Outline

- Introduction
- IP Address & MAC Address
- **TCP/UDP/ICMP**
- IP Gateway, Network Mask, TTL
- Routing Protocol
- Network Address Translation (NAT)
- Domain Name System (DNS)
- Dynamic Host Configuration Protocol (DHCP) / Asymmetric Digital Subscriber Line (ADSL)
- HyperText Transfer Protocol (HTTP) Protocol
- Virtual Private Network (VPN)

3. TCP/UDP/ICMP

- TCP (Transmission Control Protocol)：
  - 提供**Connection Oriented (**連結導向**)** 並達成**End-to-End (**兩端通訊端點對端點**) Process-to-Process(**程序對程序**)**、**Reliable Data Delivery (**可信賴的資料遞送**)**。
- UDP (User Datagram Protocol)：
  - 提供不可靠**(Unreliable)**的傳輸，也就是它並不能擔保資料是否能確實地從來源傳送到目的地，因爲**UDP** 除了自己的**Checksum** 以外，並沒有做任何確保封包能確實送達目的地的機制。
- ICMP (Internet Control Message Protocol)：
  - 提供主機或是網路設備之間交換錯誤訊息及其他重要資訊。

Question:

- What is point-to-point connection?
- What is end-to-end connection?
- What is unicast / multicast / broadcast?
- What is multiplexing?

補充1

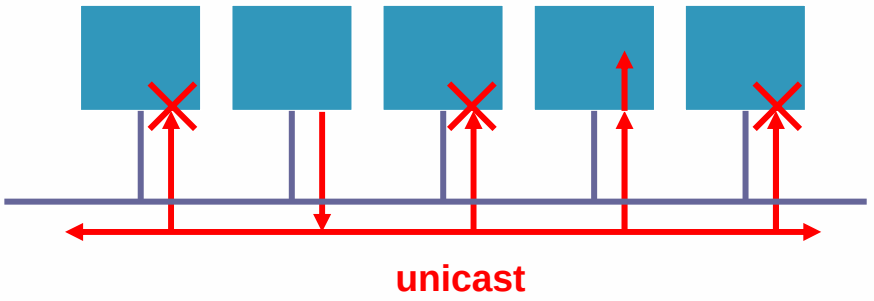
❖連接方式

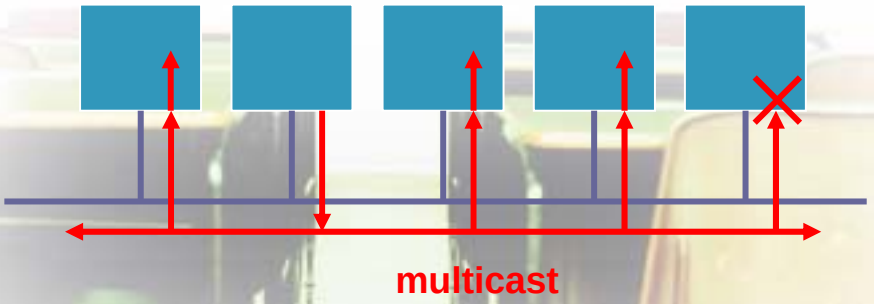
- 點對點(point-to-point)：直接相連(實體相連)

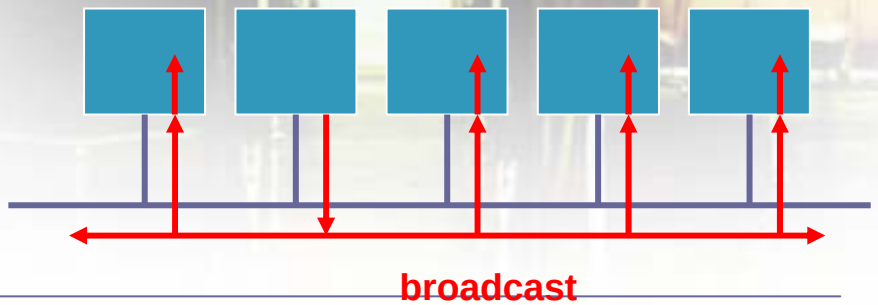
- 端點對端點(End-to-End)：間接相連(跨網路相連)

補充2（傳輸模式）

- Unicast (單點傳輸)
- Multicast (群播)
- Broadcast (廣播)

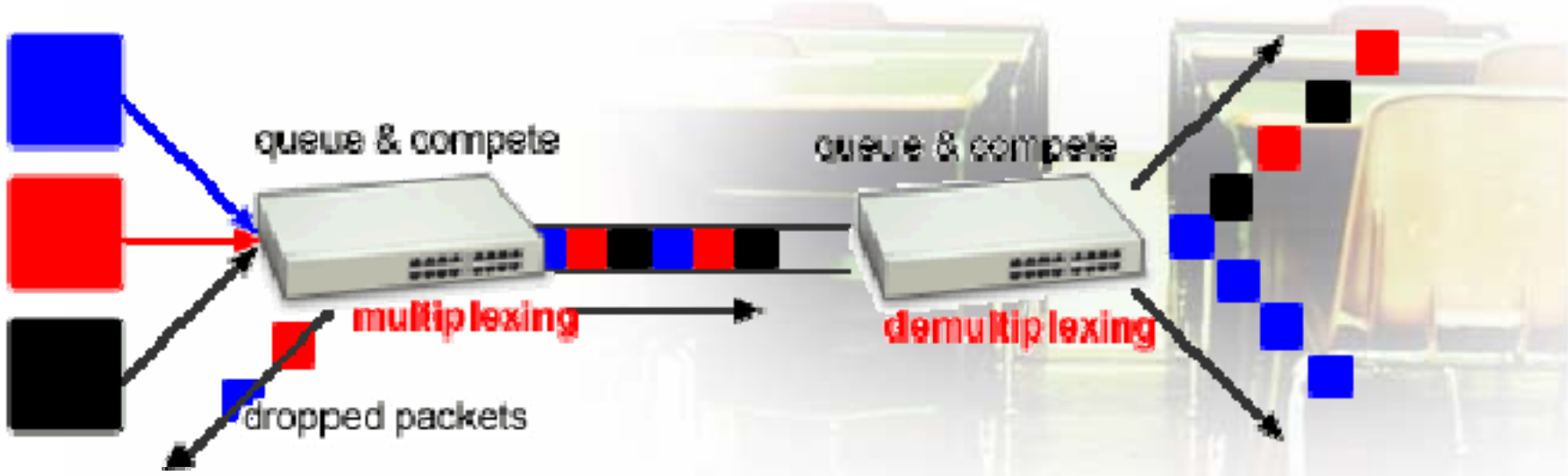


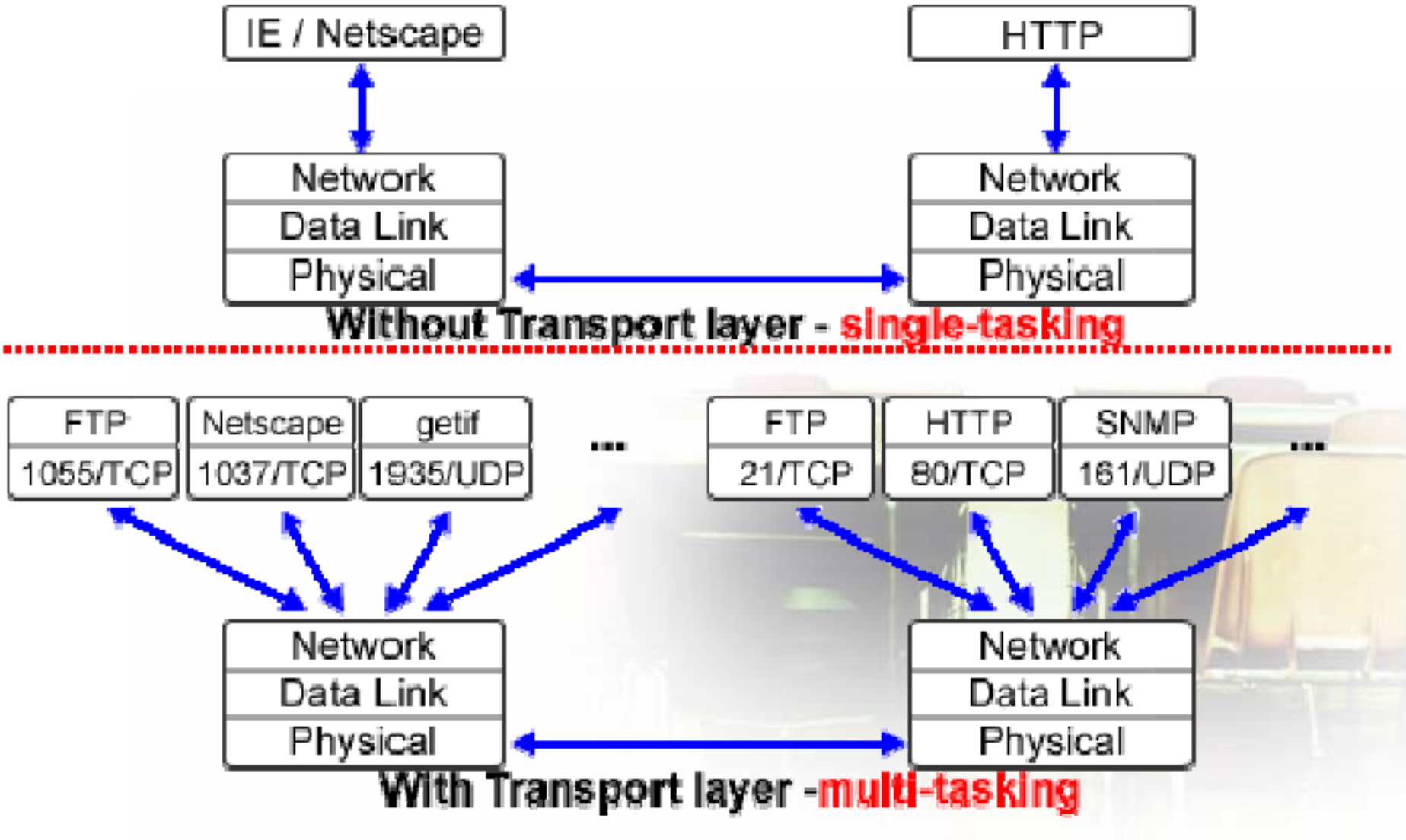


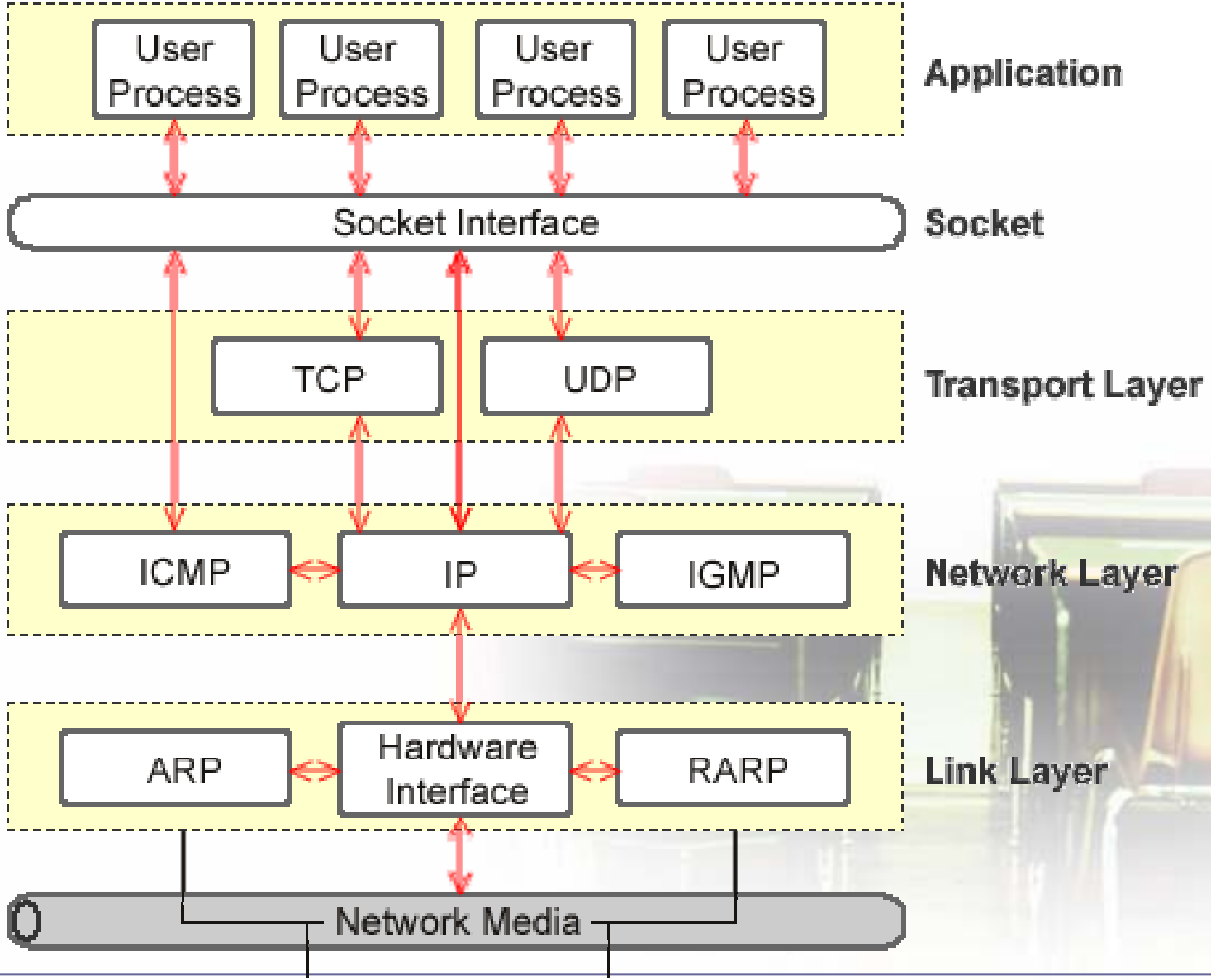


補充3 (多工:Multiplexing)

- 分時多工
  (Time-Division Multiplexing, TDM)
- 分頻多工
  (Frequency-Division Multiplexing, FDM)



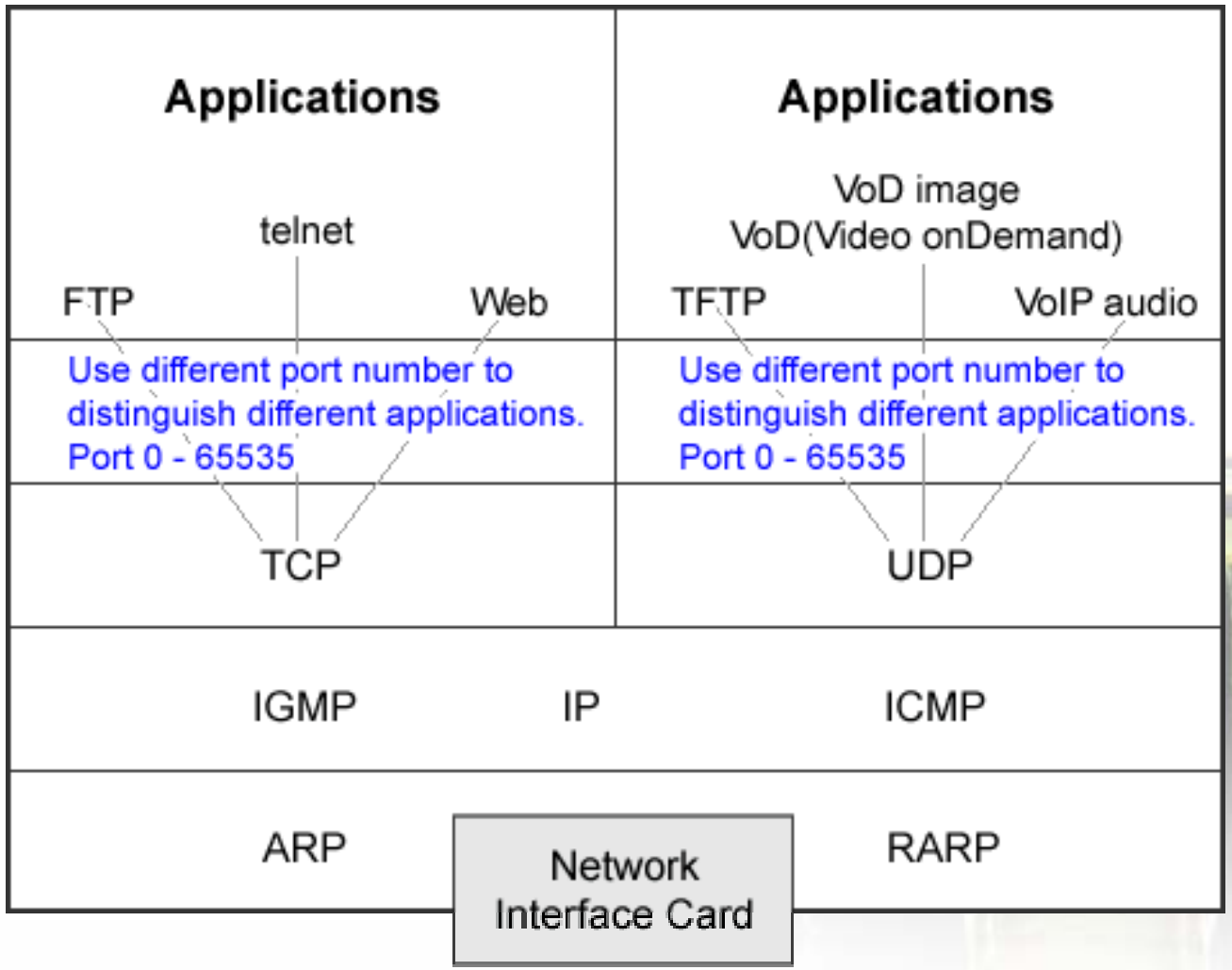

IE / Netscape
HTTP
Network
Data Link
Physical
Network
Data Link
Physical
Without Transport layer - single-tasking
FTP
1055/TCP
Netscape
1037/TCP
getif
1935/UDP
...
FTP
21/TCP
HTTP
80/TCP
SNMP
161/UDP
...
Network
Data Link
Physical
Network
Data Link
Physical
With Transport layer -multi-tasking


User Process
User Process
User Process
User Process
Application
Socket Interface
Socket
TCP
UDP
Transport Layer
ICMP
IP
IGMP
Network Layer
ARP
Hardware Interface
RARP
Link Layer
Network Media

3.1 UDP 通訊協定

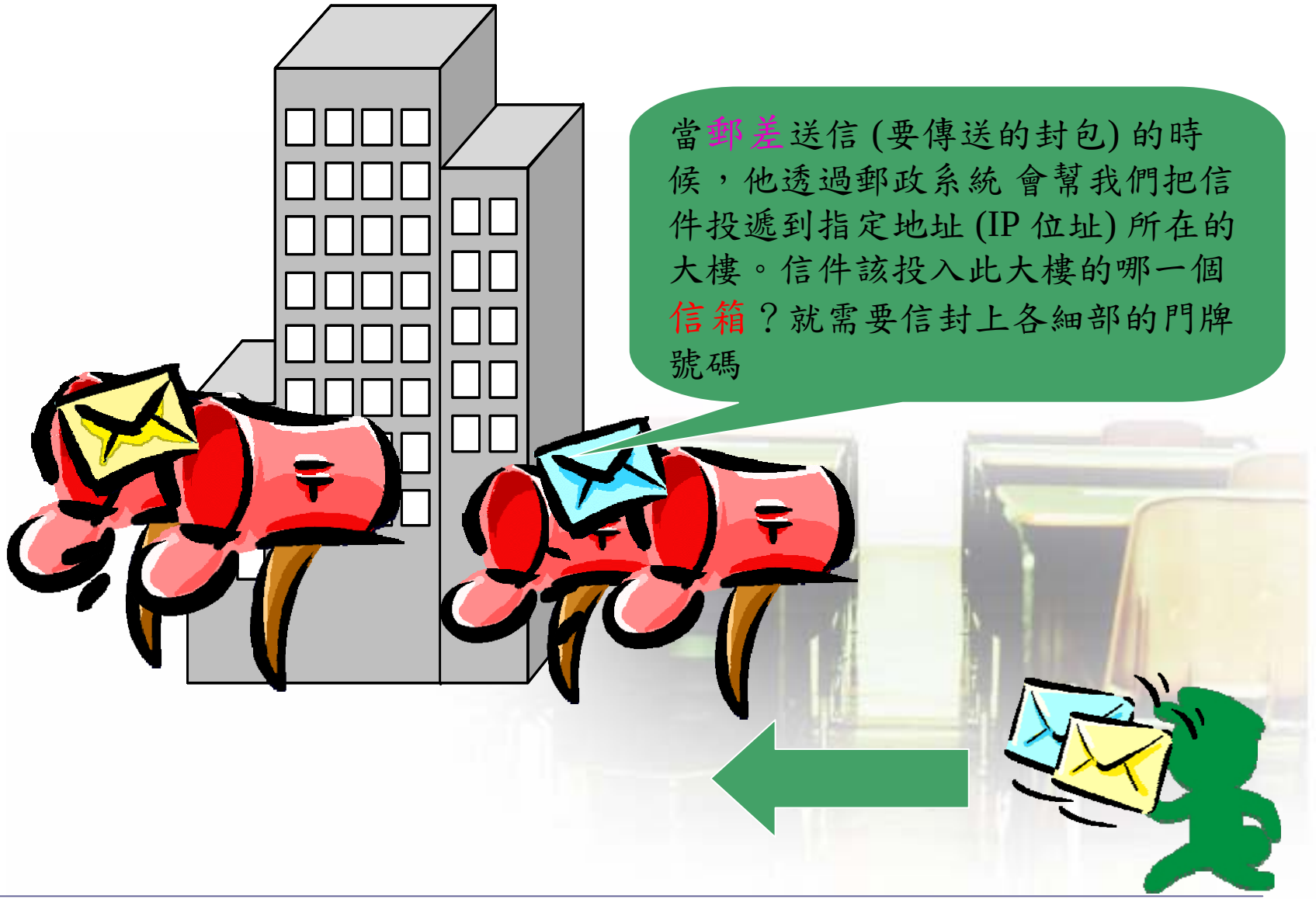
- 定義於RFC 768（User Datagram Protocol）
- 提供應用程式能夠在最低的通訊協定機制下送訊息給其他應用程式。
- 無法保證是不是傳送得到或者是會被重製或有沒有依照傳送順序到達。
- 講究傳輸速度且容許一點資料流失(Data Loss)或是封包遺失(Packet Loss) 的應用程式常用UDP 通訊協定來實作。例如NetMeeting、VoIP、MSN 使用語音等。

