打個比方，假設你發了一封標題為「全體行銷同仁請注意」的電子郵件給公司內所有員工，而不是只發給任職於行銷部門的同仁。行銷部同仁看到這封信原就是要給他們的，自然會開啟並詳加閱讀。但其他員工則會注意到這封信不是發給他們的，因而會略過不讀。但你應該也注意到，這種通訊方式既製造出大量不必要的流量，又浪費時間，然而這正是集線器的運作邏輯。

在正式及高密度的網路環境理，最適合代替集線器的裝置就是**交換器**，這是一種**全雙工裝置**（*full-duplex devices*），能夠同步收發資料。

交換器

交換器和集線器一樣，都是用來轉發封包的。但是它與集線器的不同之處在於，交換器不會把封包廣播到每一個埠，而是只把資料送到它原本就預計要抵達的電腦。交換器的外觀和集線器十分類似，如圖 1-6 所示。

圖 1-6：一台機架安裝式的 48 埠乙太網路交換器

市面上有很多款大型的交換器，例如 Cisco 所製造的型號，均可透過特殊的廠商自製軟體或網頁式介面加以管理。這種交換器常被稱為**網管型交換器**（*managed switches*）。網管型交換器具備數種有助於網管的特殊功能，例如可以啟用或停用特定網路埠、檢視網路埠統計數字、進行設定調整，或是從遠端重啟設備等等。

交換器同時也具備更為進階的封包傳輸處理功能。為了要與特定裝置直接通訊，交換器必須要能按照 MAC 位址辨識出個別的裝置，亦即它必須在 OSI 模型的資料鏈結層運作。

NOTE

除非有足夠的權限，大部分作業系統（包括 Windows）都不允許你直接以混雜模式運作網卡。若你無法依規定取得系統權限，則可能是你的職權無權在特定網路上進行任何形式的封包分析。

在集線器四週監聽

監聽一個純粹由集線器構成的網路，是所有封包分析人員的理想。第 1 章時各位已經學到，透過集線器發送的流量，會再送抵每一個連接埠上的裝置。因此若要分析集線器上連接的某台電腦的流量，只消把封包監聽器插到集線器上任一空埠就成了。你不但可以看到進出該台電腦的通訊，連同其他接在集線器上的電腦彼此之間的通訊也都一覽無遺。如圖 2-2 所示，當你的封包監聽器接上的是一個以集線器構成的網路時，你的可見範圍（visibility window）是毫不受限的。

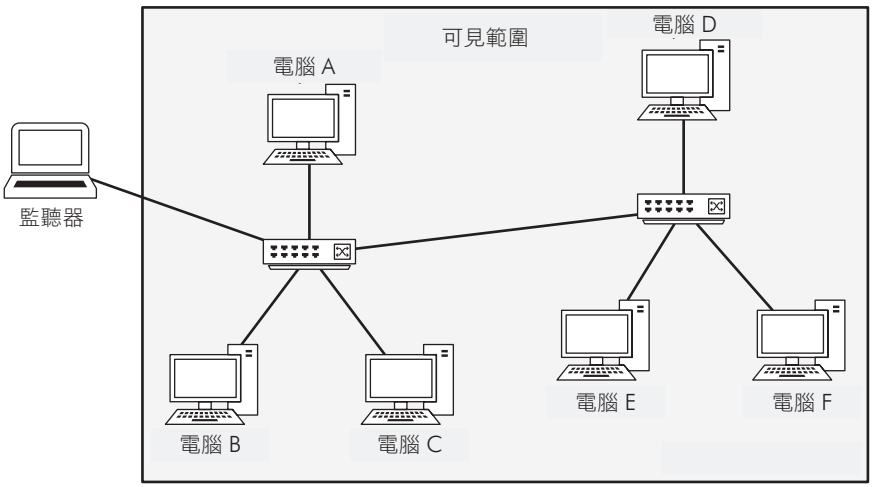


圖 2-2：在以集線器構成的網路上，監聽的可見範圍全不受限

NOTE

可見範圍（*visibility window*），正如本書其他圖解所示，代表的是在網路上可以讓你用封包監聽器看見其流量的眾多裝置。

要啟用網路埠映射，你必須對交換器下達指令，令其將流經特定埠的流量全數複製到另一埠。舉例來說，為了要捕捉交換器第 3 埠上的電腦所收發的流量，你可把監聽器插到第 4 埠上，再將第 3 埠的流量都「映射」到第 4 埠。圖 2-5 便是網路埠映射的示意圖。

