1. 實驗日期:

2020/5/14

1. 實驗名稱:

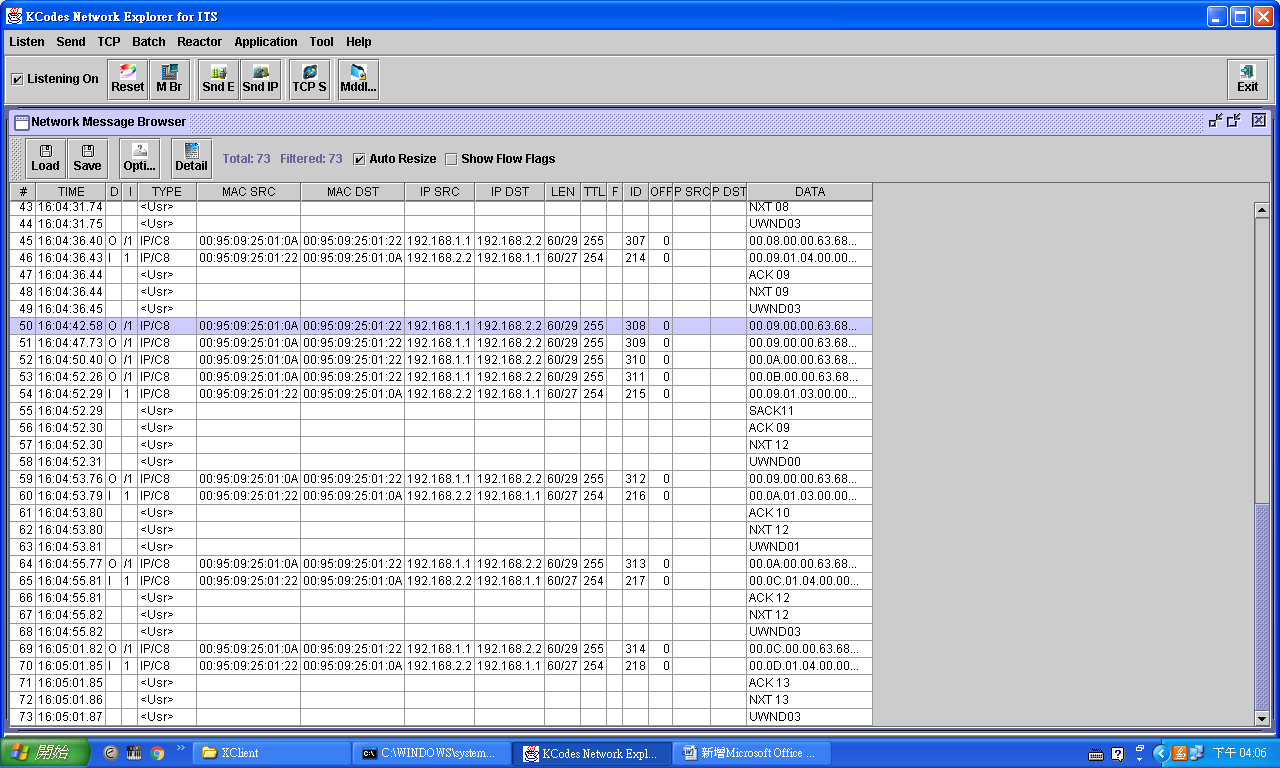
* ITS Exp. 12: Sliding Window滑動窗口
* ITS Exp. 13: Congestion Avoidance擁塞迴避

1. 問題與答案

Exp. 12: Sliding Window滑動窗口

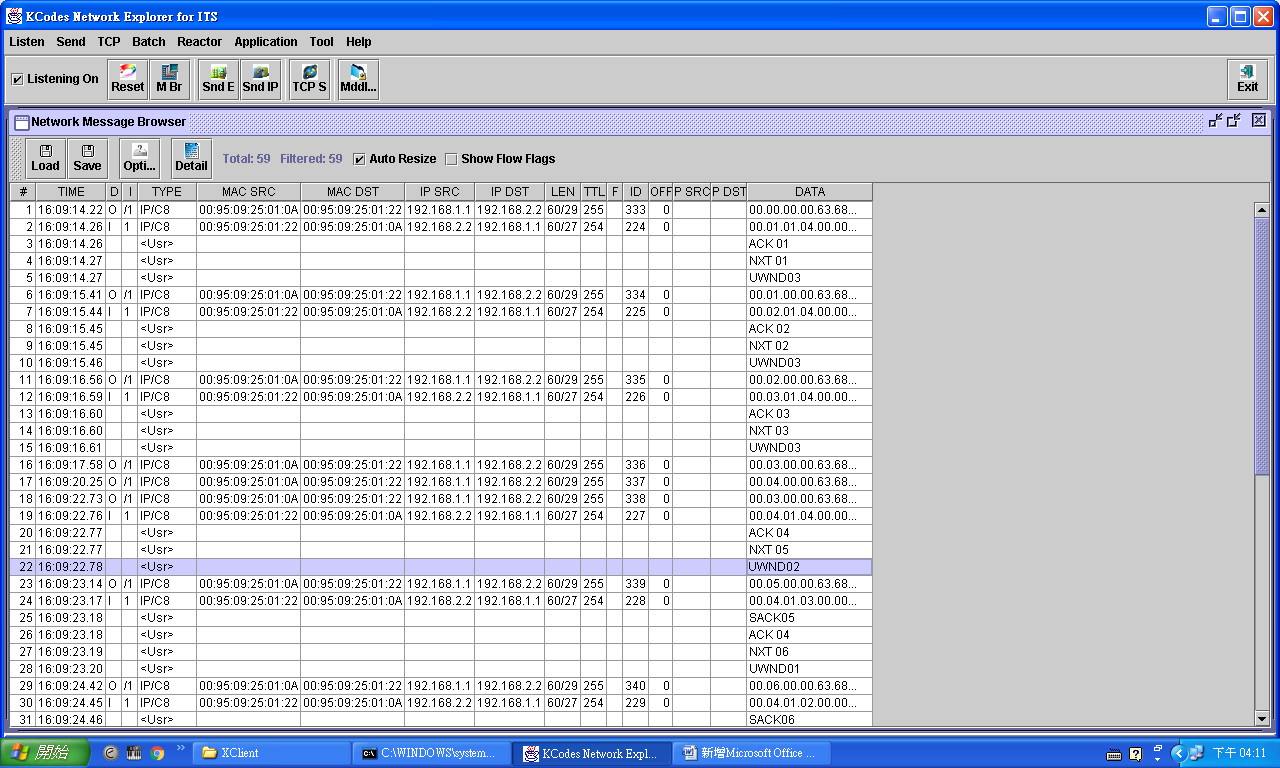
1. 在實驗中，如果封包在傳輸時我們將連接ITS的網路拔除，會發生甚麼狀況?