3.1.2 埠號(Port Number)與多工(Multitasking)的觀念

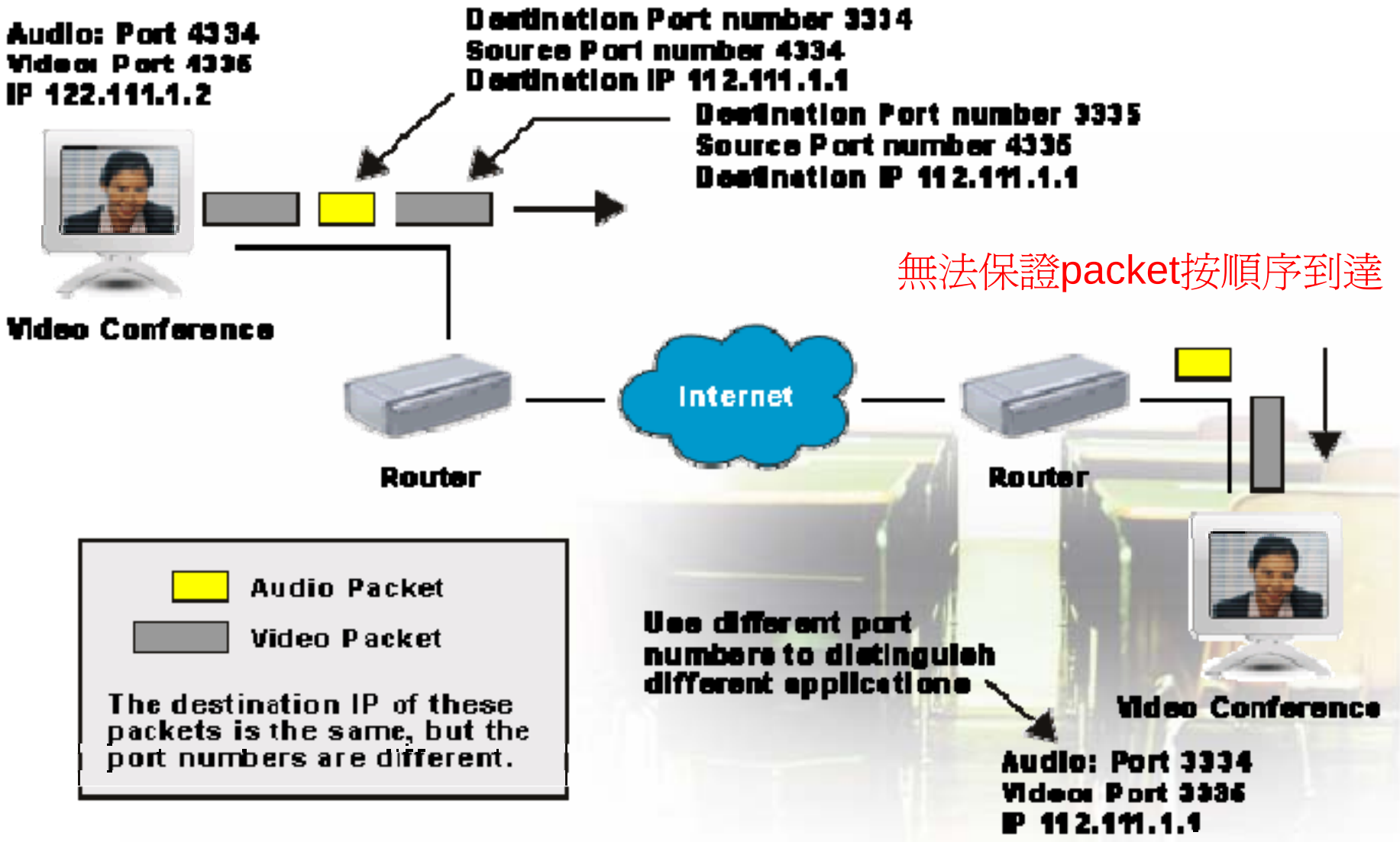


實驗 4.2 了解port number的涵意

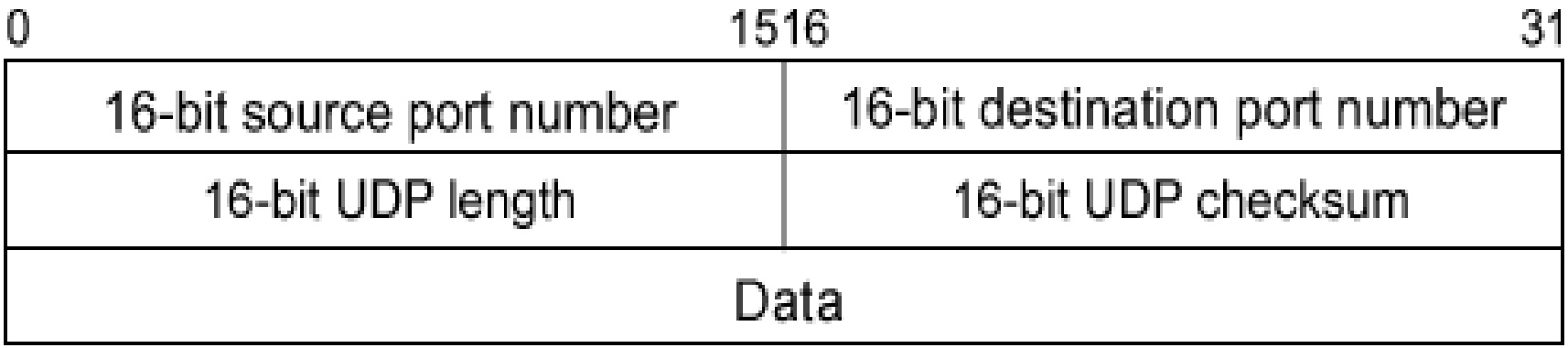
用大樓的“信箱”來解釋IP 位址和埠的概念



舉例說明 — 視訊會議


Audio: Port 4334
Video Port 4336
IP 122.111.1.2
Destination Port number 3334
Source Port number 4334
Destination IP 112.111.1.1
Destination Port number 3335
Source Port number 4335
Destination IP 112.111.1.1
無法保證packet按順序到達
Video Conference
Router
Internet
Router
Audio Packet
Video Packet
The destination IP of these packets is the same, but the port numbers are different.
Use different port numbers to distinguish different applications
Video Conference
Audio: Port 3334
Video Port 3336
IP 112.111.1.1

3.1.2 UDP資料格式

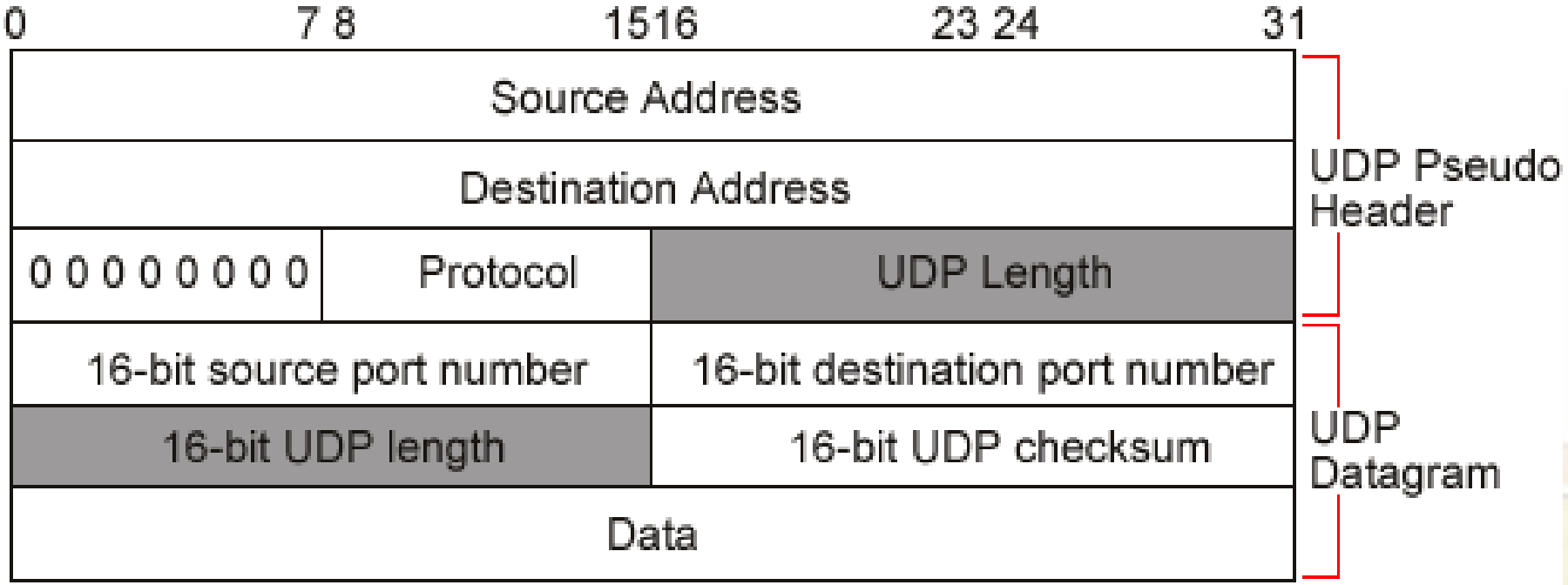


用 Ethereal 抓取到的畫面說明

UDP資料格式－ 細部說明

- **Source Port：**
  - 非必須的欄位，不使用時填零
- **Destination Port：**
  - 表示這個 Datagram 將送達的位置Port
- **UDP Length：**
  - 以八進位表示的 Datagram 的長度，包括了檔頭和資料 (min: 8 bytes, i.e. only UDP header)。
- **UDP Checksum：**
  - 將 UDP 的 Pseudo Header、UDP Header 以及 UDP 的 Data的所有資料加總之後，取一的補數，再取 16 bit的一補數，若必要時會在後面加上一個 0，以確保是兩個八進位數資料。

UDP Pseudo Header

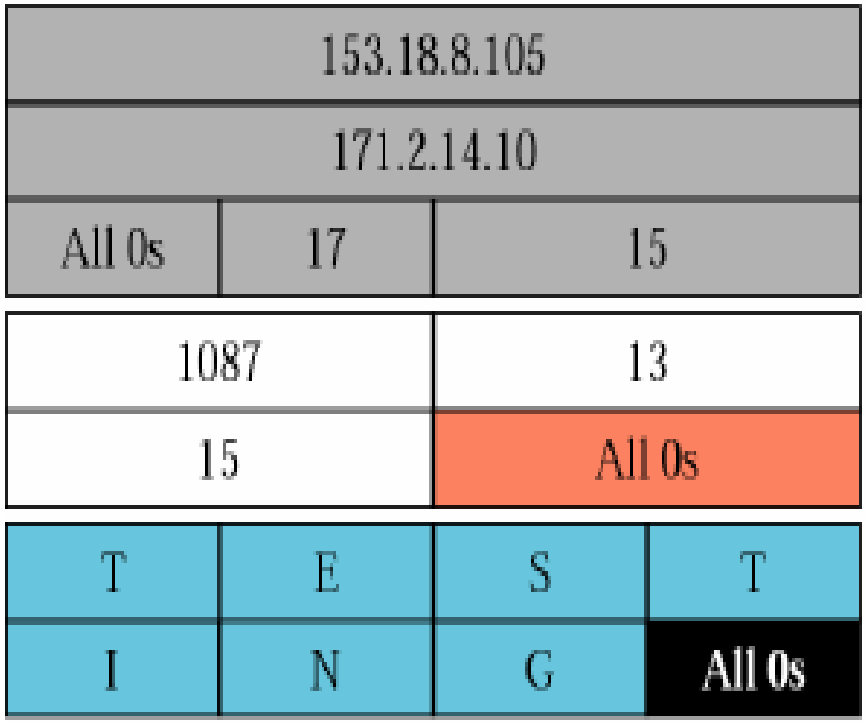


之所以稱爲虛擬(**Pseudo**)，是因爲其中的 Source IP Address 與 Destinatin IP Address 並不出現在 UDP 封包中，而是在 IP 封包中。但在進行校驗的時侯，發送端與接收端則根據此 "虛擬" 的表頭及資料進行。然而，此表頭是不會出現在任何封包中的。

UDP Pseudo Header– 細部說明

- **UDP Pseudo Header :**
  - 將部分 IP header 、UDP header看成集合成一個 header
- **UDP Datagram 的最大長度 :**
  - 理論上 : 65535 – 20(IP header) – 8(UDP header) = 65507 Bytes
  - 一般 socket 在實作上存取的緩衝區 (Buffer) 有限制 8192 Bytes)
  - 因此，像是一些使用 UDP的通信協定諸如 DNS、TFTP、SNMP 等多將其 Datagram 大小限制在 512 Bytes

UDP Checksum

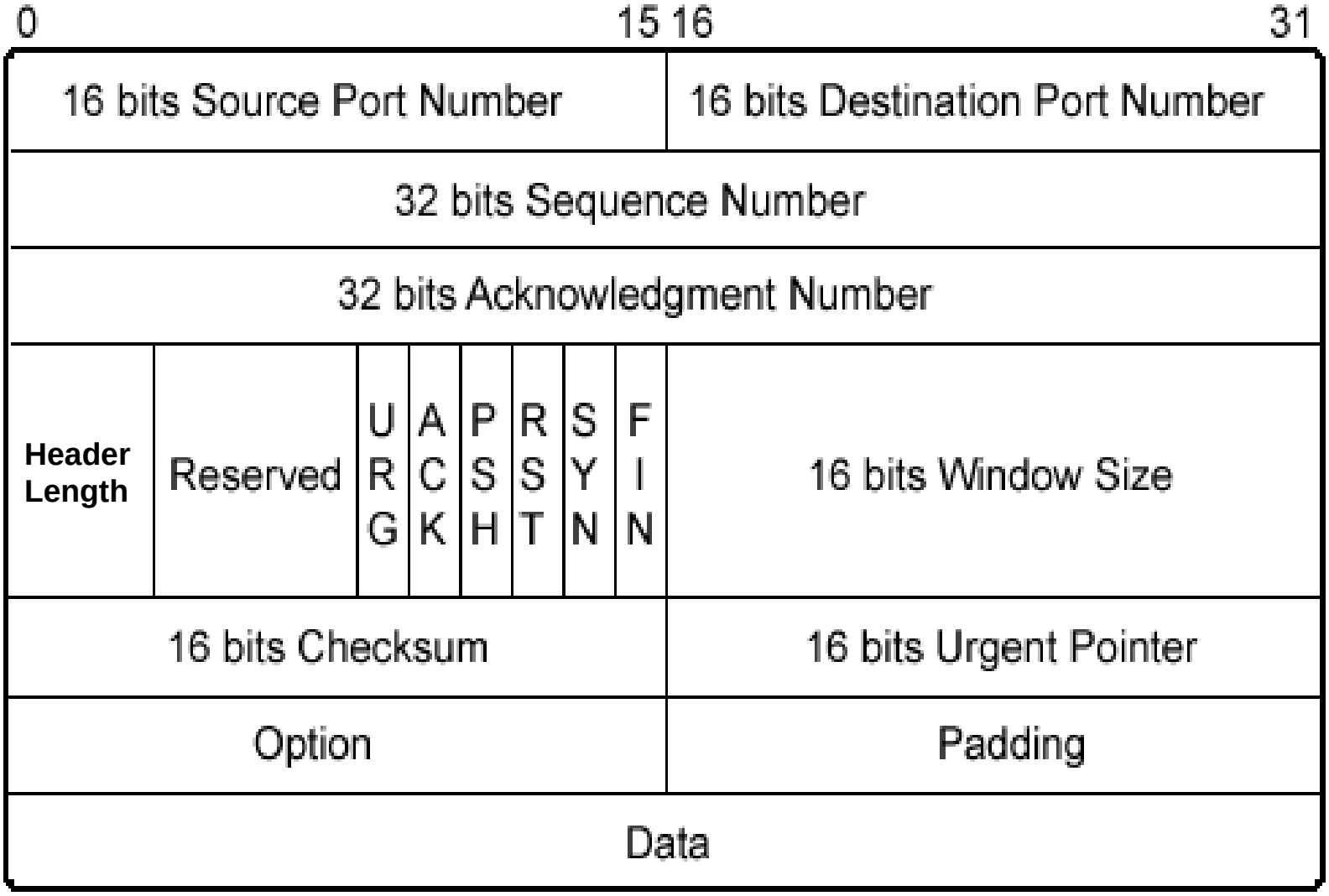


10011001 00010010	→	153.18
00001000 01101001	→	8.105
10101011 00000010	→	171.2
00001110 00001010	→	14.10
00000000 00010001	→	0 and 17
00000000 00001111	→	15
00000100 00111111	→	1087
00000000 00001101	→	13
00000000 00001111	→	15
00000000 00000000	→	0 (checksum)
01010100 01000101	→	T and E
01010011 01010100	→	S and T
01001001 01001110	→	I and N
01000111 00000000	→	G and 0 (padding)
10010110 11101011	→	Sum
01101001 00010100	→	Checksum

3.2 TCP 通訊協定

- 定義於 **RFC 793 (Transmission Control Protocol)**
  - 提供 Connection Oriented (連結導向) 並達成End-to-End (兩端通訊端點對端點) Process-to-Process (程序對程序)、Reliable Data Delivery (可信賴的資料遞送)。
- **TCP提供以下三個基本功能：**
  - Reliability：克服Packet Loss，傳送之Data 依照順序交給Process (i.e. 封包正確無誤依照順序送達應用程式 (而非正確無誤依照順序送達電腦介面))
  - Multiplexing：藉由不同的Port 可使相同的IP Address 可以提供上層不同的服務，如FTP，Telnet，HTTP 等
  - Flow Control 或 Congestion Avoidance and Control：避免網路或接收端壅塞，以便提供有效率的傳輸

TCP 資料格式



用 Ethereal 抓取到的畫面說明

TCP Header

- **Source Port & Destination Port：**來源端與目的端通訊埠號碼
- **Sequence Number：**表示此資料段在訊息中的序號，接收端依序組合資料段
- **Acknowledgment Number：**接收端希望下次收到的序號，也是回應已收到封包
- **Header Length：TCP Header**的長度
- **Reserved：**保留給未來使用

TCP Header(Cont.)

- **Window Size**：使用於流量控制，表示能接收資料的數目(以8個位元組爲單位)
- **Checksum**：錯誤偵測號碼
- **Urgent Pointer**：緊急指標。**URG flag**爲1時，此欄位才生效
- **Options**：此資料段的發送者告訴對方能接受的最大資料段長度
- **Padding**：使**header**長度以**32**個位元結束

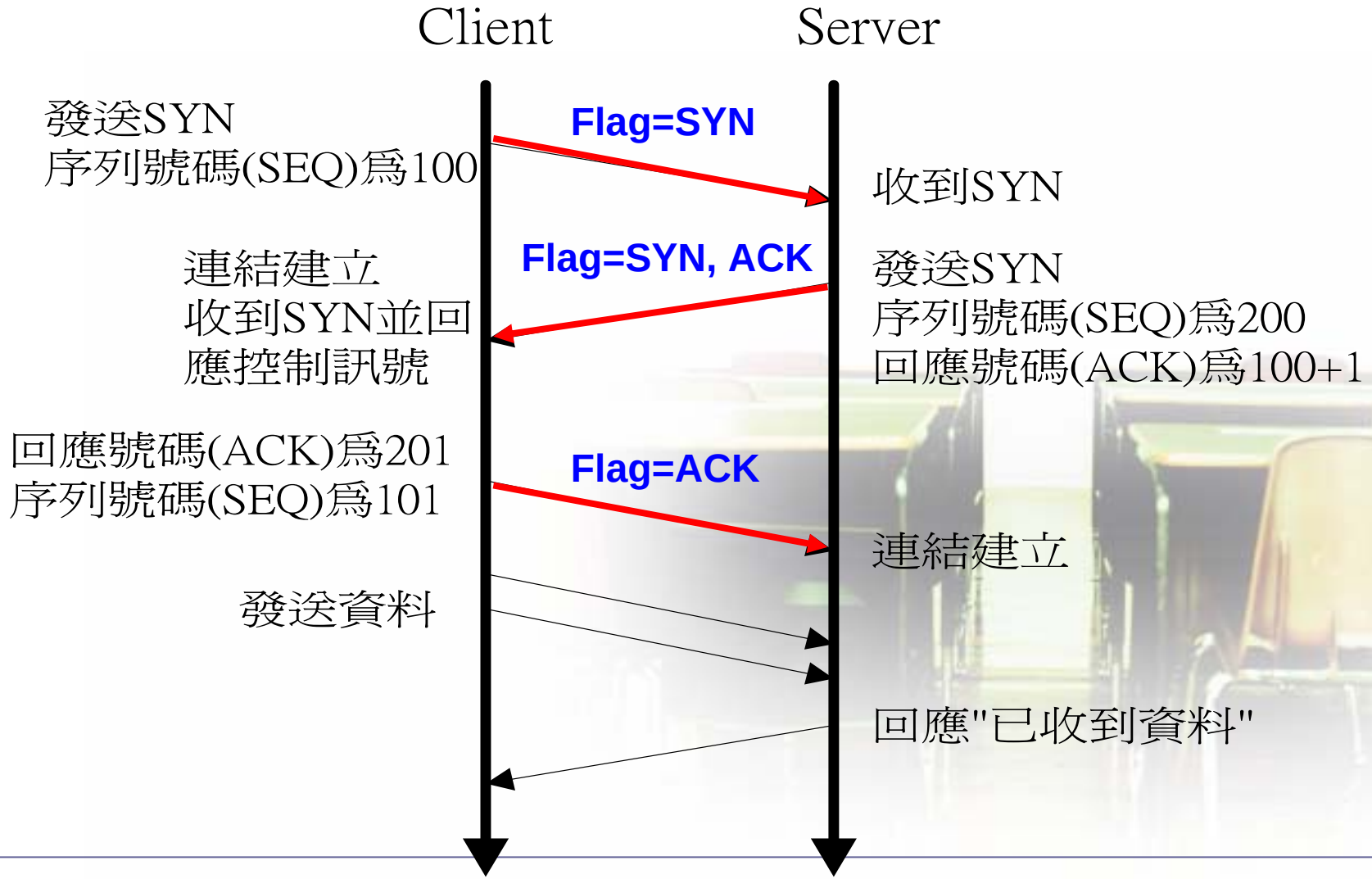
TCP flag

- **TCP** 連線都有一定之程序，並且分成幾個不同的狀態**(State)**，它是透過 **TCP Header** 欄位的 **flag** 欄位來實作

Abbr.	Name	Explanation	
ACK	Acknowledge	To verify Acknowledgement Number	如果設定，表示此封包有一個回應
SYN	Synchronize	To synchronize Sequence Number	建立順序號碼
FIN	Finish	To finish sending data	傳送資料到此為止
RST	Reset	To reset connection	重設連結
PSH	Push	To send data out	把要發送的資料發送出去
URG	**| Urgent Pointer|**	緊急指標	

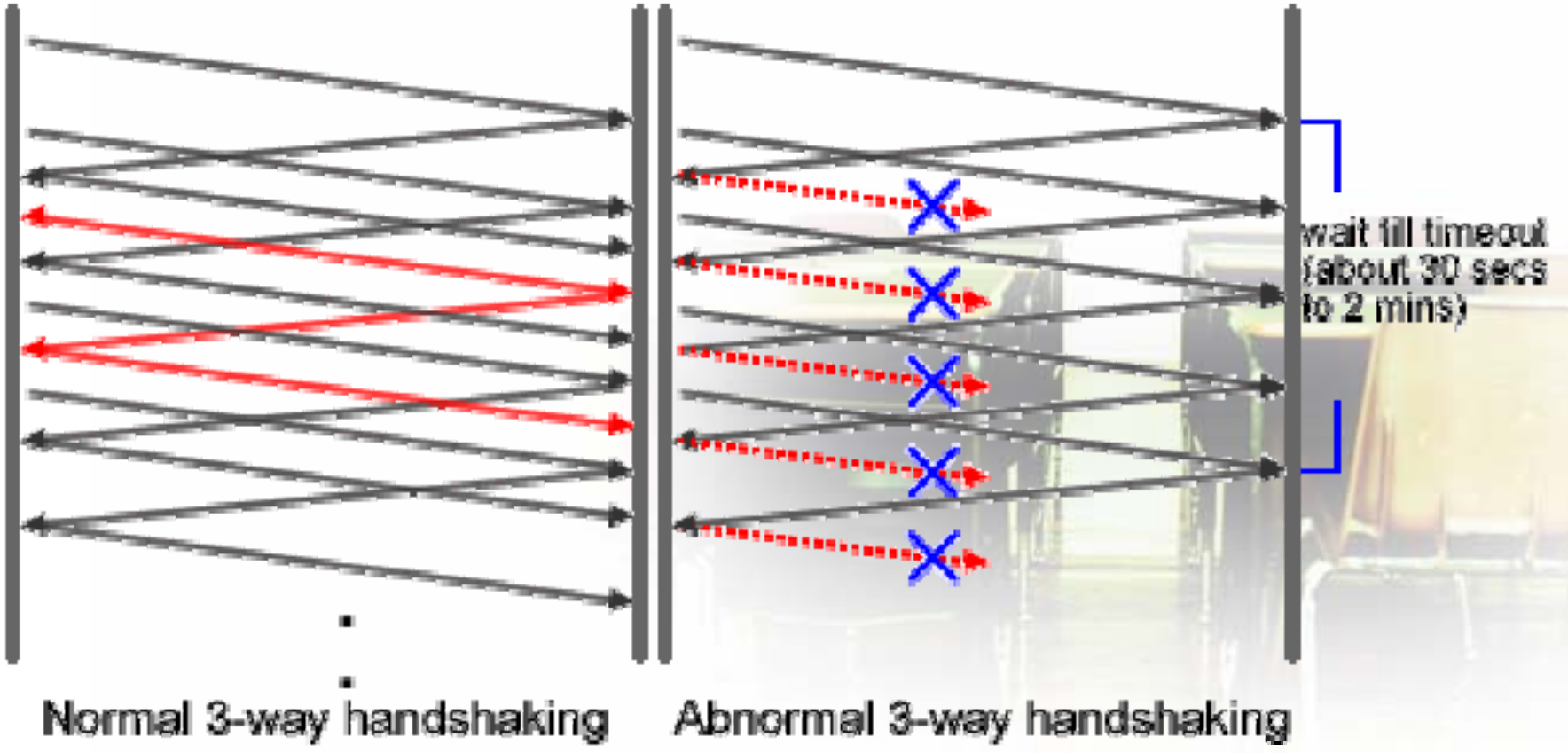
TCP 之連線建立

❖三向交握(three-way handshaking)



SYN Flooding

- **SYN Flooding是針對TCP三向式交握的一種DOS (Denial of Service) / DDOS (Distributed Denial of Service)的攻擊法**



SYN Flooding (DOS)

