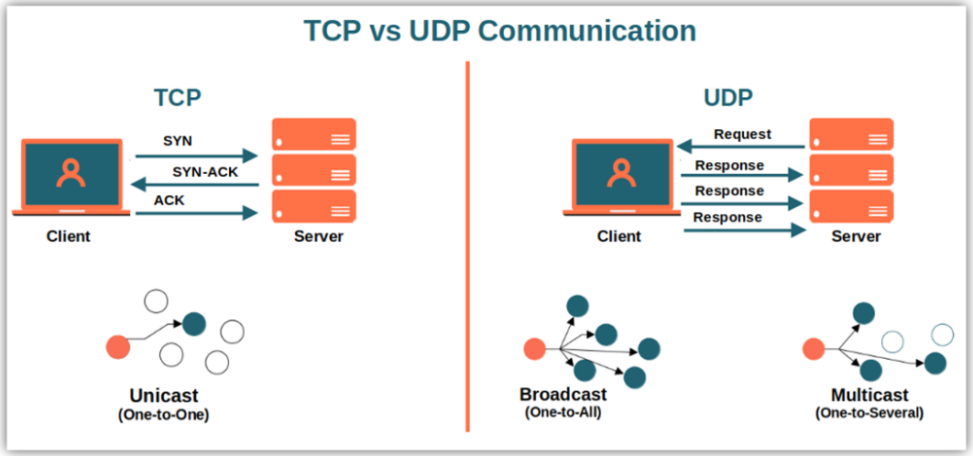


第九章 UDP與TCP

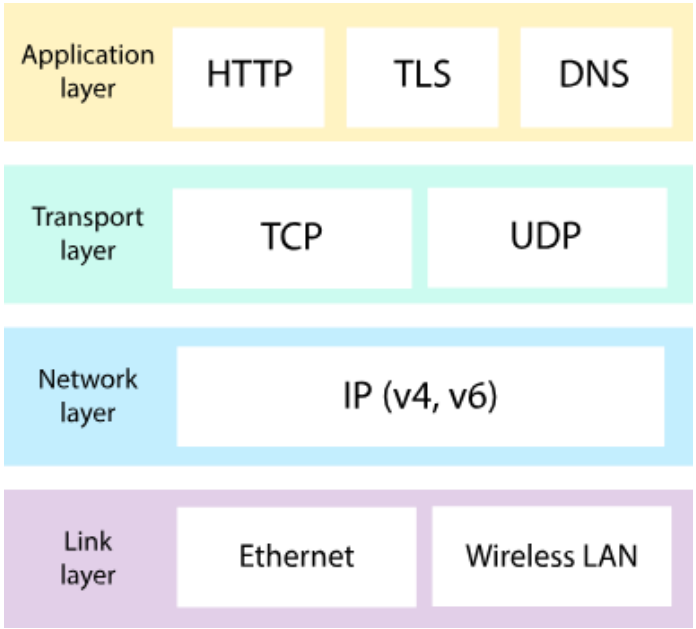
- 本章介紹傳輸層的協定 TCP/UDP，TCP屬於信賴傳輸，而UDP 屬於效能傳輸，為何這麼說，大家看完協定的內容就會了解，由於UDP比較簡單，所以我們先談UDP再來介紹TCP

本章綱要

1. UDP
2. TCP
3. TCP 傳送機制
4. TCP 連線

UDP 與 TCP

- 在 DoD 模型中,傳輸層位於互聯網路層與應用層之間,主要的功能是負責應用程式之間的通訊。



1 UDP

UDP (User Datagram Protocol) 僅提供連接埠 (Port) 處理的功能。

- 記錄封包來源端與目的端的連接埠資訊, 正確地送達目的端的應用程式。
- 非連接式 (Connectionless) 的傳送。

使用 UDP 的考量：

- 為了要降低對電腦資源的需求。
- 應用程式本身已提供資料完整性的檢查機制

UDP

- 1-1 連接埠
- 1-2 UDP 的錯誤檢查碼

1-1 連接埠

- UDP 最重要的功能是管理連接埠
- 什麼是連接埠？
  - 連接埠的英文為 Port, 是一種邏輯上的概念。每一部使用 TCP/IP 的電腦, 都會有許多連接埠, 並使用編號加以區分。
  - 應用程式若經由 TCP/IP 存取資料, 就必須獨佔一個連接埠編號。
  - IP 位址與連接埠編號兩者合起來稱為 Socket Address, 可用來定義 IP 封包最後送達的終點, 亦即目的地應用程式。

連接埠

- 連接埠編號的原則
  - 0～1023

    此段的連接埠編號稱為『Well-Known』(廣為人知的) 連接埠,主要給提供服務的伺服器應用程式使用 。
  - 1024～49151

    此部份稱為『Registered』連接埠, 保留給特定的應用程式或服務使用 。
  - 49152～65535

    稱為『Dynamic』(動態) 連接埠, 由用戶端自行使用 。

連接埠

表 11-1 常見的 Well-Known 連接埠

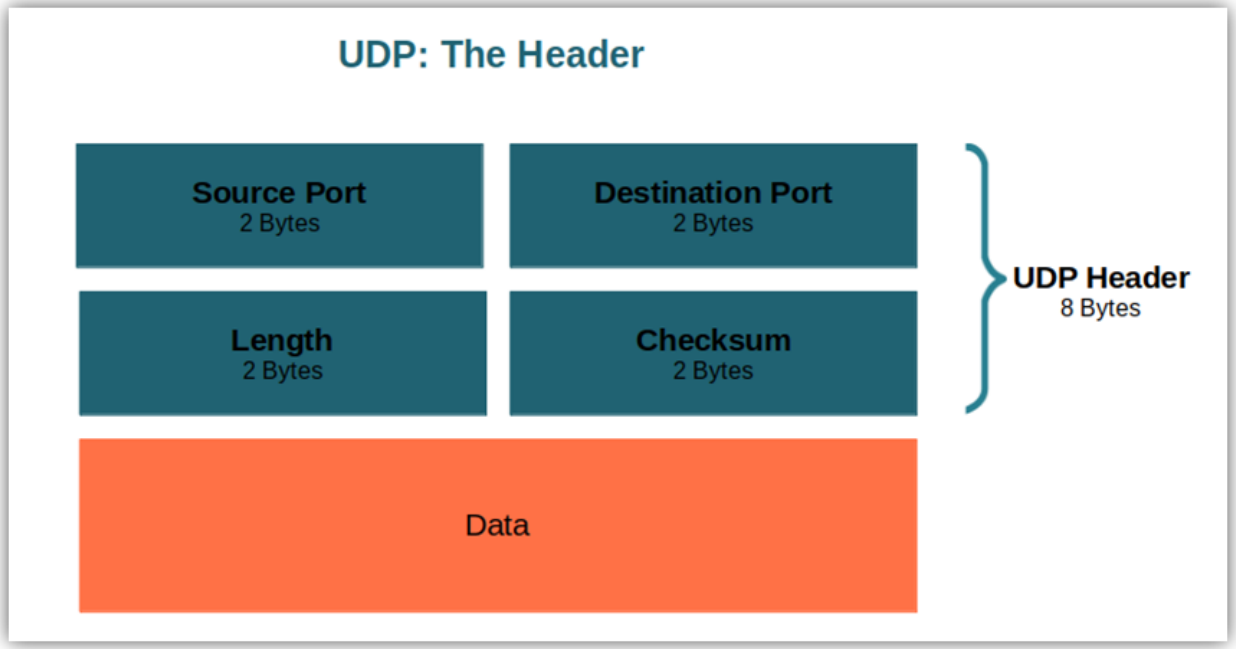
協定	連接埠編號	應用程式
UDP	53	DNS
UDP	68	DHCP Client
UDP	67	DHCP Server
UDP	520	RIP
TCP	19	NNTP
TCP	20	FTP Data
TCP	21	FTP Control
TCP	23	Telnet
TCP	25	SMTP
TCP	80	HTTP
TCP	110	POP3

用戶端有時也須要使用 Well-Known 連接埠。

有些伺服器需要 1 個以上的連接埠編號。例如：FTP 伺服器必須用到 2 個連接埠編號。

連接埠

- 使用自訂的伺服器連接埠編號
  - Well-Known 連接埠其實有點類似『約定俗成』的意思, 並不具有強制性



1-2 UDP 的錯誤檢查碼

- UDP 封包的表頭中有一個錯誤檢查碼欄位。, 除了 UDP 表頭與 UDP 資料, 計算錯誤檢查碼時, 會另外產生『Pseudo Header』(假表頭), 它包括以下的欄位
  - 來源位址：IP 表頭中來源端的 IP 位址。
  - 目的位址：IP 表頭中目的端的 IP 位址。
  - 未用欄位：長度為 8 Bits, 填入 0。
  - 上層協定：IP 表頭中紀錄上層協定編號的欄位, UDP 協定的編號為 17。
  - 封包長度：UDP 表頭中的封包長度欄位。

UDP 的錯誤檢查碼

UDP 封包傳送過程中, 計算錯誤檢查碼的步驟：

1. 在來源端電腦中, UDP 計算錯誤檢查碼時, 會暫時將 Pseudo Header 加至 UDP 封包：

圖 11-1　錯誤檢查碼的計算範圍

UDP 的錯誤檢查碼

2. 在計算完錯誤檢查碼後立即將 Pseudo Header 與 Padding 移除。
3. 從 IP 表頭讀取相關資訊, 再度產生 Pseudo Header 與 Padding, 而後計算錯誤檢查碼, 然後與 UDP 表頭中的錯誤檢查碼比對。

2 TCP

- TCP 為傳輸層的協定, 也具備處理連接埠的功能。提供了一種『可靠』的傳送機制。
- TCP 『可靠』的傳輸方式, 大致有以下幾種特性：
  - 資料確認與重送
  - 流量控制
  - 連線導向

3 TCP 傳送機制

- 3-1 確認與重送
- 3-2 Sliding Window
- 3-3 Send / Receive Window
- 3-4 Window Size 與流量控制
- 3-5 TCP 將資料視為 Byte Stream
- 3-6 雙向傳輸
- 3-7 傳送機制小結