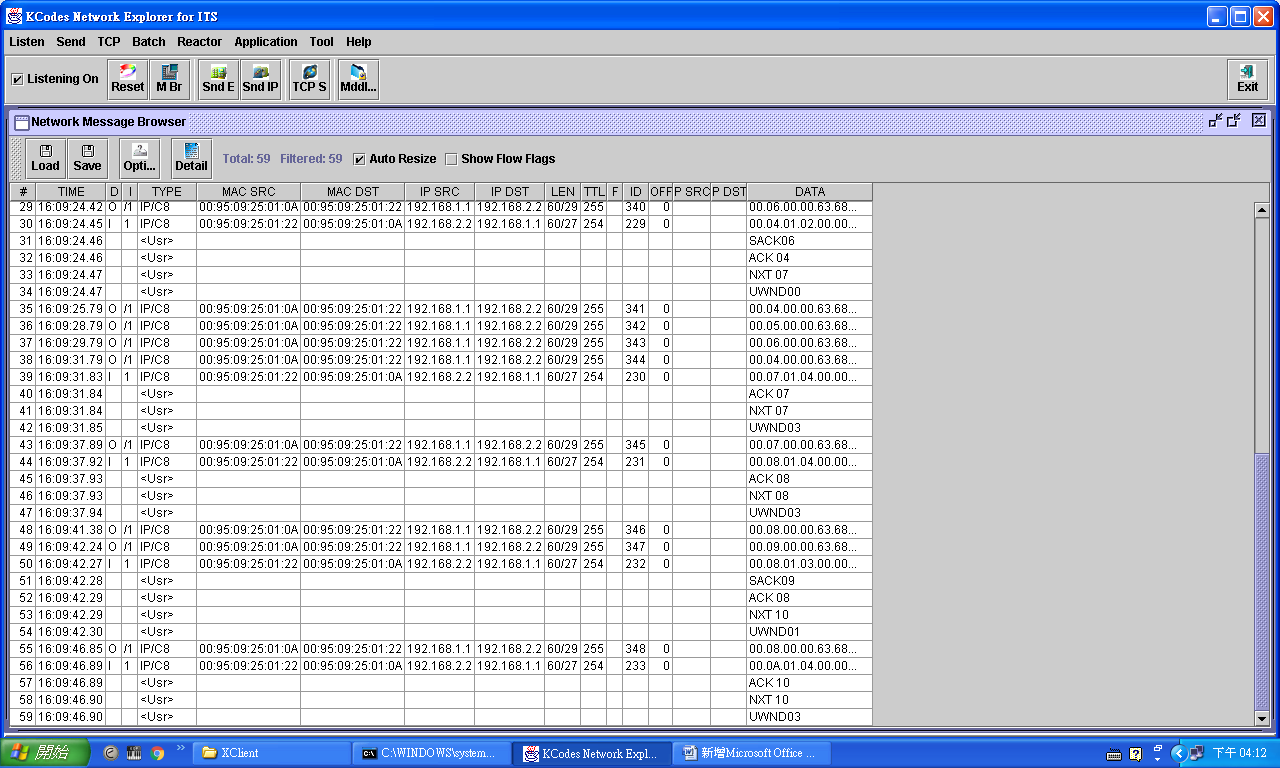
封包無法傳送，並顯示找不到傳送位置的錯誤，並會持續重新發送封包到目的地，但經過此次的封包lost之後，則會收到communication Aborted 的訊息，但停止重新傳送封包。



1. 改寫PkLost4.mddl 程式讓連續兩個IP datagram 都會 lost，如果在ITS1 尚未retransmission之前，再多送一個IP datagram，此時的window size 大小為何?