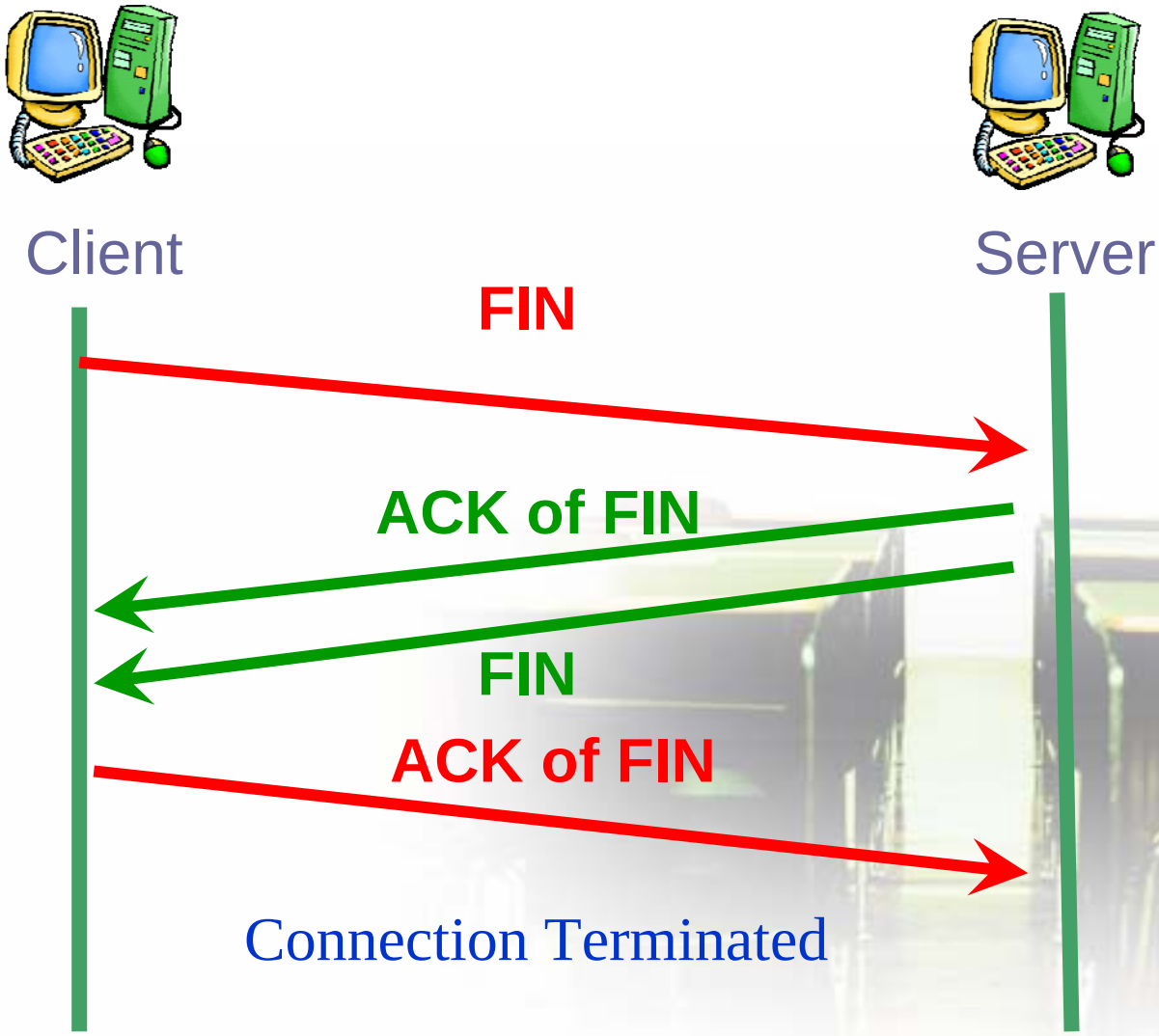
- **Q: TCP**連接的三次握手中，假設一個用戶向伺服器發送了**SYN**報文後突然死機或掉 線，那麼伺服器在發出**SYN+ACK**應答報文後是無法收到用戶端的**ACK**報文的（第三次握手無 法完成），這種情況下伺服器端一般會重試（再次發送**SYN+ACK**給用戶端）並等待一段時 間後丟棄這個未完成的連接，這段時間的長度我們稱爲**SYN Timeout**，一般來說這個時間 是分鐘的數量級（大約爲**30**秒**-2**分鐘）；一個用戶出現異常導致伺服器的一個線程等待**1** 分鐘並不是什麼很大的問題，但如果有一個惡意的攻擊者大量類比這種情況，伺服器端將 爲了維護一個非常大的半連接列表而消耗非常多的資源**----**

- 幾種簡單的解決方法 ：
  - 第一種是縮短SYN Timeout時間，由於SYN Flood攻擊的效果取決於伺服器上保持的SYN半連接數，這個值=SYN攻擊的頻度 x SYN Timeout，所以通過縮短從接收到SYN報文到確定這個報文無效並丟棄改連接的時間，例如 設置爲20秒以下（過低的SYN Timeout設置可能會影響客戶的正常訪問），可以成倍的降 低伺服器的負荷。
  - 第二種方法是設置SYN Cookie，就是給每一個請求連接的IP位址分配一個Cookie，如果短 時間內連續受到某個IP的重複SYN報文，就認定是受到了攻擊，以後從這個IP地址來的包 會被一概丟棄。
- 但上述的方法只能對付較原始的**SYN Flood**攻擊，縮短**SYN Timeout**時間僅在對方 攻擊頻度不高的情況下生效，**SYN Cookie**更依賴于對方使用真實的**IP**位址，如果攻擊者以 數萬/秒的速度發送**SYN**報文，同時利用**SOCK_RAW**隨機改寫**IP**報文中的來源位址，以上的方法 將毫無用武之地。

TCP 之連線結束

- 由於 **TCP** 爲全雙工（**Full-duplex**）的通訊協定，兩邊之連線都可以獨立進行結束連線
- 如果只有單方面結束連線稱之爲**Half-Close**，在 **Half-Close** 之後過一陣子，**TCP** 會要求另外一個方向的連線也結束

TCP 之連線結束

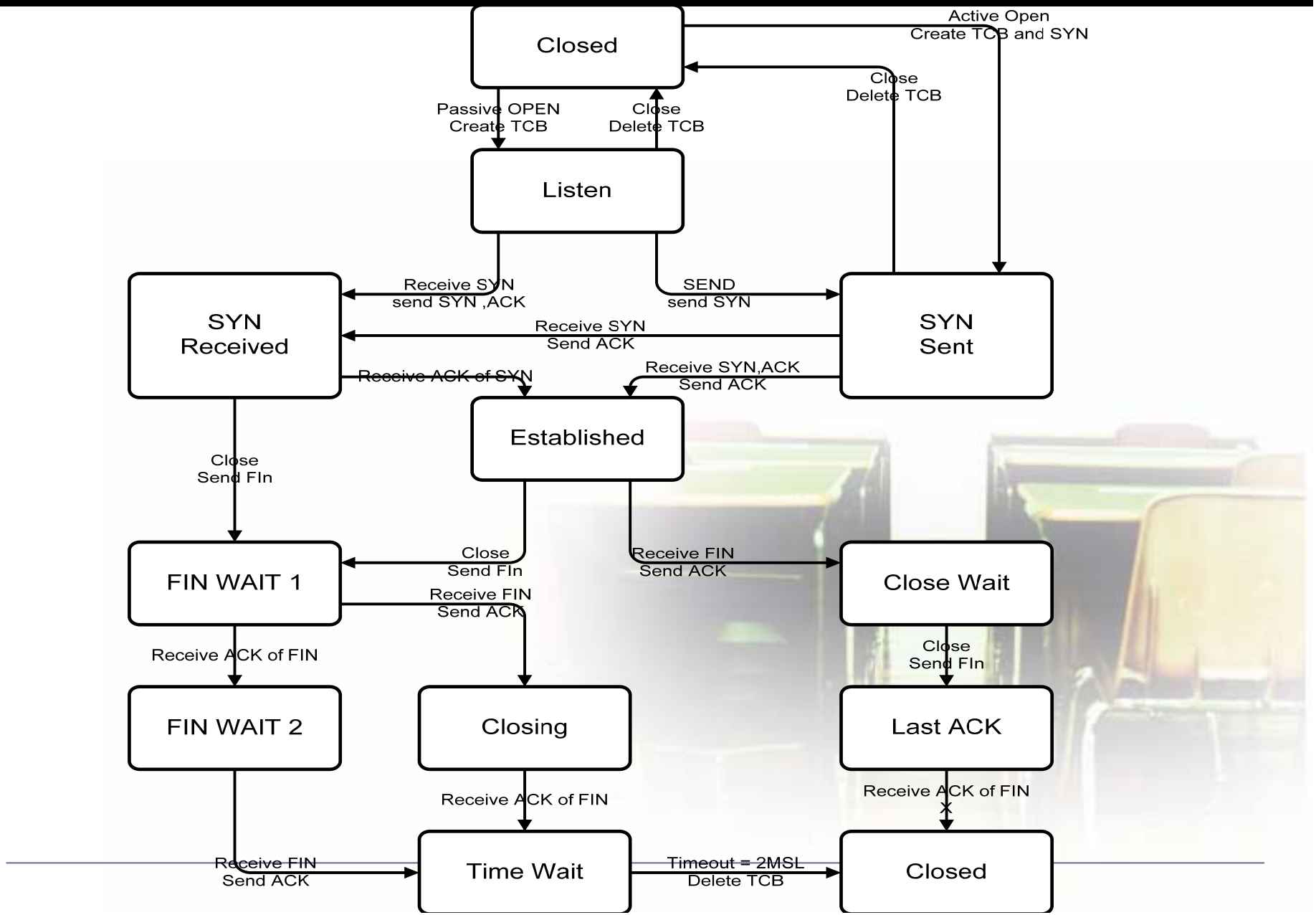

Client
Server
FIN
ACK of FIN
FIN
ACK of FIN
Connection Terminated

用一個完整的例子來說明 TCP

No	Source	Destination	Protocol	Info
1	192.168.0.1	192.168.0.2	TCP	32784 > 9999 [SYN] Seq=514208937 Ack=0 Win=5840 Len=0
2	192.168.0.2	192.168.0.1	TCP	9999 > 32784 [SYN,ACK] Seq=1256563805 Ack=514208938 Win=5792 Len=0
3	192.168.0.1	192.168.0.2	TCP	32784 > 9999 [ACK] Seq=514208938 Ack=1256563806 Win=5840 Len=0
4	192.168.0.1	192.168.0.2	TCP	32784 > 9999 [PSH,ACK] Seq=514208938 Ack=1256563806 Win=5840 Len=13
5	192.168.0.2	192.168.0.1	TCP	9999 > 32784 [ACK] Seq=1256563806 Ack=514208951 Win=5792 Len=0
6	192.168.0.2	192.168.0.1	TCP	9999 > 32784 [PSH,ACK] Seq=1256563806 Ack=514208951 Win=5792 Len=12
7	192.168.0.1	192.168.0.2	TCP	32784 > 9999 [ACK] Seq=514208951 Ack=1256563818 Win=5840 Len=0
8	192.168.0.2	192.168.0.1	TCP	9999 > 32784 [FIN,ACK] Seq=1256563818 Ack=514208951 Win=5792 Len=0
9	192.168.0.1	192.168.0.2	TCP	32784 > 9999 [FIN,ACK] Seq=514208951 Ack=1256563819 Win=5840 Len=0
10	192.168.0.2	192.168.0.1	TCP	9999 > 32784 [ACK] Seq=1256563819 Ack=514208952 Win=5792 Len=0

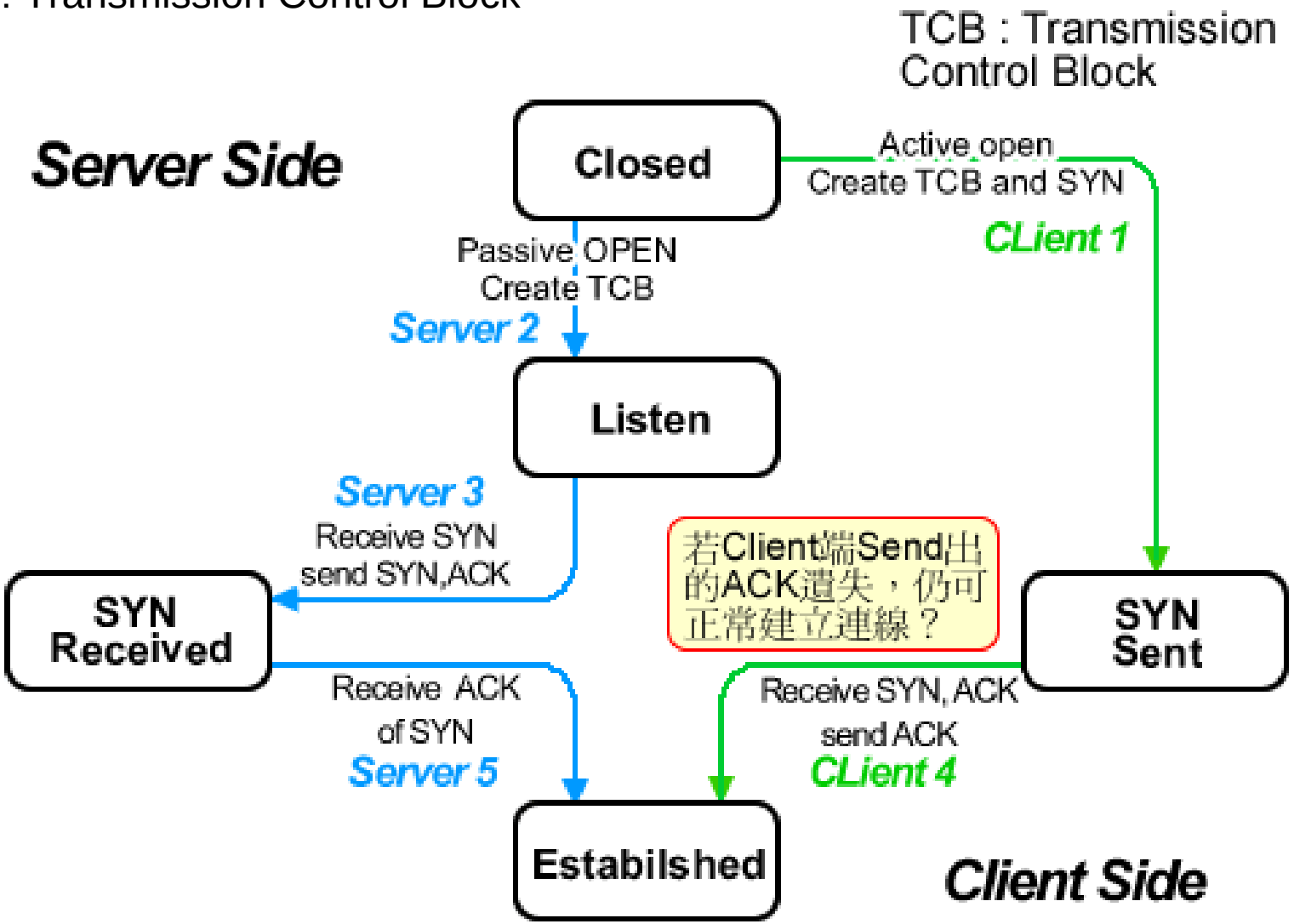
實驗 4.3 觀察TCP之連線及中斷連線

TCP Connection State Diagram



TCP Connection State Diagram 的 Connection Establish 部分

TCB: Transmission Control Block



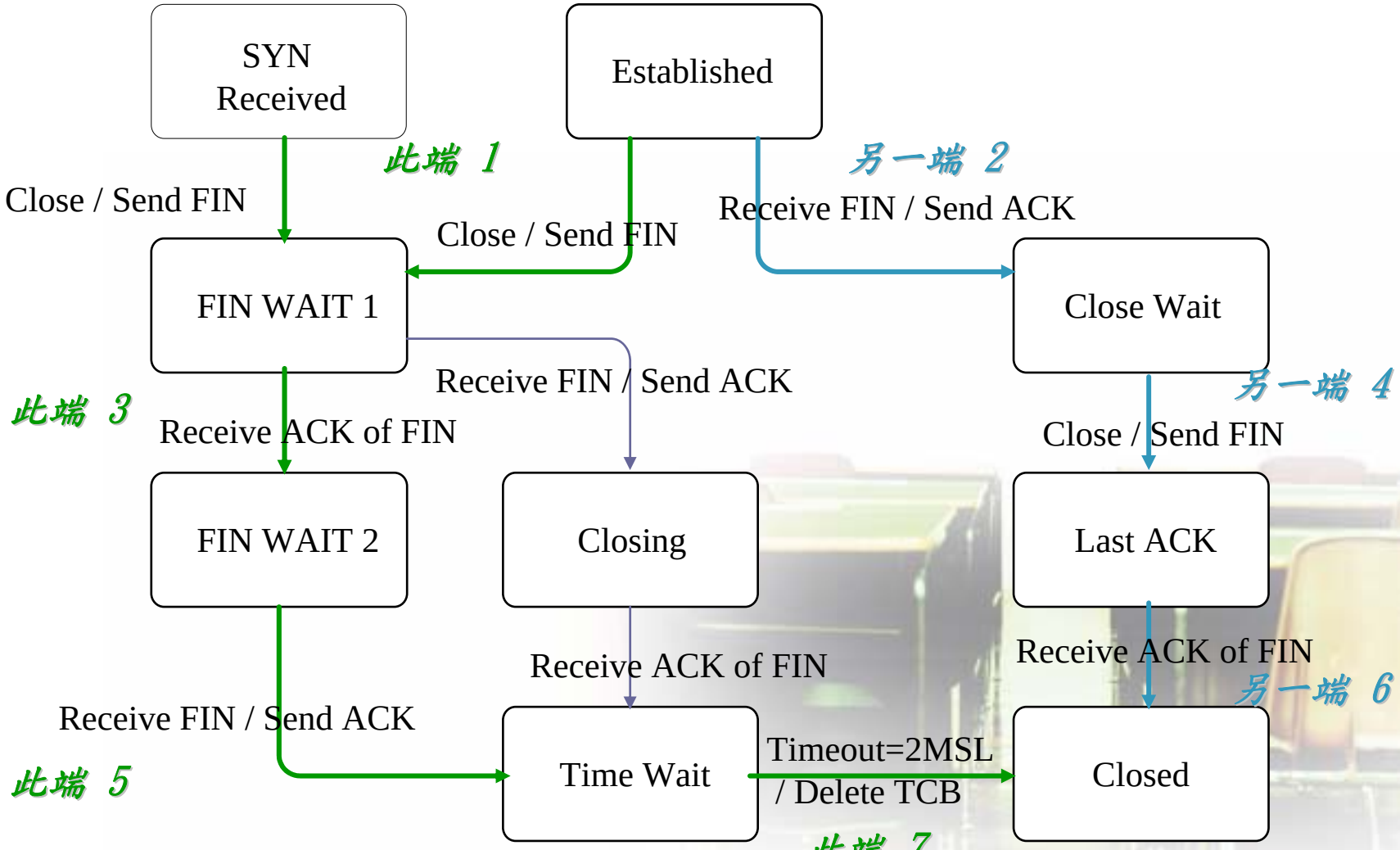
TCP Connection Establishment步驟

Client Side		Server Side	
1	Client initiates a connection to server	1	Server listens on a fixed port
2	Client sends SYN to server and then transit to SYN Sent state		
		2	Server receives SYN and then sends SYN/ACK to client, and transit to SYN Received state
3	Client receives SYN/ACK and then sends ACK to server, and transit to Established state		
4	Client connection is established	3	Server receives ACK (response) of SYN and then transit to Established state
		4	Server connection is established

TCP Connection State Diagram 的 Disconnection Establish 部分

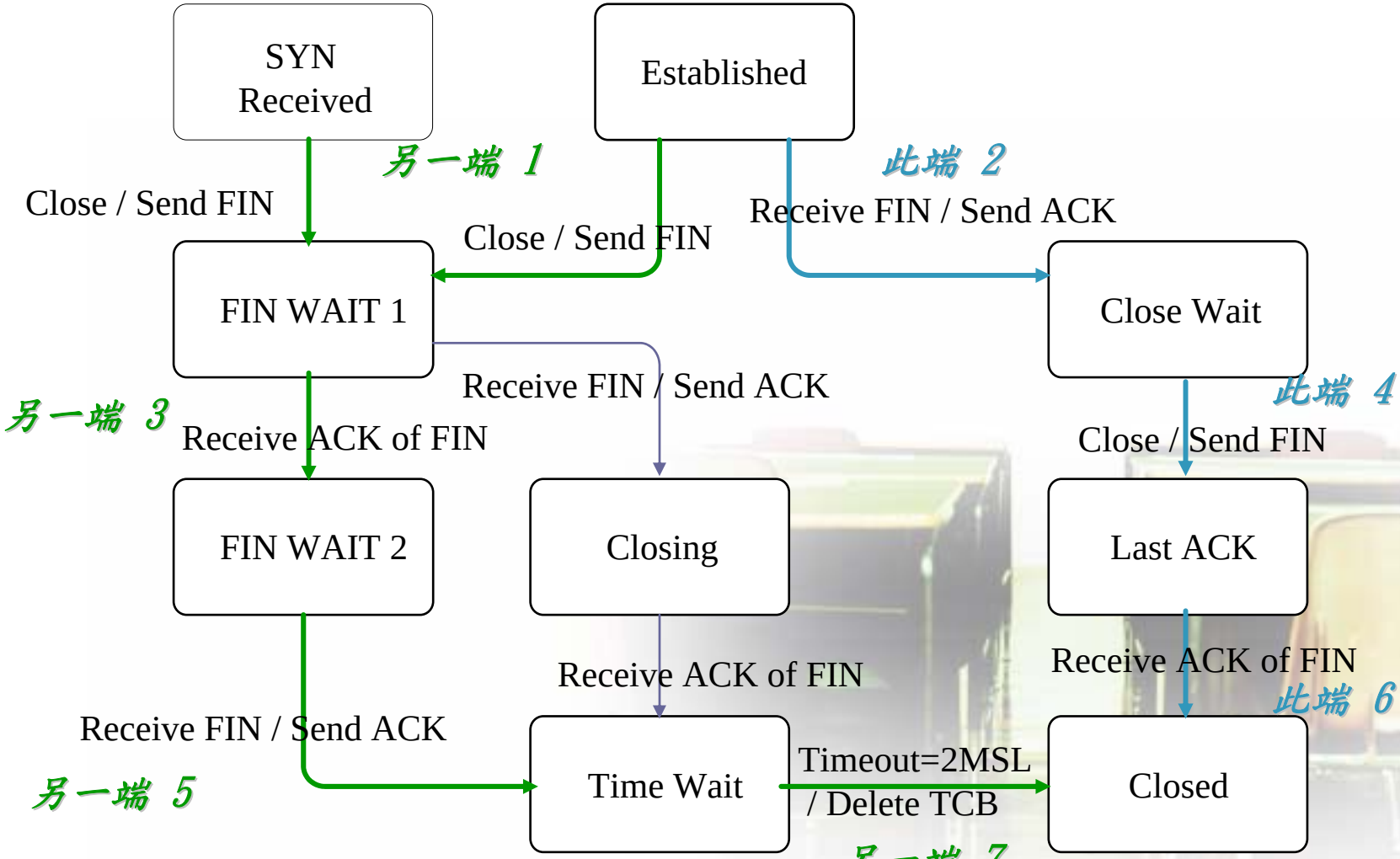
- 此端先關閉
  - Established -> FIN_WAIT_1 -> FIN_WAIT_2 -> Time_Wait -> Closed
- 另一端先關閉
  - Established -> CLOSE_WAIT -> LAST_ACK -> Closed
- 兩端同時關閉
  - Established -> FIN_WAIT_1 -> Closing -> Time_Wait -> Closed

此端先關閉



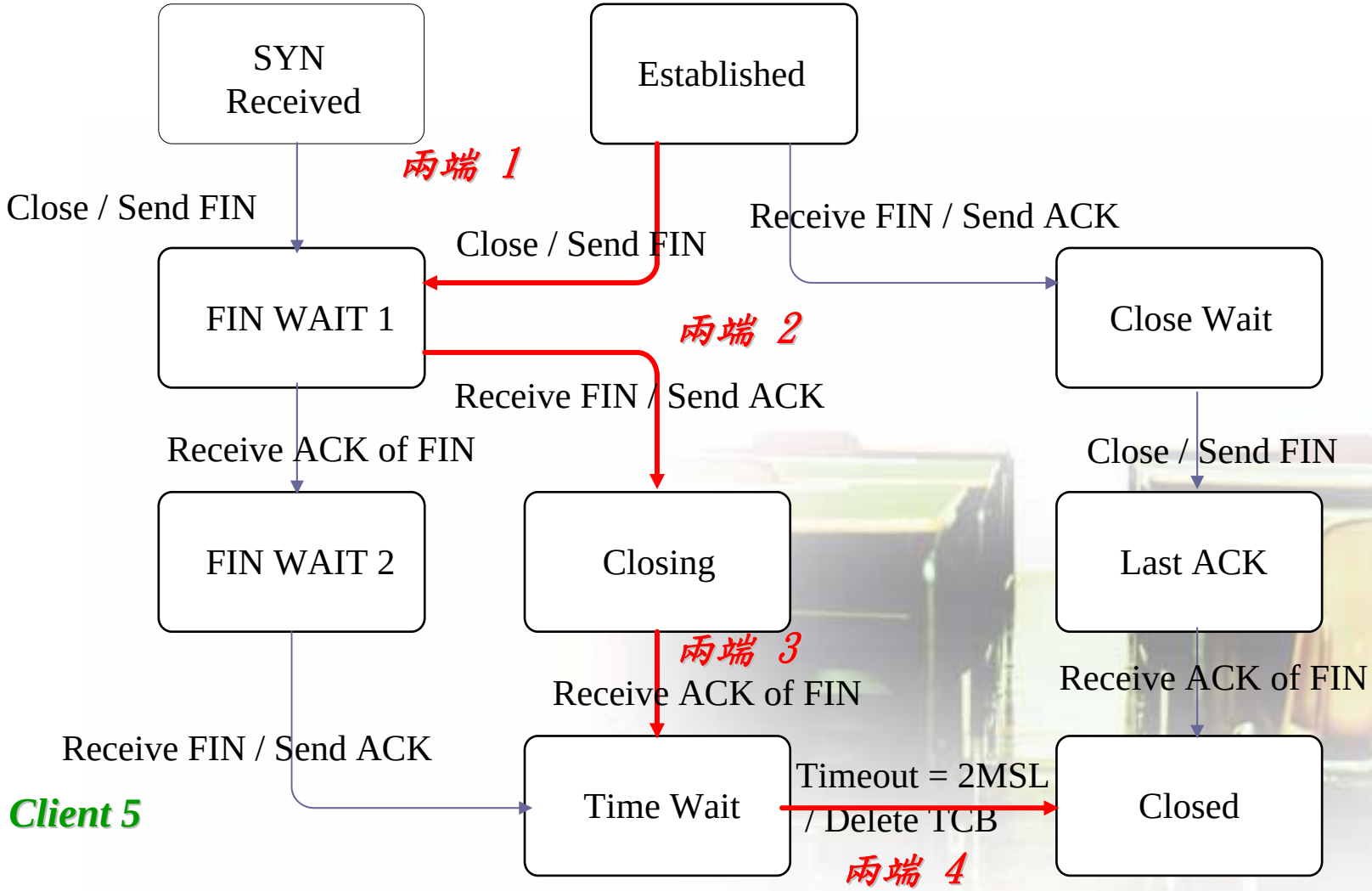
MSL : Maximum Segment Lifetimes

另一端先關閉



MSL : Maximum Segment Lifetimes

兩端同時關閉



MSL : Maximum Segment Lifetimes

TCP 通訊協定的可靠性

- **Checksum**：確保**Packet** 正確
- **Sequence Number**：用來確保**Packet** 依照順序
- **Positive Acknowledgment (ACK)**：**Packet Loss**或錯誤重傳的策略
- **Window Size**：作流量控制**(Flow Control)**