3-1 確認與重送

- TCP 『可靠』的傳送機制簡而言之，就是『確認與重送』

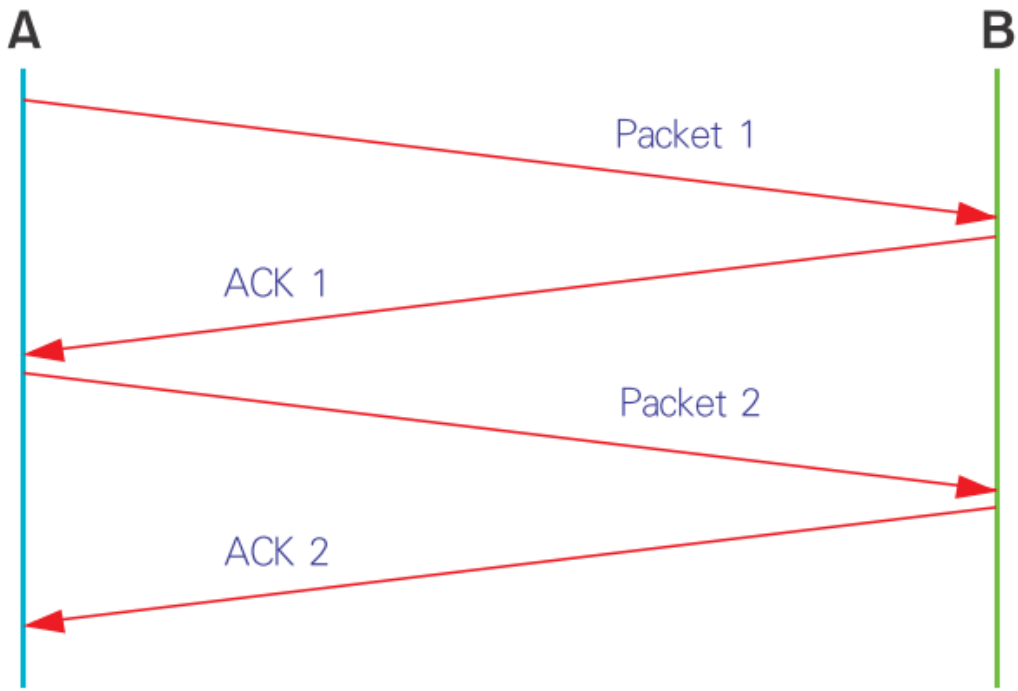


圖 11-2 利用確認機制可得知對方收到了封包

確認與重送

1. A 首先傳送 Packet 1 封包給 B
2. B 收到 Packet 1 封包後, 傳送 ACK 1 封包給 A
3. A 收到 ACK 1 封包, 接著即可傳送Packet 2 封包給 B
4. B 收到 Packet 2 封包後, 傳送 ACK 2 封包給 A

確認與重送

- 在 TCP 傳送過程中，即使發生錯誤，仍可藉由重送封包的方式來補救，維持資料的正確性與完整性。

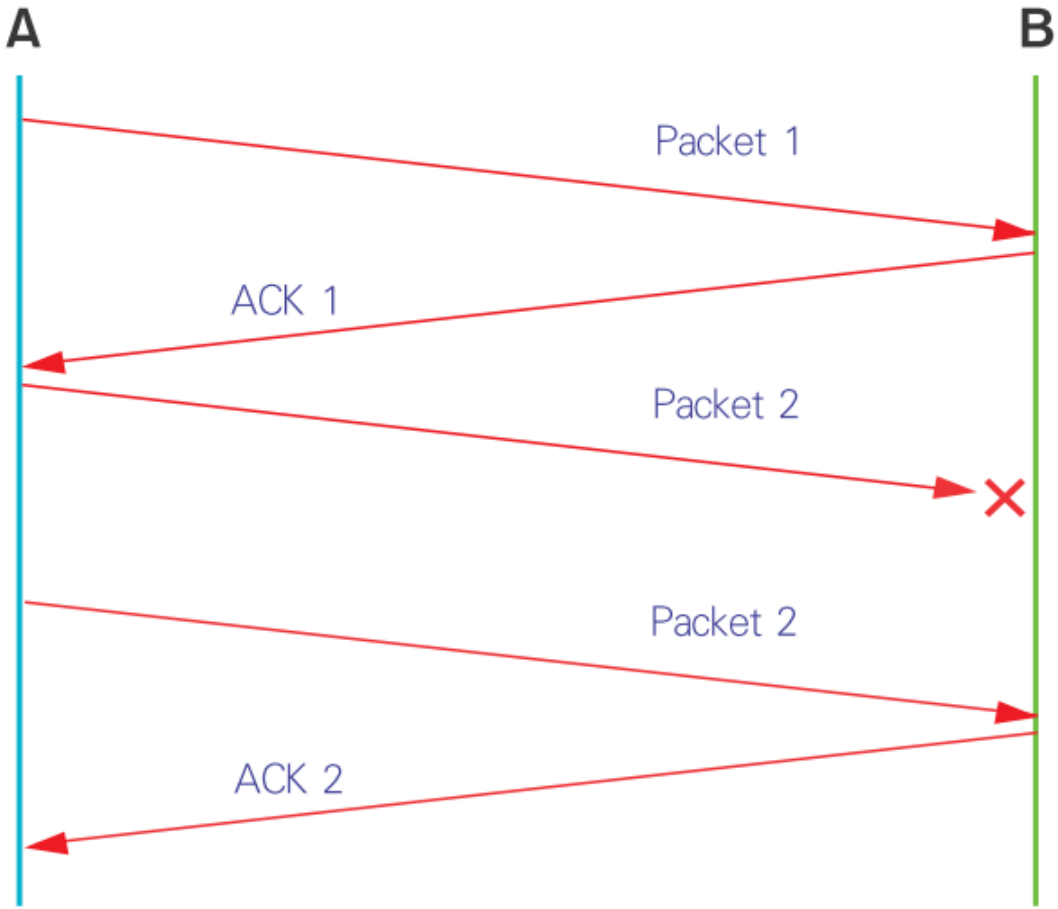


圖 11-3　利用重送機制來處理傳送過程中的錯誤

3-2 滑動窗(Sliding Window)

- 上述封包傳送的過程, 大部份時間都在等 ACK 封包。『Sliding Window』技術可改善此問題

3-2滑動窗(Sliding Window)

- 在傳送一開始時, A 的 Sliding Window 應該是像這個樣子：

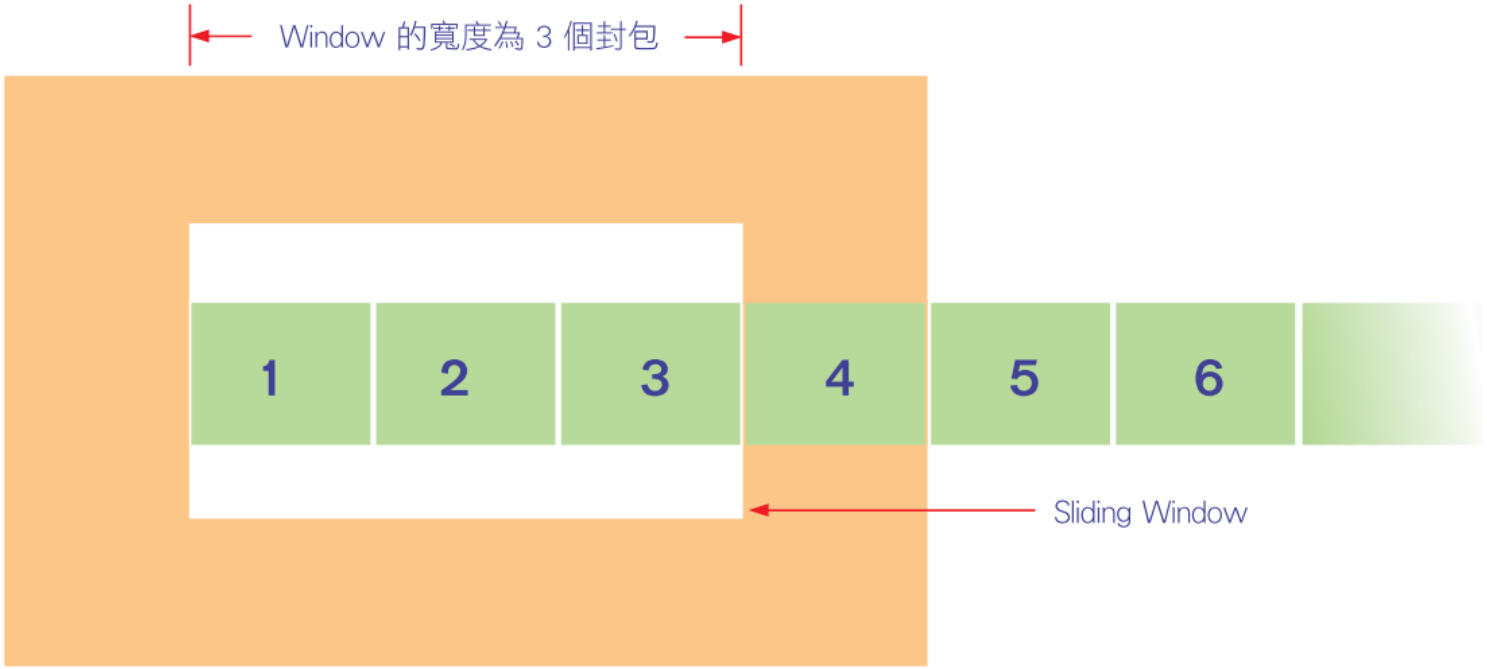


圖 11-4　A 剛開始傳送時, Sliding Window 的狀態

Sliding Window

- 1.將 Packet 1 標示為『完成』(以「深綠色」表示)

