中的資料區塊（簡而言之；封包）都會被傳送。上述條件會限制傳送端未經確認的資料量，而因此間接地限制了傳送端的傳送速率。如此一來，在每一輪封包的傳送之開始，上述限制就會允許傳送端送出cwnd個MSS(maximum segment size)到連線中；而在經過RTT(Run Trip Time)時間後，傳送端則會收到這些資料的確認訊息。因此，傳送端的速率大約等於cwnd/RTT bytes/sec。藉由調整cwnd值，傳送端便可以調整將資料送入連線的速率。

TCP傳送端要如何察知自己和目的端之間路徑上的擁塞狀況？很多版本的TCP將逾時未收到確認(ACK)事件，或收到三次重複的ACK視為封包遺失。在發生嚴重的擁塞情況時，傳輸路徑上的路由器會發生緩衝區溢位，導致封包被丟棄，此即遺失事件，傳送端可能因此逾時未收到ACK或收到三次重複的ACK。此外，當封包耽誤太久也會觸發類似情況，但傳送端無從判斷真實情況，只能視之為「從傳送端到接收端路徑上發生擁塞」的徵兆。擁塞控制對於網路的正常運作極為重要。沒有擁塞控制，網路就會很容易癱瘓，端點到端點之間就只能傳輸極少的資料，甚至無法傳輸資料。網路之擁塞控制極為不易，以致擁塞控制演算法的細節極為細緻複雜，初學者往往需要花上大量的時間去理解，更有甚者，常導致錯誤的理解，複雜的控制機制更阻礙演算法的創新。

本研究的目的在於讓使用者透過TCP Simulator，快速且精確的瞭解TCP擁塞控制演算各個狀態之間的變化。藉由互動式的模擬過程，使用者可以深切體驗狀態轉移的細節，因而更細緻地瞭解TCP擁塞控制演算法有限狀態機的流程運作，以達到最完整的學習效果。[4]

2. TCP 擁塞控制演算法

TCP擁塞控制演算法，此演算法首見於[3]，在[2]中得到標準化。此種演算法有三項主要狀態，分別為：
(1) Slow Start (SS)
(2) Congestion Avoidance (CA)
(3) Fast Recovery (FR)
SS、CA都是TCP必須執行的元素，兩者的差異只在於回應收到ACK時它們增加cwnd大小的方式。相較於CA，SS會較快速地增加cwnd的大小。FR對於TCP傳送端則是建議性的，而非強制性的。

2.1 Slow Start

在TCP開始建立連線時，cwnd值通常會被初始化為某個小數值 1 MSS(maximum segment size) [1]，使得初始的傳送速率大約是MSS/RTT。由於TCP傳送端可以取得的頻寬可能遠大於MSS/RTT，所以TCP傳送端會想要快速地得知可取得的頻寬數量為何。因此，在Slow Start狀態中，cwnd會從 1 MSS開始，每當所傳輸的封包得到第一次確認時，便增加 1 MSS。圖 2 的例子中，TCP會將第一筆封包送入網路，然後等待確認。當這筆確認訊息抵達時，TCP傳送端便會將其擁塞窗格增加 1 MSS，然後送出兩個MSS封包。接著這兩份封包也得到確認，於是TCP傳送端會針對兩筆得到確認的封包都各將擁塞窗格增加 1 MSS，從而得到大小為 4 MSS的擁塞窗格，依此類推。此一過程會造成每回RTT中傳送速率都會倍增。因此，在Slow Start階段中，TCP的傳送速率會從低速開始，但是呈指數性地成長。但是這種指數性的增長要到何時才會結束？首先，如果發生了由逾時事件所指示的遺失事件，亦即擁塞狀況，TCP會將cwnd值驟降為 1，然後重新開始Slow Start程序，同時也會將第二個狀態變數，ssthresh (slow start threshold)的數值設定為cwnd/2。第二種終止Slow Start 的時機，跟ssthresh的數值有直接密切的相關。因為ssthresh等於上次偵測到擁塞狀況時cwnd值的一半，所以當cwnd值達到或超過ssthresh值時，還持續將其倍增，可能會再次塞爆網路。因此，當cwnd值等於ssthresh值時，Slow Start會終止，TCP會轉轉換進入Congestion Avoidance。TCP在Congestion Avoidance模式中，會更加謹慎地增加cwnd。最後一種會終止Slow Start的情況是偵測到三次重複的ACK時，(接收端接到跳序的封包時，會重覆送回上次的ACK)；在此情況下TCP會執行Fast Retransmit然後進入Fast Recovery。[4]