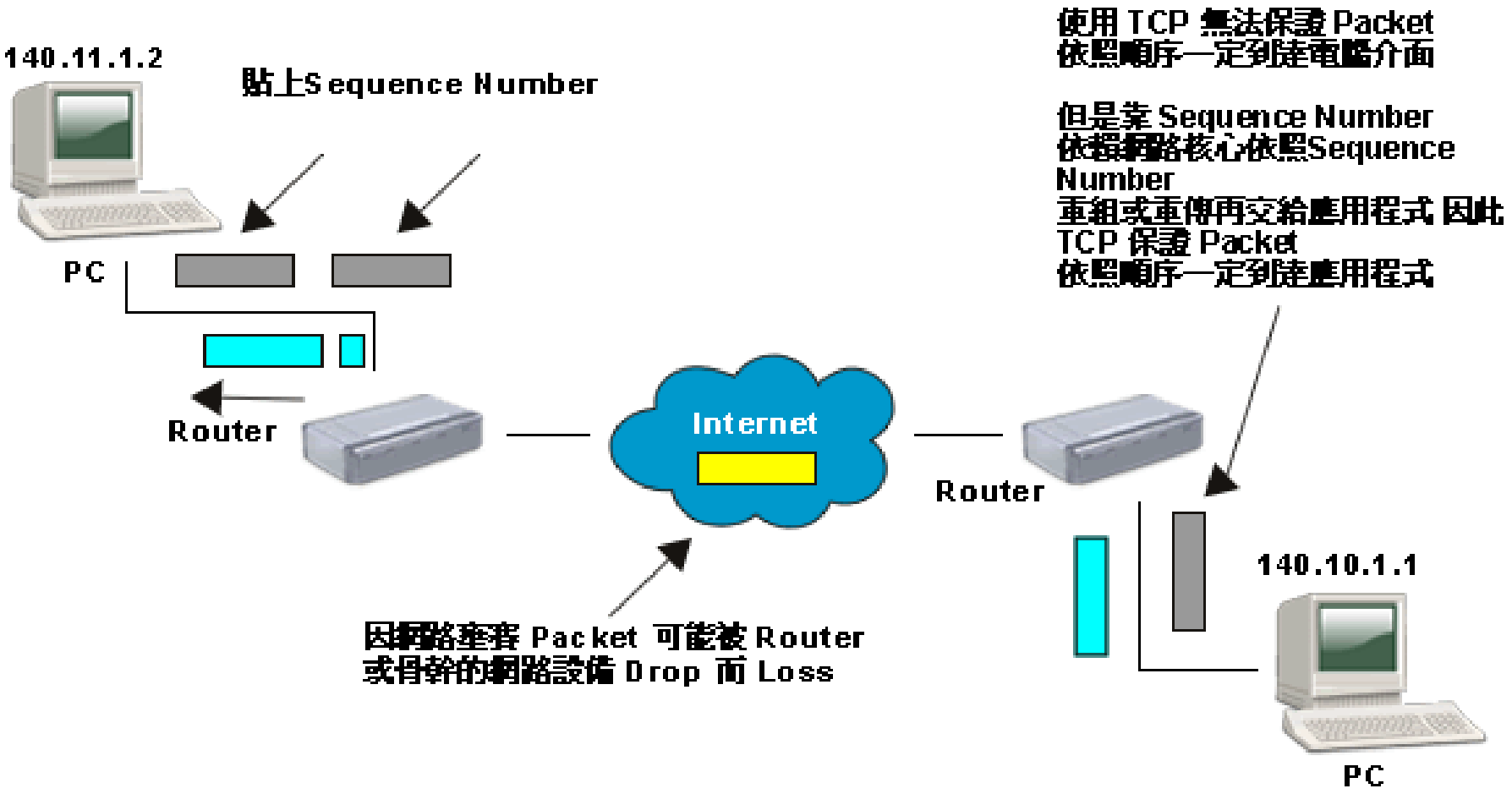
- **Checksum**：確保**Packet** 正確 (由**Pseudo Header**、**TCP Header**與**Data** 一起計算而得，如果接收端計算結果與**Checksum** 不同，就會認爲這**TCP Packet** 錯誤而要求傳送端重傳)
- **Sequence Number**：用來確保**Packet** 依照順序 (每個**Packet**都必須加上**Sequence Number**，如此才知道**Packet** 重組時的順序)
- **Positive Acknowledgment (ACK)**：**Packet Loss** 或錯誤重傳的策略
- **Window Size**：作流量控制**(Flow Control)** 目的是當**Server** 或**Client** 端使用**TCP** 連線時，可以控制傳送的速率。因爲**Packet**傳送過程當中會因爲傳輸和通過中間**Switch/Router** 等因設備處理**Packet**產生延遲或因處理能力不足而造成**Packet Loss**，如此一來若設計成每傳送一**Packet** 都須等待**ACK**才傳下一**Packet**，將會大幅降低傳輸的效率，如此的做法稱爲**Stop-and-Go ARQ**（**Automatic Repeat Request**）。

TCP 通訊協定的可靠性 – Checksum

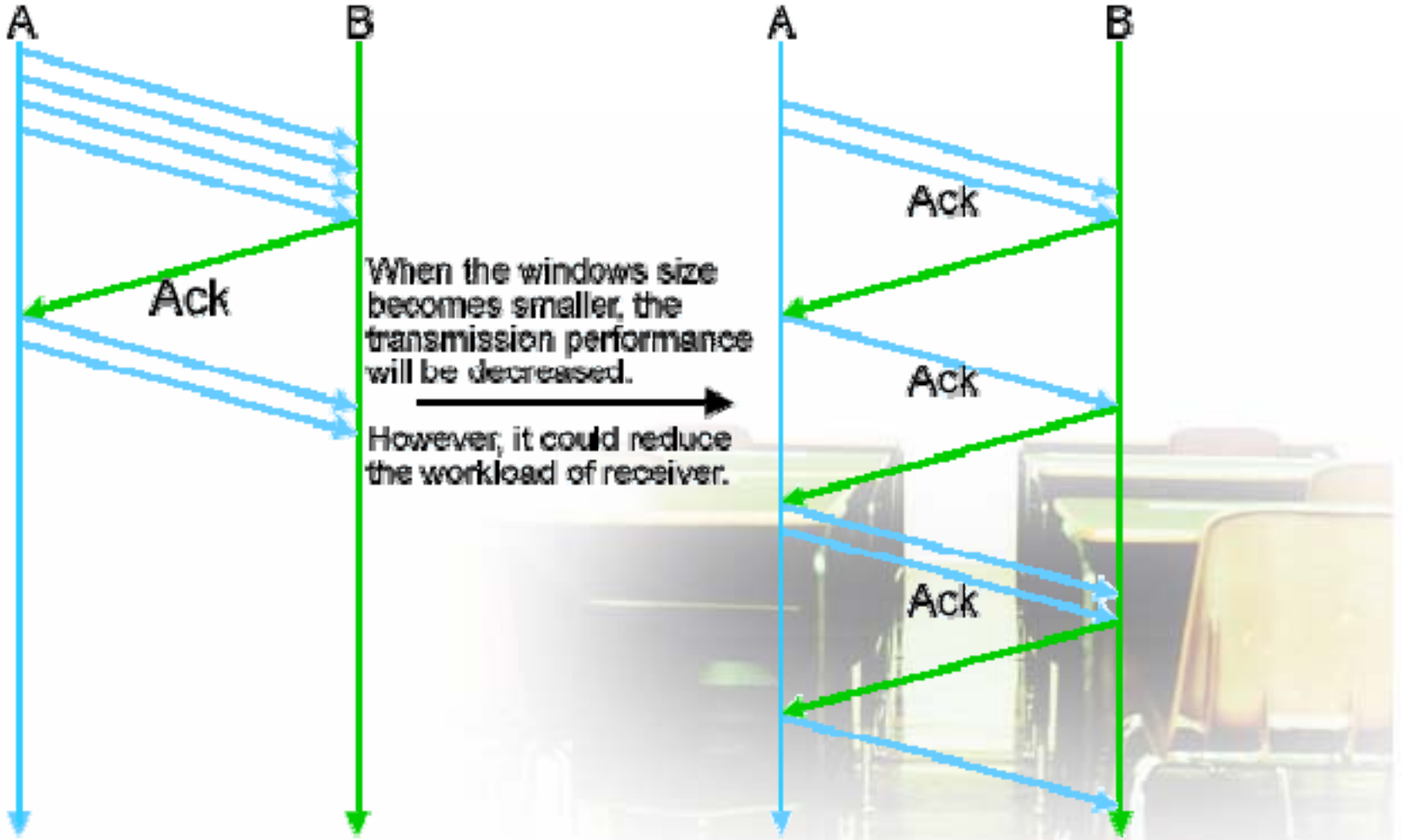
- 如同UDP，TCP 的 Checksum 也是由 Pseudo Header、TCP Header與Data 一起計算而得
- 如果接收端計算結果與Checksum 不同，就會認爲這TCP Packet 錯誤而要求傳送端重傳

實驗 4.1 比較IP checksum及TCP checksum

TCP 通訊協定的可靠性 – Sequence Number 的運用及ACK



TCP 通訊協定的可靠性
– Window Size 流量控制



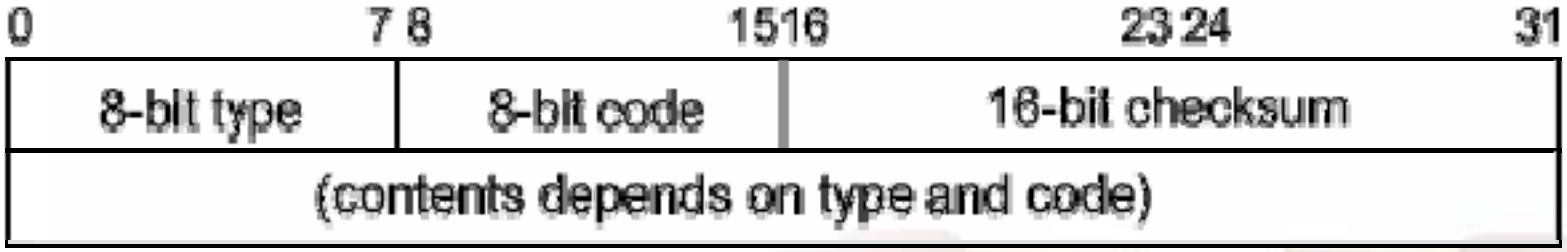
TCP 流量控制

3.3 ICMP 通訊協定

- 定義於 **RFC 792 (Internet Control Message Protocol)**
- 設計用來處理網路設備之間的錯誤的，報告網路環境中錯誤狀態的發生
- 沒辦法確保訊息確實送達或是轉回發送地
- **ICMP** 訊息基本上是用 **Datagram** 來傳送錯誤訊息，爲了避免無限制地傳送錯誤訊息，**ICMP** 訊息傳送的錯誤並不會產生另一個**ICMP** 訊息
- 訊息被分成幾個 **Fragmentation** 時，**ICMP** 訊息只會送第一個 **Fragmentation** 的錯誤訊息

ICMP 訊息

- 訊息格式是依照訊息種類不同而異，共通部分
  - TOS 欄位爲 0，Protocol 的欄位爲 1
  - ICMP 共通訊息部分



- Type : ICMP 訊息的類別
- Code 和下面的部分則是依各個訊息類別不同 (pp. 3-20~3-21)
- Checksum 的計算方式則是和 IP Checksum 相同

參考 http://www.iana.org/assignments/icmp-parameters

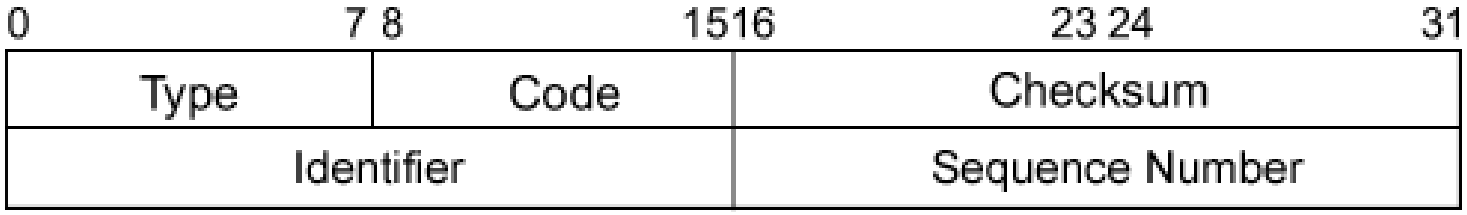
用 Ethereal 抓取到的畫面說明

ICMP 應用– ping

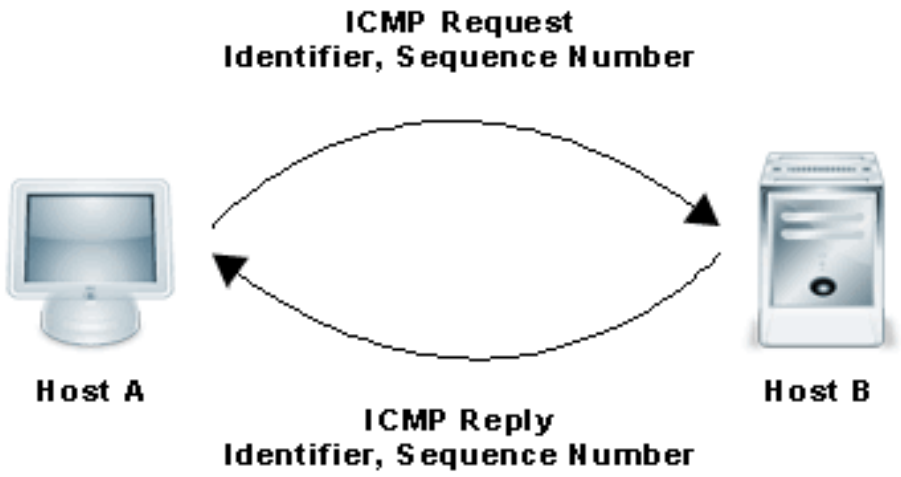
- 利用**ICMP** 訊息裡的 **Type 0** 和 **8(Echo Reply/Request)**
- 測試本機到網路某一端的機器是否有連線，或是測試遠端機器是否有開啓 **ping** 的回應

Type 0/8 : Echo Reply/Request

- **Echo Reply 回應回去的 Identifier 和 Sequence Number 會和收到 Echo Request 的相同 。**



ICMP Echo Reply/Request Datagram



舉一個ping例子說明

1. 192.168.0.80 對 192.168.0.10 發出 ICMP Type 8 Requset Packet

```
⊞ Frame 8 (74 bytes on wire, 74 bytes captured)
⊞ Ethernet II, Src: 00:40:f4:4c:1d:e5, Dst: 00:
⊞ Internet Protocol, Src Addr: 192.168.0.80 (19
⊟ Internet Control Message Protocol
    Type: 8 (Echo (ping) request)
    Code: 0
    Checksum: 0x465c (correct)
    Identifier: 0x0200
    Sequence number: 05:00
    Data (32 bytes)
```

2. 192.168.0.80 對 192.168.0.10 發出 ICMP Type 8 Requset Packet

```
⊞ Frame 9 (74 bytes on wire, 74 bytes captured)
⊞ Ethernet II, Src: 00:e0:4c:00:00:36, Dst: 00:4
⊞ Internet Protocol, Src Addr: 192.168.0.10 (19
⊟ Internet Control Message Protocol
    Type: 0 (Echo (ping) reply)
    Code: 0
    Checksum: 0x4e5c (correct)
    Identifier: 0x0200
    Sequence number: 05:00
    Data (32 bytes)
```

ICMP 應用– traceroute

- 利用 **IP Packet** 的 **TTL** 欄位來進行測試網路 **Packet** 經過哪些路徑
  - router 會在 IP Packet 的 TTL欄位爲 0 的時候，回送 ICMP Type 11 Time-to-Live Exceed 訊息的功能
- 在實作上有兩種方法
  - 送 UDP 封包製造 TTL = 0 而回應的狀況
  - 送 ICMP Echo Request 封包製造 TTL = 0 回應的狀況

traceroute範例

 # traceroute 168.95.1.1

traceroute to 168.95.1.1 (168.95.1.1), 30 hops max, 38 byte packets

1 61-218-177-193.HINET-IP.hinet.net (61.218.177.193) 0.917 ms 0.885 ms 0.844 ms

2 10.218.177.254 (10.218.177.254) 32.017 ms 34.033 ms 33.376 ms

3 tc-c6r1.router.hinet.net (168.95.144.202) 31.748 ms 32.338 ms 31.771 ms

4 tc-b-c12r1.router.hinet.net (168.95.254.129) 31.740 ms 32.338 ms 32.927 ms

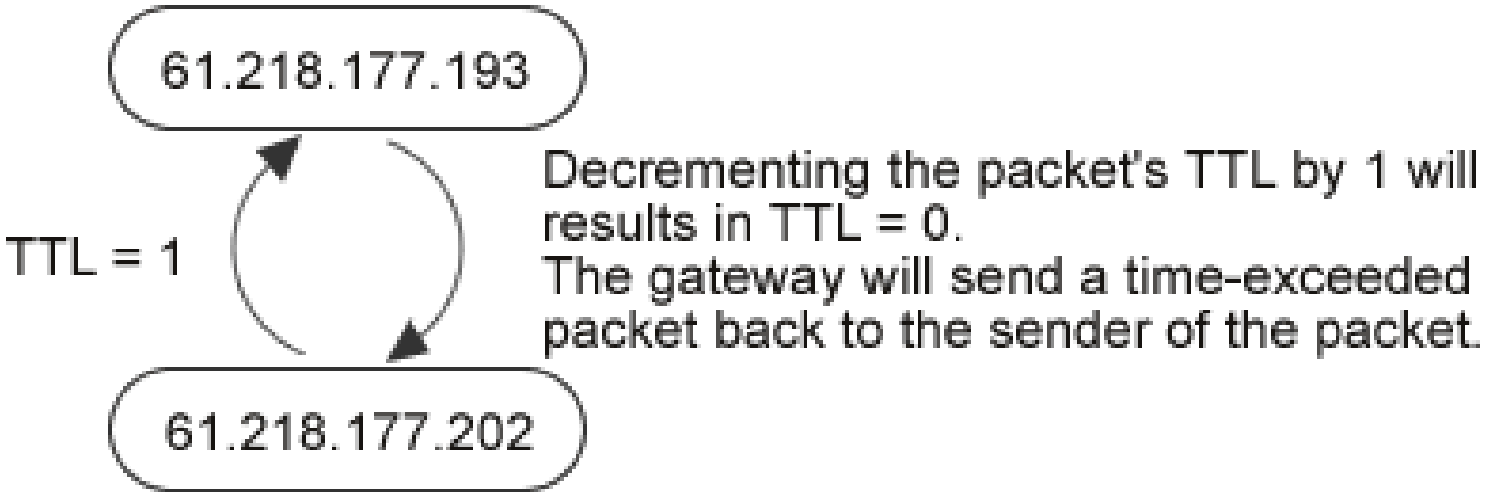
5 210.65.2.22 (210.65.2.22) 36.282 ms 35.720 ms 35.018 ms

6 tp-s2-c6r9.router.hinet.net (211.22.35.1) 33.380 ms 34.851 ms 35.006 ms

7 tp-b-c6r2.router.hinet.net (168.95.1.61) 33.369 ms 33.940 ms 33.387 ms

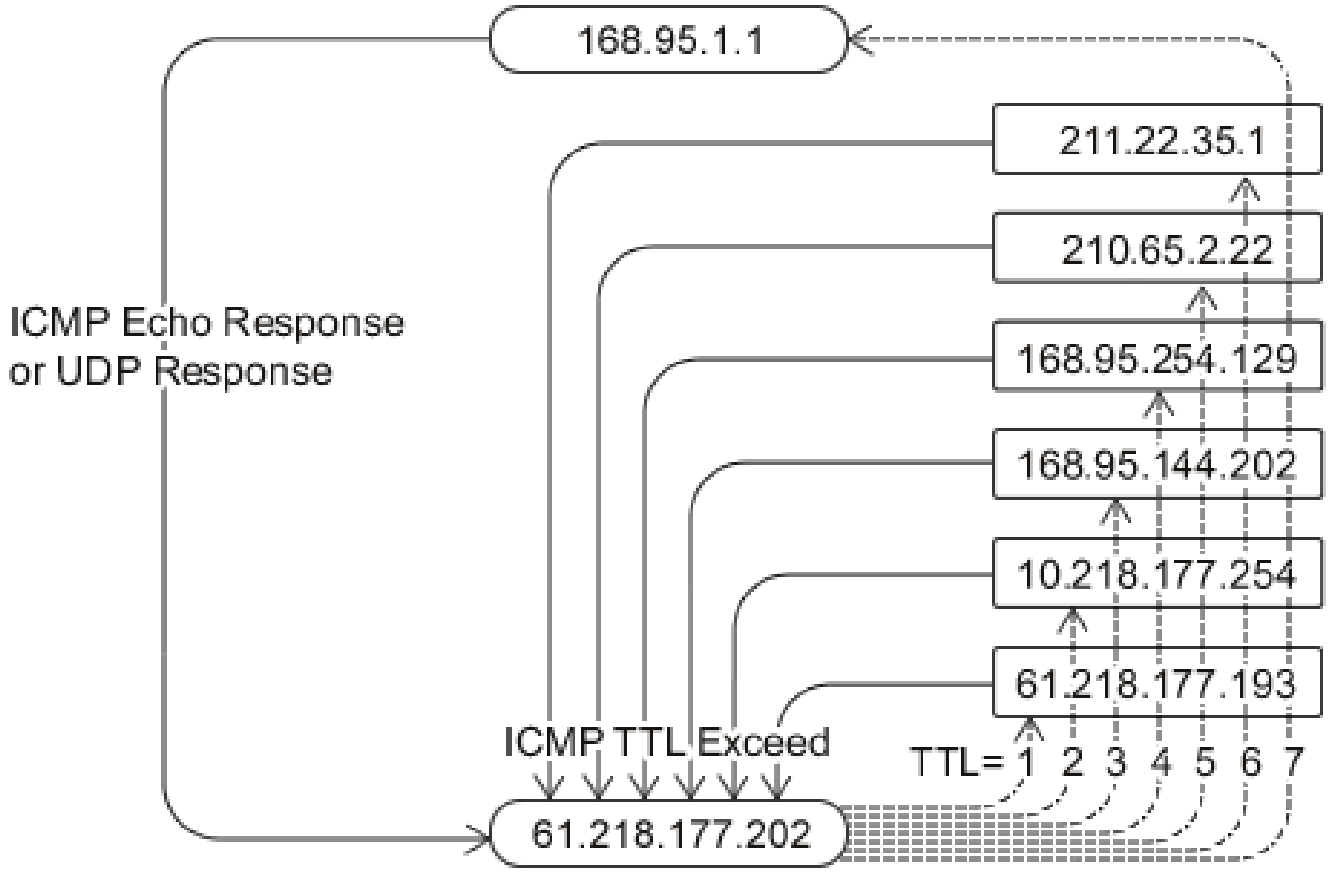
承上例，解釋第一個封包

- 第一個 **Packet** 的 **TTL** 設爲 **1**，往外傳送
- 第一個 **Router** 是 **61.218.177.193**，它收到 **Packet** 後將 **TTL** 減 **1** 後(變成 **0**)
- 於是 **61.218.177.193** 送一個 **ICMP Time Exceed Packet** 給 **61.218.177.202**