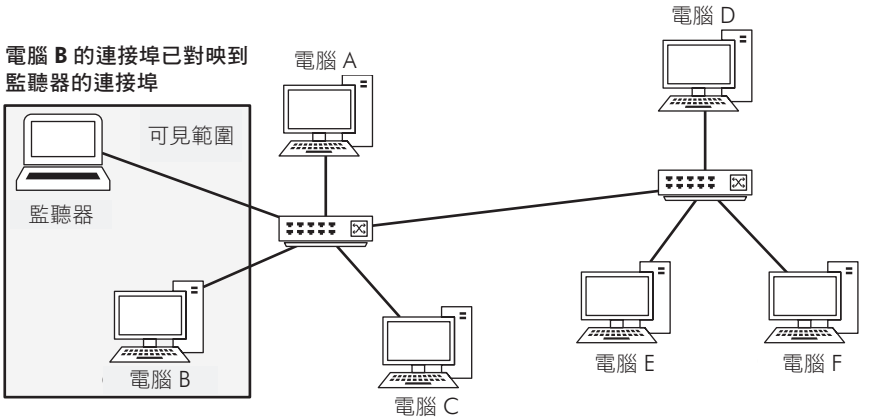


圖 2-5：在交換式網路上，透過網路埠映射法，就可以擴大可見範圍

至於網路埠映射要如何設定，端看交換器製造商的設計而定。對大部分的企業用交換器來說，你得先登入它的指令列介面，再用特定指令設定網路埠映射。表 2-1 列舉幾款常見品牌的網路埠映射設定指令範例。

表 2-1：啟用網路埠映射的指令

製造商	指令
Cisco	`set span <source port><destination port>`[1]
Enterasys	`set port mirroring create <source port> <destination port>`
Nortel	`port-mirroring mode mirror-port <source port> monitor-port <destination port>`

1 舊型 Cisco 交換器才支援 set 一族的指令。目前的 IOS 版本已可改用 monitor session 指令定義映射來源與目的埠。

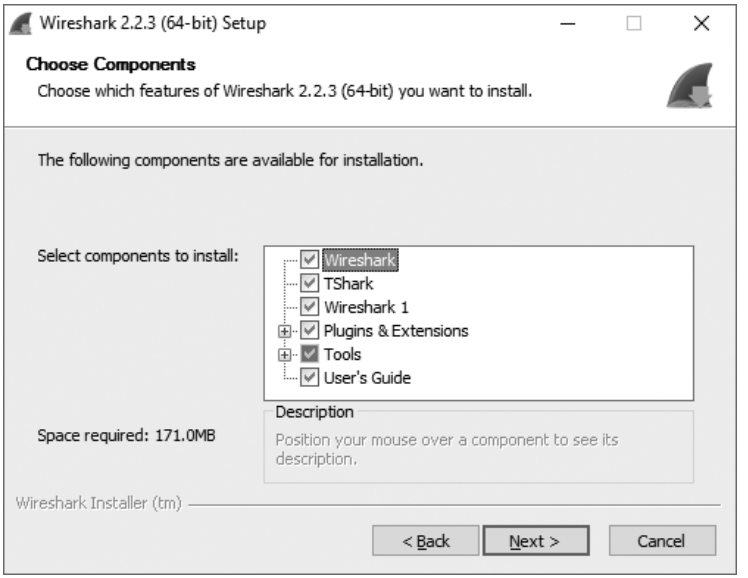


圖 3-1：勾選你想安裝的 Wireshark 元件

4. 在 Additional Tasks 畫面點選 **Next** 繼續。

5. 選擇你要安裝 Wireshark 的位置，然後點選 **Next** 繼續。

6. 當對話框詢問是否安裝 WinPcap 時，請確認已如圖 3-2 所示一般勾選 **Install WinPcap 4.1.3** 旁的小方格。然後點選 **Next**。這時安裝程序應會繼續。

