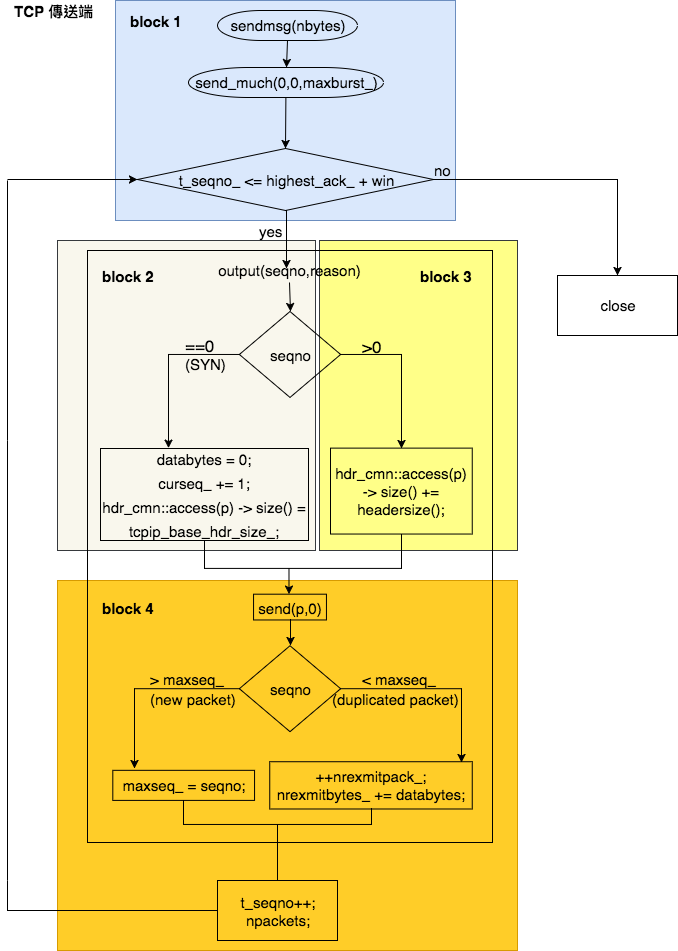


圖一.TCP: Connection to Application and Low-Level Network

說明：套用OSI模型的基礎，NS2也針對各個層提供相對應的方法讓大家去模擬使用，而這次我們是針對傳輸層中的TCP傳輸層做分析。



圖二.傳送端

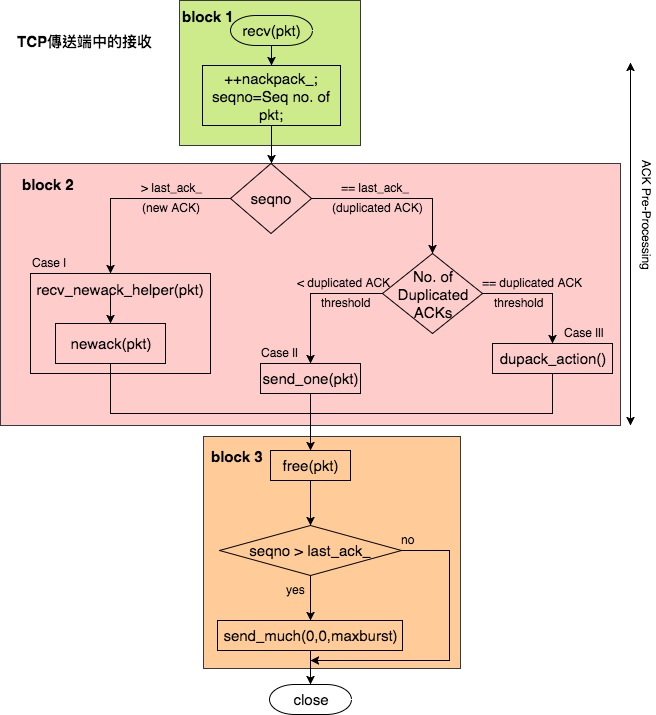
|  |  |  |
| --- | --- | --- |
| 函數名稱 | 介紹 | 備註 |
| sendmsg(nbytes) | 接收來自應用層的資料 | nbytes=-1時，表示允許發送的無限多數據 |
| send\_much(0,0,maxburst)  內部重要變數：  t\_seqno  highest\_ack\_  win\_  npackets | 用於在發送窗口範圍內把盡可能多的數據發送出去。  下一個要傳送的seqno  目前最大的ACK  windows窗口大小  本次發送分組數紀錄，每次執行send\_much()都會歸0 | maxburst是一次最多發送的分組數，默認為0，表示不限制數量。 |
| output(seqno,reason)  內部重要函式：  send(p,0)  內部重要變數：  maxseq\_ | 用於發送單個分组。被send\_much()（發送多個分組）、send\_one()（發送單個）等調用。  將封包往下層送  目前最大的seqno | seqno要發送的序號  reason表示重發原因 |