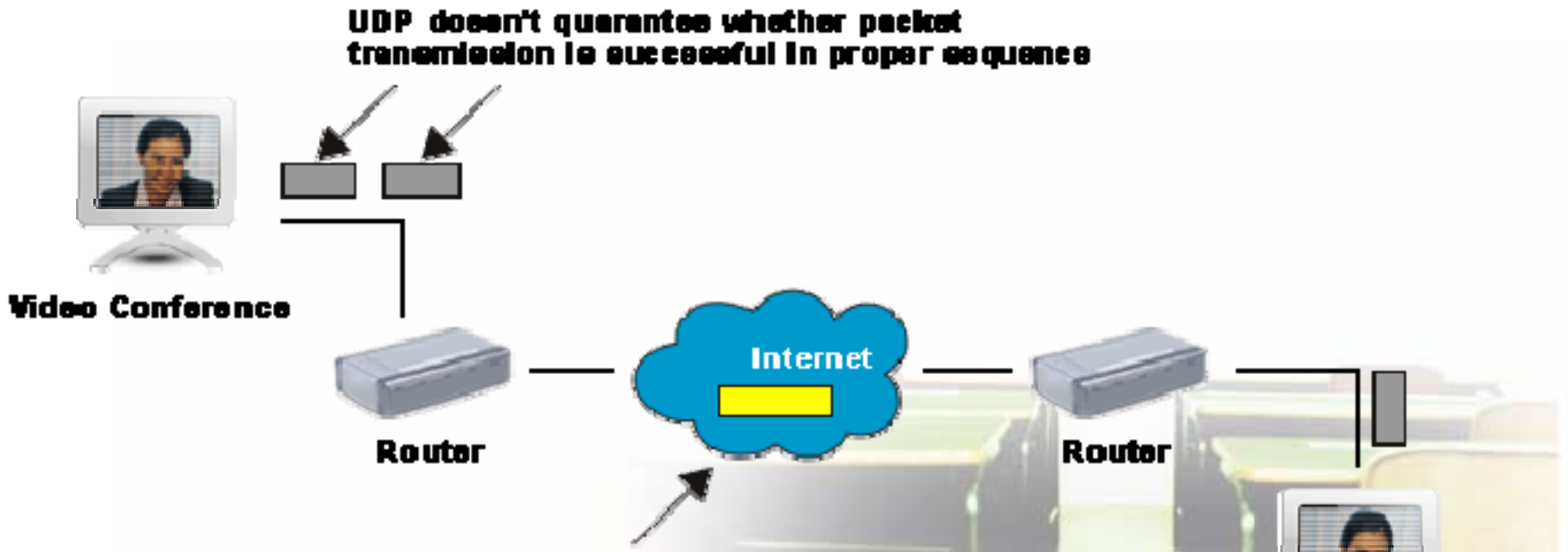
traceroute之packet 流程圖

- 每個 **Packet** 的 **TTL** 會累加，直到抵達 **168.95.1.1**




額外補充說明
CYUT NC

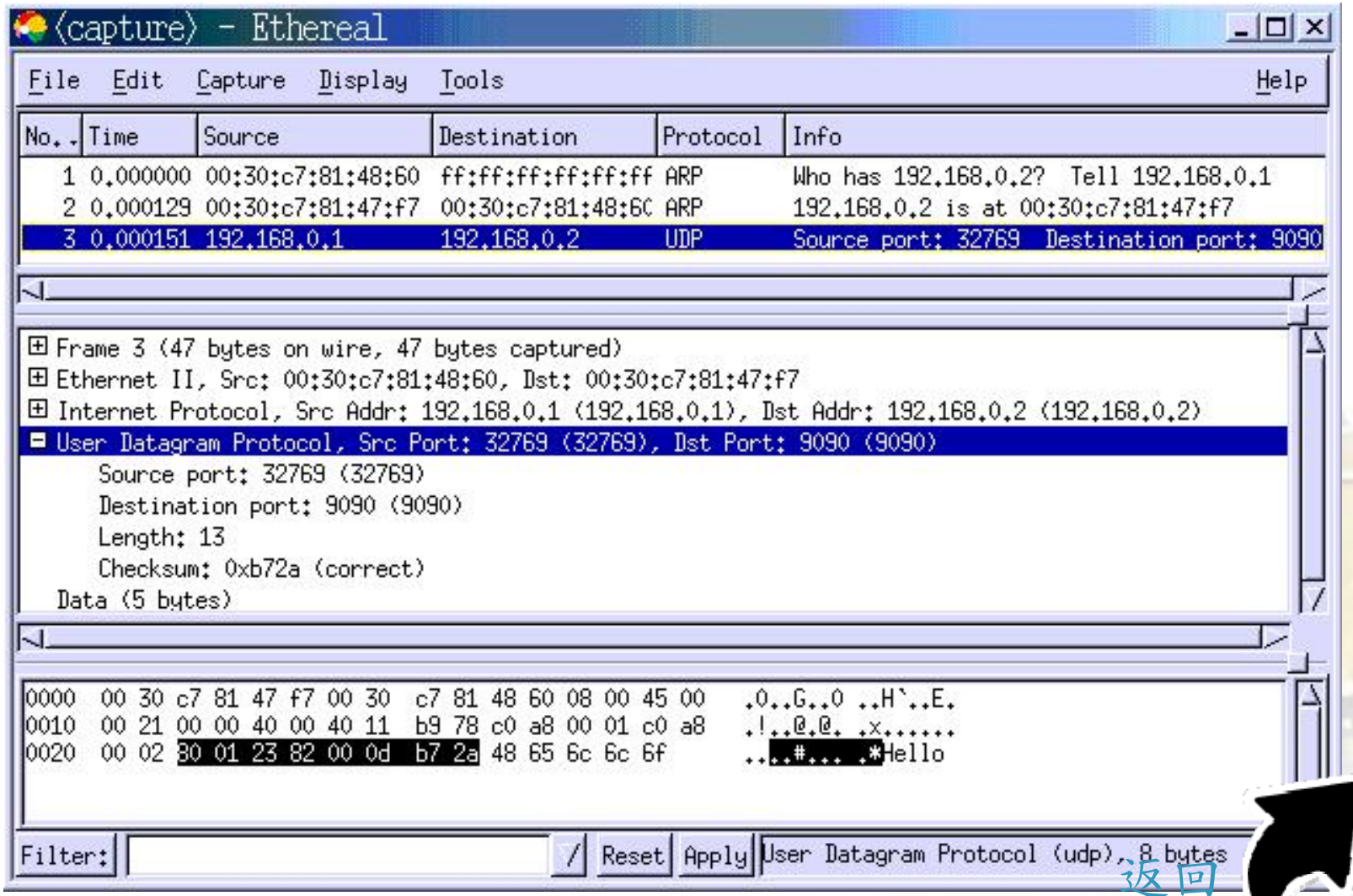
非連結導向(Connection-less oriented)



- Due to the Internet traffic jam, packet loss might occur. Data might be dropped by Router or other network device on the backbone.
- It is not necessary to retransmit the audio and video data if users may not mind some minor image losses.

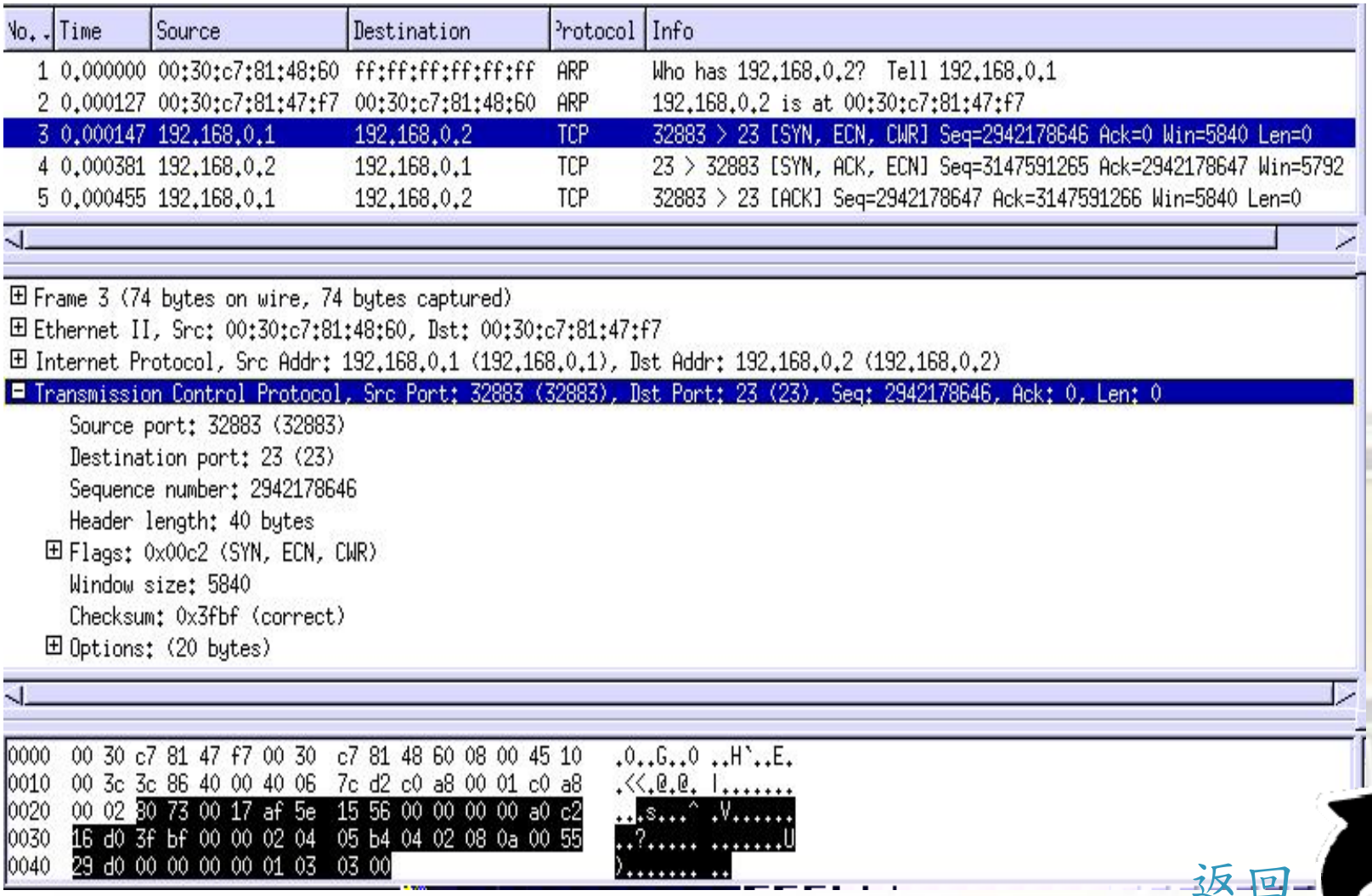
返回

UDP Header



返回

TCP Header



No.	Time	Source	Destination	Protocol	Info
1	0.000000	00:30:c7:81:48:60	ff:ff:ff:ff:ff:ff	ARP	Who has 192.168.0.2? Tell 192.168.0.1
2	0.000127	00:30:c7:81:47:f7	00:30:c7:81:48:60	ARP	192.168.0.2 is at 00:30:c7:81:47:f7
3	0.000147	192.168.0.1	192.168.0.2	TCP	32883 > 23 [SYN, ECN, CWR] Seq=2942178646 Ack=0 Win=5840 Len=0
4	0.000381	192.168.0.2	192.168.0.1	TCP	23 > 32883 [SYN, ACK, ECN] Seq=3147591265 Ack=2942178647 Win=5792
5	0.000455	192.168.0.1	192.168.0.2	TCP	32883 > 23 [ACK] Seq=2942178647 Ack=3147591266 Win=5840 Len=0

```
⊞ Frame 3 (74 bytes on wire, 74 bytes captured)
⊞ Ethernet II, Src: 00:30:c7:81:48:60, Dst: 00:30:c7:81:47:f7
⊞ Internet Protocol, Src Addr: 192.168.0.1 (192.168.0.1), Dst Addr: 192.168.0.2 (192.168.0.2)
⊟ Transmission Control Protocol, Src Port: 32883 (32883), Dst Port: 23 (23), Seq: 2942178646, Ack: 0, Len: 0
    Source port: 32883 (32883)
    Destination port: 23 (23)
    Sequence number: 2942178646
    Header length: 40 bytes
  ⊞ Flags: 0x00c2 (SYN, ECN, CWR)
    Window size: 5840
    Checksum: 0x3fbf (correct)
  ⊞ Options: (20 bytes)
```

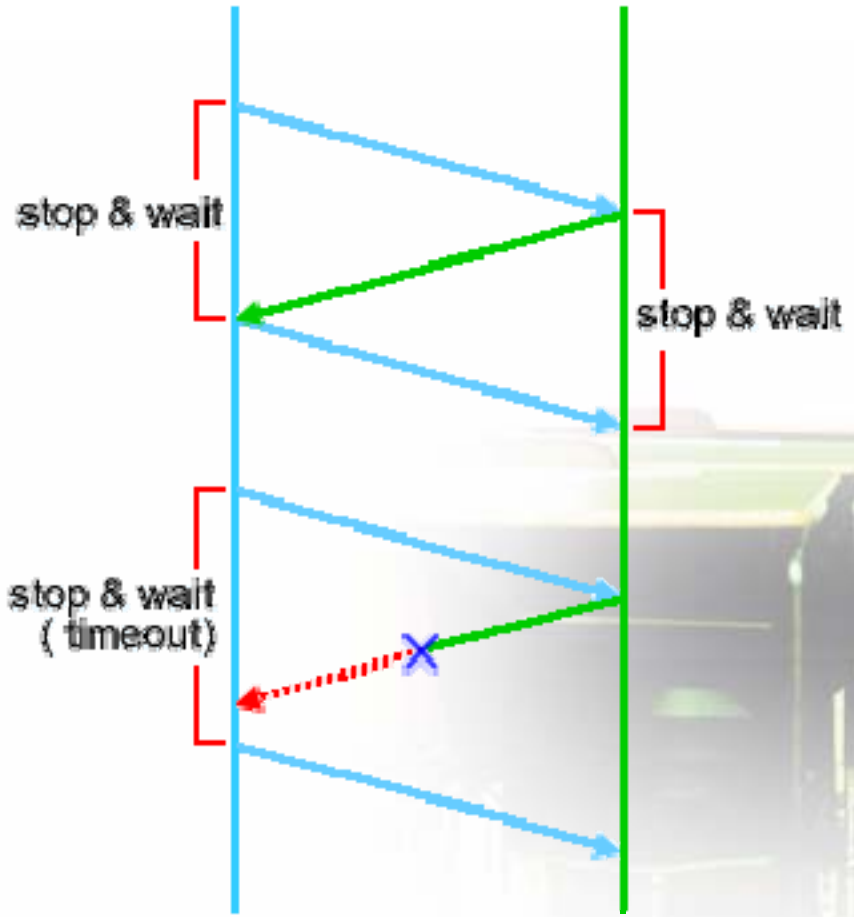
```
0000  00 30 c7 81 47 f7 00 30  c7 81 48 60 08 00 45 10   .0..G..0 ..H`..E.
0010  00 3c 3c 86 40 00 40 06  7c d2 c0 a8 00 01 c0 a8   .<<.@.@. |.......
0020  00 02 80 73 00 17 af 5e  15 56 00 00 00 00 a0 c2   ...s...^ .V......
0030  16 d0 3f bf 00 00 02 04  05 b4 04 02 08 0a 00 55   ..?..... .......U
0040  29 d0 00 00 00 00 01 03  03 00                     )....... ..
```

返回

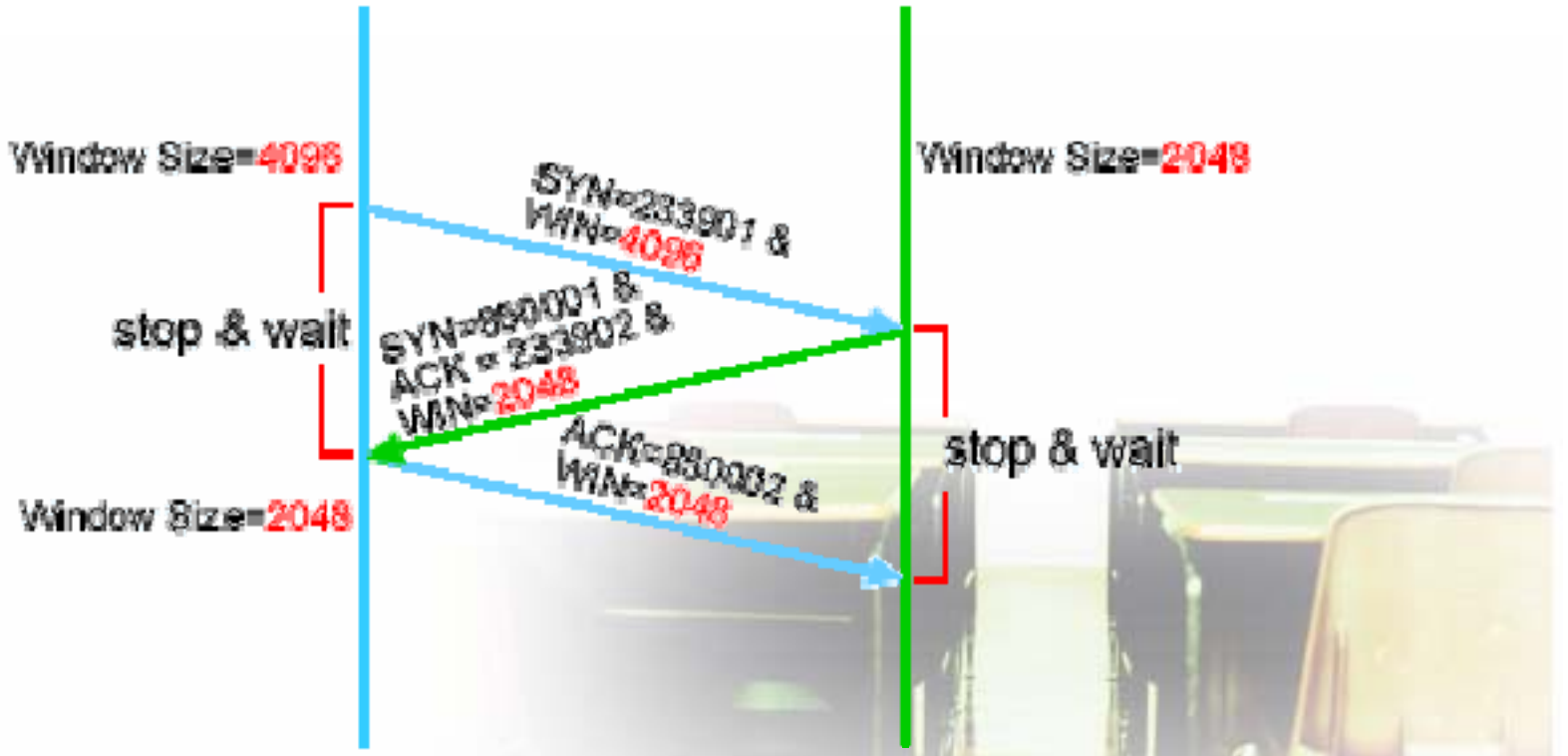
TCP 流量控制機制

- 停止與等待法 **(Stop-and-Wait)**
- 視窗大小 **(Window Size)**
- 滑動視窗法 **(Sliding Window)**

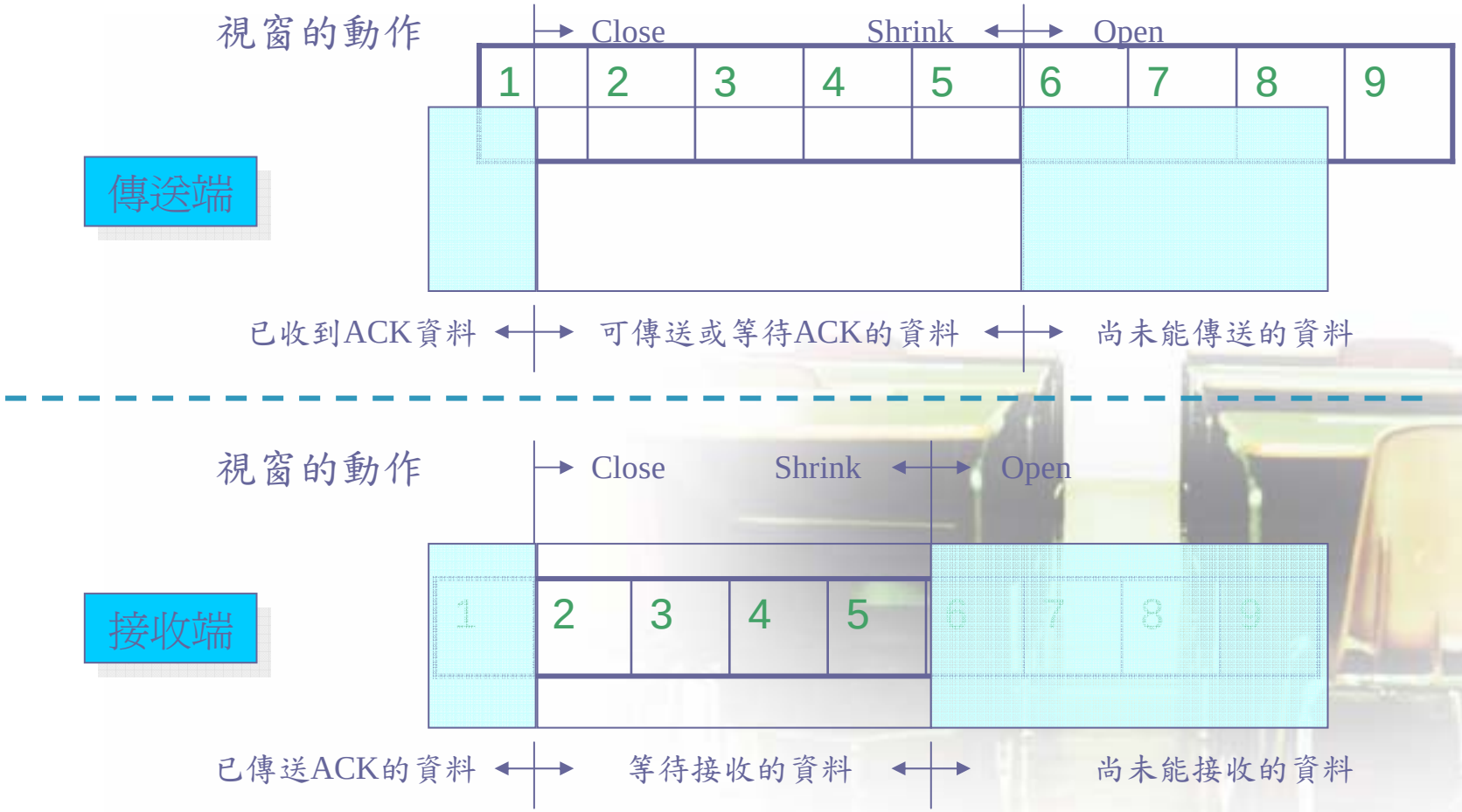
TCP 流量控制 - Stop-and-Wait



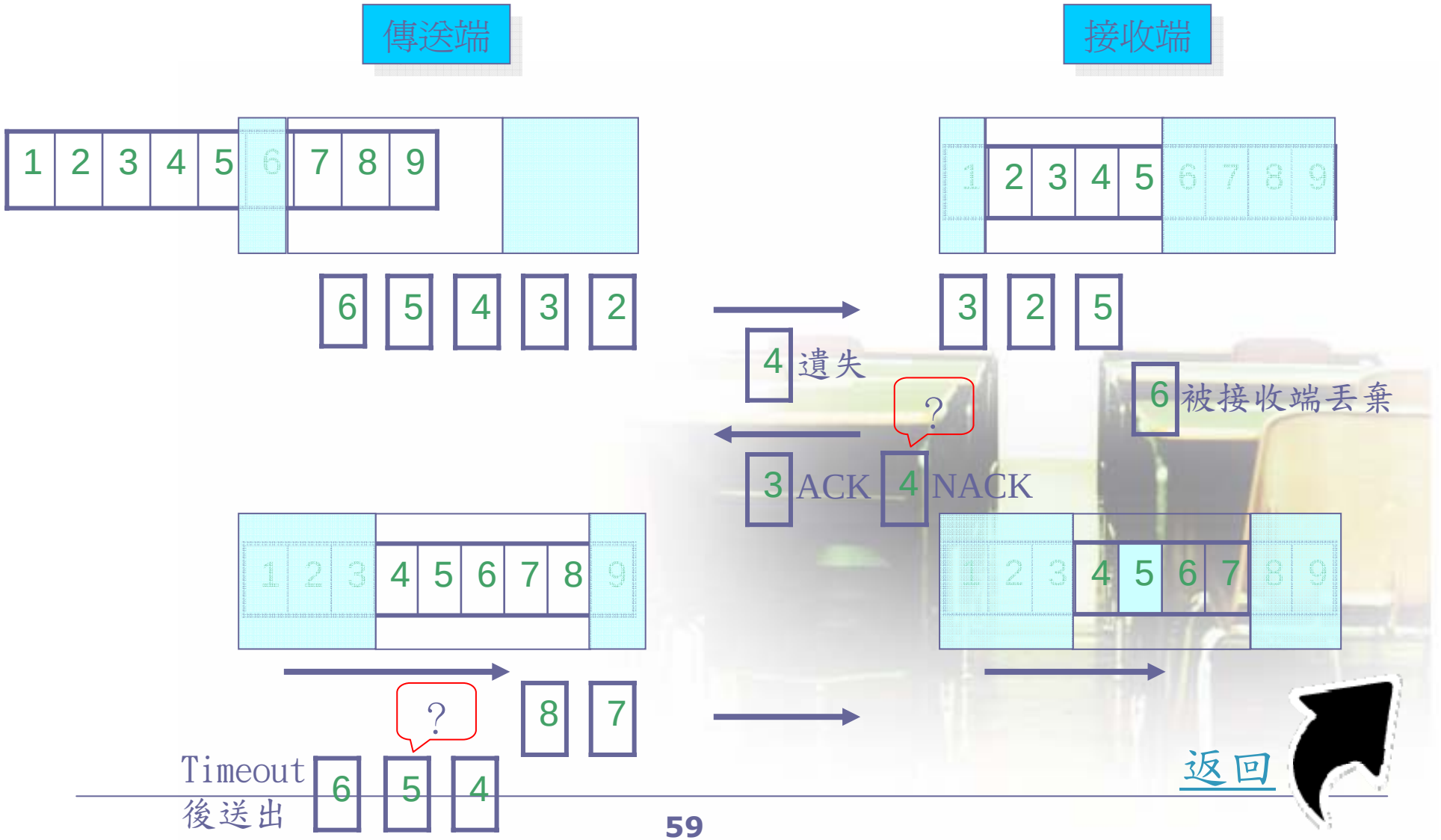
TCP 流量控制 – Window Size



Sliding Window名詞說明



TCP 流量控制 - Sliding Window



ICMP Header

No.	Time	Source	Destination	Protocol	Info
7	1.700550	192.168.251.22	192.168.251.251	ICMP	Echo (ping) request
8	1.701079	192.168.251.251	192.168.251.22	ICMP	Echo (ping) reply
9	2.695662	192.168.251.22	192.168.251.251	ICMP	Echo (ping) request
10	2.696145	192.168.251.251	192.168.251.22	ICMP	Echo (ping) reply
11	3.697765	192.168.251.22	192.168.251.251	ICMP	Echo (ping) request
12	3.698241	192.168.251.251	192.168.251.22	ICMP	Echo (ping) reply
13	4.698194	192.168.251.22	192.168.251.251	ICMP	Echo (ping) request
14	4.698613	192.168.251.251	192.168.251.22	ICMP	Echo (ping) reply

```
⊞ Frame 7 (74 bytes on wire, 74 bytes captured)
⊞ Ethernet II, Src: 00:e0:98:81:16:0d, Dst: 00:30:c7:81:48:5d
⊞ Internet Protocol, Src Addr: 192.168.251.22 (192.168.251.22), Dst Addr:
⊟ Internet Control Message Protocol
    Type: 8 (Echo (ping) request)
    Code: 0
    Checksum: 0x495c (correct)
    Identifier: 0x0300
    Sequence number: 01:00
    Data (32 bytes)
```

```
0000  00 30 c7 81 48 5d 00 e0  98 81 16 0d 08 00 45 00   .0..H].. ......
0010  00 3c 00 dc 00 00 80 01  c1 81 c0 a8 fb 16 c0 a8   .<...... ........
0020  fb fb 08 00 49 5c 03 00  01 00 61 62 63 64 65 66   ....I\.. ..abcdef
0030  67 68 69 6a 6b 6c 6d 6e  6f 70 71 72 73 74 75 76   ghijklmn opqrstuv
0040  77 61 62 63 64 65 66 67  68 69                    wabcdefg hi
```