預設值window size為3，當連續兩個IP datagram 遺失，便開始不斷變小，直到變成0。

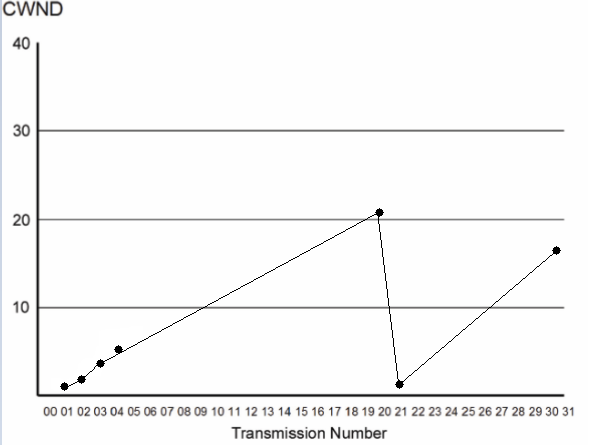


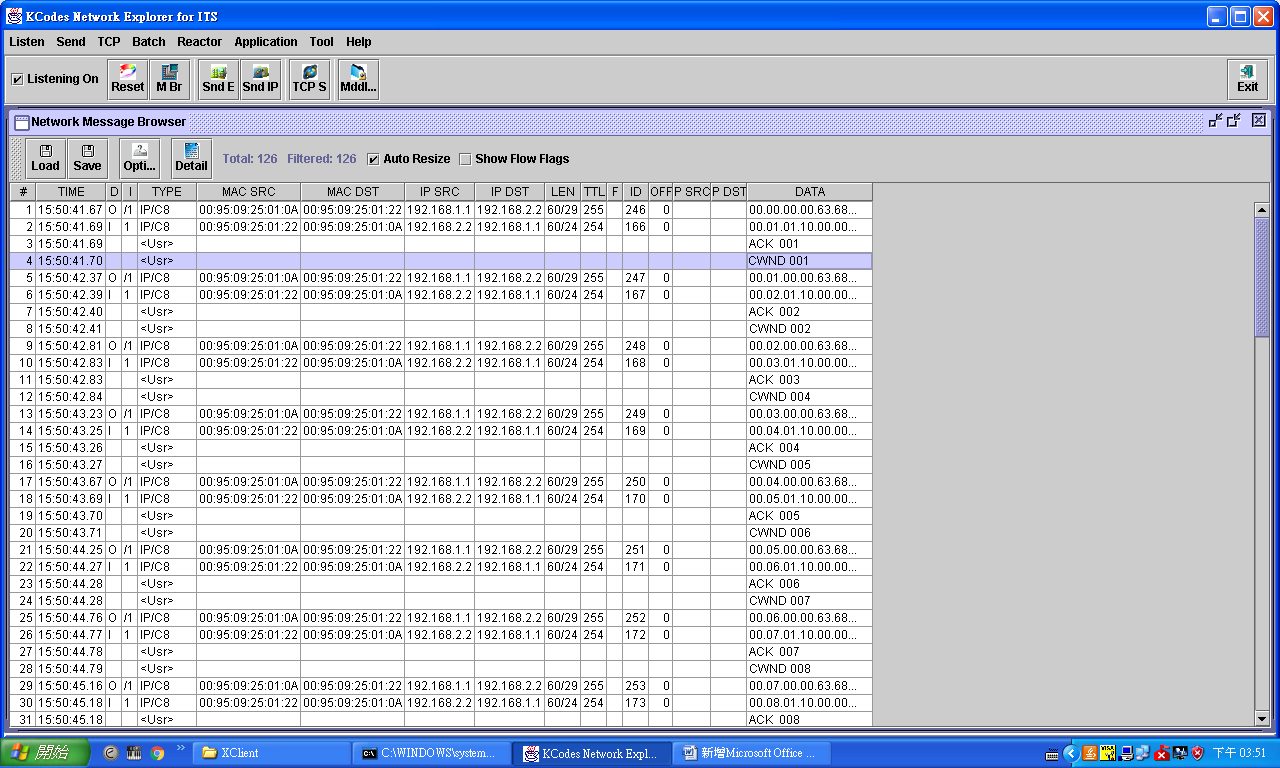


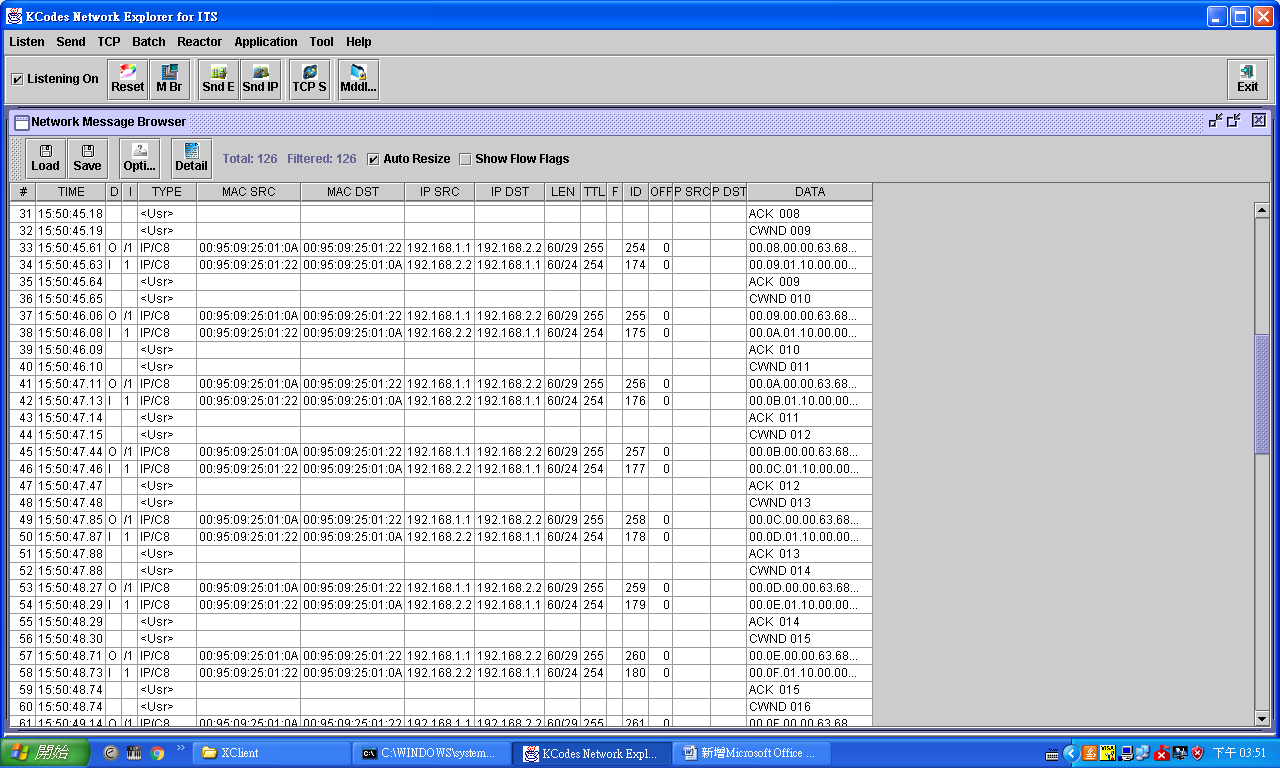