7. 在安裝過程中，WinPcap 也會開始安裝。當它開始安裝時，只需點選 **Next** 跳過歡迎安裝畫面，同樣略讀授權同意書後按 **I Agree** 繼續。

8. 你將會看到安裝 USBPcap 的選項[4]，這是一個可以透過 USB 裝置蒐集資料的工具。如果你要用到它，請勾選旁邊的小方格然後按 **Next** 繼續安裝。

4 譯者嘗試安裝較新版（2.2.6）時，USBPcap 的安裝抉擇是在第 6 步後就先出現了，然後才會出現 WinPcap 安裝提示，與上述順序略有不同。

視窗一樣可以啟用名稱解譯、用顯示篩選器過濾想關注的會話，或是用右鍵點選特定會話後反向篩選出封包清單窗框裡與該會話相關的封包。當你想要挖掘出某連續通訊的細節時，會話篩選器是非常好用的。

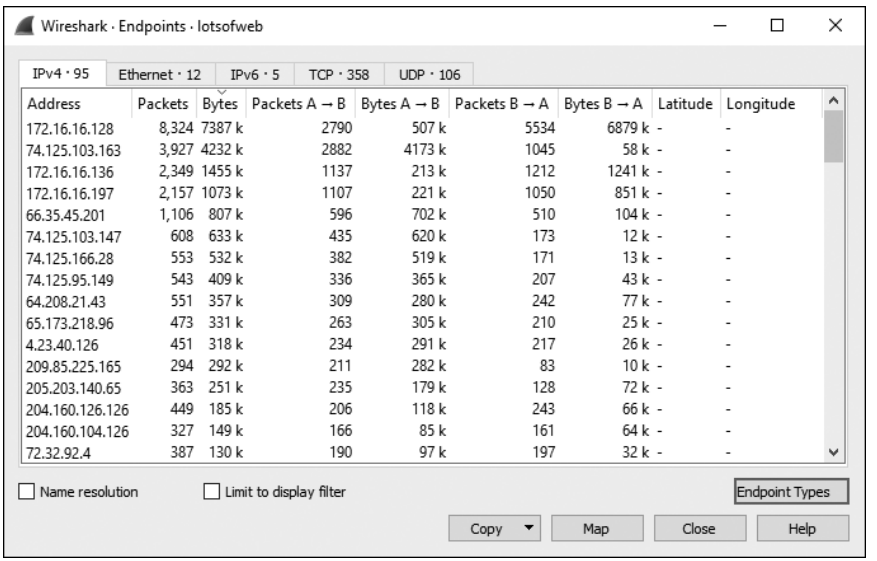
從端點與會話看出最饒舌的是誰

lotsofweb.pcapng

在為網路除錯時，Endpoints 和 Conversations 視窗非常好用，尤其是當你想找出網路上流量特別大的始作俑者時。

還是以 *lotsofweb.pcapng* 檔案為例。檔案名稱已經明示，這個捕捉結果檔案內含有來自各個用戶端瀏覽網際網路時所產生的 HTTP 流量。圖 5-4 顯示出檔案中的所有端點，並依位元組數量由大到小排序。

請注意，跟最多流量（以位元組計算）有關的端點是 IP 位址 172.16.16.128。這是一個內部網路位址（第 7 章將會解釋如何判別），而且由於該裝置是檔案裡通訊量最大的一方，我們因而得知它是**最饒舌的一個**。