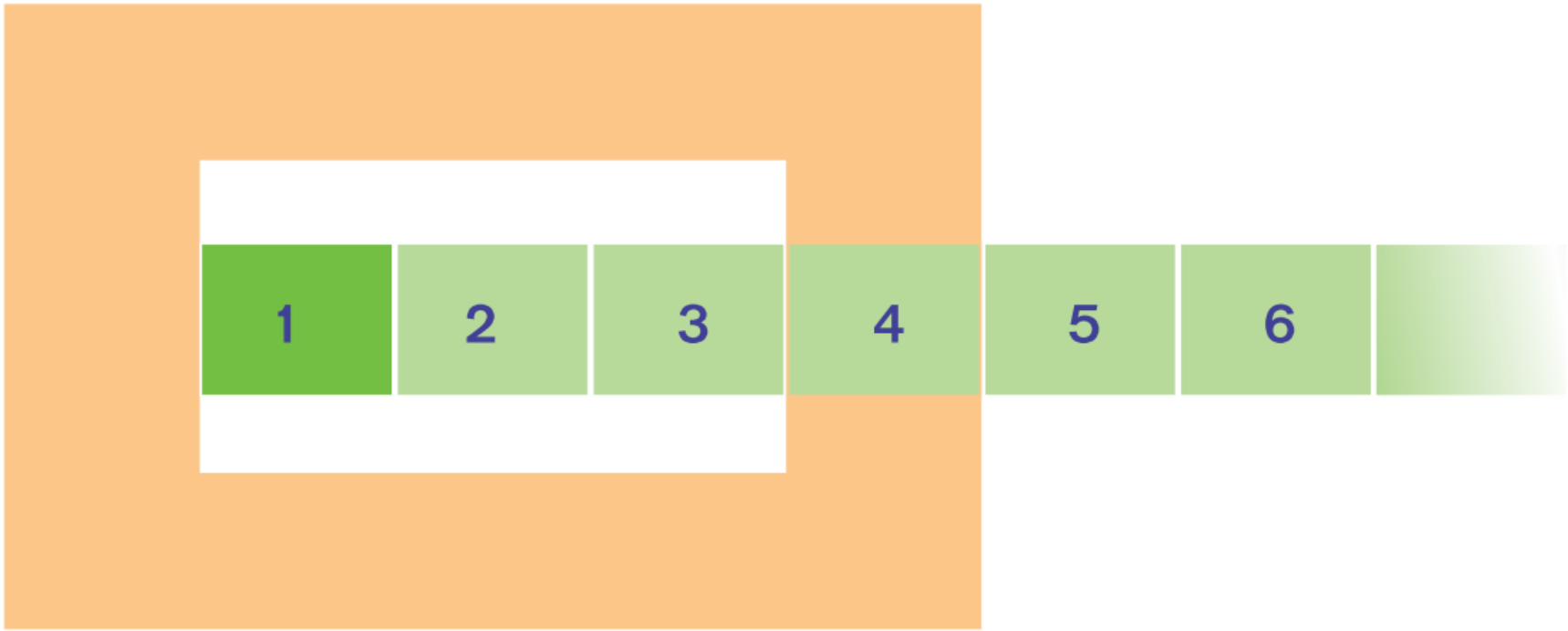


圖 11-5　收到 ACK 1 後, A 的 Sliding Window 首先將 Packet 1 標示為『完成』

Sliding Window

- 2. 將 Sliding Window 往右滑動 1 格。

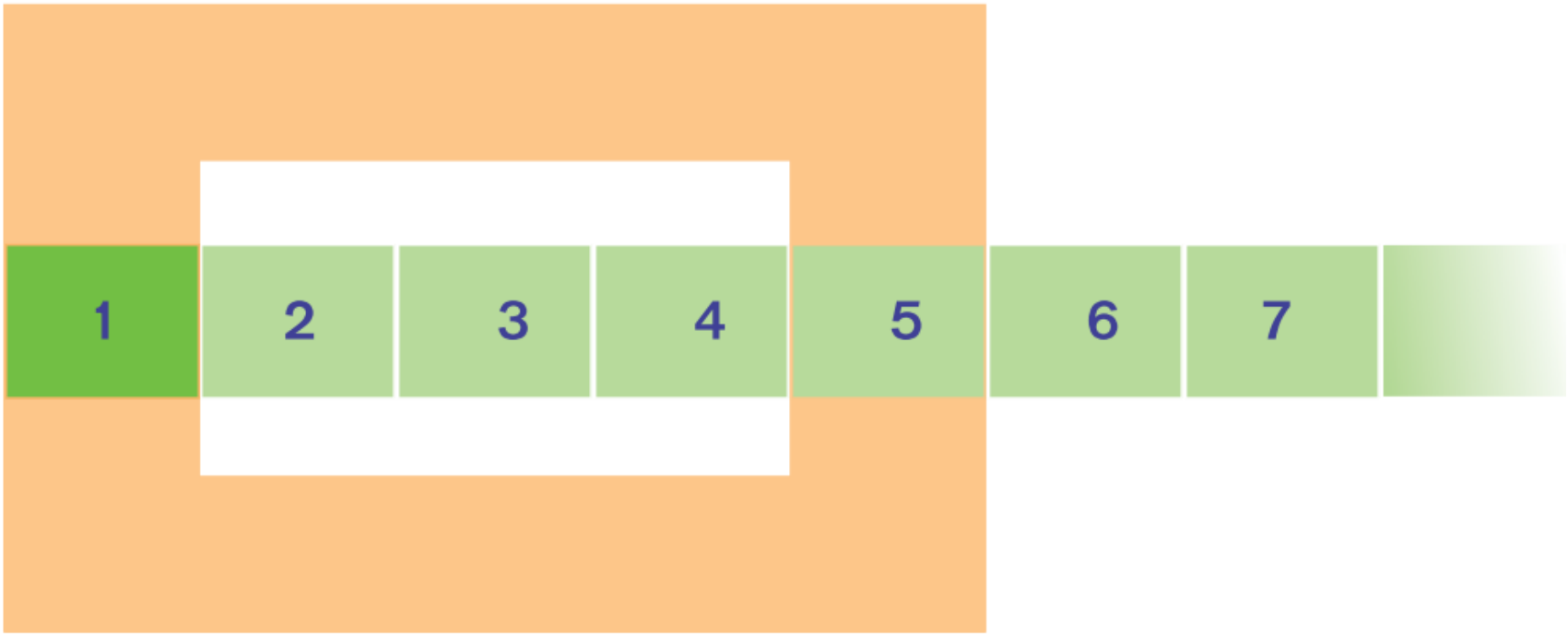


圖 11-6　A 的 Sliding Window 往右滑動

Sliding Window

- 3. 將新進入 Sliding Window 的 Packet 4 (位於 Window 的最右邊) 送出。

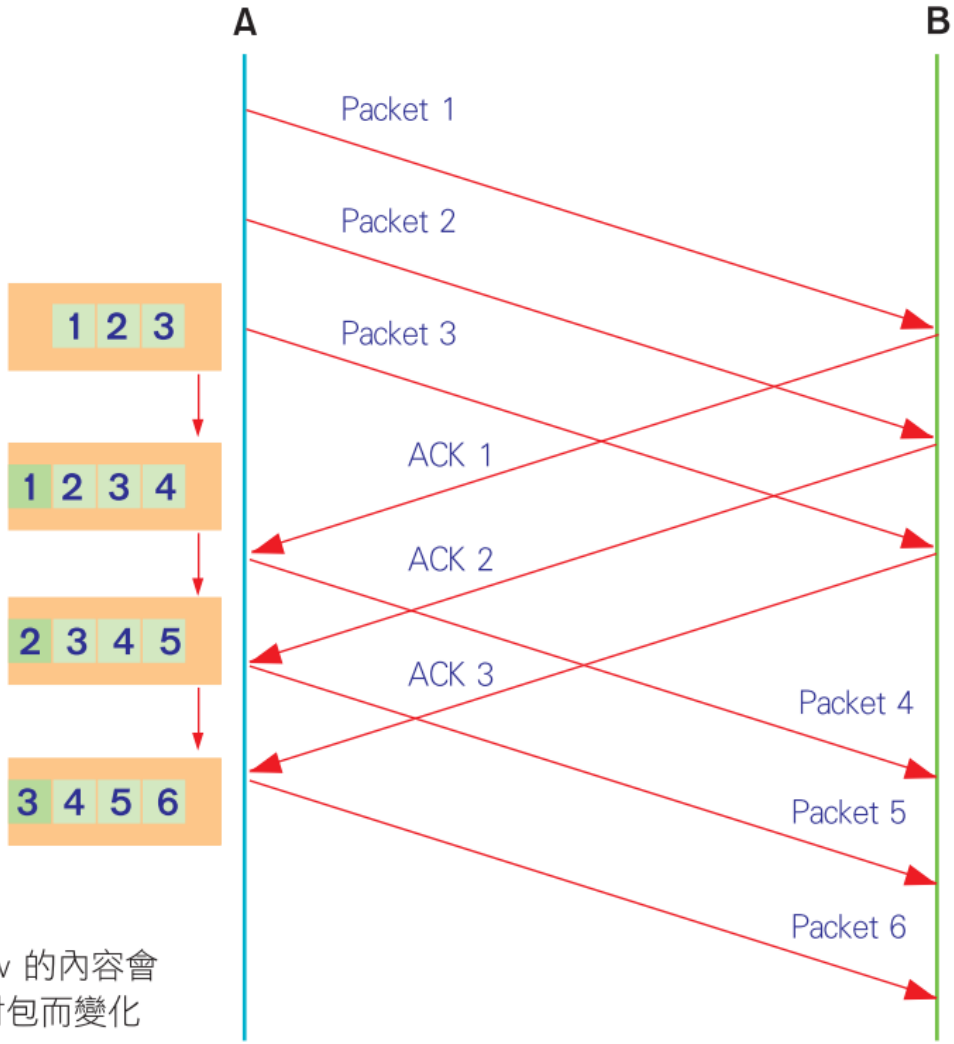


圖 11-7 A 的 Sliding Window 的內容會隨著收到的 ACK 封包而變化

Sliding Window

- 相較於每送出一個封包便要等待回應的 ACK 封包, Sliding Window可以迅速送出多個封包。

11-3-3 Send / Receive Window

- TCP 的來源端與目的端會有各自的 Sliding Window。為了方便區分, 我們將來源端的 Sliding Window 稱為 Send Window (傳送窗口), 目的端的 Sliding Window 稱為 Receive Window (接收窗口)