實驗導引

實驗 4.1 IP checksum與TCP checksum

實驗 4.2 了解port number的涵意

實驗 4.3 觀察TCP之連線及中斷連線

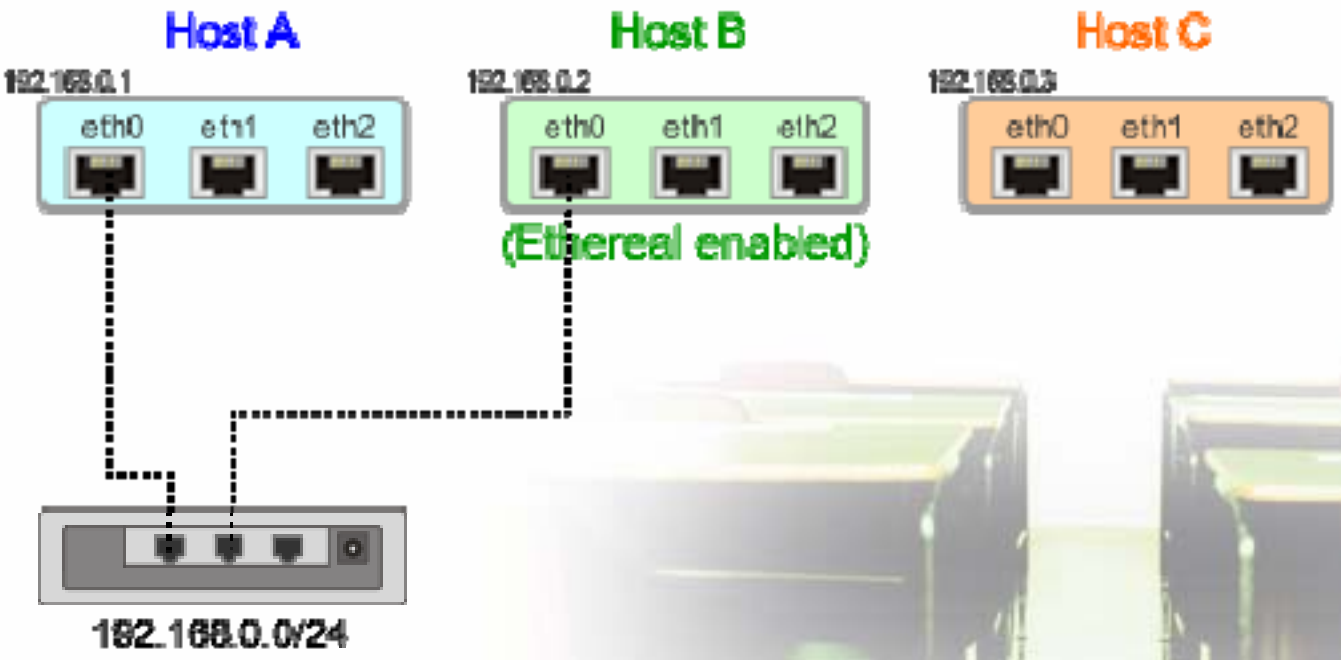
CYUT NC

實驗 4.1 IP checksum與TCP checksum

實驗目的

- 計算**IP checksum**
- 計算**TCP checksum**
- 比較**IP checksum**及**TCP checksum**

實驗架構圖


Host A
192.168.0.1
eth0
eth1
eth2
Host B
192.168.0.2
eth0
eth1
eth2
(Ethereal enabled)
Host C
192.168.0.3
eth0
eth1
eth2
192.168.0.0/24

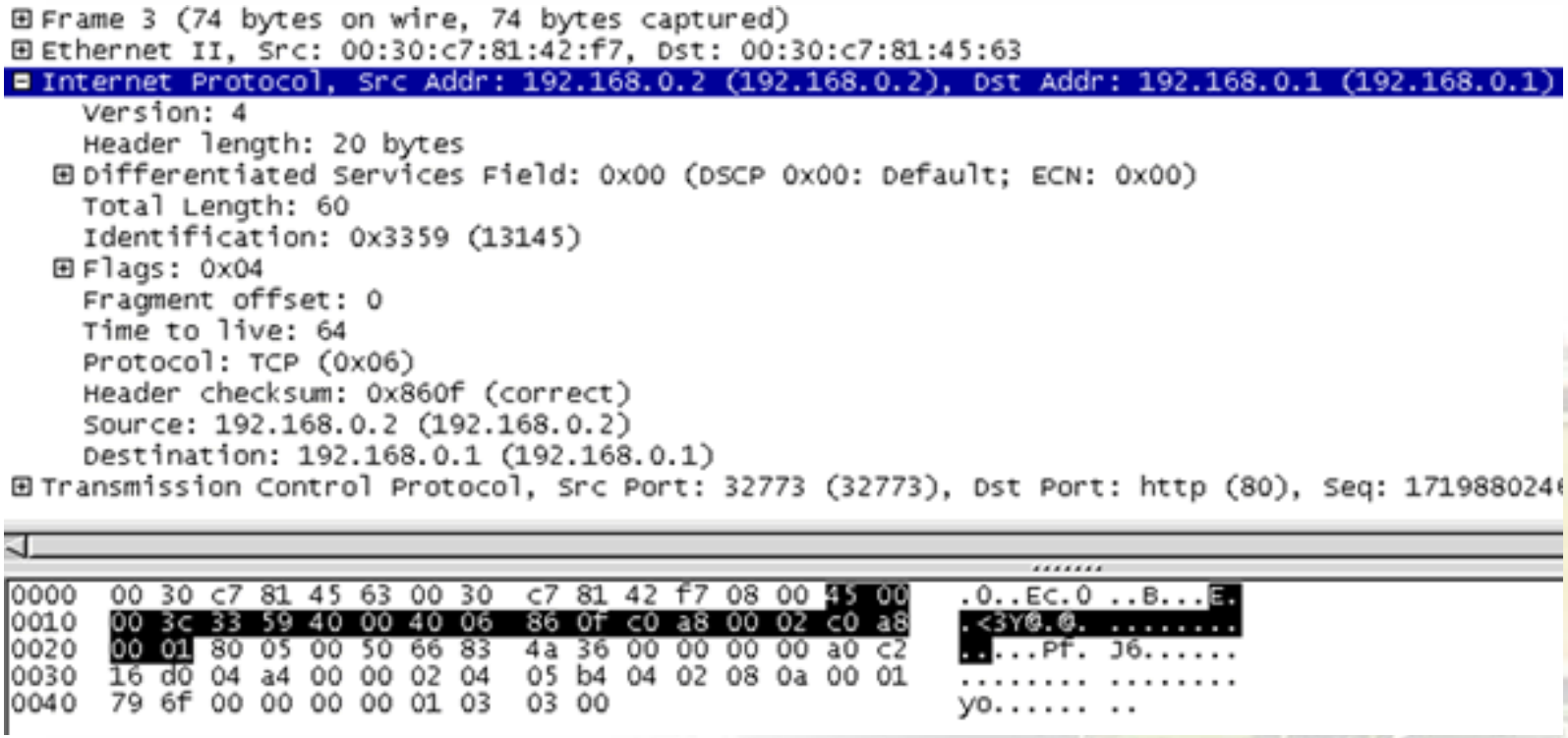
Step 1: 抓取TCP封包

❖ Host B:

- 開啓 Ethereal ，interface選eth0，以觀察下列步驟之封包
- 開啓browser，於網址列輸入192.168.0.1 (在NetGuru中每個Host皆內建Web server)
- 利用抓取到的封包來計算IP checksum及TCP checksum

Step 2: 計算IP checksum

- 抓取的封包展開**Internet Protocol**如下圖：



IP checksum計算方式

- **IP header** 的欄位，除了**checksum** 不列入計算，以**16 bits** 爲單位計算，如下表

4	5	0	0
0	0	3	c
3	3	5	9
4	0	0	0
4	0	0	6
c	0	a	8
0	0	0	2
c	0	a	8
0	0	0	1

4500+003c+3359+4000+4006+c0a8+0002+c0a8+0001=279ee

- 進位的2加回79ee中，2+79ee=79f0，換成二進位爲0111 1001 1111 0000，取其1‘s補數1000 0110 0000 1111，換成16進位爲860f(即爲checksum)

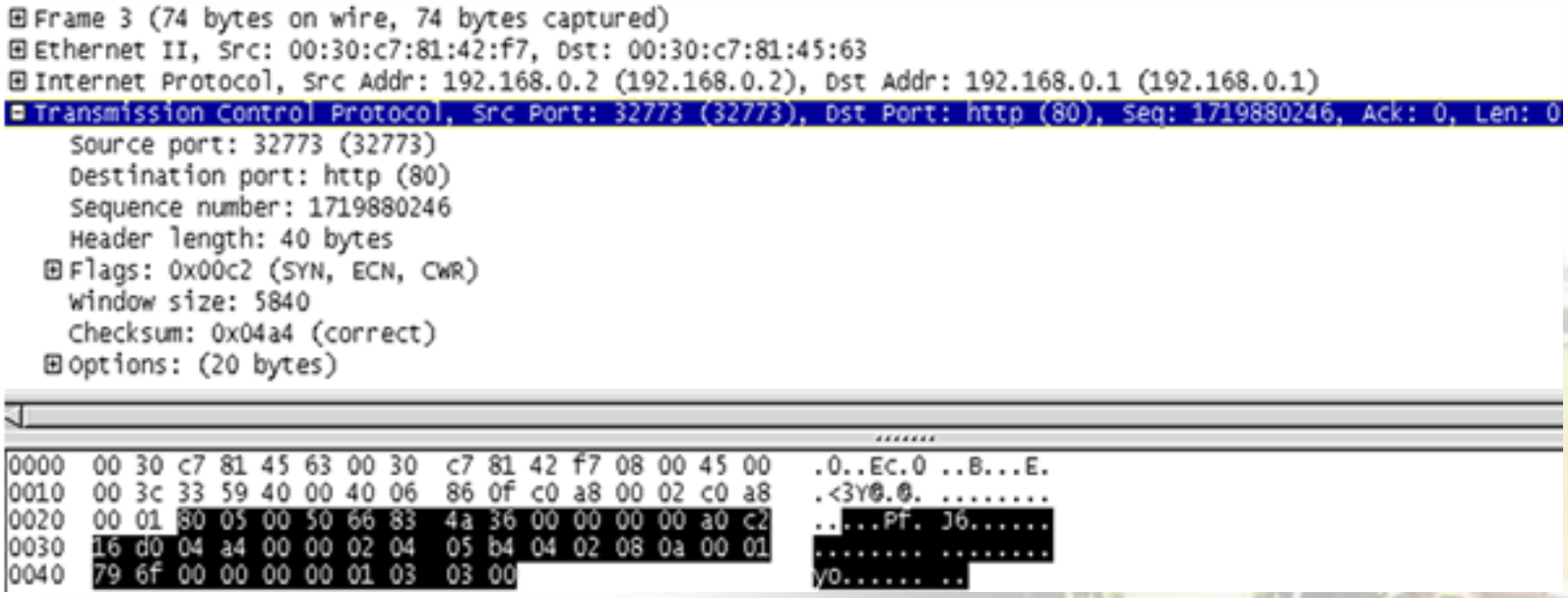
計算結果與原封包內容對照

```
⊞ Frame 3 (74 bytes on wire, 74 bytes captured)
⊞ Ethernet II, Src: 00:30:c7:81:42:f7, Dst: 00:30:c7:81:45:63
⊟ Internet Protocol, Src Addr: 192.168.0.2 (192.168.0.2), Dst Addr: 192.168.0.1 (192.168.0.1)
    Version: 4
    Header length: 20 bytes
  ⊞ Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00)
    Total Length: 60
    Identification: 0x3359 (13145)
  ⊞ Flags: 0x04
    Fragment offset: 0
    Time to live: 64
    Protocol: TCP (0x06)
    Header checksum: 0x860f (correct)
    Source: 192.168.0.2 (192.168.0.2)
    Destination: 192.168.0.1 (192.168.0.1)
⊞ Transmission Control Protocol, Src Port: 32773 (32773), Dst Port: http (80), Seq: 171988024
```

```
0000  00 30 c7 81 45 63 00 30  c7 81 42 f7 08 00 45 00   .0..Ec.0 ..B...E.
0010  00 3c 33 59 40 00 40 06  86 0f c0 a8 00 02 c0 a8   .<3Y@.@. ........
0020  00 01 80 05 00 50 66 83  4a 36 00 00 00 00 a0 c2   .....Pf. J6......
0030  16 d0 04 a4 00 00 02 04  05 b4 04 02 08 0a 00 01   ........ ........
0040  79 6f 00 00 00 00 01 03  03 00                     yo...... ..
```

Step 3: 計算TCP checksum

- 抓取的封包展開**Transmission Control Protocol**如下圖：



Pseudo Header

32-bit source IP address		
32-bit destination IP address		
zero	8-bit protocol	16-bit TCP length

- Source IP address = c0 a8 00 02 (192.168.0.2)
  Destination IP address = c0 a8 00 01(192.168.0.1)
  Protoco
  TCP len bytes)

c	0	a	8
0	0	0	2
c	0	a	8
0	0	0	1
0	0	0	6
0	0	2	8

TCP header及TCP data

8	0	0	5
0	0	5	0
6	6	8	3
4	a	3	6
0	0	0	0
0	0	0	0
a	0	c	2
1	6	d	0
0	0	0	0
0	2	0	4
0	5	b	4
0	4	0	2
0	8	0	a
0	0	0	1
7	9	6	f
0	1	0	3
0	3	0	0

Checksum欄位以0000取代

TCP checksum的計算方式

- 將 **TCP** 的 **Pseudo Header**、**TCP Header** 及其 **Data**加總後(**TCP checksum**欄位不列入計算)，若必要時會在後面補零，以確保是偶數長度
- 在本例，上列數字加起來爲**3fb58**，將溢位的”**3**”加回**fb58**結果爲 **fb5b**，換成二進位爲 **1111 1011 0101 1011**，取其補數**0000 0100 1010 0100**，轉換成**16**進位結果爲 **04a4**

計算結果與原封包內容對照

```
⊞ Frame 3 (74 bytes on wire, 74 bytes captured)
⊞ Ethernet II, Src: 00:30:c7:81:42:f7, Dst: 00:30:c7:81:45:63
⊞ Internet Protocol, Src Addr: 192.168.0.2 (192.168.0.2), Dst Addr: 192.168.0.1 (192.168.0.1)
⊟ Transmission Control Protocol, Src Port: 32773 (32773), Dst Port: http (80), Seq: 1719880246, Ack: 0, Len: 0
    Source port: 32773 (32773)
    Destination port: http (80)
    Sequence number: 1719880246
    Header length: 40 bytes
  ⊞ Flags: 0x00c2 (SYN, ECN, CWR)
    Window size: 5840
    Checksum: 0x04a4 (correct)
  ⊞ Options: (20 bytes)
```

```
0000  00 30 c7 81 45 63 00 30  c7 81 42 f7 08 00 45 00   .0..Ec.0 ..B...E.
0010  00 3c 33 59 40 00 40 06  86 0f c0 a8 00 02 c0 a8   .<3Y@.@. ........
0020  00 01 80 05 00 50 66 83  4a 36 00 00 00 00 a0 c2   .....Pf. J6......
0030  16 d0 04 a4 00 00 02 04  05 b4 04 02 08 0a 00 01   ........ ........
0040  79 6f 00 00 00 00 01 03  03 00                     yo...... ..
```

❖ 問題與討論

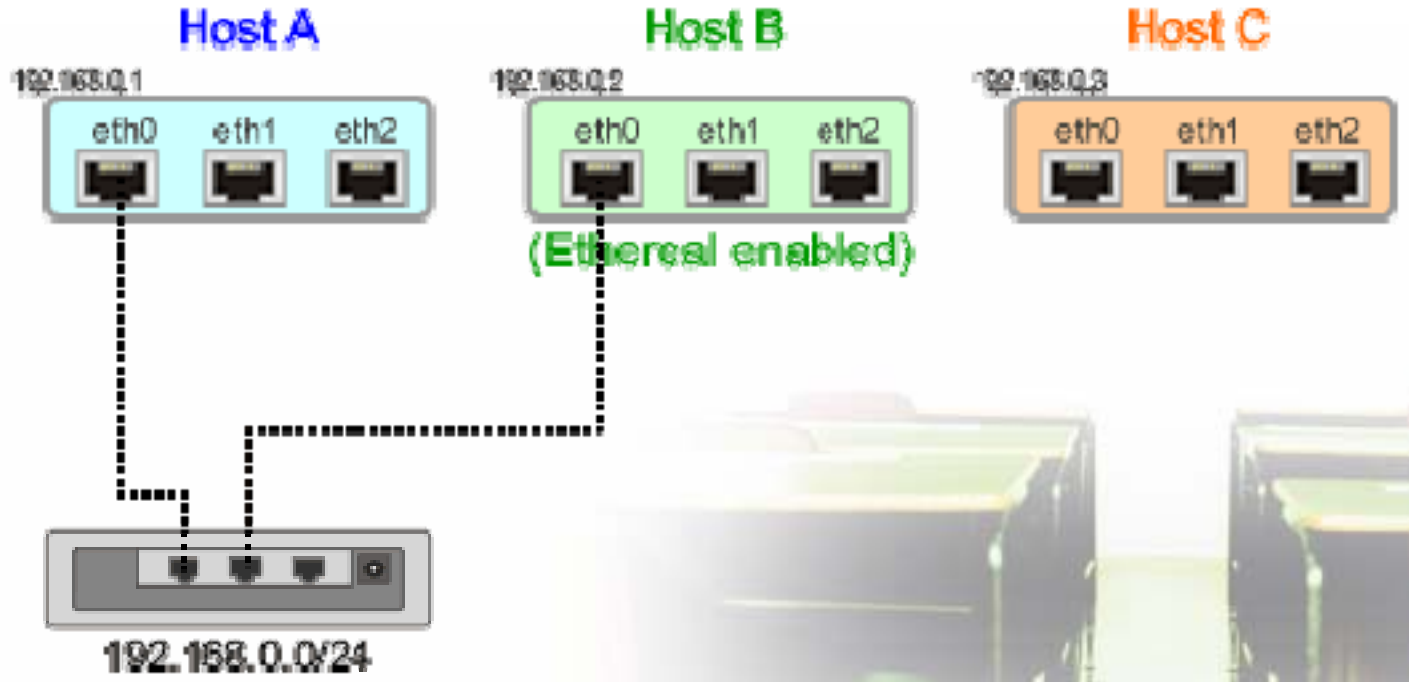
- 比較IP checksum與TCP checksum計算方式有何不同?

實驗 4.2 了解port number的涵意

實驗目的

- 了解如何查閱常用的**port number**
- 觀察**TCP**封包
- 觀察**UDP**封包
- 觀察**IP Fragment**

實驗架構圖


Host A
192.168.0.1
eth0
eth1
eth2
Host B
192.168.0.2
eth0
eth1
eth2
(Ethereal enabled)
Host C
192.168.0.3
eth0
eth1
eth2
192.168.0.0/24

Step 1: 觀察/etc/services 的內容

- 任選**NetGuru**任一主機
  - 以命令列執行以下指令

    more /etc/services
  - 請寫下pop3d, ftpd, telnetd的port number爲何?

Step 2: 觀察TCP封包

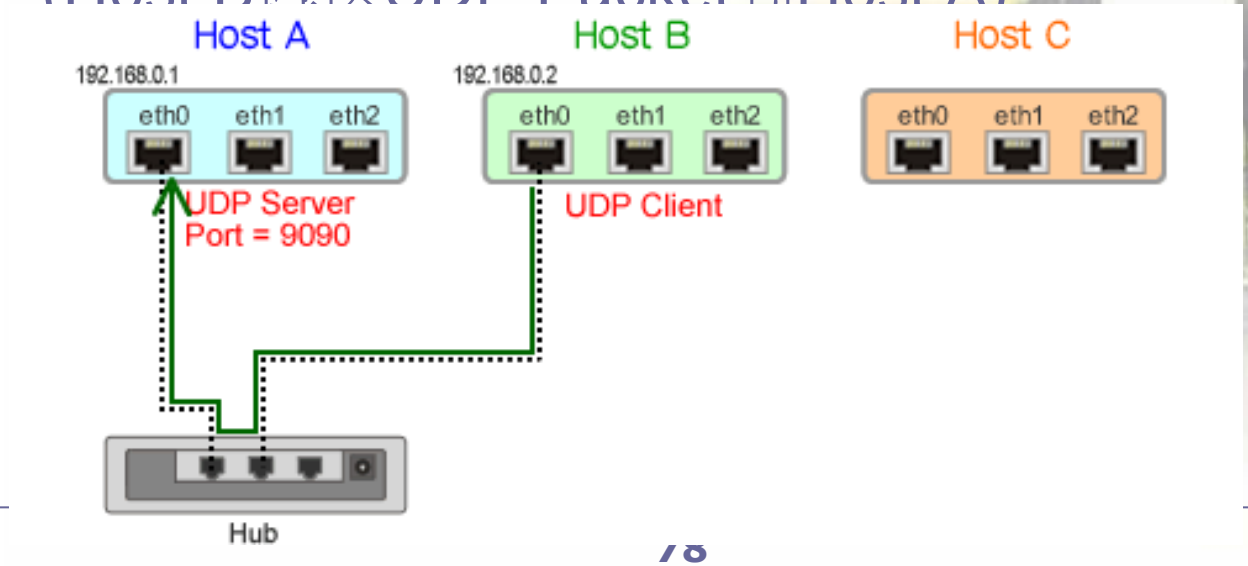
- **Host B：**
  - 開啓 Ethereal ，interface選eth0
  - telnet 192.168.0.1 21
- 另於**Host B：**
  - 再次開啓 Ethereal ，interface選eth0
  - telnet 192.168.0.1 ftp
- 分析相關封包
  - 觀察上述兩者的封包有何不同?
  - 觀察telnet packet之source port爲何?destination port爲何?