Exp. 13: Congestion Avoidance擁塞迴避

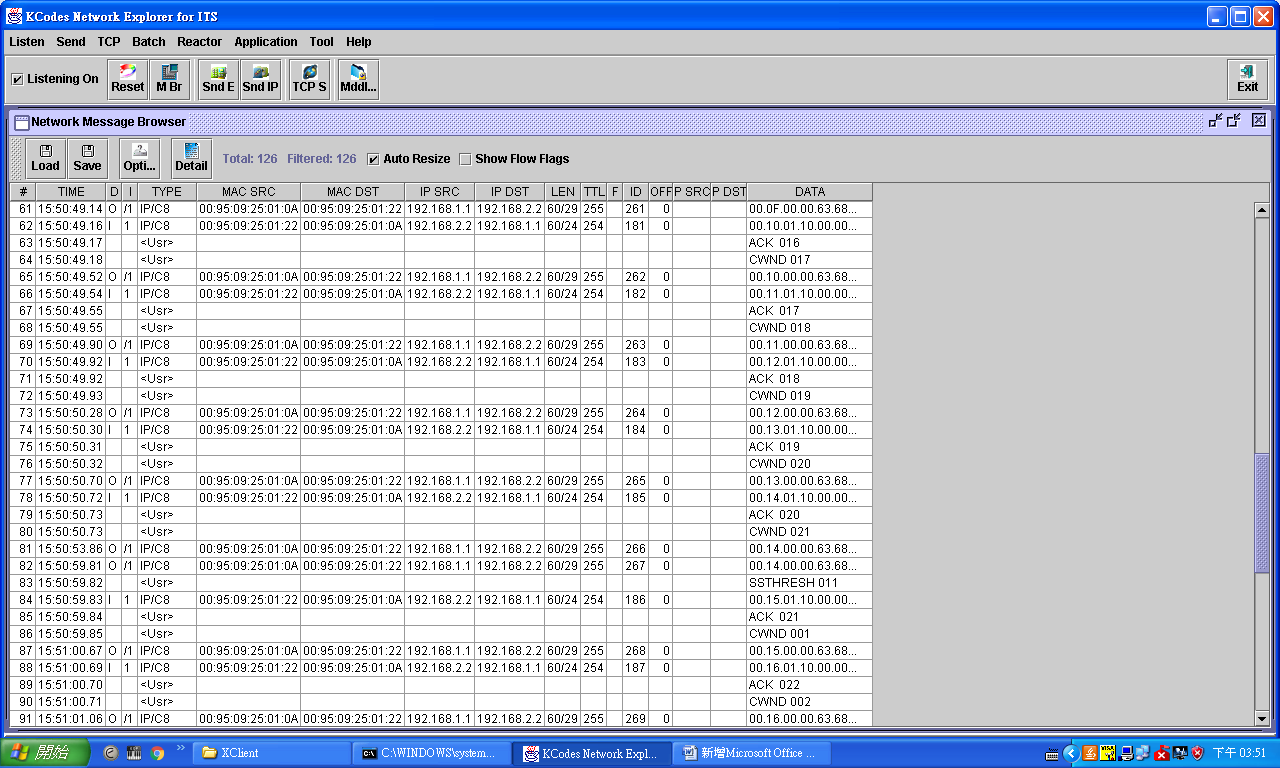
1. 連續發出20個IP datagram 後製造1個packet lost，完成後再多送10個可以順利抵達目的端的IP datagram ，參考每一個IP datagram 發出後CWND值的大小，完成Figure 13.10