Send / Receive Window

- Receive Window會記錄連續收到的封包與非連續收到的封包。只會將連續收到的封包轉交給上層應用程式；同樣地, 也只會針對連續收到的封包發出 ACK。

圖 11-8 目的端只會將連續收到的封包轉交給上層應用程式, 並發出對應的 ACK

Send / Receive Window

- 在傳送一開始時, 目前端的 Receive Window 應該是像這個樣子：

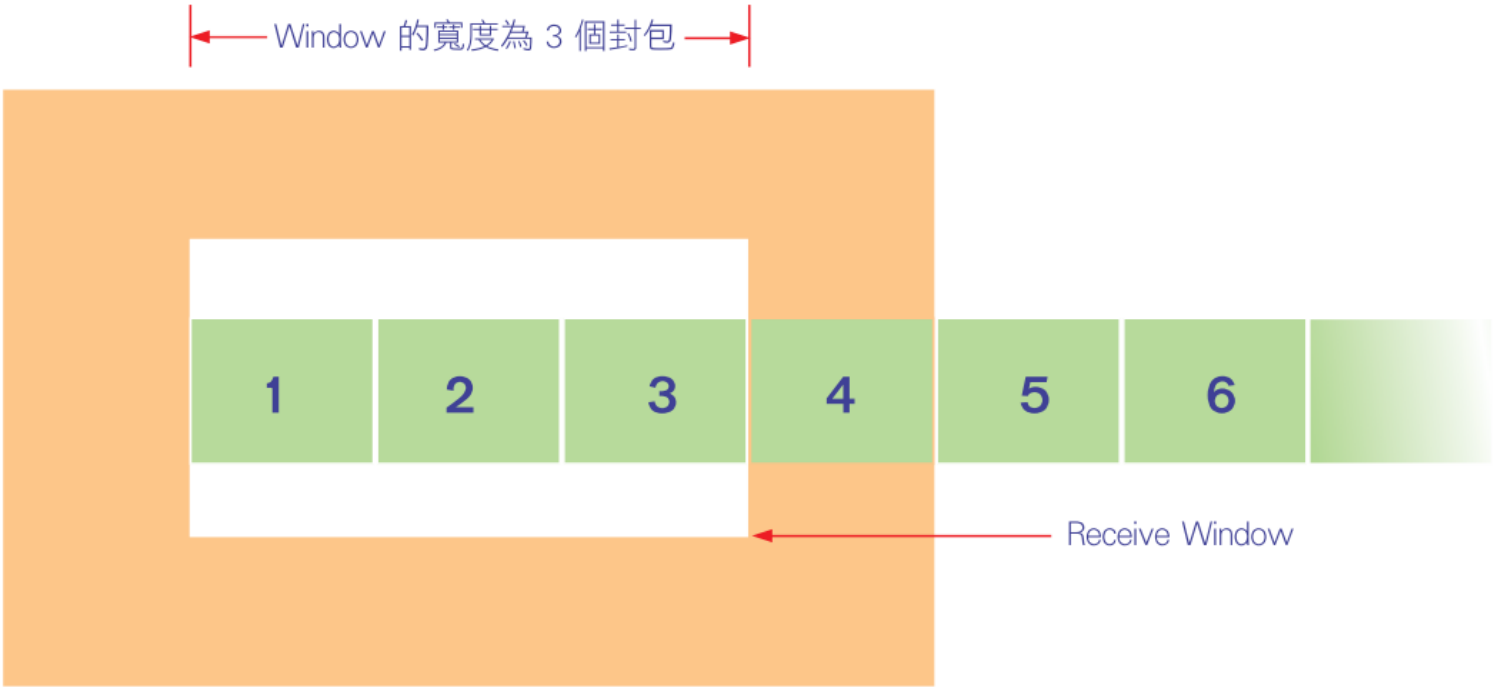


圖 11-9　開始傳送時, B 的 Receive Window 的狀態

Send / Receive Window

- 1.將收到的封包加以標示。

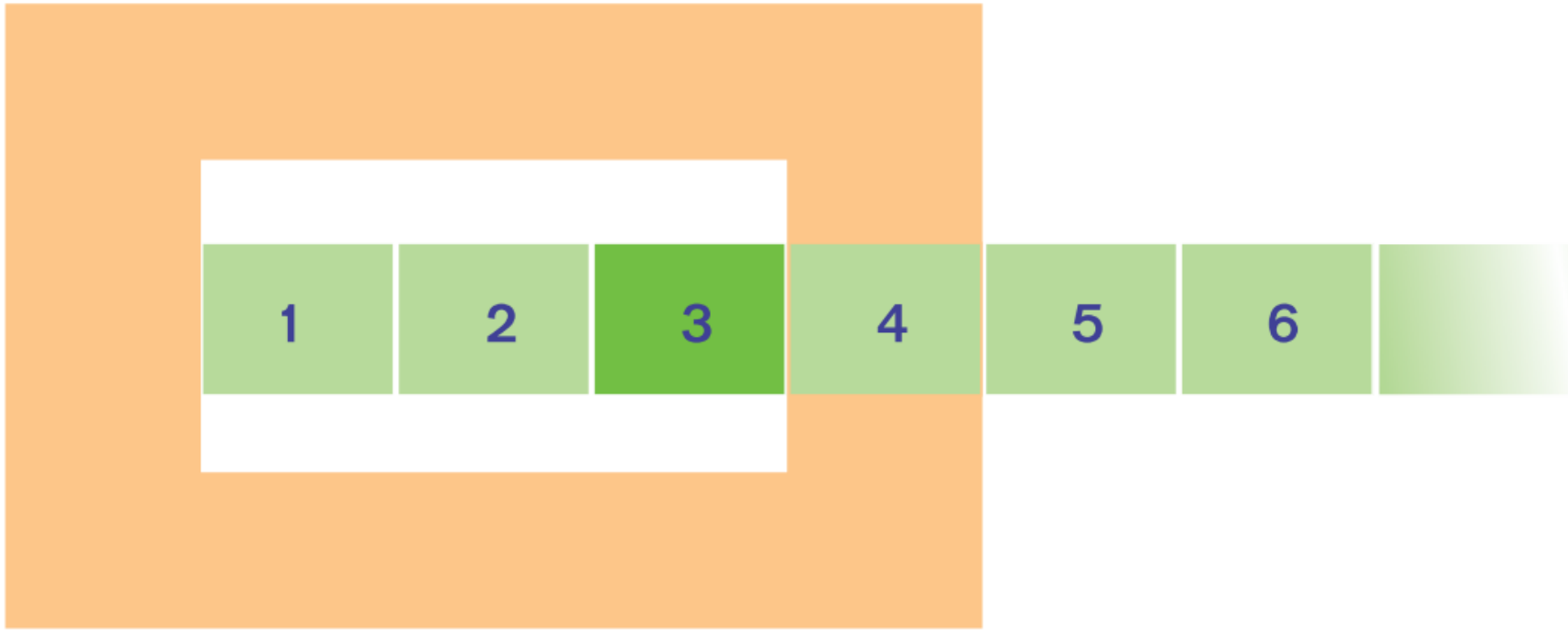
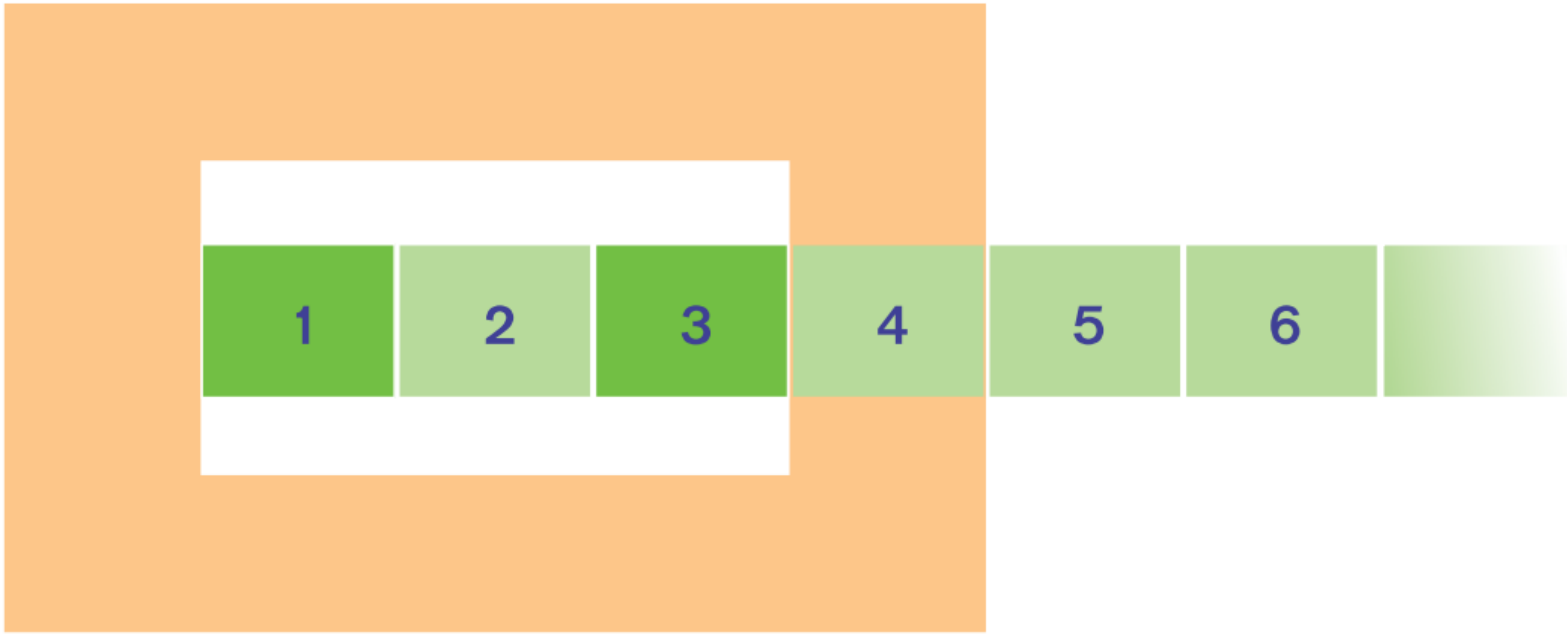