Address	Packets	Bytes	Packets A → B	Bytes A → B	Packets B → A	Bytes B → A	Latitude	Longitude
172.16.16.128	8,324	7387 k	2790	507 k	5534	6879 k	-	-
74.125.103.163	3,927	4232 k	2882	4173 k	1045	58 k	-	-
172.16.16.136	2,349	1455 k	1137	213 k	1212	1241 k	-	-
172.16.16.197	2,157	1073 k	1107	221 k	1050	851 k	-	-
66.35.45.201	1,106	807 k	596	702 k	510	104 k	-	-
74.125.103.147	608	633 k	435	620 k	173	12 k	-	-
74.125.166.28	553	532 k	382	519 k	171	13 k	-	-
74.125.95.149	543	409 k	336	365 k	207	43 k	-	-
64.208.21.43	551	357 k	309	280 k	242	77 k	-	-
65.173.218.96	473	331 k	263	305 k	210	25 k	-	-
4.23.40.126	451	318 k	234	291 k	217	26 k	-	-
209.85.225.165	294	292 k	211	282 k	83	10 k	-	-
205.203.140.65	363	251 k	235	179 k	128	72 k	-	-
204.160.126.126	449	185 k	206	118 k	243	66 k	-	-
204.160.104.126	327	149 k	166	85 k	161	64 k	-	-
72.32.92.4	387	130 k	190	97 k	197	32 k	-	-

圖 5-4：從 Endpoints 視窗可以看出有哪些主機在通話

現在，資料就能用 FTP 流量的屬性來解析了，你可以在封包清單裡正確地分析它們，不必再深入到位元組內容才能判讀（圖 5-13）。

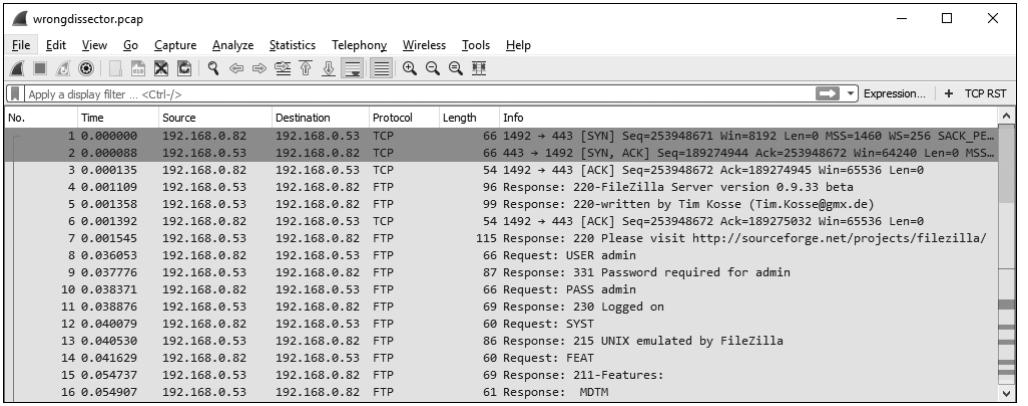


圖 5-13：正確地檢視解析出來的 FTP 流量

同一個捕捉結果檔裡可以重複套用同一個強制解碼設定。Wireshark 會記得你在 Decode As…對話框裡設置過的強制解碼紀錄，在該對話框裡你可以隨意檢視和編輯任何曾經建立過的強制解碼設定。

根據預設模式，一旦關閉 Wireshark，它不會記得你套用的強制解碼設定。但是你仍然可以在 Decode As…對話框裡點選 Save 按鍵，儲存強制解碼設定。此解碼設定會留在你個人的 Wireshark 組態設定檔裡；只要你是以這個組態設定檔啟動 Wireshark 並開啟捕捉結果檔，解碼設定就會生效。要刪除儲存的解碼設定，只需在 Decode As…對話框裡點選減號按鍵即可。