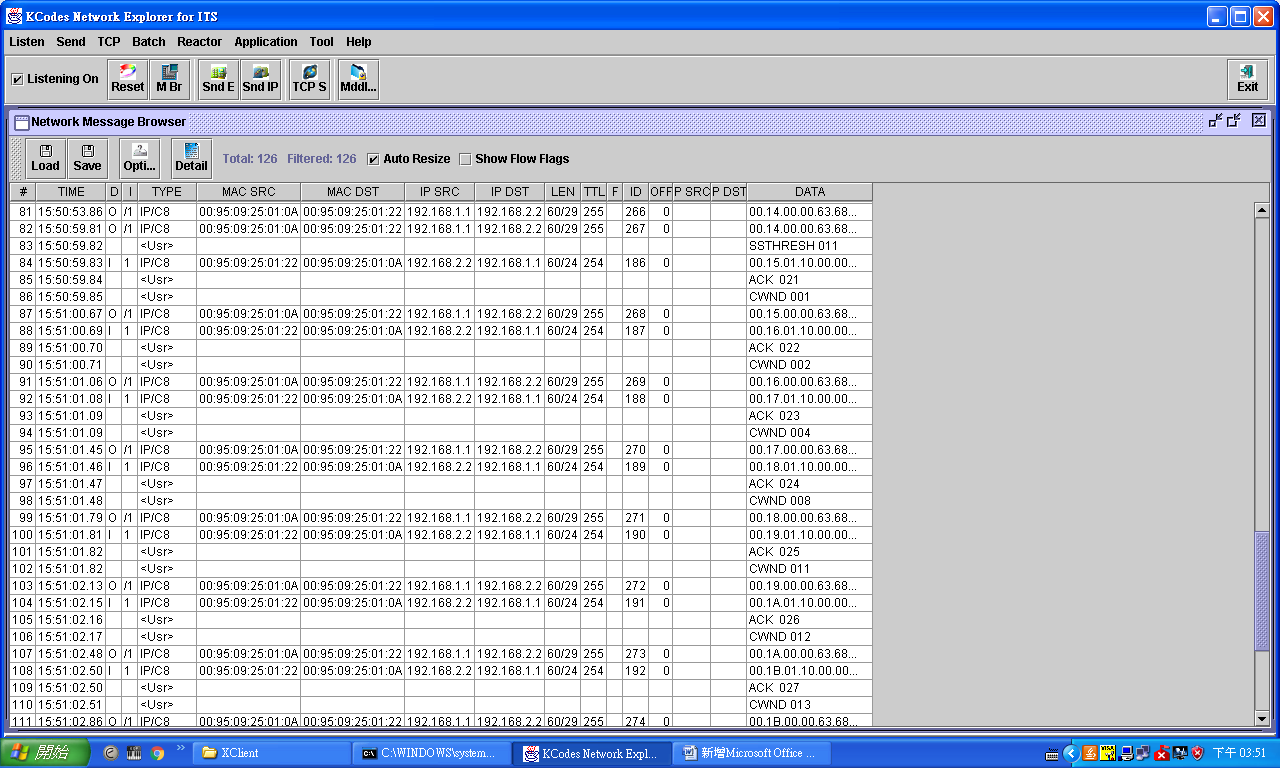
一開始為指數成長UWND 04為程式預設直，之後便開始線性成長，從圖中可以知道在封包21的時候封包阻塞，所以window size 變為1開始指數成長，SSTHRESH為UWND/2所以答案為(前一個封包UWND 21)21/2四捨五入為11，超過SSTHRESH=11之後便開始線性成長。