圖 11-10　收到 Packet 3 後, B 的 Receive Window 的狀態

Send / Receive Window

- 2. 如果收到的封包位於 Window 的最左邊, 則發出對應的 ACK 封包, 並將 Receive Window 往右滑動 1 格。



Send / Receive Window

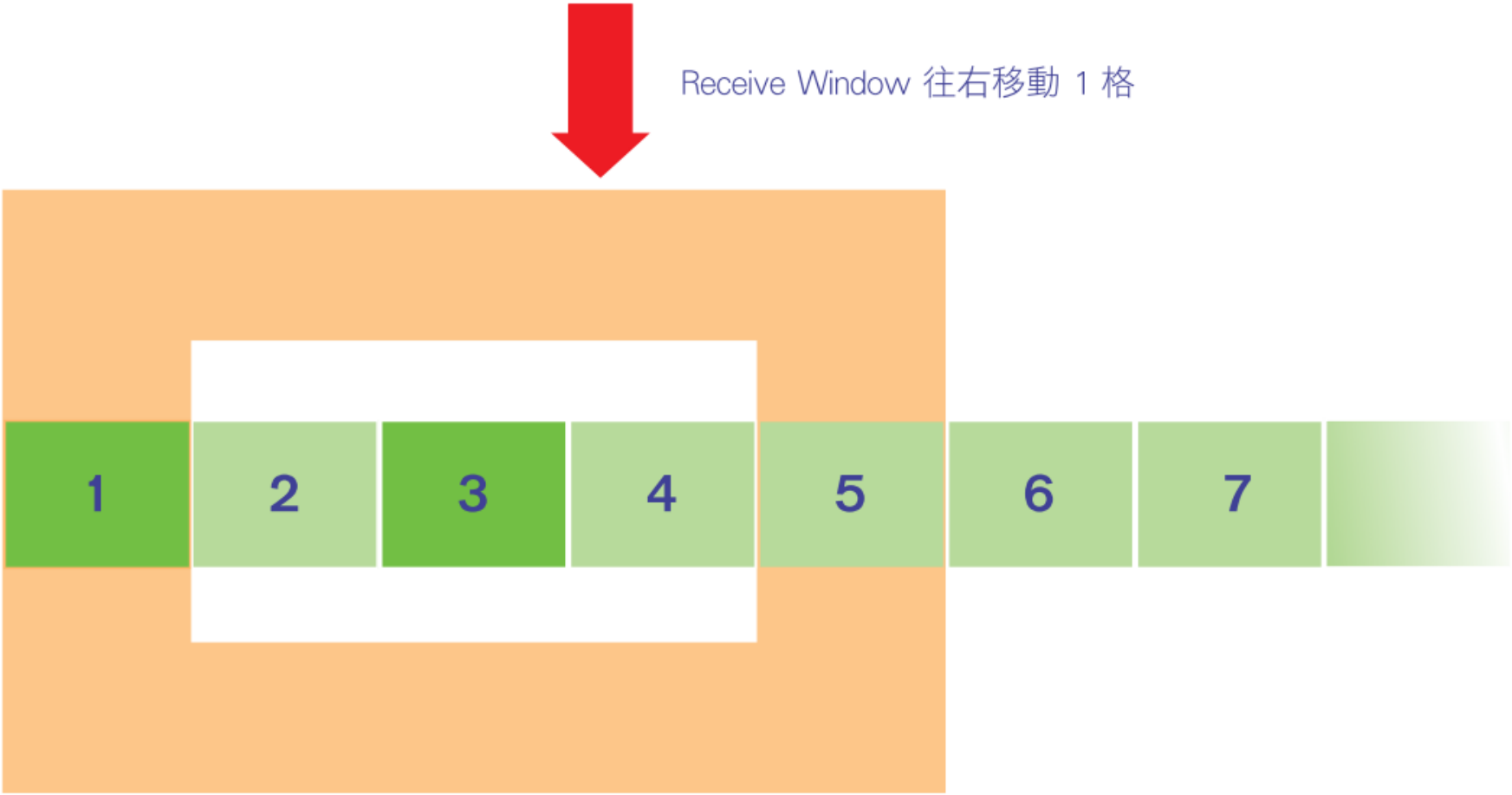
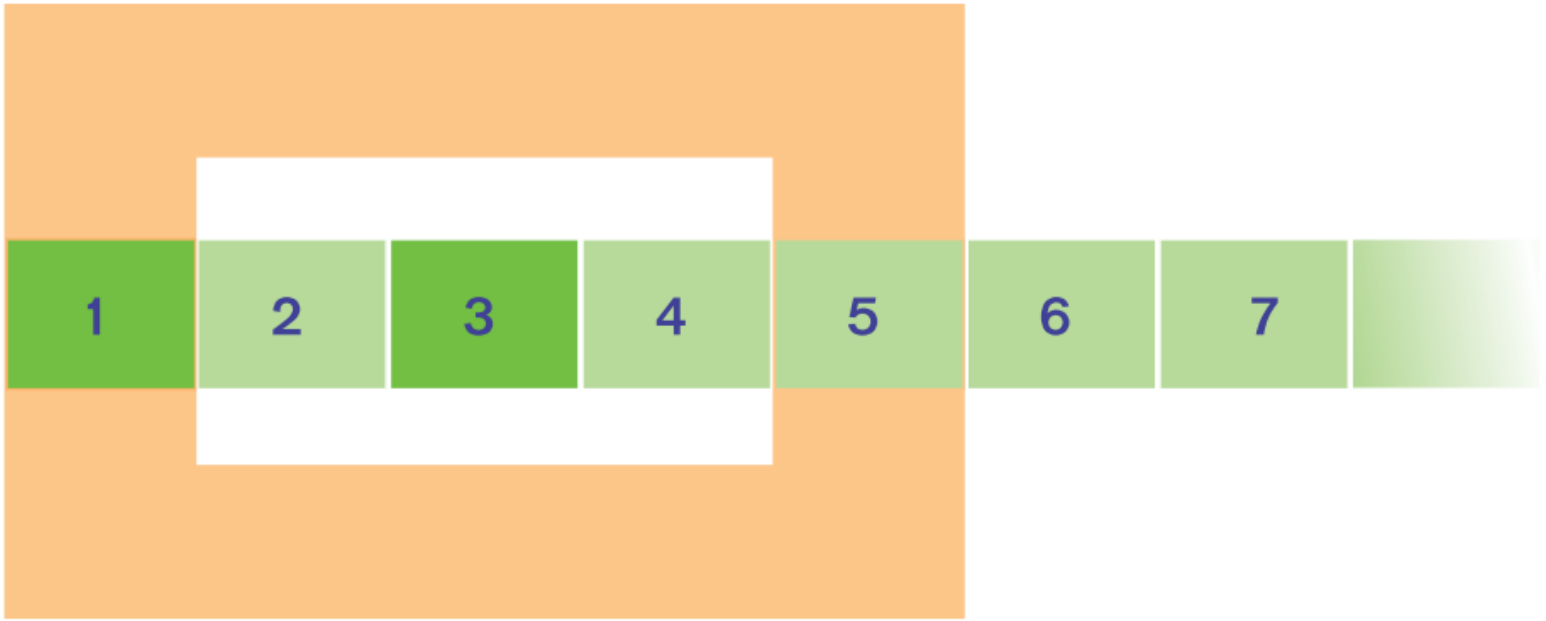


圖 11-11　B 收到 Packet 1 後, 其 Receive Window 的變化

Send / Receive Window

- 3.如果 Window 最左邊的封包也已經標示為『收到』,則再往右 1 格, 直到 Window 最左邊的是尚未標示為『收到』的封包。



Send / Receive Window

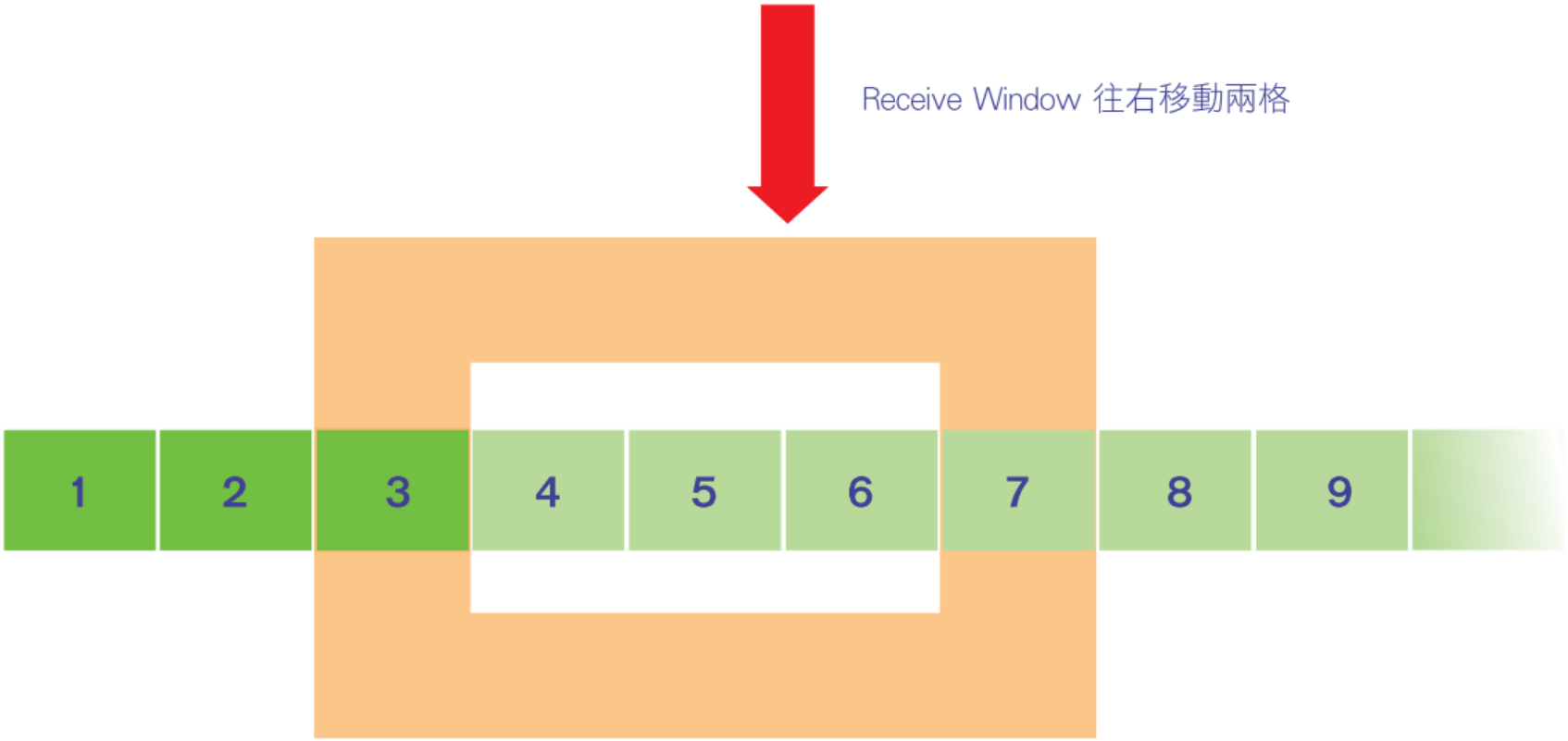


圖 11-12　收到 Packet 2 後, B 的 Receive Window 的變化

Send / Receive Window

- 總結Send Window 與 Receive Window 在上述步驟中的變化

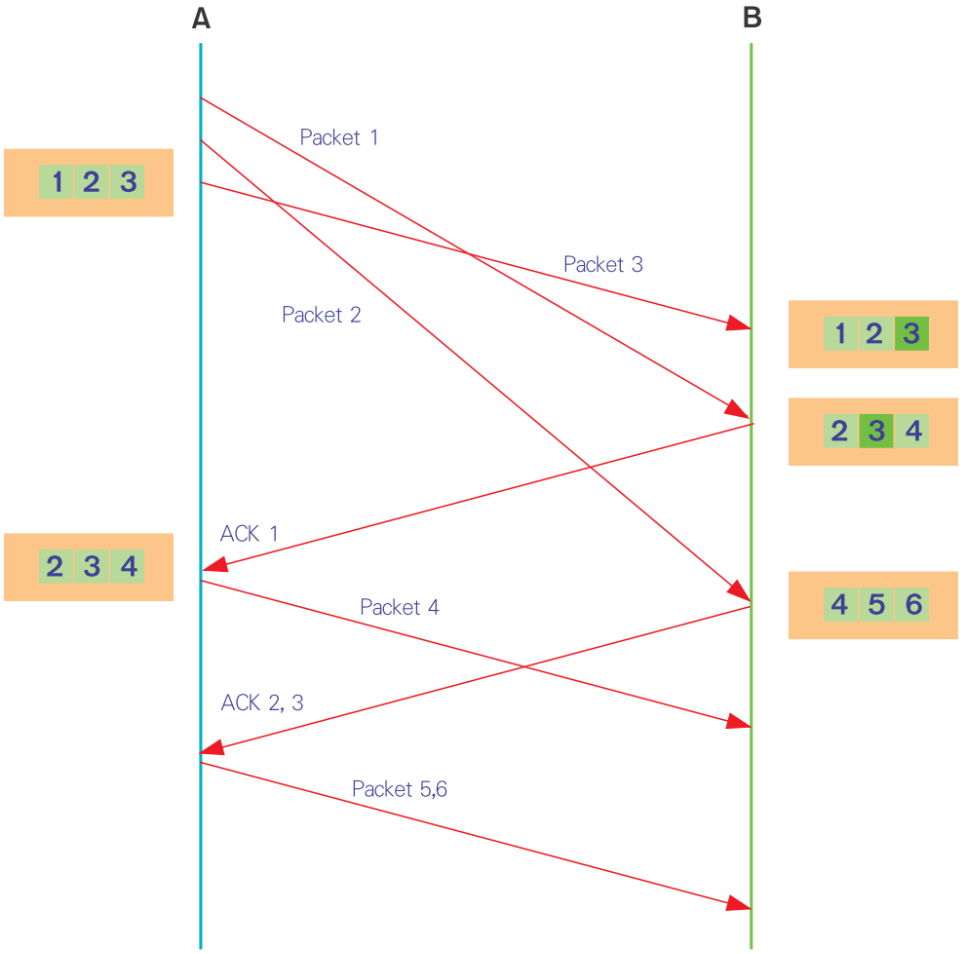


圖 11-13 Send / Receive Window 的變化情形

3-4 Window Size 與流量控制

- TCP 具有一項重要的功能, 便是流量控制 (Flow Control),主要是靠 Sliding Window 的大小 (稱為 Window Size) 來調整
  - 當Window Size 變小時, 流量也會變慢。常需要等待確認, 傳輸效率極差。
  - 當 Window Size 變大時, 可連續傳送多個封包, 流量也會變快。會耗費較多的電腦資源。
- Window Size 是由目的端決定,在整個傳送過程中, Receive Window 大小會隨著客觀條件不斷變化