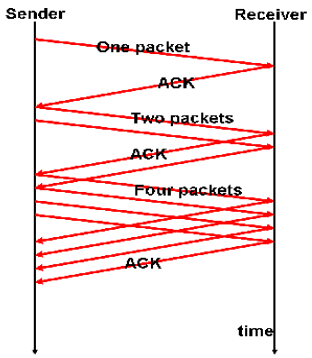

圖2 TCP建立連線過程

2.2 Congestion Avoidance

由圖 3 可看出在進入 Congestion Avoidance 狀態時，cwnd 值大約等於上次遇到擁塞狀況時 cwnd 值的一半，擁塞隨時都可能會發生。因此，相較於每回 RTT 都倍增 cwnd 值，TCP 會採取較為保守的策略，在每回 RTT 中只會將 cwnd 值增加一個 MSS[4]。現存有幾種方式可完成這項任務。其中一種常見的方法是讓 TCP 傳送端在每次有新的確認訊息抵達時，就將 cwnd 增加 MSS 個位元組 (MSS/cwnd)。假設 MSS 等於 14,600 位元組，則在每一回 RTT 中會有 10 筆封包被送出。每筆抵達的 ACK 都會將擁塞窗格的大小增加 1/10 MSS，因此在收到 10 筆封包的 ACK 後，擁塞窗格的數值便會增加 1 MSS。然而，在發生逾時事件時，TCP 的 Congestion Avoidance 演算法會採取相同的行為。就像 Slow Start 的狀況一樣：cwnd 值會被設定為 1 MSS，ssthresh 值則會備更新為當發生遺失事件時 cwnd 值的一半。然而，遺失事件也可能是由三次重複 ACK 事件所觸發的。在此狀況下，由收到重複的 ACK 可知網路仍然能從傳送端投遞封包到接收端。因此在這種類型的遺失事件發生時，TCP 的行為應該要比面對由逾時事件所指示的遺失事件時要來得緩和一點。TCP 會將 cwnd 值減半並且再加 3 個 MSS，然後將 ssthresh 值紀錄為收到三次重複 ACK 時 cwnd 值的一半。接著 TCP 的狀態便會由先前的狀態進入 Fast Recovery 狀態。[4]

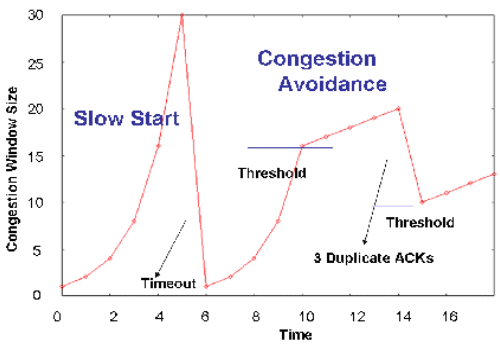

圖3 TCP擁塞控制狀態圖

2.3 Fast Recovery

在 Fast Recovery 狀態下，每次收到代表遺失封包須要重送的重複 ACK，cwnd 的值變會增加 1 個 MSS。最後，當遺失封包的 ACK 抵達時，TCP 便會降低 cwnd 之後進入 Congestion Avoidance 狀態。如果發生逾時事件，Fast Recovery 狀態則會在採取與 Slow Start 及 Congestion Avoidance 狀態相同的舉動之後，進入 Slow Start 狀態。cwnd 值會被設定為 1 MSS，ssthresh 值則會被設定為遺失事件發生時 cwnd 值的一半。而此狀態是非強制性的 TCP 元件[2]。在 TCP Tahoe 的早期 TCP 版本中，不管是在由逾時事件所指示的遺失事件，或是在由三次重複 ACK 所指示的遺失事件之後，都會無條件的將擁塞窗格砍至 1 MSS，然後進入 Slow Start 階段。新版本的 TCP，TCP Reno，則加入 Fast Recovery。 [2，4]

圖4 TCP擁塞控制演算法狀態圖

3. 用 FSM 模擬 TCP 擁塞控制

雖然 TCP 擁塞控制的運作，可以透過 FSM 來描述。然而一般 TCP 擁塞控制演算法的 FSM 常常是詰屈聱牙，初學者常常須要長時間的研讀才能徹底瞭解。但不幸的是，很多人的疑惑都是源於對於個個狀態間轉換的不瞭解。因此如果能開發一個具 GUI 互動能力的模擬器，將有助於初學者快速瞭解 TCP 擁塞控制演算法，且在學習的過程中能輕易瞭解各個不同的狀態下，不同的事件會對應到哪些不同的對應動作。