說明：

1. block 1主要是接收來自應用層(Application)送出的資料，sendmsg()函式是被上層應用所調用、用來將數據按照size的大小計算總共的序列數。

舉例:該筆資料串流包含一份大小為500,000個位元組的檔案，size為1000位元組，總共會有500份區段，curseq\_變數是儲存算出的最大分組號也就是500。send\_much(0,0,maxburst)，被recv()、sendmsg()等調用，用於在發送窗口範圍內把盡可能多的數據發出去，maxburst默認為0，表示不限制數量，(t\_seqno\_ <= highest\_ack + win && t\_seqno\_ < curseq\_)分組號在發送窗口內，且小於最大分組號，滿足條件就盡量發送，不滿足條件才退出。

1. block 2 主要是用來建立連線時傳送最一開始的SYN區段，當seqno=0代表當前要發送的序號是第一個分組時，設定一些有關SYN區段的內容。
2. block 3 主要是開始將每個區段加上IP+TCP的表頭。
3. block 4 藉由send(p,0)將封包往下層送，判斷是新封包或者是重傳的封包，新封包的話，將目前最大序號=發送序號，重傳的封包，只是統計重傳的分組數，並將下一個要發送的序列號+1。



圖三.TCP傳送端中的接收

|  |  |  |
| --- | --- | --- |
| 函數名稱 | 介紹 | 備註 |
| recv(pkt)  重要函式：  newack(pkt)  send\_one(pkt)  dupack\_action()  free(pkt)  send\_much(0,0,maxburst)  內部重要變數：  seqno  last\_ack\_  dupacks\_  numdupacks\_ | 將接收到的封包做處理  newack()函數主要用於處理收到新ACK時紀錄的更新、RTT的更新等  只單傳一個封包，遇到需要重傳的封包時  快速重傳  回收該封包  接收處理完，繼續發送封包  來自pkt中的seqno  上一個ack序號  重傳的ack次數  接收到相同的ack次數上限，到達時開始執行快速重傳 | pkt接收到的封包 |

說明：

1. block 1 當接收到來自接收端的ack封包時就會進入recv(pkt)函式，nackpack只是統計ack的包數，seqno的值是來自pkt中的seqno。
2. block 2 檢查seqno如果大於上次的ack序號，表示是新的ACK，等於上次的ack序號，如果等於上次的ack序號，表示是重複ack，dupacks\_代表重傳的次數duplicated ACK預設是3，重傳次數如果等於3次，就執行快速重傳，小於3次就直接重傳。

Case l：newack()函數主要用於處理收到新ACK時紀錄的更新、RTT的

更新等。Pkt為接收ACK分组

Case ll：send\_one()收到一個或兩個重複ack，就重傳分組。

Case lll：在recv()中，用於收到3個重複ACK後的快速重傳，减少

CWND和SSTHRESH,並重啟重傳定時器。

1. block 3 處理接收完後的封包，free(pkt)函式回收該分組內存，判斷這次的ack序號是否大於或等於上次的ack，表示是新ACK或重複ACK，是合法的。繼續發送盡可能多的封包，也就是開始執行圖二中的send\_much(0,0,maxburst)函

式。