3-5 TCP 將資料視為 Byte Stream

- TCP 處理傳送或接收的資料時, 都是『以 Byte 為計量單位』。用術語來說, 這叫做『將資料視為 Byte Stream (位元組串流)』

TCP 將資料視為 Byte Stream

- 序號 (Sequence Number)
  - TCP 封包內所記錄的序號 (Sequence Number), 是指 TCP Payload 的第 1 個 Byte 的編號, 不是封包本身的編號
  - 在連線一開始, A 會隨機選取一個數字作為 Initial Sequence Number (ISN, 初始序號), 並以 1 個同步封包 通知 B, 俟獲得 B 回覆 ACK 之後, A 開始將 TCP Payload 打包、編號送出。但是所編的第 1 個號碼不是 ISN, 而是 ISN+1。
  - 所有 A 送給 B 的封包, 都會標示所載送資料的第 1 Byte 的編號。

TCP 將資料視為 Byte Stream

表 11-2 Packet 1、2、3 的序號

封包編號	Payload 長度	序號
1	100	1001 (ISN+1)
2	200	1101 (1001 + 100, 亦即 Packet 1 的 Sequence Number 再加上 Packet 1 之 Payload 的長度)
3	300	1301 (1101 + 200, 亦即 Packet 2 的 Sequence Number 再加上 Packet 2 之 Payload 的長度)

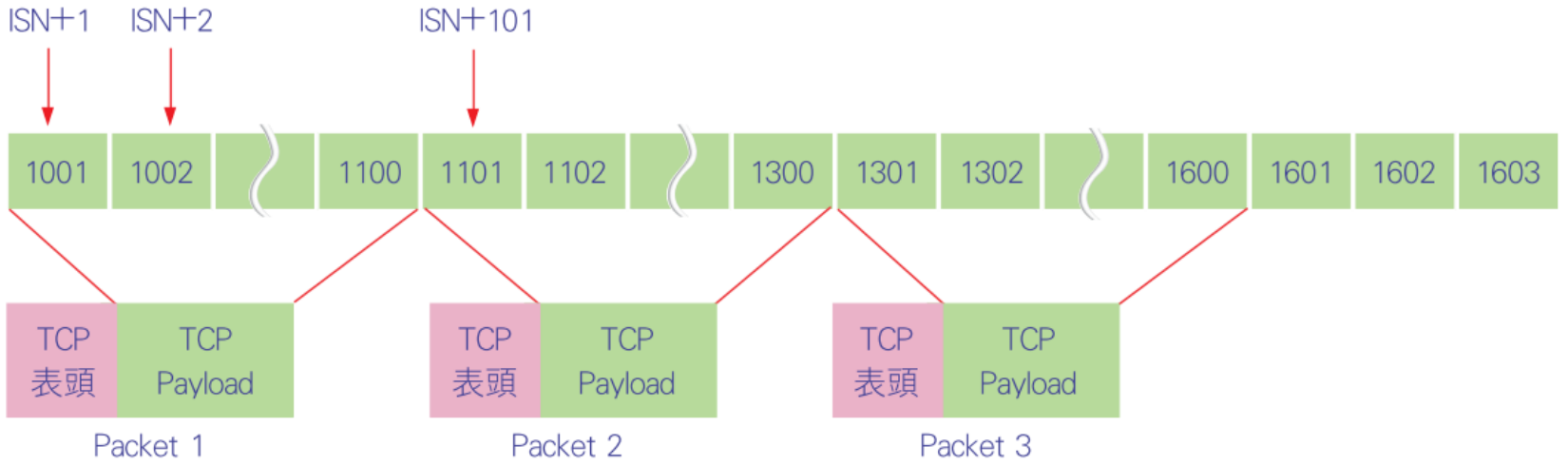


圖 11-14 序號與封包之間的關係

TCP 將資料視為 Byte Stream

- 回應序號 (Acknowledge Number)
  - 回應序號代表期望自己下一次收到的封包序號。
  - 回應序號還有另一種意義－用來確認之前的封包已經收到