Port number說明

- **Port** 是用來區分同一台主機上面不同服務的機制
- **Port** 的大小是 **16 bits**，也就是範圍從 **0** 到 **65535**
- 在網路世界裡，有許多已經約定成俗的 **Port Number**，例如：**Port 21** 爲 **FTP**； **Port 23** 爲**Telnet**
  - 這些通用的Port Number可以參考 Linux 或Unix 系統下 /etc/services 檔案的內容
  - 一般而言，0到1023保留給上述常用的服務，而 1024到 65535可以供其他的程式或是使用者自行使用

Step 3: 使用udpsend與udpserver觀察UDP packets

- **Host B:**
  - 開啓 Ethereal ，interface選eth0，以觀察下列步驟之封包
- **Host A:**
  - 執行#udpserver -p 9090
- **Host B:**
  - 執行#udpsend -d 192.168.0.1 -dport 9090 -m Hello (Host B傳送UDP Packet到Host A)



觀察UDP封包

- 觀察source port，destination port等欄位
- 問題與討論：
  - 比較UDP封包與TCP封包的欄位有何差別，爲什麼?
  - 假如udpserver尚未啓動，但udpsend已送出UDP封包，會有什麼現象?

說明：udpserver與udpsend

- 此二指令是**NetGuru**中特有的指令(非**Linux**標準指令)，其目的爲了方便老師講解及幫助學生了解**UDP**，相關參數如下：
- **udpserver**
  - –p：指要開啓UDP Server 之 Port
  - –s：預計跟系統 “要” 一個的空間，以方便讀取接收到的 UDP Data
    倘若沒有輸入此參數，就會執行預設值(1024 bytes)

(直接執行udpserver就會顯示所有參數的簡介)

udpsend

–dport：要傳送到 UDP Server 的哪一個 Port
倘若沒有輸入此參數，就會執行預設值(9090)

–d：UDP Server IP 此參數一定要設定

–b：UDP Client 要傳送多少個 '*' 字元的Data給UDP Server
例如： udpsend –b 4000 ，是指送出 UDP Data 爲 4000 bytes ， 內容都是 '*'

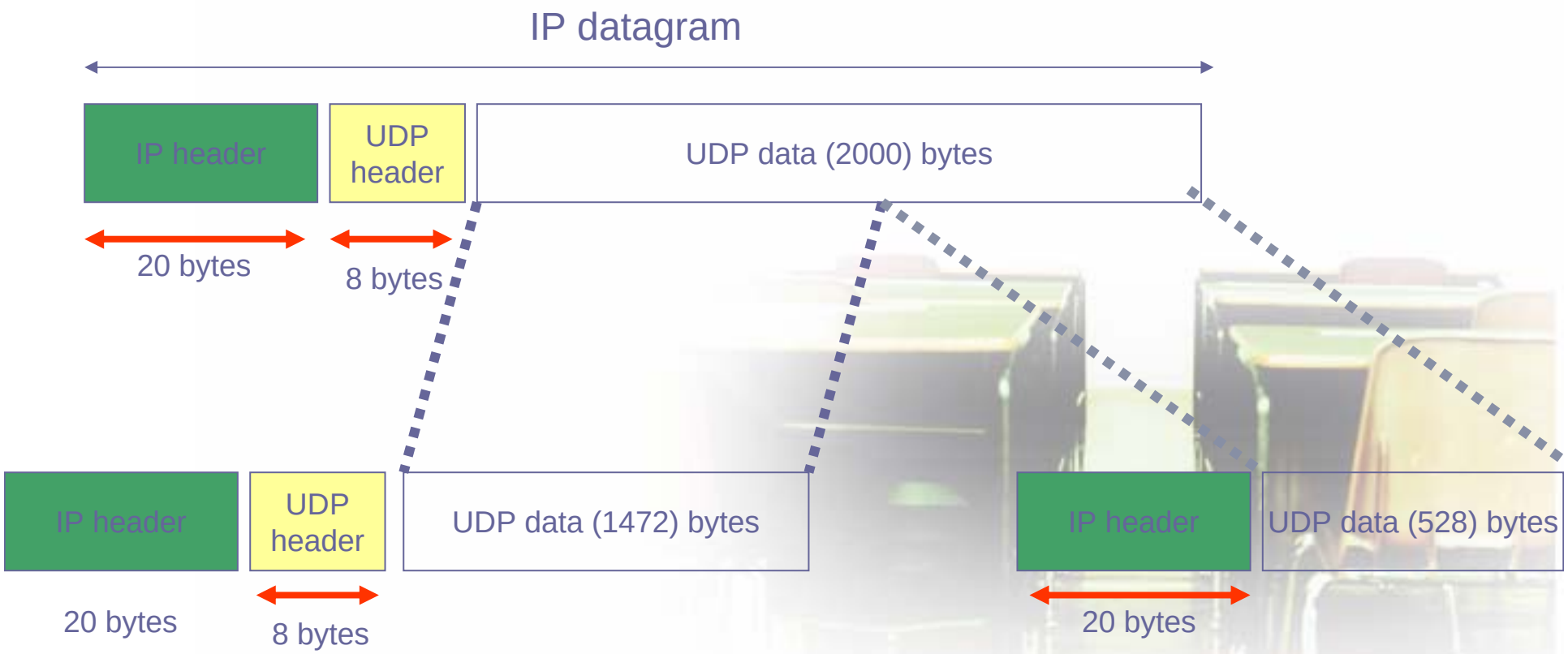
–m ：UDP Client 要傳送什麼的Data給UDP Server
例如：udpsend –m bye ，是指送出 UDP Data 內容爲 'bye'

(直接執行udpsend就會顯示所有參數的簡介)

Step 4: 觀察IP fragment

- **Host B：**
  - 開啓 Ethereal ，interface選eth0，以觀察下列步驟之封包
- **Host A:**
  - 執行#udpserver –p 9090
- **Host B:**
  - 執行#udpsend -d 192.168.0.1 -dport 9090 -b 2000 (Host B傳送一個data長度2000 bytes的UDP Packet到Host A)
- **分析相關封包**
  - 在IP fragment的封包中，於第二個封包中有無port資訊?

IP fragment圖解

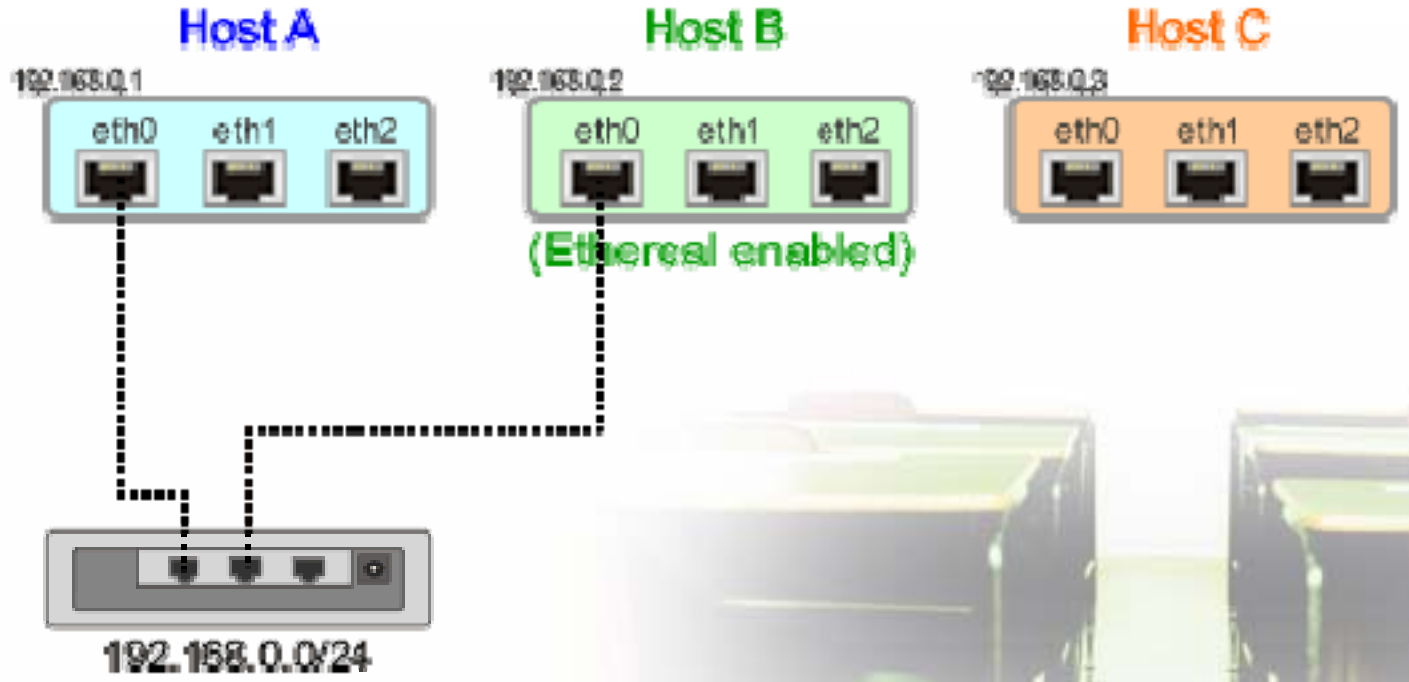


實驗 4.3 觀察TCP之連線及中斷連線

實驗目的

- 了解**TCP**連線流程
- 了解**TCP**中斷連線流程
- 了解**netstat**指令用法

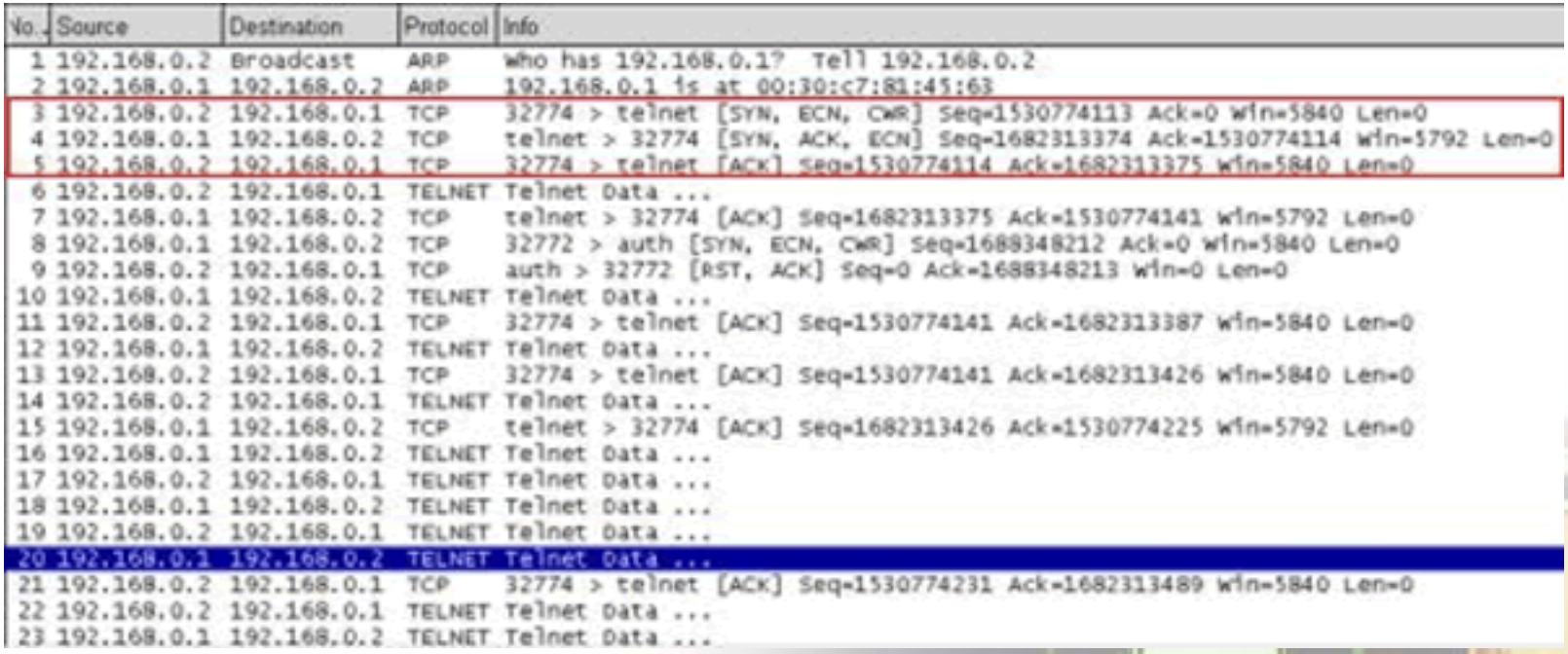
實驗架構圖


Host A
Host B
Host C
eth0
eth1
eth2
eth0
eth1
eth2
eth0
eth1
eth2
(Ethereal enabled)
192.168.0.0/24

Step 1: 觀察telnet封包

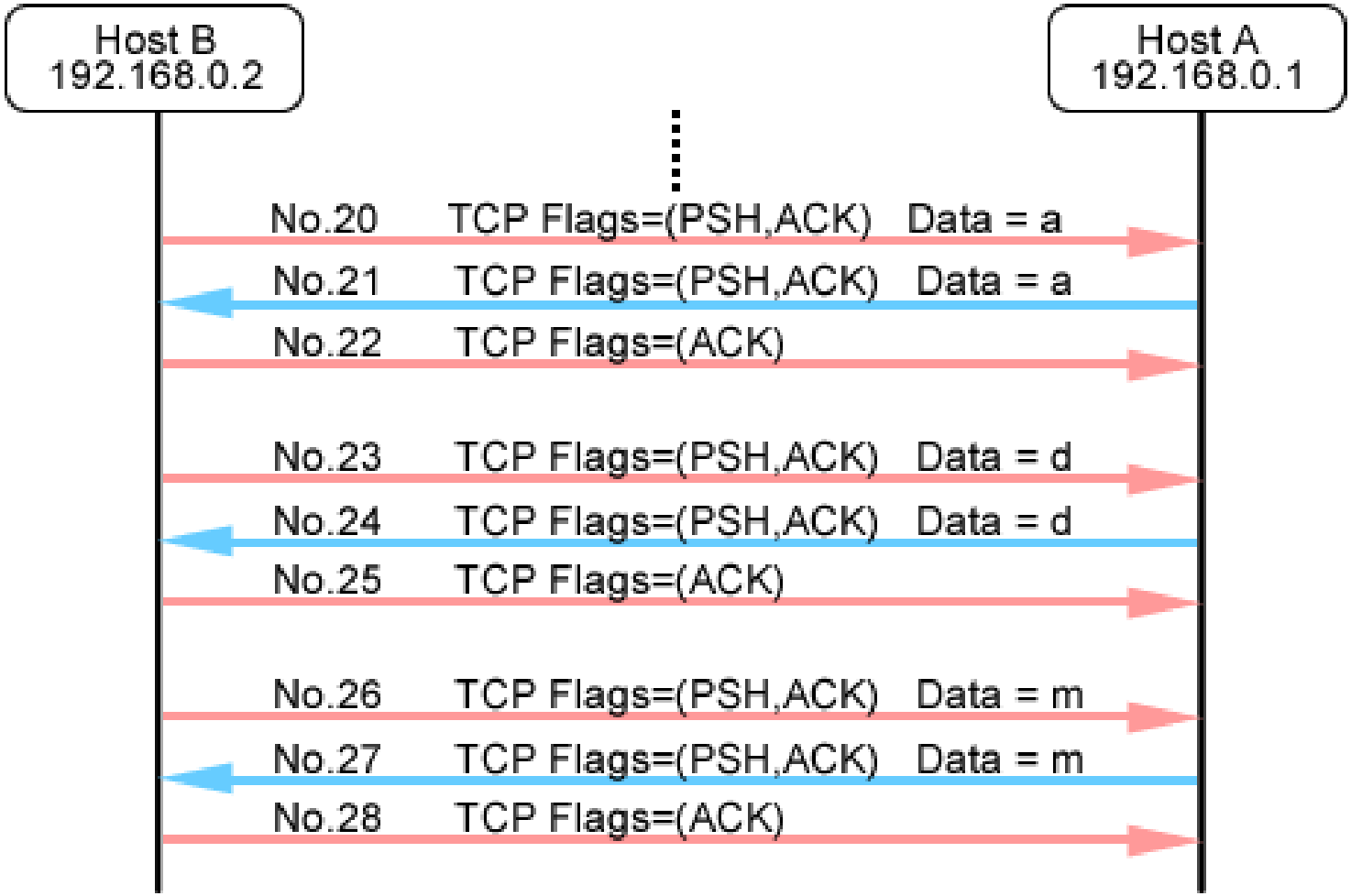
- **Host B：**
  - 開啓 Ethereal ，interface選eth0，以觀察下列步驟之封包
- **Host B:**
  - telnet 192.168.0.1
  - 帳號admin，密碼123456
  - exit
- **分析相關封包**
  - 觀察三向交握之過程(觀察Flag之變化)
  - 觀察輸入帳號和密碼的過程
  - 觀察exit的過程(觀察Flag之變化)

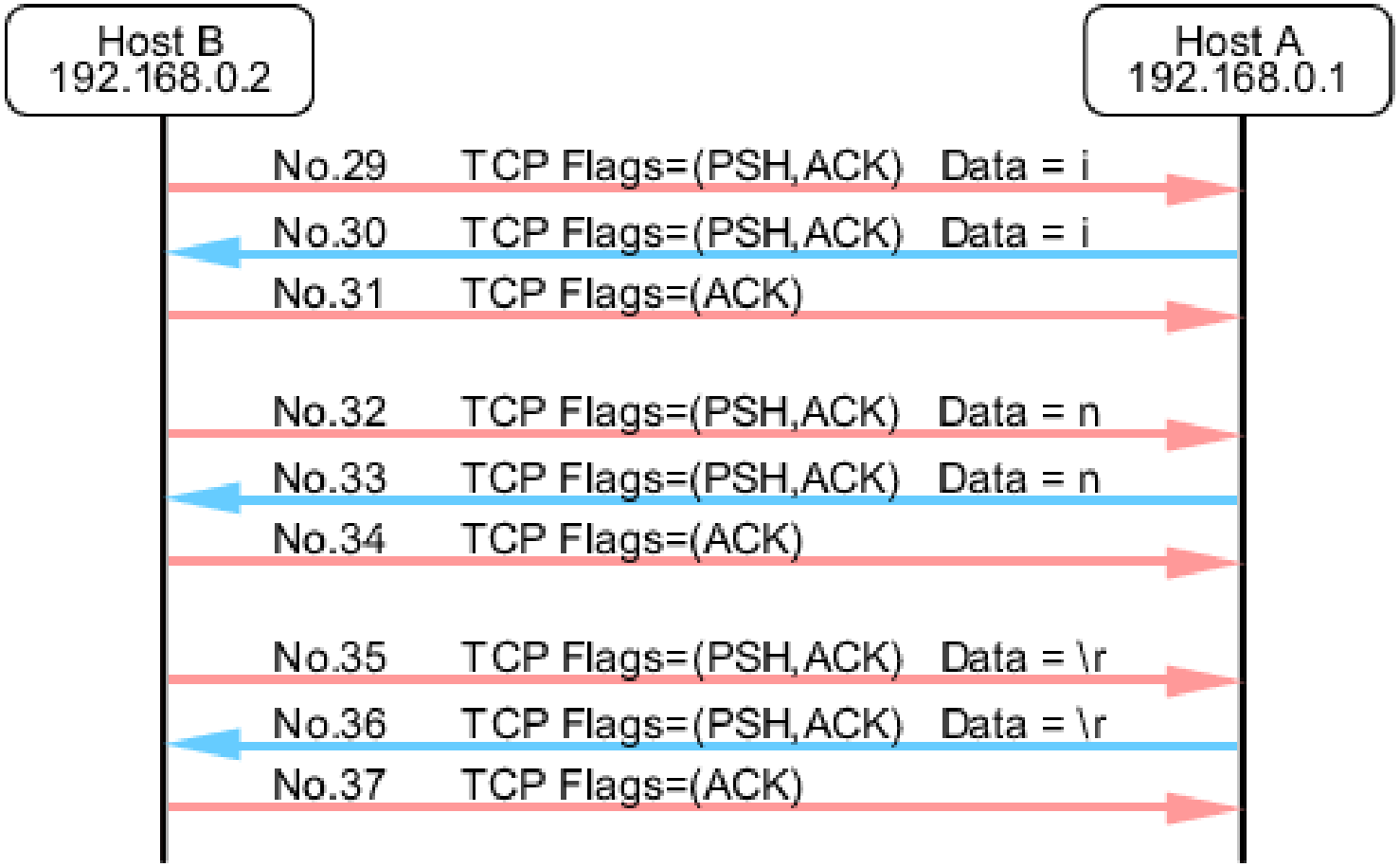
觀察三向交握之過程



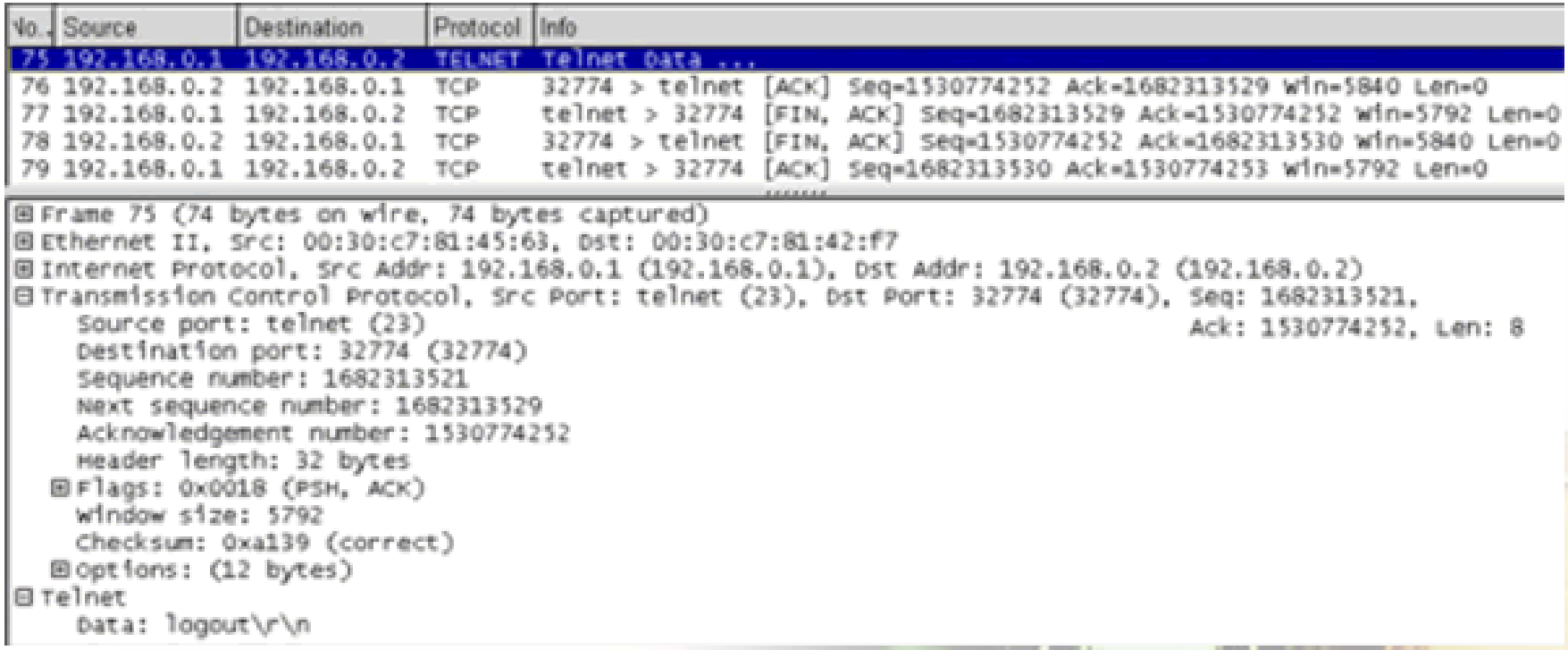
No.	Source	Destination	Protocol	Info
1	192.168.0.2	Broadcast	ARP	who has 192.168.0.1? Tell 192.168.0.2
2	192.168.0.1	192.168.0.2	ARP	192.168.0.1 is at 00:30:c7:81:45:63
3	192.168.0.2	192.168.0.1	TCP	32774 > telnet [SYN, ECN, CWR] Seq=1530774113 Ack=0 Win=5840 Len=0
4	192.168.0.1	192.168.0.2	TCP	telnet > 32774 [SYN, ACK, ECN] Seq=1682313374 Ack=1530774114 Win=5792 Len=0
5	192.168.0.2	192.168.0.1	TCP	32774 > telnet [ACK] Seq=1530774114 Ack=1682313375 Win=5840 Len=0
6	192.168.0.2	192.168.0.1	TELNET	Telnet Data ...
7	192.168.0.1	192.168.0.2	TCP	telnet > 32774 [ACK] Seq=1682313375 Ack=1530774141 Win=5792 Len=0
8	192.168.0.1	192.168.0.2	TCP	32772 > auth [SYN, ECN, CWR] Seq=1688348212 Ack=0 Win=5840 Len=0
9	192.168.0.2	192.168.0.1	TCP	auth > 32772 [RST, ACK] Seq=0 Ack=1688348213 Win=0 Len=0
10	192.168.0.1	192.168.0.2	TELNET	Telnet Data ...
11	192.168.0.2	192.168.0.1	TCP	32774 > telnet [ACK] Seq=1530774141 Ack=1682313387 Win=5840 Len=0
12	192.168.0.1	192.168.0.2	TELNET	Telnet Data ...
13	192.168.0.2	192.168.0.1	TCP	32774 > telnet [ACK] Seq=1530774141 Ack=1682313426 Win=5840 Len=0
14	192.168.0.2	192.168.0.1	TELNET	Telnet Data ...
15	192.168.0.1	192.168.0.2	TCP	telnet > 32774 [ACK] Seq=1682313426 Ack=1530774225 Win=5792 Len=0
16	192.168.0.1	192.168.0.2	TELNET	Telnet Data ...
17	192.168.0.2	192.168.0.1	TELNET	Telnet Data ...
18	192.168.0.1	192.168.0.2	TELNET	Telnet Data ...
19	192.168.0.2	192.168.0.1	TELNET	Telnet Data ...
20	192.168.0.1	192.168.0.2	TELNET	Telnet Data ...
21	192.168.0.2	192.168.0.1	TCP	32774 > telnet [ACK] Seq=1530774231 Ack=1682313489 Win=5840 Len=0
22	192.168.0.2	192.168.0.1	TELNET	Telnet Data ...
23	192.168.0.1	192.168.0.2	TELNET	Telnet Data ...