要把強制解碼設定儲存起來很容易，但更容易把它拋到腦後。如果你忘記自己做過這件事，之後分析其他資料時很可能會感到一頭霧水，所以最好還是小心使用強制解碼。為了不讓自己變成粗心的受害者，我都會避免把強制解碼設定儲存到我自己的 Wireshark 組態設定檔裡。

這個下載動作的傳輸速率在每秒 0 到 100 個封包之間劇烈變動，有時還近乎完全停滯。如果把兩張 IO 圖表並排比較（如圖 5-19 一般），會更容易看出後者傳輸速率有明顯的不一致性。當你比較兩張圖表時，請留意 X 軸與 Y 軸所採用的計量單位，確保合理的比較。由於圖表的比例會視封包的數目或資料傳輸量而自動調整，這會成為圖 5-19 中兩張圖表的關鍵差異。下載緩慢的範例圖，其每秒封包數是在 0 到 100 間跳動，但下載迅速的範例圖，代表下載速率的 Y 軸比例則是介於每秒封包數 0 到 700 個之間，比例相差甚鉅。

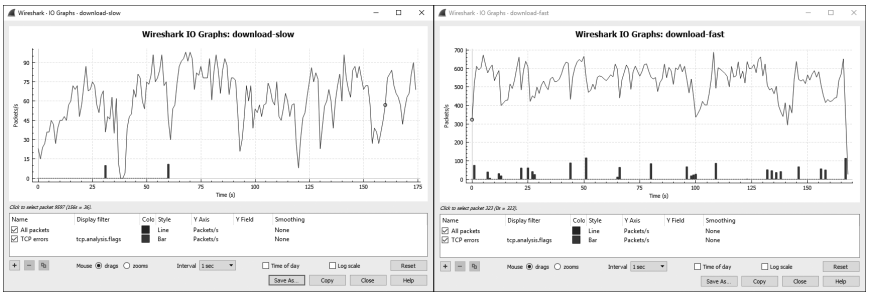


圖 5-19：並排比較 IO Graph，有助於看出趨勢的差異

IO Graph 視窗裡可以調整的選項都位在視窗下方，你可以從中設定篩選器（其語法與顯示或捕捉用篩選器一樣），並指定篩選後顯示的色彩。舉個例子，你可以建立一個特定 IP 位址的篩選器並指定其顯示用的顏色，就可以開始觀察每個裝置的吞吐量變化。我們來動手試試看。

請開啟 *http_espn.pcapng*，這是在某裝置瀏覽 ESPN 首頁時捕捉的紀錄。如果你細看 Conversation 視窗，就會注意到流量最大的外部 IP 位址是 205.234.218.129。從這一點可以推斷出，它應該就是使用者造訪 *espn.com* 時，從中接收資料的主要內容供應者。但是會話中看起來還有好幾個其他的 IP 位址參與，有可能是因為首頁中有部分內容是重導向自其他外部內容或是廣告業者所引起的。透過圖 5-20 所示的 IO 圖表，我們可以看出瀏覽時接收的直接與第三方間內容的差別。

這是一個很有趣的練習，我建議大家拿自己最常去的網站多分析幾次看看，它同時也是比較不同網路主機 IO 分佈的好辦法。

往返時間圖表

download-fast .pcapng

Wireshark 還有另一個圖表繪製功能，就是可以檢視捕捉結果檔案中的往返時間分佈圖（round-trip time graphing）。*round-trip time*（簡稱 *RTT*）是指收到一個封包抵達後確認回覆所需的時間。實際上，就是你的封包抵達目的地所需的時間，再加上確認封包已抵達的回覆送到你這邊的全部時間。通常都是要找出通訊途中的緩慢或瓶頸之處或想確認是否有任何遲滯現象時，就會著手分析 RTT。

我們來試用這個功能。請開啟 *download-fast.pcapng*，任選一個 TCP 封包，然後點選 **Statistics ▸ TCP Stream Graphs ▸ Round Trip Time Graph**。*download-fast.pcapng* 的 RTT 圖表會如圖 5-21 所示。