本研究使用了一個能自動產生具 GUI 互動能力的模擬器的軟體—FSMGEN 製作了一個 TCP 模擬器，本系統的使用非常簡單，使用者只需將所設計的有限狀態機以簡單的文字格式描述作為輸入，本系統即可自動產生模擬器。所產生的模擬器包含 HTML 及 Javascript 程式碼，使用者可利用一般的網頁瀏覽器操控並觀察其狀態變化。藉由互動式的模擬過程，使用者可以深切體驗狀 態轉移的細節，因而更細緻的瞭解有限狀態機的運作。雖然本系統所產生的模擬器並未能畫出動態的狀態圖，但其所具有的 GUI 及互動能力，已經足以協助初學者一個淺顯易懂的 TCP 擁塞控制流程及架構圖，學習 TCP 擁塞控制演算法的使用者能夠簡單地瞭解其中複雜的運作流程。

3.1 FSMGEN

系統特點：

- 流程簡化—使用者只要將狀態機包含的狀態(state)，事件(event)與輸出動作(action)撰寫成標準的文字檔案，作為輸入檔案。
- 圖形介面—此工具產生的檔案為標準 HTML 網頁檔案，並採用 Javascript 方式執行程式。此網頁圖形介面可於任何網頁瀏覽器瀏覽操作，無預安裝額外程式，且可放置於網頁伺服器，可供多人同時於伺服器執行。
- 操作簡單，即時呈現—使用者操作時只需點選滑鼠，即可隨時看到狀態機即時變化，不需輸入特殊指令，無預記憶學習。
- 支援階層式架構—每個狀態中允許包含子狀態機(sub-FSM)，只需使用者按照子目錄方式建立輸入檔案，產生器即可自動完成子狀態機間的連結。
- 彈性化—使用者操作狀態機模擬器的過程中，若對當前的執行狀態不滿意，可隨時回到上一步，不需從頭開始，節省時間。
- 完整紀錄—使用者可隨時點選顯示完整流程紀錄，檢視整個流程是否符合需求。

輸入格式：
輸入檔案以使用ASCII(英文)或 Big5(中文)編碼純文字檔。檔案內容欄位格式如下:

```
_state   狀態名稱 1
_event   事件 1_1    下一狀態    輸出 1_1
_event   事件 1_2    下一狀態    輸出 1_2
_state   狀態名稱 2
_event   事件 2_1    下一狀態    輸出 2_1
_event   事件 2_2    下一狀態    輸出 2_2
```

檔案中，以底線(_)開頭的_state 與_event 是關鍵字標籤，其餘欄位用途與說明如表 1：

表 1 FSMGEN 輸入欄位說明

欄位	用途	說明
狀態名稱	FSM 內之 state 宣告	同一輸入檔案中，狀態名稱不可重複
事件	此 state 內能觸發的 event 名稱	在某一狀態下，可觸發的事件(event)，不同狀態下可容許觸發同一事件
下一狀態	事件觸發後，將轉移至下一個狀態的名稱	此狀態名稱必須使用於同一檔案內有定義過的狀態
輸出	事件觸發後，所應採取的 action 之說明	在該狀態下，發生特定事件後所對應的輸出動作之描述，可為任意文字敘述

3.2 實作成果

```
state Slow_Start
event duplicate_ACK(if_dupACKcount<3) Slow_Start cwnd=cwnd+1,dupACKcount++
event new_ACK Slow_Start cwnd=cwnd+1,dupACKcount=0
event timeout Slow_Start ssthresh=cwnd/2,cwnd=1 MSS,dupACKcount=0
event new_ACK(if_cwnd>=ssthresh) Congestion_Avoidance terminate_Slow_Start
event dupACKcount==3 Fast_Recovery
ssthresh=cwnd/2,cwnd=ssthresh+3,execute_Fast_Retransmit

state Congestion_Avoidance
event new_ACK Congestion_Avoidance cwnd=cwnd+1*(MSS/cwnd),dupACKcount=0
event duplicate_ACK(if_dupACKcount<3) Congestion_Avoidance
cwnd=cwnd+1*(MSS/cwnd),dupACKcount++
event timeout Slow_Start ssthresh=cwnd/2,cwnd=1 MSS,dupACKcount=0
event dupACKcount==3 Fast_Recovery
ssthresh=cwnd/2,cwnd=ssthresh+3,execute_Fast_Retransmit

state Fast_Recovery
event duplicate_ACK Fast_Recovery cwnd=cwnd+1
event timeout Slow_Start ssthresh=cwnd/2,cwnd=1,dupACKcount=0
event new_ACK(recovered_the_lost_packet) Congestion_Avoidance
cwnd=ssthresh,dupACKcount=0
```