觀察輸入帳號和密碼的過程




Host B
192.168.0.2
Host A
192.168.0.1
No.29 TCP Flags=(PSH,ACK) Data = i
No.30 TCP Flags=(PSH,ACK) Data = i
No.31 TCP Flags=(ACK)
No.32 TCP Flags=(PSH,ACK) Data = n
No.33 TCP Flags=(PSH,ACK) Data = n
No.34 TCP Flags=(ACK)
No.35 TCP Flags=(PSH,ACK) Data = \r
No.36 TCP Flags=(PSH,ACK) Data = \r
No.37 TCP Flags=(ACK)

觀察exit的過程



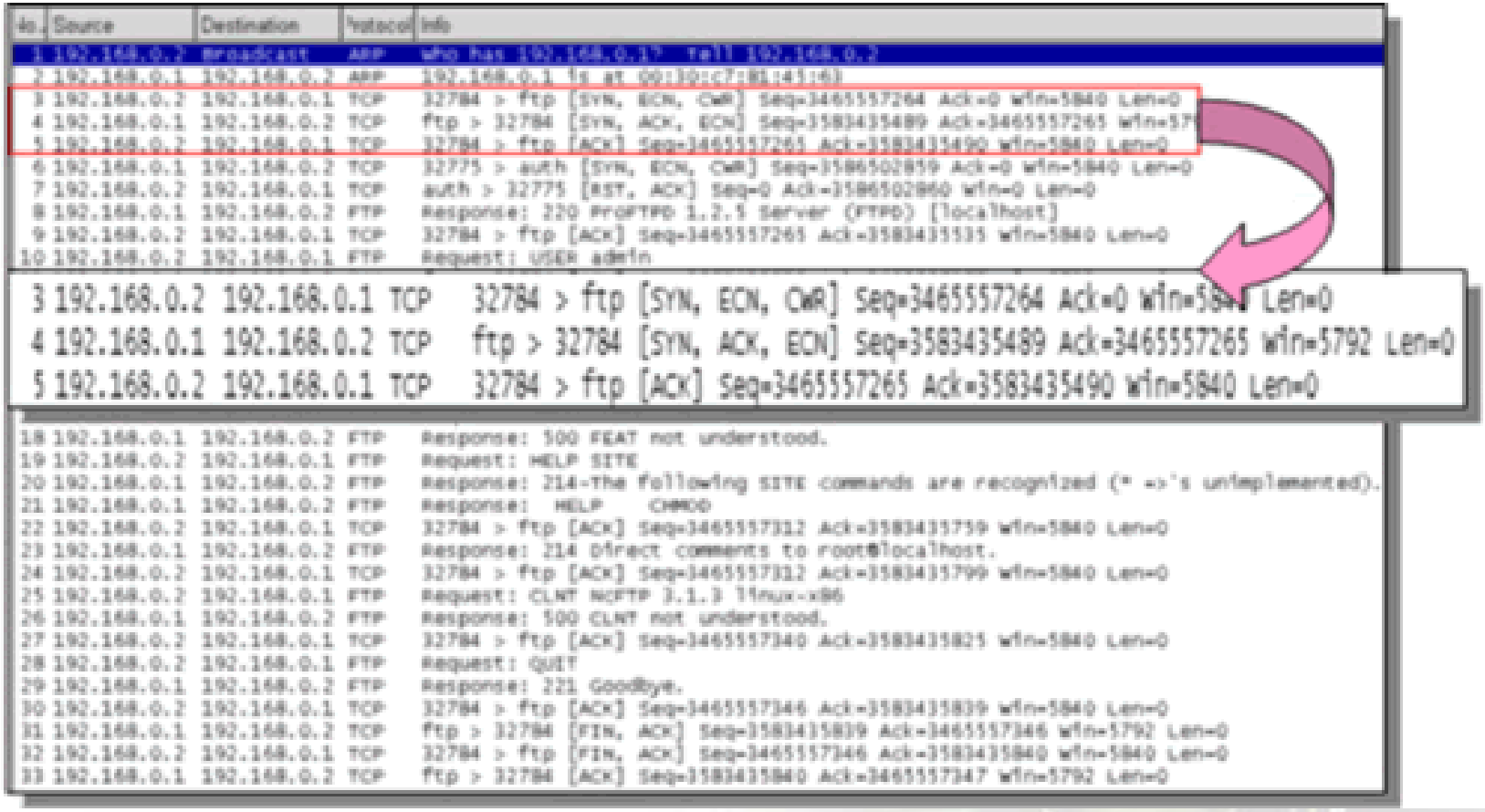
No.	Source	Destination	Protocol	Info
75	192.168.0.1	192.168.0.2	TELNET	Telnet Data ...
76	192.168.0.2	192.168.0.1	TCP	32774 > telnet [ACK] Seq=1530774252 Ack=1682313529 Win=5840 Len=0
77	192.168.0.1	192.168.0.2	TCP	telnet > 32774 [FIN, ACK] Seq=1682313529 Ack=1530774252 Win=5792 Len=0
78	192.168.0.2	192.168.0.1	TCP	32774 > telnet [FIN, ACK] Seq=1530774252 Ack=1682313530 Win=5840 Len=0
79	192.168.0.1	192.168.0.2	TCP	telnet > 32774 [ACK] Seq=1682313530 Ack=1530774253 Win=5792 Len=0

```
⊞ Frame 75 (74 bytes on wire, 74 bytes captured)
⊞ Ethernet II, Src: 00:30:c7:81:45:63, Dst: 00:30:c7:81:42:f7
⊞ Internet Protocol, Src Addr: 192.168.0.1 (192.168.0.1), Dst Addr: 192.168.0.2 (192.168.0.2)
⊟ Transmission Control Protocol, Src Port: telnet (23), Dst Port: 32774 (32774), Seq: 1682313521,
                                                                                  Ack: 1530774252, Len: 8
    Source port: telnet (23)
    Destination port: 32774 (32774)
    Sequence number: 1682313521
    Next sequence number: 1682313529
    Acknowledgement number: 1530774252
    Header length: 32 bytes
  ⊞ Flags: 0x0018 (PSH, ACK)
    Window size: 5792
    Checksum: 0xa139 (correct)
  ⊞ Options: (12 bytes)
⊟ Telnet
    Data: logout\r\n
```

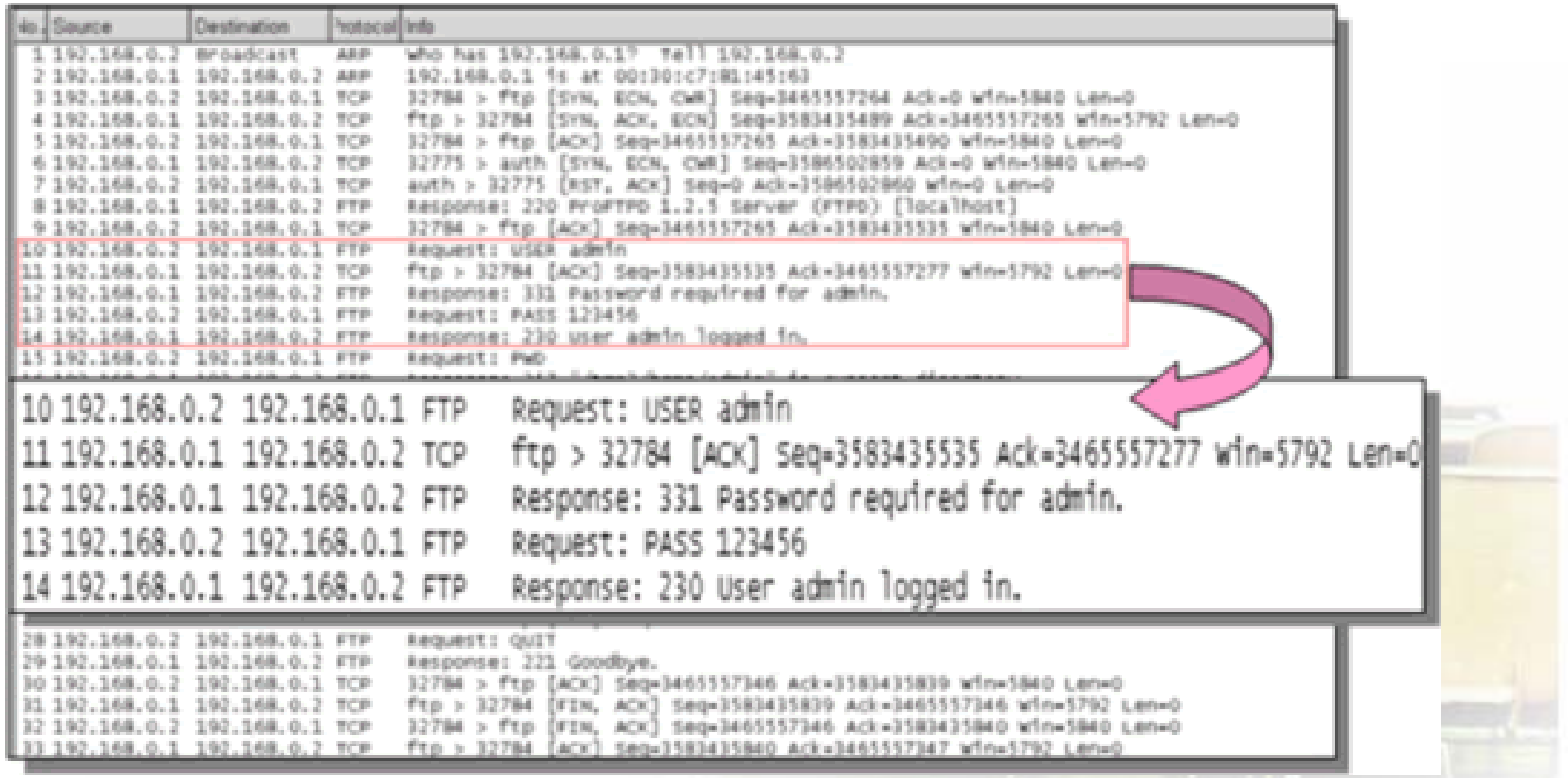
Step 2: 觀察ftp封包

- **Host B：**
  - 開啓Ethereal，interface選eth0，以觀察下列步驟之封包
- **Host B:**
  - ncftp –uadmin –p123456 192.168.0.1
  - exit
- 分析相關封包
  - 觀察三向交握之過程(觀察Flag之變化)
  - 觀察輸入帳號和密碼的過程
  - 觀察logout(exit)的過程(觀察Flag之變化)

觀察三向交握之過程

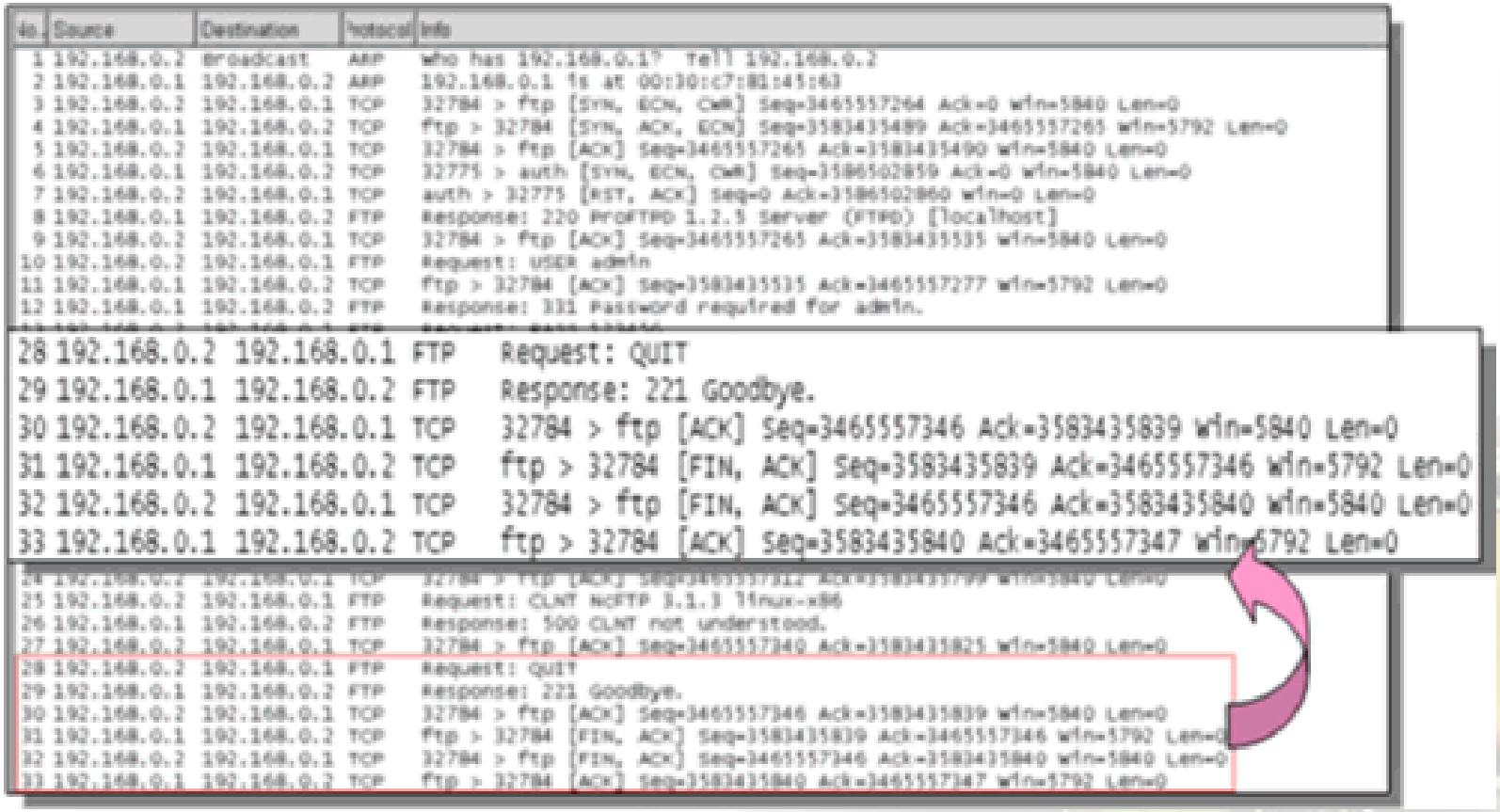

No. Source Destination Protocol Info
1 192.168.0.2 Broadcast ARP Who has 192.168.0.1? Tell 192.168.0.2
2 192.168.0.1 192.168.0.2 ARP 192.168.0.1 is at 00:30:c7:81:45:60
3 192.168.0.2 192.168.0.1 TCP 32784 > ftp [SYN, ECN, CWR] Seq=3465557264 Ack=0 Win=5840 Len=0
4 192.168.0.1 192.168.0.2 TCP ftp > 32784 [SYN, ACK, ECN] Seq=3583435489 Ack=3465557265 Win=5
5 192.168.0.2 192.168.0.1 TCP 32784 > ftp [ACK] Seq=3465557265 Ack=3583435490 Win=5840 Len=0
6 192.168.0.1 192.168.0.2 TCP 32775 > auth [SYN, ECN, CWR] Seq=3586502859 Ack=0 Win=5840 Len=0
7 192.168.0.2 192.168.0.1 TCP auth > 32775 [RST, ACK] Seq=0 Ack=3586502860 Win=0 Len=0
8 192.168.0.1 192.168.0.2 FTP Response: 220 ProFTPD 1.2.5 Server (FTPD) [localhost]
9 192.168.0.2 192.168.0.1 TCP 32784 > ftp [ACK] Seq=3465557265 Ack=3583435535 Win=5840 Len=0
10 192.168.0.2 192.168.0.1 FTP Request: USER admin
3 192.168.0.2 192.168.0.1 TCP 32784 > ftp [SYN, ECN, CWR] Seq=3465557264 Ack=0 Win=5840 Len=0
4 192.168.0.1 192.168.0.2 TCP ftp > 32784 [SYN, ACK, ECN] Seq=3583435489 Ack=3465557265 Win=5792 Len=0
5 192.168.0.2 192.168.0.1 TCP 32784 > ftp [ACK] Seq=3465557265 Ack=3583435490 Win=5840 Len=0
18 192.168.0.1 192.168.0.2 FTP Response: 500 FEAT not understood.
19 192.168.0.2 192.168.0.1 FTP Request: HELP SITE
20 192.168.0.1 192.168.0.2 FTP Response: 214-The following SITE commands are recognized (* =>'s unimplemented).
21 192.168.0.1 192.168.0.2 FTP Response: HELP CHMOD
22 192.168.0.2 192.168.0.1 TCP 32784 > ftp [ACK] Seq=3465557312 Ack=3583435739 Win=5840 Len=0
23 192.168.0.1 192.168.0.2 FTP Response: 214 Direct comments to root@localhost.
24 192.168.0.2 192.168.0.1 TCP 32784 > ftp [ACK] Seq=3465557312 Ack=3583435799 Win=5840 Len=0
25 192.168.0.2 192.168.0.1 FTP Request: CLNT NcFTP 3.1.3 linux-x86
26 192.168.0.1 192.168.0.2 FTP Response: 500 CLNT not understood.
27 192.168.0.2 192.168.0.1 TCP 32784 > ftp [ACK] Seq=3465557340 Ack=3583435825 Win=5840 Len=0
28 192.168.0.2 192.168.0.1 FTP Request: QUIT
29 192.168.0.1 192.168.0.2 FTP Response: 221 Goodbye.
30 192.168.0.2 192.168.0.1 TCP 32784 > ftp [ACK] Seq=3465557346 Ack=3583435839 Win=5840 Len=0
31 192.168.0.1 192.168.0.2 TCP ftp > 32784 [FIN, ACK] Seq=3583435839 Ack=3465557346 Win=5792 Len=0
32 192.168.0.2 192.168.0.1 TCP 32784 > ftp [FIN, ACK] Seq=3465557346 Ack=3583435840 Win=5840 Len=0
33 192.168.0.1 192.168.0.2 TCP ftp > 32784 [ACK] Seq=3583435840 Ack=3465557347 Win=5792 Len=0

觀察輸入帳號和密碼的過程



- 比較**telnet**與**ftp**在輸入帳號和密碼過程有何差異?

觀察logout的過程



No.	Source	Destination	Protocol	Info
1	192.168.0.2	Broadcast	ARP	Who has 192.168.0.1? Tell 192.168.0.2
2	192.168.0.1	192.168.0.2	ARP	192.168.0.1 is at 00:30:c7:81:45:63
3	192.168.0.2	192.168.0.1	TCP	32784 > ftp [SYN, ECN, CWR] Seq=3465557264 Ack=0 Win=5840 Len=0
4	192.168.0.1	192.168.0.2	TCP	ftp > 32784 [SYN, ACK, ECN] Seq=3583435489 Ack=3465557265 Win=5792 Len=0
5	192.168.0.2	192.168.0.1	TCP	32784 > ftp [ACK] Seq=3465557265 Ack=3583435490 Win=5840 Len=0
6	192.168.0.1	192.168.0.2	TCP	32775 > auth [SYN, ECN, CWR] Seq=3586502859 Ack=0 Win=5840 Len=0
7	192.168.0.2	192.168.0.1	TCP	auth > 32775 [RST, ACK] Seq=0 Ack=3586502860 Win=0 Len=0
8	192.168.0.1	192.168.0.2	FTP	Response: 220 ProFTPD 1.2.5 Server (FTPD) [localhost]
9	192.168.0.2	192.168.0.1	TCP	32784 > ftp [ACK] Seq=3465557265 Ack=3583435535 Win=5840 Len=0
10	192.168.0.2	192.168.0.1	FTP	Request: USER admin
11	192.168.0.1	192.168.0.2	TCP	ftp > 32784 [ACK] Seq=3583435535 Ack=3465557277 Win=5792 Len=0
12	192.168.0.1	192.168.0.2	FTP	Response: 331 Password required for admin.
28	192.168.0.2	192.168.0.1	FTP	Request: QUIT
29	192.168.0.1	192.168.0.2	FTP	Response: 221 Goodbye.
30	192.168.0.2	192.168.0.1	TCP	32784 > ftp [ACK] Seq=3465557346 Ack=3583435839 Win=5840 Len=0
31	192.168.0.1	192.168.0.2	TCP	ftp > 32784 [FIN, ACK] Seq=3583435839 Ack=3465557346 Win=5792 Len=0
32	192.168.0.2	192.168.0.1	TCP	32784 > ftp [FIN, ACK] Seq=3465557346 Ack=3583435840 Win=5840 Len=0
33	192.168.0.1	192.168.0.2	TCP	ftp > 32784 [ACK] Seq=3583435840 Ack=3465557347 Win=5792 Len=0
25	192.168.0.2	192.168.0.1	FTP	Request: CLNT NcFTP 3.1.3 linux-x86
26	192.168.0.1	192.168.0.2	FTP	Response: 500 CLNT not understood.
27	192.168.0.2	192.168.0.1	TCP	32784 > ftp [ACK] Seq=3465557340 Ack=3583435825 Win=5840 Len=0
28	192.168.0.2	192.168.0.1	FTP	Request: QUIT
29	192.168.0.1	192.168.0.2	FTP	Response: 221 Goodbye.
30	192.168.0.2	192.168.0.1	TCP	32784 > ftp [ACK] Seq=3465557346 Ack=3583435839 Win=5840 Len=0
31	192.168.0.1	192.168.0.2	TCP	ftp > 32784 [FIN, ACK] Seq=3583435839 Ack=3465557346 Win=5792 Len=0
32	192.168.0.2	192.168.0.1	TCP	32784 > ftp [FIN, ACK] Seq=3465557346 Ack=3583435840 Win=5840 Len=0
33	192.168.0.1	192.168.0.2	TCP	ftp > 32784 [ACK] Seq=3583435840 Ack=3465557347 Win=5792 Len=0

Step 3:利用netstat指令，觀察TCP連線狀態

- **Host A**：
  - netstat –l (察看目前在LISTEN State有哪些 Socket)
  - netstat –tcp
- **Host B**：
  - ncftp –uadmin –p123456 192.168.0.1
- **Host A**：
  - Ctrl + C中斷netstat，觀察連線程序

netstat –l

- 察看目前在**LISTEN State**有哪些 **Socket**

```
bash# netstat -l
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 *:7878                  *:*                     LISTEN
tcp        0      0 *:11111                 *:*                     LISTEN
tcp        0      0 *:5801                  *:*                     LISTEN
tcp        0      0 *:5901                  *:*                     LISTEN
tcp        0      0 *:9998                  *:*                     LISTEN
tcp        0      0 *:pop3                  *:*                     LISTEN
tcp        0      0 *:sunrpc                *:*                     LISTEN
tcp        0      0 *:imap2                 *:*                     LISTEN
tcp        0      0 *:6000                  *:*                     LISTEN
tcp        0      0 *:www                   *:*                     LISTEN
tcp        0      0 *:6001                  *:*                     LISTEN
tcp        0      0 *:ftp                   *:*                     LISTEN
tcp        0      0 *:17878                 *:*                     LISTEN
tcp        0      0 *:telnet                *:*                     LISTEN
tcp        0      0 *:8888                  *:*                     LISTEN
udp        0      0 *:sunrpc                *:*
```

netstat –tcp

- 觀察連線程序

```
Active Internet connections (w/o servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
PID/Program name
tcp        0      0 192.168.0.1:ftp         192.168.0.2:32778       ESTABLISHE
3564/proftpd: admin
tcp        0      0 localhost:6001          localhost:32785         ESTABLISHE
308/Xvnc
tcp        0      0 localhost:32774         localhost:6000          ESTABLISHE
515/mozilla-bin
tcp        0      0 localhost:32769         localhost:6000          ESTABLISHE
310/qvwm
tcp        0      0 localhost:32768         localhost:6000          ESTABLISHE
297/qvwm
tcp        0      0 localhost:32777         localhost:6000          ESTABLISHE
614/rxvt.cjk
tcp        0      0 localhost:6000          localhost:32774         ESTABLISHE
296/Xvesa
tcp        0      0 localhost:32770         localhost:6001          ESTABLISHE
316/qvwm
tcp        0      0 localhost:32771         localhost:6001          ESTABLISHE
327/qvwm
tcp        0      0 localhost:32788         localhost:6001          ESTABLISHE
```

TCP Connection State Diagram