TCP 將資料視為 Byte Stream

表 11-3　ACK 1、2、3 的回應序號

ACK 封包	回應序號
1	1101
2	1301
3	1601

- Packet 1 的序號= ISN+1
- Packet 2 的序號= ACK 1 的回應序號
- Packet 3 的序號= ACK 2 的回應序號

TCP 將資料視為 Byte Stream

- 從傳送端的角度來看

  序號代表此 TCP 封包的 Payload 中, 第 1 個 Byte 的編號；回應序號代表對方回應 ACK 時, 該回應封包的序號。

- 從接收端的角度來看

  根據序號可知此 TCP 封包的 Payload 從哪個號碼開始；根據回應序號, 可期望自己下一次接收序號為何的封包。

TCP 將資料視為 Byte Stream

- 定義 Window 的邊界
  Sliding Window 同樣是以 Byte 為單位來界定 Window, 而非以封包編號。

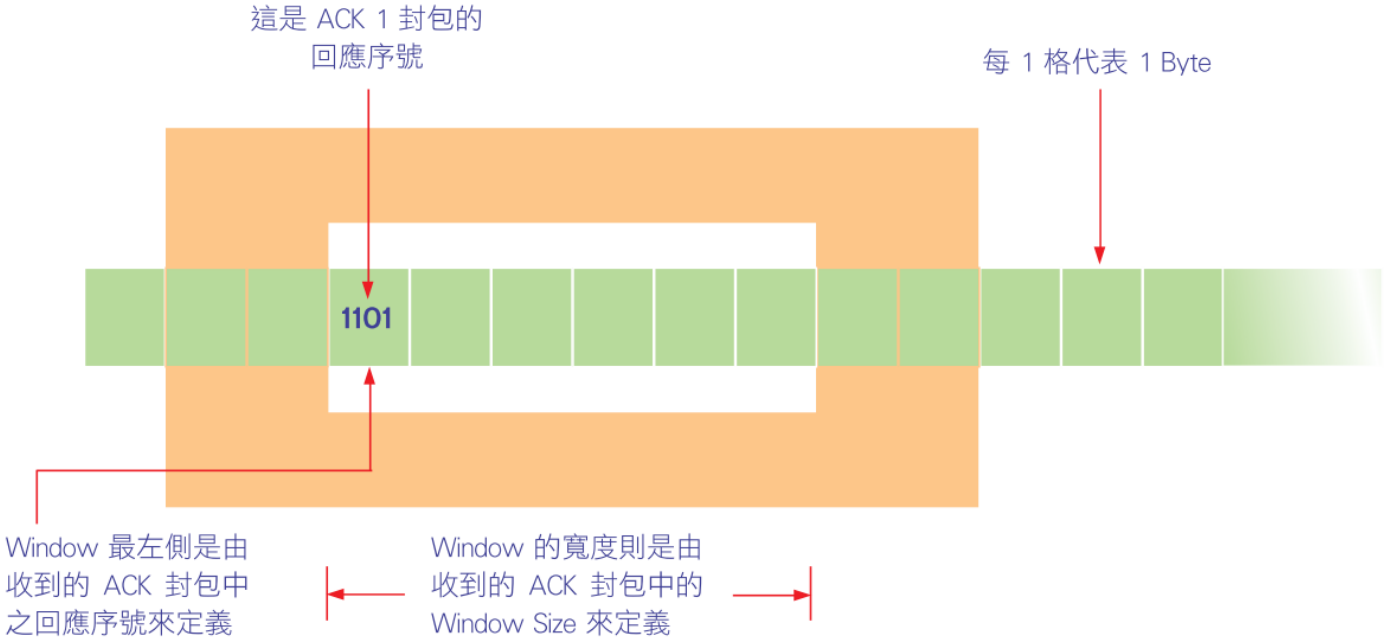


圖 11-15 以 Byte 為單位定義 Send Window 的邊界

TCP 將資料視為 Byte Stream

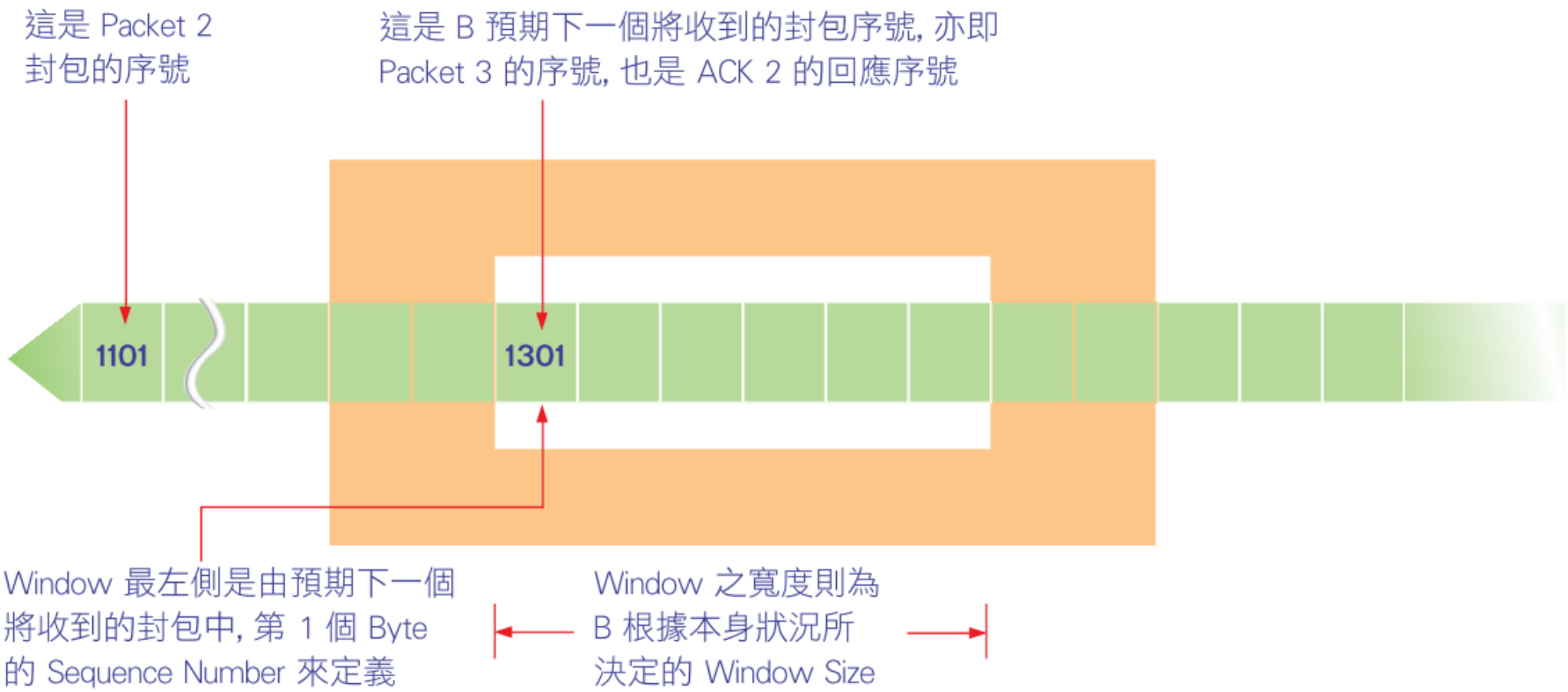


圖 11-16　以 Byte 為單位定義 Receive Window 的邊界

3-6 雙向傳輸

- TCP 是一個雙向的協定,可想像成由兩條單向管道所構成的雙向傳輸

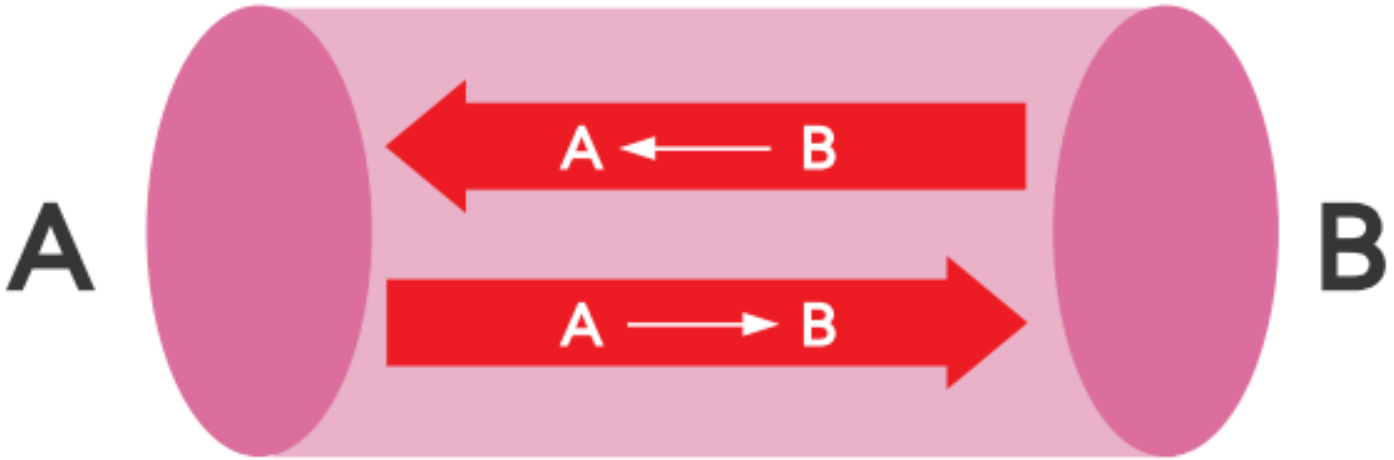


圖 11-17 TCP 連線是由兩條單向傳輸的管道結合而成

雙向傳輸

- A、B 之間互傳的封包，都扮演『傳送兼回應、回應兼傳送』雙重角色

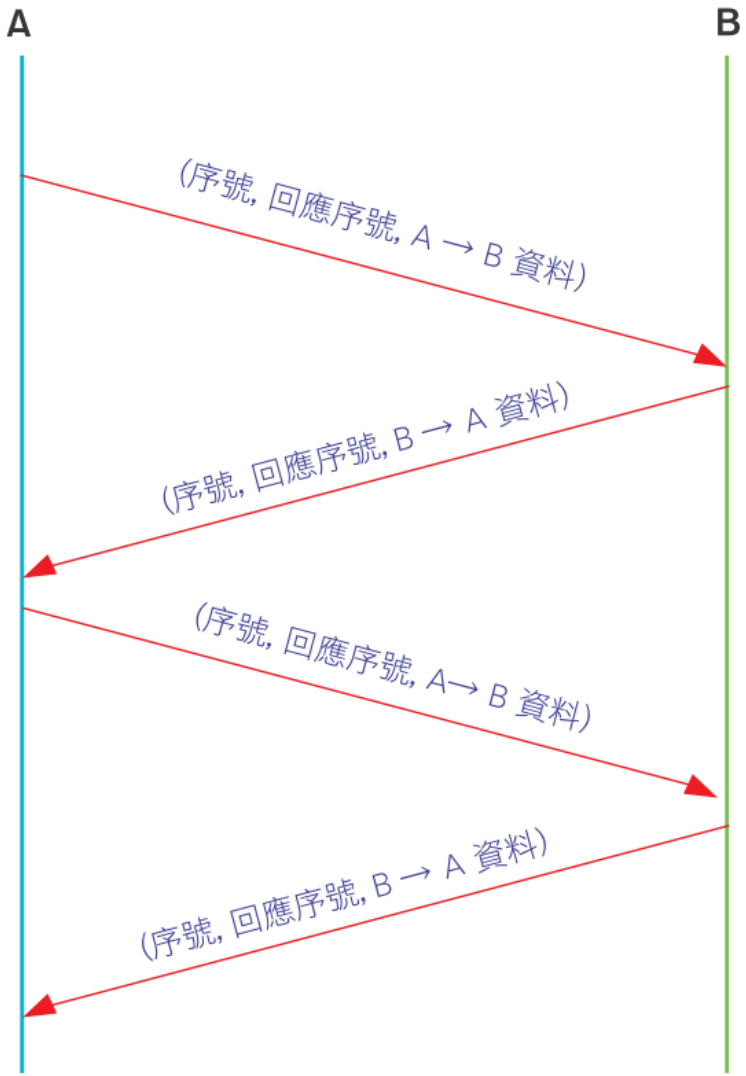


圖 11-18　每個 TCP 封包可能包含雙向傳輸的資訊

3-7 TCP傳送機制小結

- TCP 傳送包含確認與重送的機制
- TCP 傳送包含流量控制的機制
- TCP 將資料視為 Bytes Stream
- TCP 為雙向傳輸的協定

4 TCP 連線

- 4-1 識別連線
- 4-2 建立連線
- 4-3 中止連線

4-1 識別連線

- TCP 連線是由連線兩端的 IP 位址與連接埠編號所定義，伺服器可以和同一用戶端的不同連接埠建立多條連線，或是和不同的用戶端同時建立連線

圖 11-19　TCP 連線是由連線兩端的 IP 位址與連接埠編號所定義

識別連線

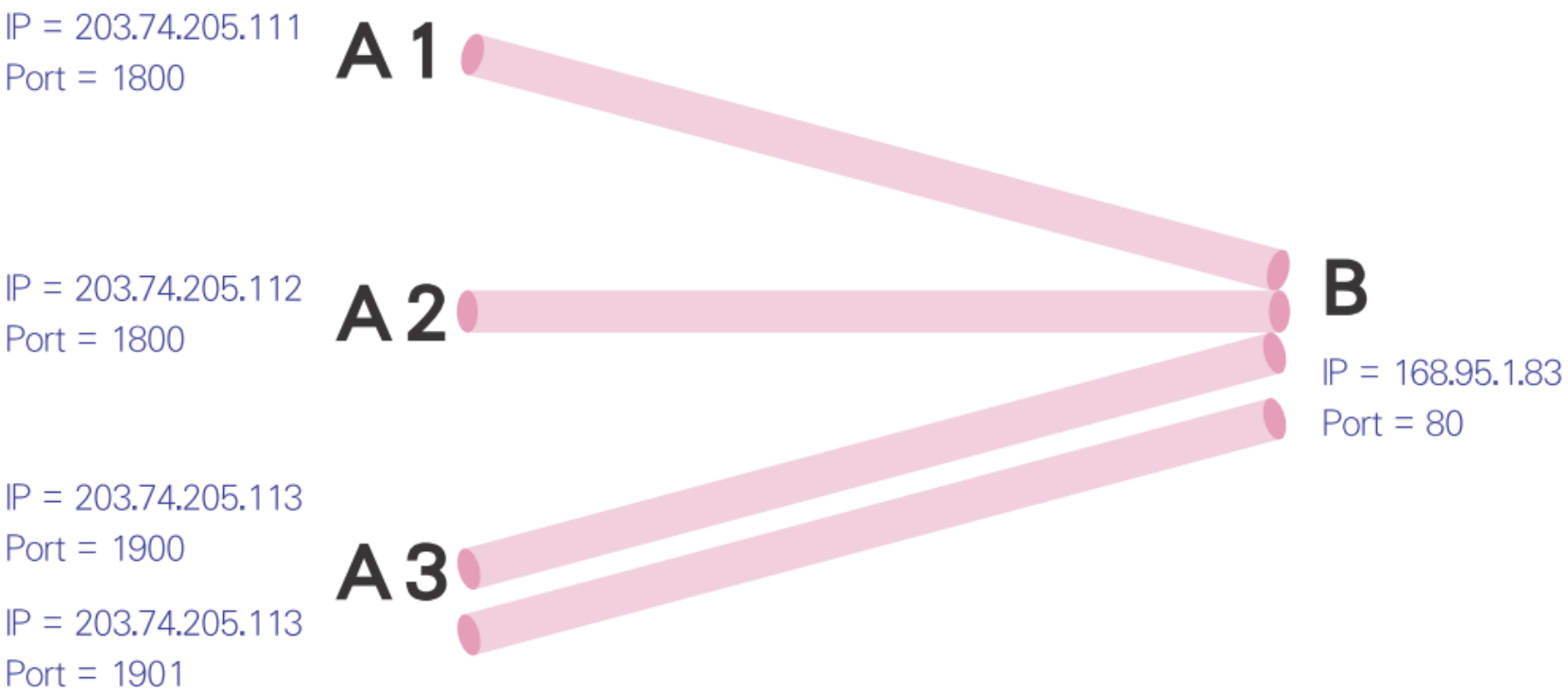


圖 11-20　伺服器可以和多個用戶端, 或同一用戶端的不同連接埠建立多條連線

4-2 建立連線

- 開始建立連線時, 一定會有一方為主動端 (Active), 另一方為被動端 (Passive)。
- 在建立連線時, 必須交換以下資訊：
  - 雙方的 ISN (初始序號)
  - 雙方的 Window Size