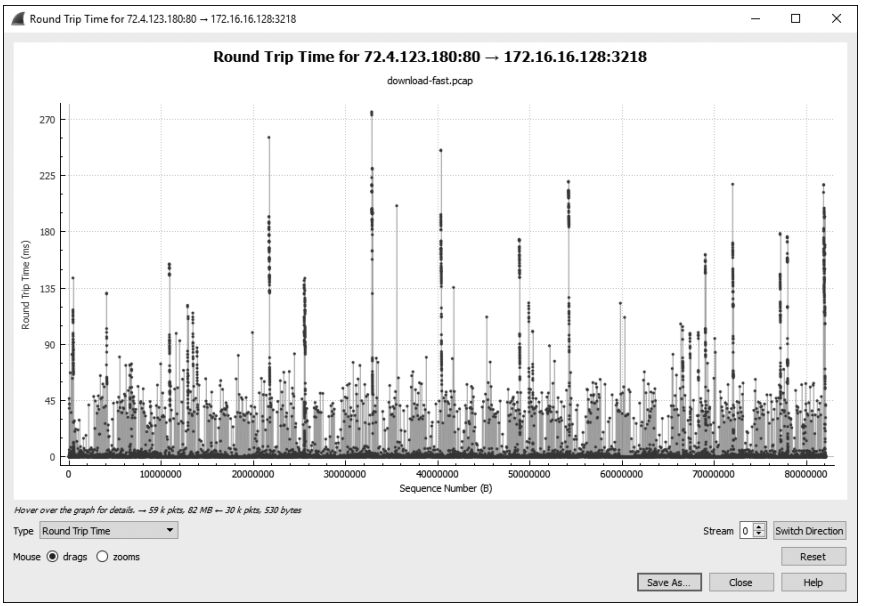


圖 5-21：快速下載的 RTT 圖表趨勢看來十分一致，只有少數的異常值

- 封包的動作碼（opcode）現在變成 0x0002 ❶，代表這是一個回應而非請求。
- 位址資訊都顛倒易位，發送端的 MAC 位址和 IP 位址在這個封包裡變成了目標的 MAC 位址和 IP 位址 ❸。
- 最重要的部分，所有資訊都到齊了，即表示現在我們已經知道目標主機 192.168.0.1 的 MAC 位址，就是 00:13:46:0b:22:ba ❷。

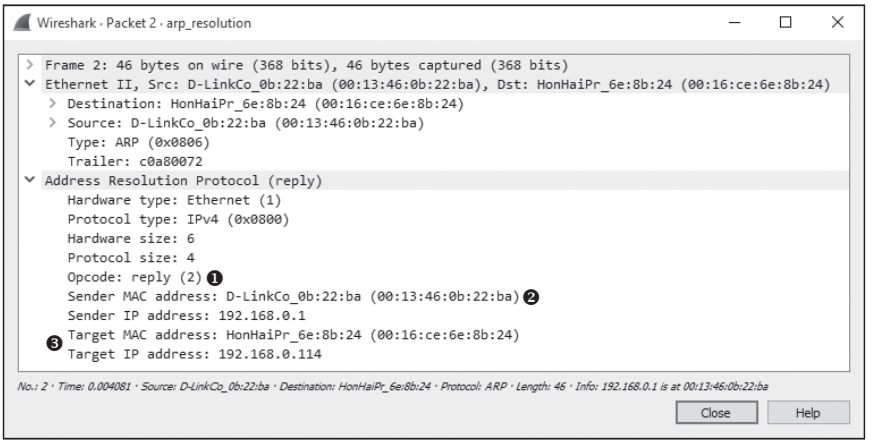


圖 7-4：ARP 回應封包

無償 ARP

arp_gratuitous .pcapng

在我老家如果有人說某事是「無緣無故完成的」（gratuitously done），通常都帶有負面意義。但是一個 *gratuitously ARP* 卻無疑是個好玩意。