圖 5 TCP 擁塞控制程式碼

圖 5 是輸入於 FSMGEN 的 TCP 擁塞控制狀態機的程式碼，產生的 Simulator 如圖 6、7、8 所示，透過 TCP Simulator，即可針對三個主要狀態觸發任一事件驅動 Simulator 進行狀態轉移，讓初學者能夠輕鬆的瞭解 TCP 擁塞控制演算法其狀態之間的流程。

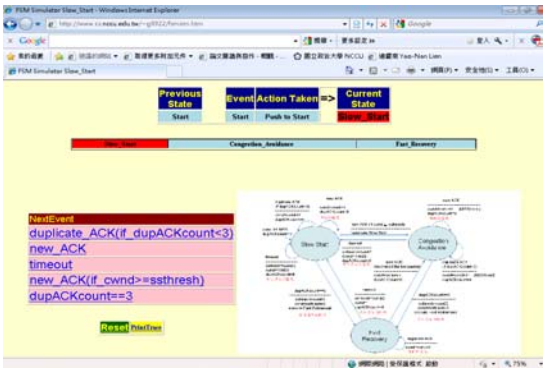

圖 6 TCP Simulator

http://www.cs.nccu.edu.tw/~g9922/fsmsim.htm

經由上述網址選取 Start 之後，開始進入 TCP 擁塞控制演算法的模擬，圖 7 是狀態網頁，其中：

- Previous State—前一個狀態。
- Event—所觸發的事件。
- Action Taken—事件觸發後，所應採取的動作。
- Current State—目前的狀態。

圖 7 TCP Simulator 狀態網頁

圖 8 是觸發事件網頁，使用者可以透過選取不同的 NextEvnt 來觸發不同的事件。

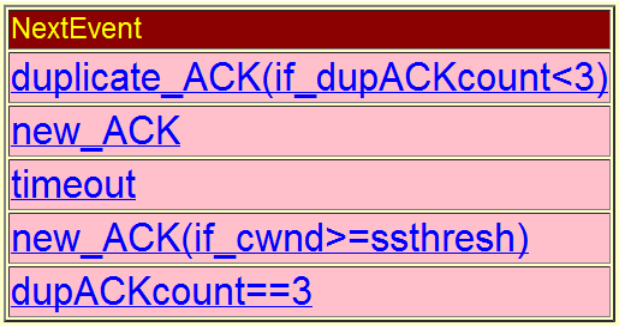

圖 8 TCP Simulator 觸發事件網頁

透過選取不同的觸發事件，使用者可以驅動 Simulator，逐步進行狀態轉移，使用者可以清楚的瞭解，不同的觸發事件會採取哪些不同的動作，而狀態又會是如何的改變。使用 Simulator，就像程式語言開發環境中的 Debugger，可以一步一步的驅動程式往前執行，使用者可以輕易的了解程式執行中的每一步細節。

4. 結論

TCP 擁塞控制演算方法背後的細節其實是很複雜的，初學者往往都要花上大量的時間去理解，於新創演算法時也造成很大的困擾。因此本研究透過有限狀態機(Finite State Machine, FSM)模擬器，提供初學者一個淺顯易懂的 TCP 擁塞控制流程及學習工具，讓初學者能夠輕鬆的瞭解 TCP 擁塞控制演算法。

使用者可利用一般的網頁瀏覽器操控 TCP Simulator，觀察 TCP 擁塞控制演算各個狀態之間的變化。藉由互動式的模擬過程，使用者可以深切體驗狀態轉移的細節，因而更細緻地瞭解 TCP 擁塞控制演算法有限狀態機的流程運作，以達到最完整的學習效果，也有助於新創演算法時的驗證與除錯。

參考文獻

[1] M.Allman, S. Floyd and D. Partridge, “Increasing TCP’s Initial window,” RFC 3390, Oct. 2002.

[2] M.Allman, V.Paxson and W.Stevens,”TCP Congestion Control,”RFC 2581,Ape.1999.

[3] V.Jacobson,”Vongestion Avoidance and Control”,Proc. 1988 ACM SIGCOMM (Stanford,CA,Auug.1988).

[4] James F. Kurose and Keith W. Ross. Computer Networking: A Top-Down Approach 5/E

[5] Yao-Nan Lien and Li-Cheng Chi, "A Generator for Finite State Machine Simulators - FSMGEN", Proceedings of 22th Conference on Object-Oriented Technology and Applications (OOTA), Jul 8-9, 2011, Taipei, Taiwan, NSC 982A04.