建立連線

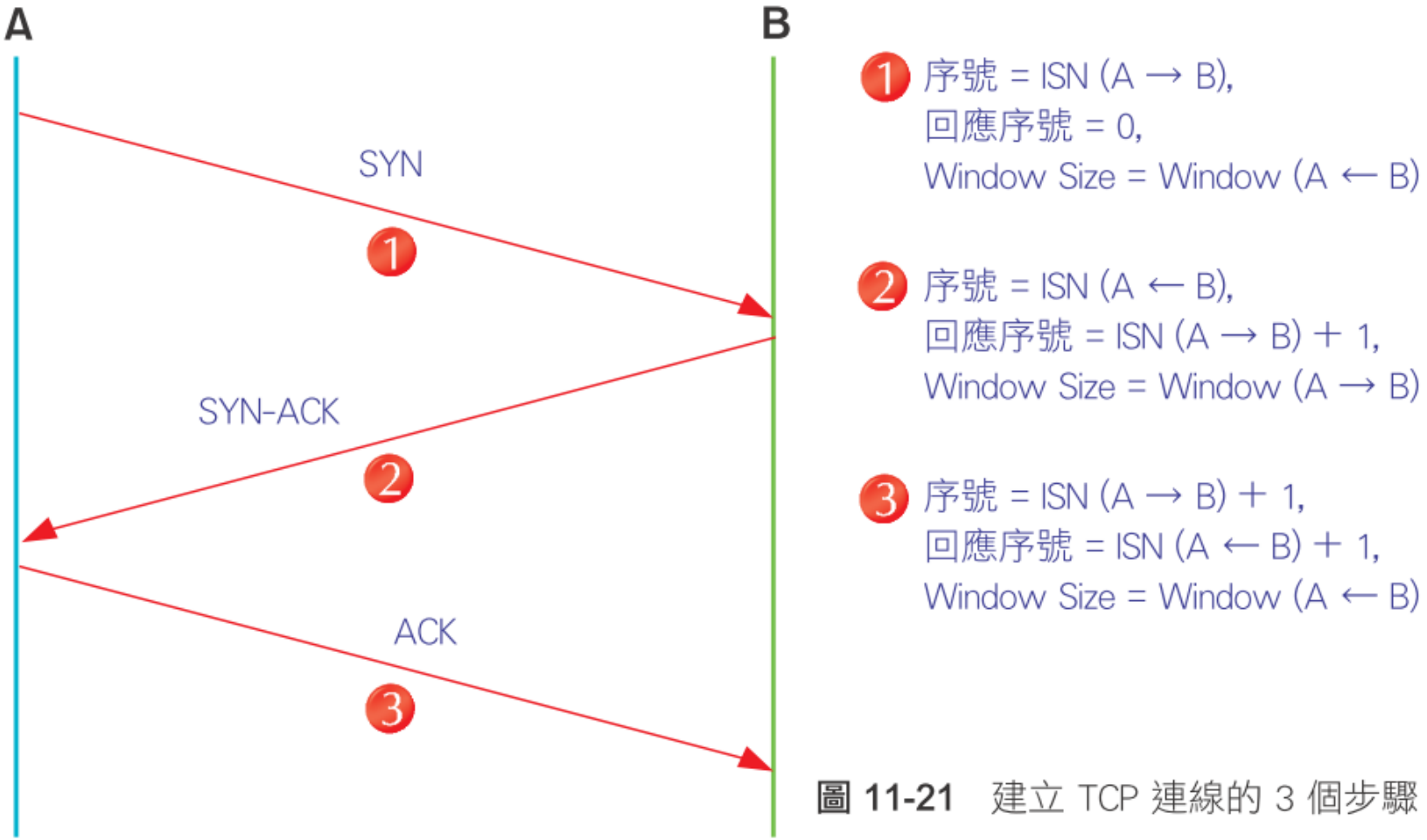


圖 11-21　建立 TCP 連線的 3 個步驟

建立連線

- 第 1 步驟
  - 序號
    指定 A → B 的 ISN
  - 回應序號
    指定 A ← B 的回應序號。
  - SYN Flag
    用來表示這是同步封包
  - Window Size

    代表A預設Receive Window的大小

建立連線

- 第 2 步驟
  - 序號
    代表 A ← B 的 ISN
  - 回應序號
    SYN-ACK 封包的回應序號等於 ISN (A → B) 再加上 1
  - SYN-ACK Flag
    表示此封包是回應先前收到的同步封包，所以它的序號就是 ISN (A ← B)。
  - Window Size

  代表 B 預設Receive Window的大小

建立連線

- 第 3 步驟
  - 序號

    也就是 SYN-ACK 封包的回應序號, 等於 ISN (A → B) + 1。
  - 回應序號

    此處的序號等於 ISN (A ← B) 再加上 1, 表示 A 期望 B 下次送來的資料, 是以 ISN (A ← B) + 1 為第 1 Byte 的編號。

建立連線

- 第 3 步驟
  - ACK Flag

    表示回應序號包含了確認收到的資訊。
  - Window Size

    A 的 Receive Window 大小, 亦即 Window (A ← B)。

4-3 中止連線

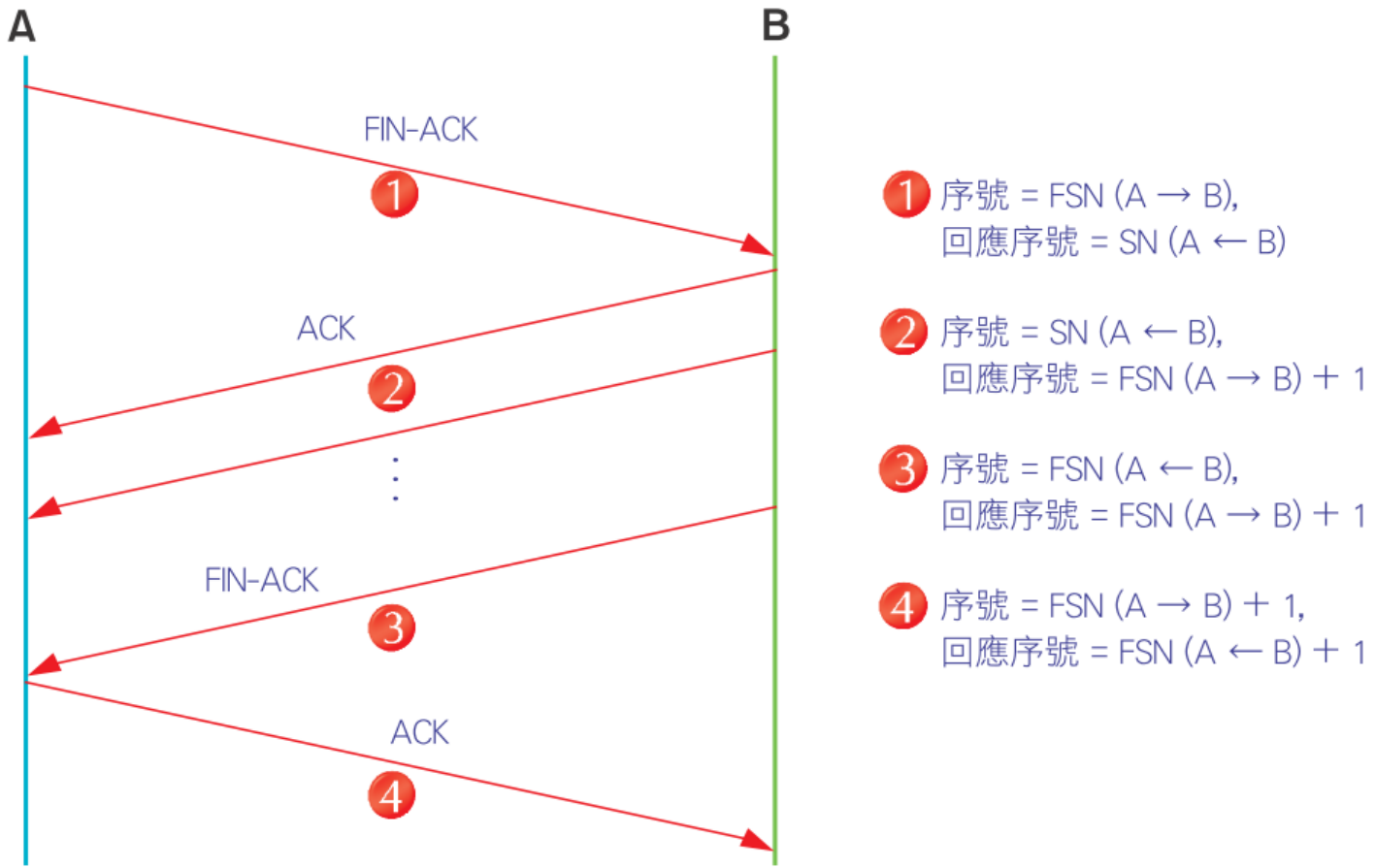


圖 11-22　結束 TCP 連線的 4 個步驟

4-3 中止連線

- 第 1 步驟
  - 序號
    指定 A → B 的序號
  - 回應序號
    指定 A ← B 的回應序號。

4-3 中止連線

- FIN-ACK Flag

  A → B 已經傳輸完畢, ACK 表示回應序號包含了確認收到的資訊。
- Window Size

  與一般相同。

中止連線

- 第 2 步驟
  - 序號

    指定 A ← B 的序號。
  - 回應序號

    此處回應序號等於第 1 步驟 FIN-ACK 封包的 FSN (A → B) 再加上 1。
  - ACK Flag

    表示回應序號包含了確認收到的資訊。

中止連線

- 第 3 步驟
  - 序號

    指定 A ← B 的序號
  - 回應序號

    指定 A → B 的回應序號。此處的回應序號與第 2 步驟的回應序號相同, 皆為 FSN (A → B) + 1。
  - FIN-ACK Flag

    代表 A ← B 已經傳輸完畢

中止連線

- 第 4 步驟
  - 序號

    指定 A → B 的序號, 此處序號等於第 1 步驟 FIN-ACK 封包的 FSN (A → B) 再加上 1。
  - 回應序號

    指定 A ← B 的回應序號, 此處回應序號等於第 3 步驟 FIN-ACK 封包的 FSN (A ← B) 再加上 1。
  - ACK Flag

    表示回應序號包含了確認收到的資訊。

本章完結