在很多狀況下，裝置的 IP 位址都有可能變更。一旦發生這種事，網路上所有其他主機原本存在快取區裡的 IP 與 MAC 位址對應就會立即變成過時資訊。為了避免這個現象造成進一步的通訊錯誤，這時網路上就會傳遞一個無償 ARP 封包（gratuitous ARP packet），強迫所有收到它的裝置都去更新自己的快取區，把 IP 與 MAC 位址對應修正為變更過的 IP 位址（參見圖 7-5）。

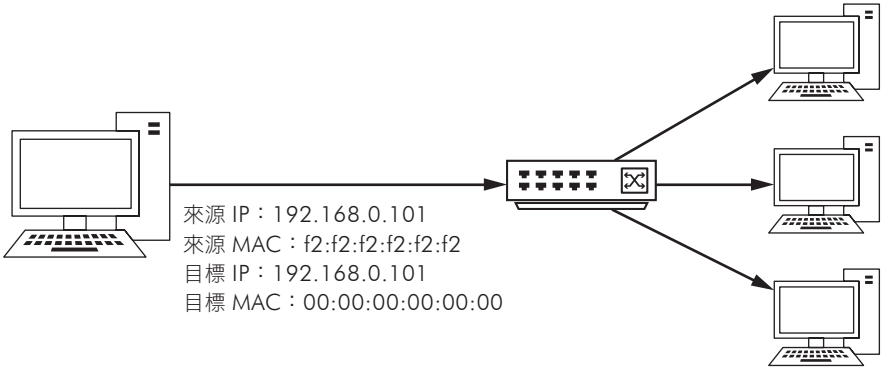


圖 7-5：無償 ARP 處理程序

會產生無償 ARP 封包的情境有好幾種。其中最常見的，就是 IP 位址異動。請開啟 *arp_gratuitous.pcapng* 捕捉結果檔案，你會看到它的行動模式。這個檔案裡只有獨一無二的一個封包（參見圖 7-6），因為無償 ARP 只需這一個封包就可以達成任務。

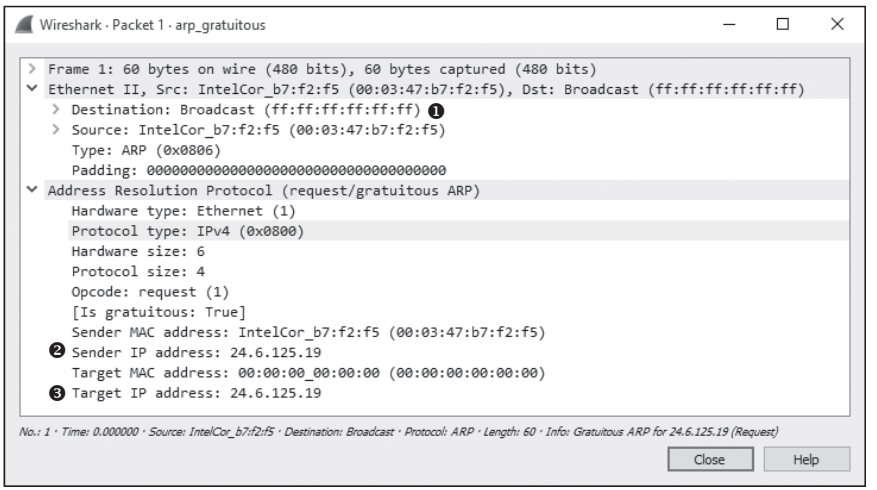


圖 7-6：無償 ARP 封包

請檢視它的乙太網路表頭，你馬上可以看出這也是一個廣播，所以網路上所有裝置都會收到它 ❶。其 ARP 表頭看起來跟 ARP 請求很相似，唯一不同之處是它的發送端 IP 位址 ❷ 和目標 IP 位址 ❸ 居然是一樣的。當

