Computer Networking Project

施信宏 m10915045

林紹瑜 m10915050

一、Introduction ns3 - network simulator 3

**ns3(**network simulator 3)為一個離散網路模擬器，前代還包括**ns-1**、**ns-2**。是自由軟體，供社群的各個開發者和用戶更新，於2006年開發第一版。

ns3模擬大致可分為以下幾個步驟:

1. 拓撲定義：創建基本的設備，與其相關聯的裝置。
2. 模型使用：添加模型（例如UDP、IPv4、點對點設備和連結、應用）
3. 節點和連接配置：設置模型預設值（例如，發送的封包的大小和點對點連接的MTU值)。
4. 執行：模擬事件，使用者輸入相關數據。
5. 性能分析：在模擬完成後，可以根據內建的function追蹤封包及程式。
6. 圖形可視化：原始或處理過的數據能被gunplot、matplotlib等工具製成圖片，利於觀測。

二 、TCP Congestion Control

運作方法 :

TCP會為每條連接建立一個 congestion window 來限制在端對端間傳輸的未確認分組總數量。類似於TCP flow control 中使用的滑動窗口。TCP在一個連接初始化或逾時後使用slow start 機制來增加 congestion window 的大小。它的起始值一般為Maximum segment size(MSS) 的兩倍，雖然初始值不高，但其增長速度很快。當每個分段得到確認時，congestion window 會增加一個MSS，使得在每次round-trip time(RTT) 內congestion window能高效地增長。

當congestion window 超過 slow start threshold (ssthresh) 時，演算法就會進入congestion avoidance階段。在congestion avoidance階段，只要未收到重複確認(Dup ACK)，congestion window則在每次RTT內線性增加一個MSS大小。

此 project 對 4 種不同的TCP Congestion Control作探討，分別介紹TCP tahoe、TCP Reno、TCP NewReno及Westwood之特點。

TCP tahoe、TCP Reno : 演算法大致相同，對於丟包事件判斷都是以retransmission timeout(RTO)和重複確認為條件，但是對於重複確認的處理，兩者有所不同：

* Tahoe：如果收到三次重複確認——即第四次收到相同的分段確認，並且分段對應的封包無負載分段和無改變接收窗口，Tahoe則進入fast retransmit，將ssthresh改為當前congestion window 的一半，將擁塞窗口降為1個MSS，並進入slow start。
* Reno：如果收到三次重複確認，Reno演算法則進入fast retransmit，只將congestion window減半來跳過slow start，將ssthresh設為當前新的congestion window的數值，進入Fast recovery，是Reno演算法引入的方法，將遺失的分段重傳後，啟動一個timer，並等待該遺失分段包的分段確認後，再進入擁塞控制階段。如果仍然逾時，則回到slow start。

對於RTO，兩個演算法都是將congestion window降為1個MSS，然後進入slow start。

TCP New Reno : New Reno的快速恢復中，一旦出現3次重複確認，TCP傳送方先記下3次重複確認時已傳送但未確認的分段的最大序列號，然後重發重複確認對應序列號的分段包。如果只有該重複確認的分段遺失，則接收方接收該重發分段包後，會立即返回最大序列號的分段確認包，從而完成重發；但如果重複確認期間的傳送包有多個遺失，接收方在接收該重發分段後，會返回非最大序列號的分段確認包，從而傳送方繼續保持重發這些遺失的分段，直到最大序列號的分段確認包的返回，才登出快速恢復階段。

New Reno在低錯誤率時執行效率和「選擇確認」（Selective ACKnowledgement，SACK）相當，在高錯誤率仍優於Reno。

TCP Westwood : 改良於TCP NewReno，用以，相較於前面提到的三種協定，TCP Westwood對於cwnd的調整並不是使用loss做測量，而是制定了新的演算法-Agile Probing，來偵測bandwidth，藉此調整cwnd與ssthresh以使用更多的bandwidth，下面簡介一下TCP Westwood計算方式，TCP Westwood會在每個RTT後計算bandwidth，假設delta為經過的時間，完成的傳輸量為bk，則採樣值是bk / delta，並且像RTT一樣，經過一次平滑處裡才得出測量的bandwidth：