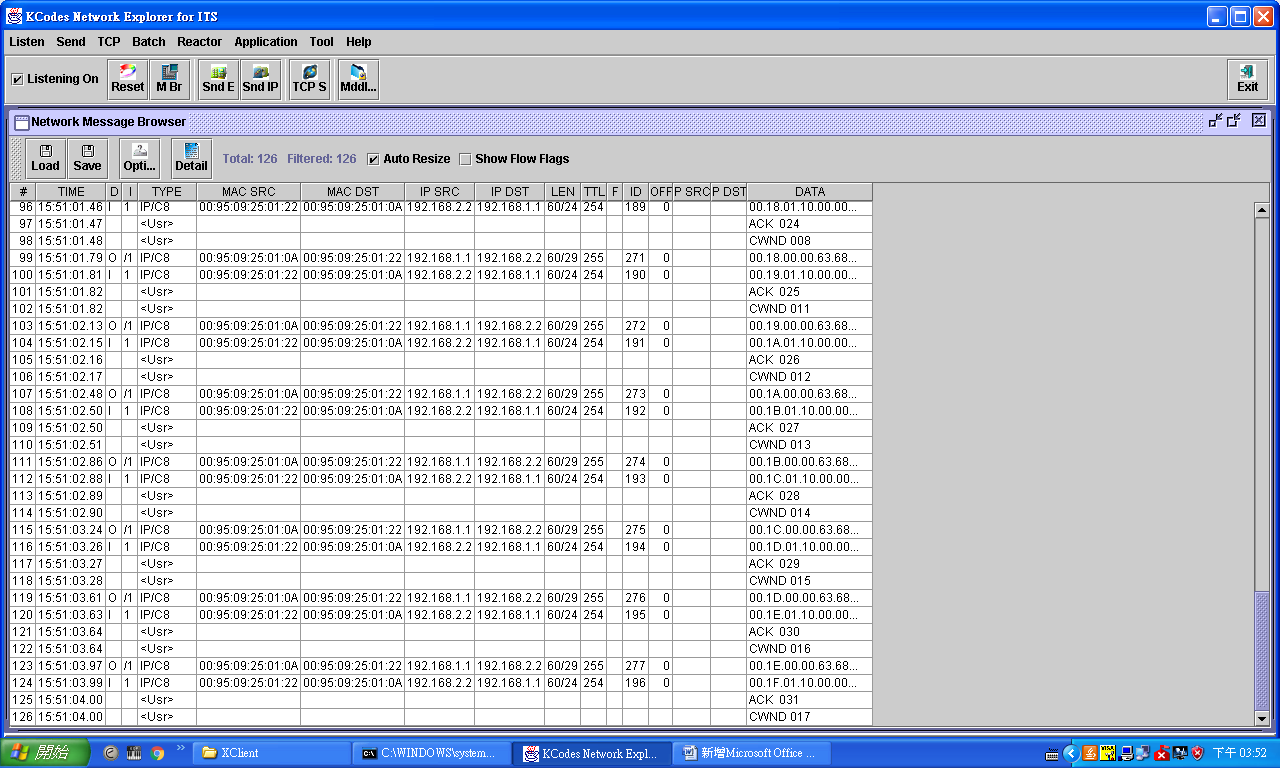






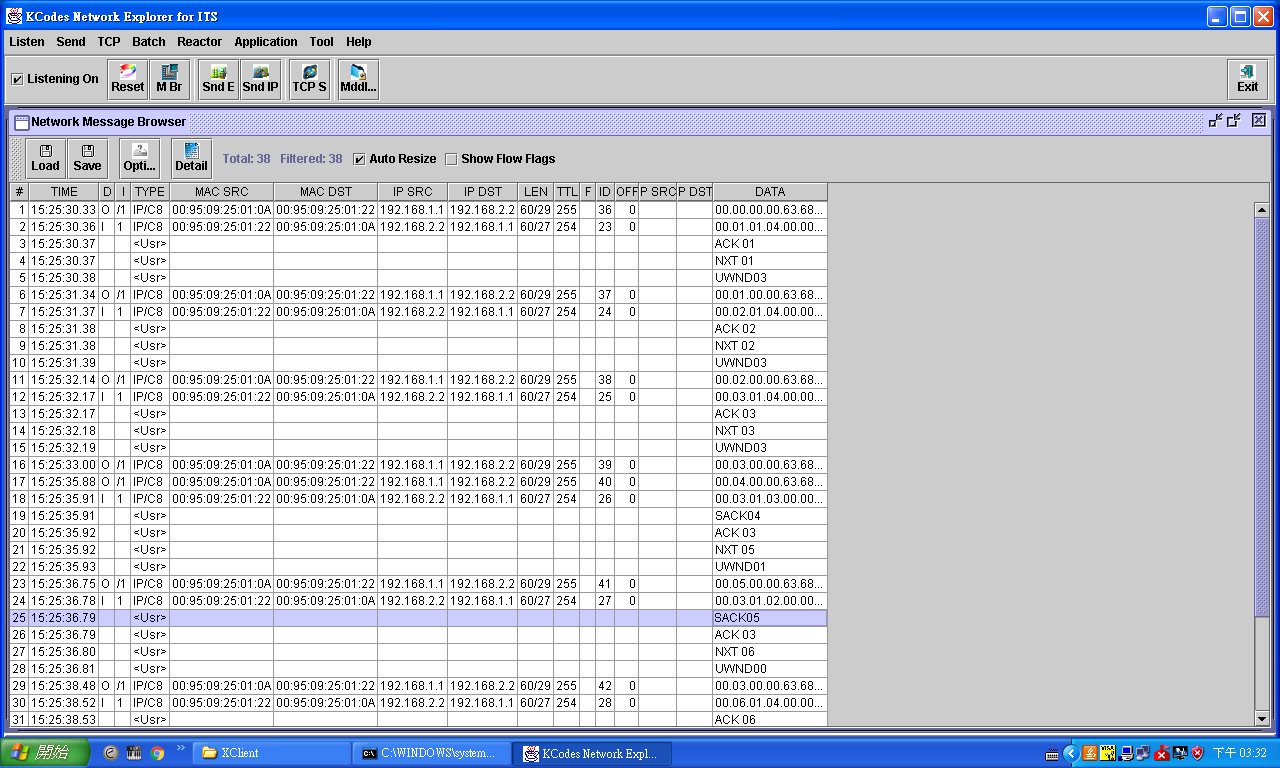


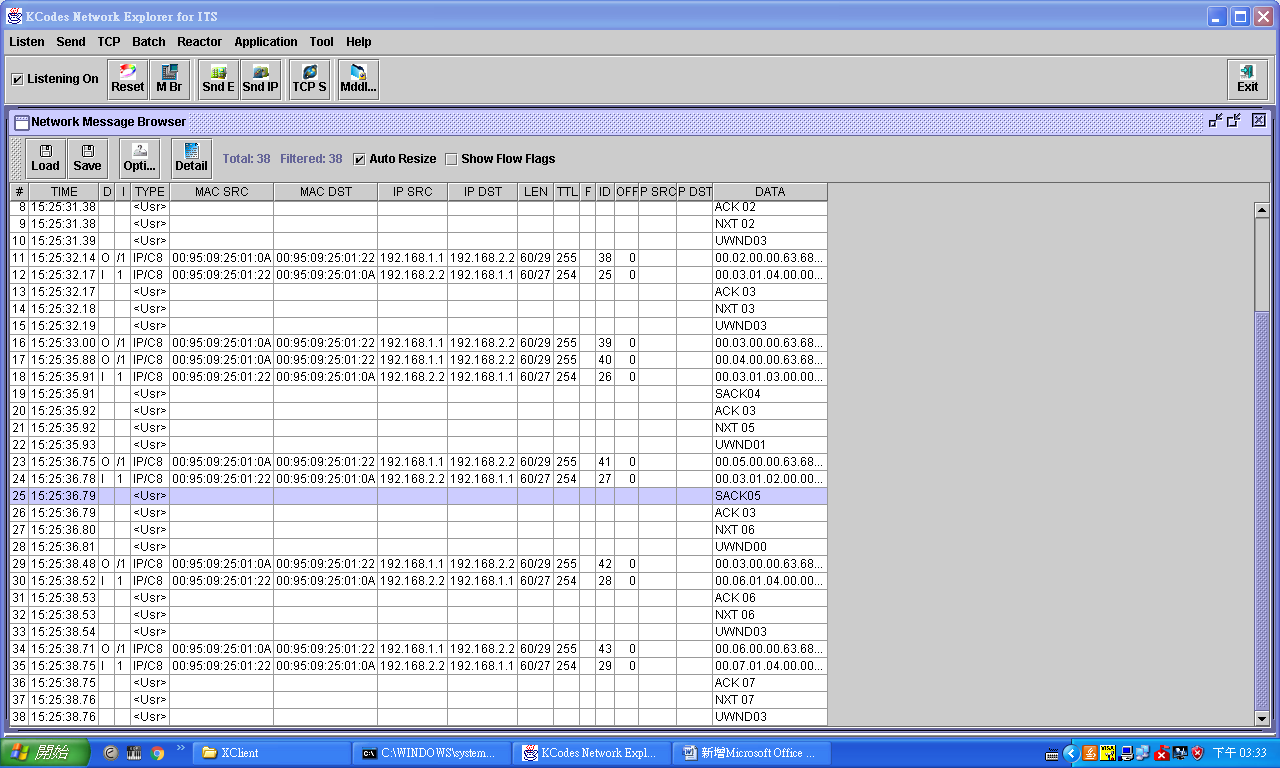




1. 結果討論

EXP12:Sliding window(驗收)