IPv4 位址

IPv4 **位址**是一串長 32 位元的數字，用來作為網路連線裝置獨一無二的辨識方式。如果要求記憶長達 32 個位元的零與一數字組合未免強人所難，因此就有人把 IP 位址改寫成**四個以點區分的數值表示法**（*dotted-quad notation* 或 *dotted-decimalnotation*）。

在四組數值表示法裡，每一組數值都含有 0 與 1 的組合，各組轉換成十進位數值後，就會變成四組介於 0 ～ 255 之間的數字，格式像是 *A.B.C.D*（如圖 7-7 所示）。舉例來說，請看 11000000 10101000 00000000 00000001 這個 IP 位址，任誰也難以記誦這樣的數值。但若改成四個以點區分的十進位數值表示法，就可以寫成 192.168.0.1。

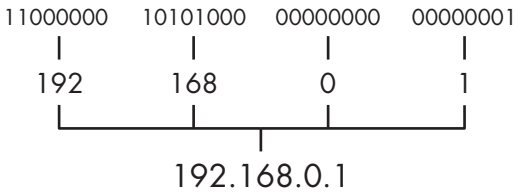


圖 7-7：四個以點區分的 IPv4 位址表示法

IP 位址包含兩個部分：亦即**網路**和**主機**兩部分。網路部分代表的是裝置所在的區域網路，主機部分則代表裝置本身在區域網路裡的身份。你無法直接判斷出 IP 位址中哪一部分屬於網路、哪一部分又屬於主機。只能藉助於另一組定址資訊，也就是所謂的**網路遮罩值**（*network mask* 或 *netmask*），有時也稱為**子網路遮罩**（*subnet mask*）。

NOTE

本書中如果提及 IP 位址，通常都是指 IPv4 位址。稍後我會介紹 IPv6，其定址方式是大相逕庭的。如果要提及 IPv6 位址時，我會明確指出是 IPv6 的定址。

網際網路協定第 4 版（IPv4）							
偏移差	位元組	0		1	2		3
位元組	位元	0–3	4–7	8–15	16–18	19–23	24–31
0	0	版本	表頭長度	服務類型	封包全長		
4	32	識別碼			旗標	分段偏移差	
8	64	存活時間		協定	表頭校驗值		
12	96	來源 IP 位址					
16	128	目標 IP 位址					
20	160	選項					
24+	192+	資料					

圖 7-9：IPv4 封包結構

存活時間

ip_ttl_source.pcapng
ip_ttl_dest.pcapng

存活時間（*Time to Live, TTL*）數值決定的是一段可以持續的時間，或是一個封包能經過的路由器數目上限，超過這個數值時 IPv4 便會認定該封包壽命告罄。TTL 是在封包產生時就已定義好，通常只要它每經過一個路由器被轉送一次，TTL 值便減 1。舉例來說，若是某封包的 TTL 值為 2，則它抵達的第一個路由器便會將 TTL 減 1，再傳給下一個路由器。後者會再將 TTL 減 1，導致 TTL 降為 0，倘若此時封包所在網路仍非最終目的地，這封包就會自動銷毀（參見圖 7-10）。

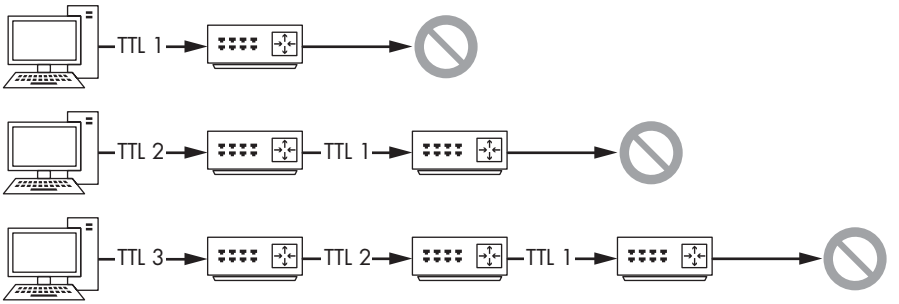


圖 7-10：封包的 TTL 每經過一個路由器便會減 1