若 K = 1 :

bw\_ns\_est(1) = bk / delta ;

bw\_est(1) = bk / delta ;

K > 1 :

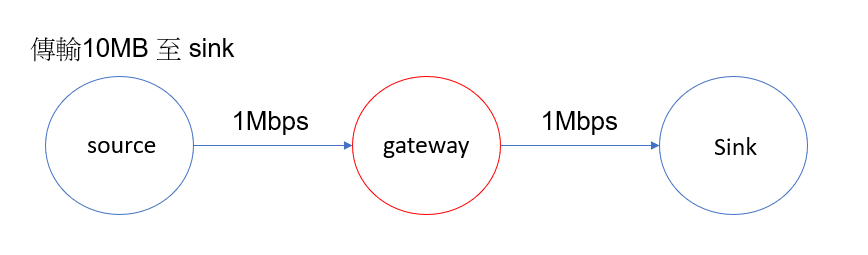




三、網路模擬情況

在模擬的部分，我們使用ns-3-24中的tcp-variants-comparison.cc檔案做測試，模擬了四種版本的TCP分別是Tahoe、Reno、NewReno與Westwood，設定了一些模型的參數如下：

| 通道bandwidth | 1Mbps |
| --- | --- |
| 通道delay | 0.01ms |
| access bandwidth | 10Mbps |
| access delay | 45ms |
| error rate | 0.1% |
| 傳送資料量 | 10MB |
| 監測時間 | 150s |



上圖為網路模擬的示意圖，從source node 傳輸10MB的資料至 sink node，其中通道所能傳輸的上限為1Mbps。最大監測時間為150秒，若尚未傳輸完畢，會強制關閉此次模擬，error rate為0.1%，若error rate 設為 0，則為完美情況，將不會造成封包遺失。

四、實驗結果與分析

在圖一中，我們展示cwnd在這模擬中的變化(前20秒)，可以看出，TCP Tahoe與TCP Reno的cwnd雖然控制方式類似，但是TCP Tahoe在loss後會重新啟動"slow start"的步驟，cwnd會從1重新開始，而TCP Reno則是進入"fast recovery"的部分，cwnd變為當前的ssthresh一半後，會以線性的方式增長，相較之下就可以以比較穩定的情況傳輸。

而TCP NewReno與TCP Reno相異的部分只是在”fast recovery"的部分，利用一個ACK來確認部分窗口，並且立刻傳送剩下的packet，主要是為了避免同個窗口丟失多個packet造成超時的問題，不過在我們模擬的情況，TCP NewReno與TCP Reno的差距非常小。

從圖一可以看出，在packet loss的之後，透過計算頻寬的方式，TCP Westwood可以從更大cwnd來執行壅塞避免的步驟，以20秒的情況為例，TCP Westwood的cwnd可以比Reno多出快百分之50的大小。

|  |
| --- |
| 圖一 Congenstion Window (前20秒的情況)  上至下分別為Tahoe、Reno、NewReno與Westwood |

圖二的部分，我們展示的是四種TCP協定的Slow Start Threshold的變化，可以看出的Tahoe, Reno與NewReno三者的變化大致上是相似的，都會在5000~10000左右震盪，然而Westwood與其他三者的差異就很大，基本上是一直維持在10000左右，從圖一也可以看到，每次發生loss之後，Westwood都是從10000左右再開始線性增長。

|  |
| --- |
| 圖二 Slow Start thresh |

最後我們就來看整體的傳輸結果，透過ns3中提供的flowmonitor工具，可以計算傳輸結果的TX與RX速率，下圖中分別展示了四種版本的傳輸結果，因為error rate是隨機設定的，所以在四種情況下都差不多。在TX的部分，我們可以看到再Tahoe中傳輸的速率是773.64 kbit/s，而Reno與NewReno則是相當接近的889.81 kbit/s與865.66 kbit/s，這裡所得到的結果NewReno反而比Reno來的慢的原因，可能是因為這裡的loss並不是真的頻寬壅塞所造成的，因此，NewReno所調整的快速恢復在這種情況下反而沒有提供作用。最後，可以看到Westwood的速度974.29 kbit/s已經相當接近模擬所設定的1Mbps。