(Ack 是接收端下一個要收的封包

Nxt 是傳送端下一個要傳送的封包編號

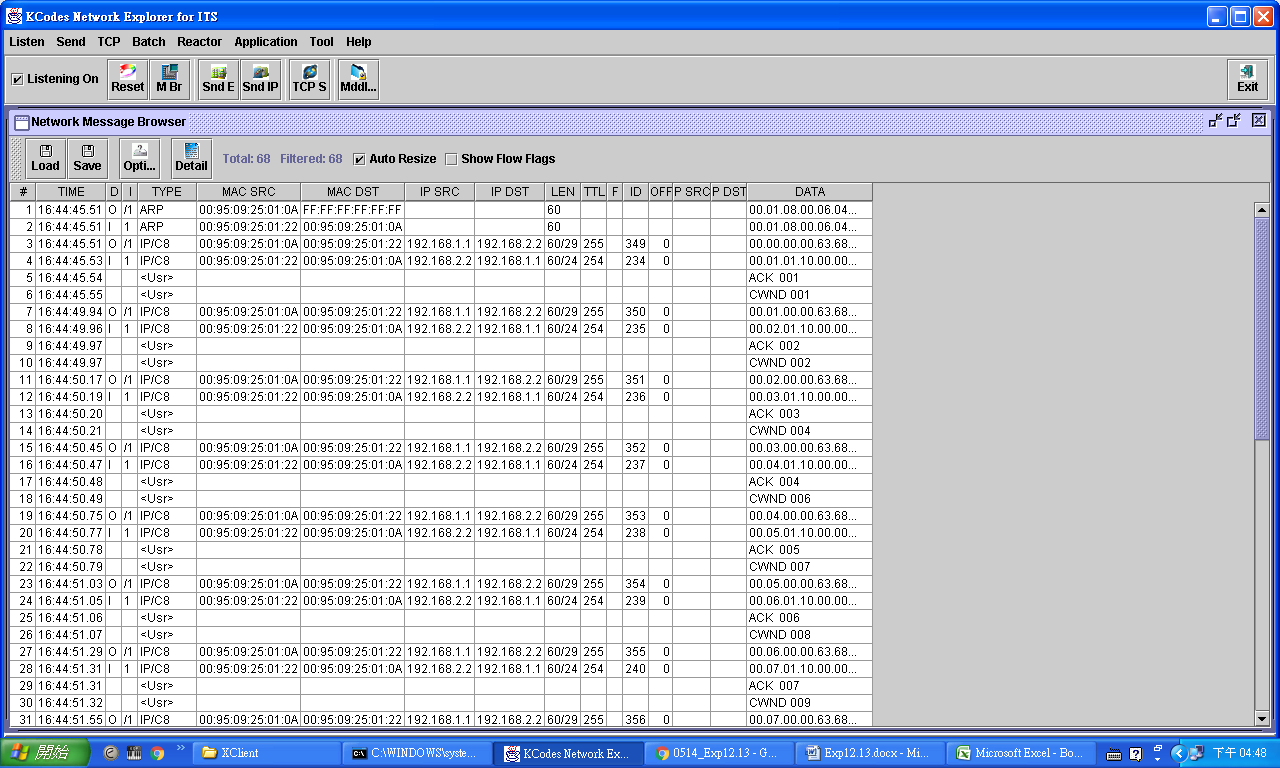
Sack 是接收端目前收到的封包編號

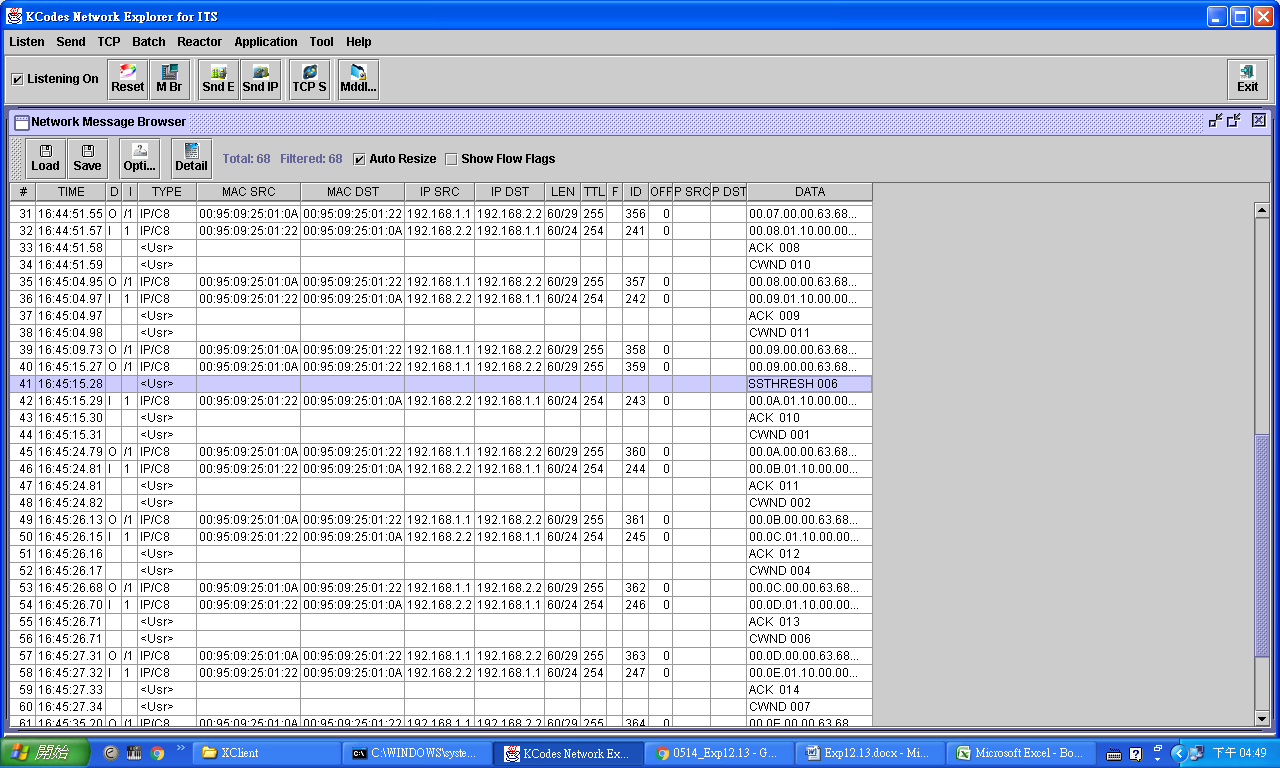
UWND 為剩餘window size數)

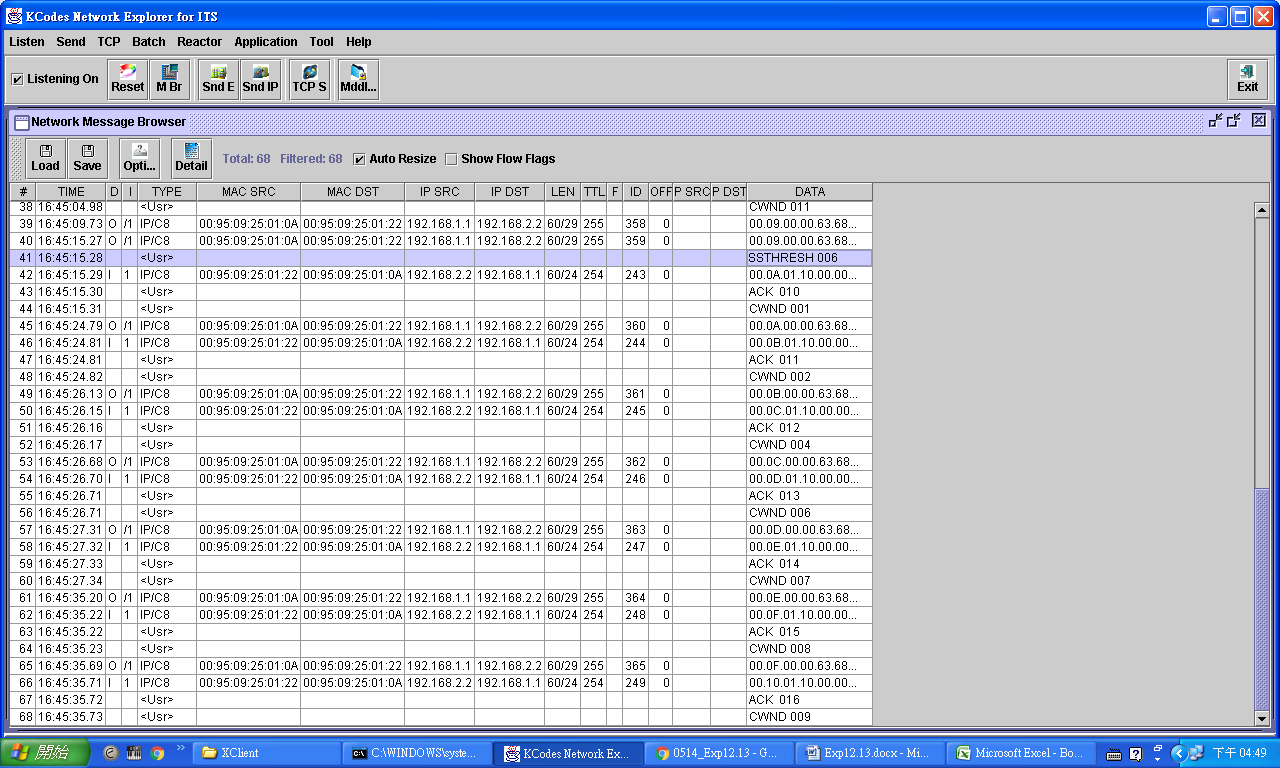
當window size 為0時，03封包剛好time out (5秒)重傳，所以依序03->04->05

傳出去，而傳完之後window size 又為3。

Exp13 congestion avoidance(驗收)







一開始window size 為指數成長，但到了06就開始線性成長，原因是middle程式所設定，原本設置為4，但為了看出差別我們把它改為6。到了window size 11時，封包堵塞，SSTHRESH為11/2四捨五入，所以為6。堵塞完window size 便又從1開始指數成長，到了window size 6之後便開始縣性成長。

1. 補充資料

<http://linux.vbird.org/linux_server/0110network_basic.php>

<https://blog.csdn.net/dog250/article/details/51439747>