計算機網路概論

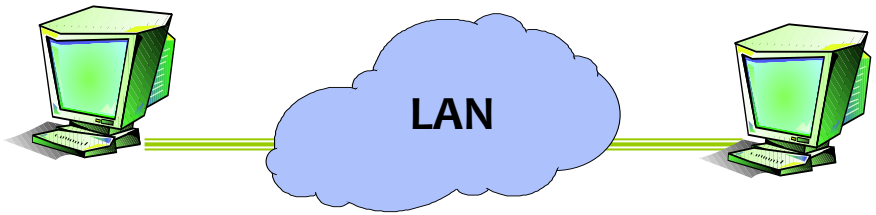
可靠傳輸機制

大綱

- **緒論**
- **停止和等候協議**
- **滑動視窗協議**
- 滑動視窗協議的問題

可靠傳輸機制

- 資料傳輸是經由通信鏈 (link) 傳輸，在傳送的過程中可能會因為訊號干擾而產生訊號錯誤
- 通常使用循環冗餘校驗 (CRC) 偵測錯誤
- 有些偵錯程式足夠強大更正錯誤的訊號，但常常需消耗龐大的資源
- 錯誤的訊框必須被丟棄
- 為了達到可靠的傳輸，我們必須重送這些被丟棄的訊框



可靠傳輸機制

- 我們可以使用以下兩種基礎方法的組合達到重送錯誤訊框的目的
  - 肯定回覆
  - 等待逾時
- 肯定回覆（縮寫 ACK）是一個小的控制訊框. 接收端用來通知傳送端其已收到稍早傳來的訊框。
  - 控制資框 是一個只有標頭沒有資料的訊框。
- 傳送端則透過接收到肯定回覆確認其訊框已傳送成功。

可靠傳輸機制

- 若傳送端在經過一段適當的等待時間還沒有收到肯定回覆，傳送端會重新傳送原本的訊框。
- **傳送端等待肯定回覆一段適當的時間的動作就叫等待逾時。**
- 一般使用肯定回覆 與 等待逾時 實作出的可靠傳輸方法, 有時又稱為 自動重覆請求 (ARQ, Automatic Repeat ReQuest)。

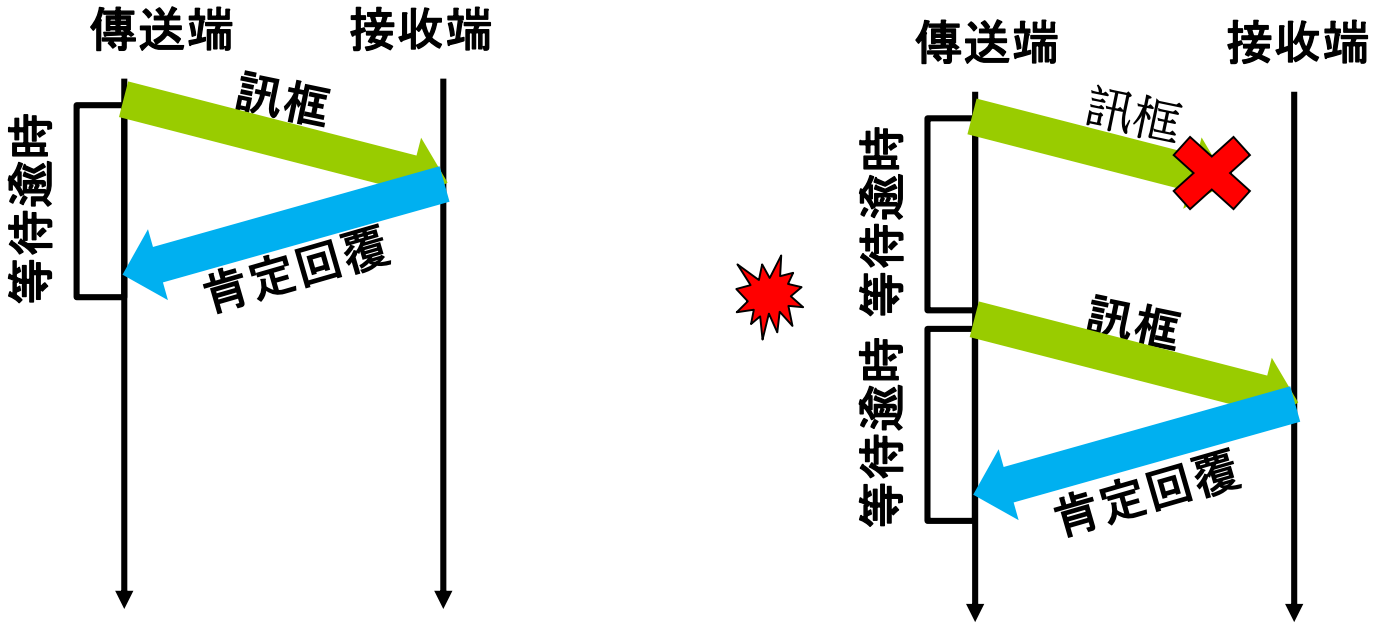
大綱

- 緒論
- 停止和等候協議
- 滑動視窗協議
- 滑動視窗協議的問題

停止和等候行協議

- 停止和等候 (Stop-and-Wait)協議的直觀概念
  - 傳送端在傳送完一個訊框後必須等收到肯定回覆後才能傳送下一個訊框
  - 當傳送端等待一段時間後仍未收到肯定回覆，傳送端發生等待逾時 (timeout)，就會重送原本的訊框

停止和等待協議

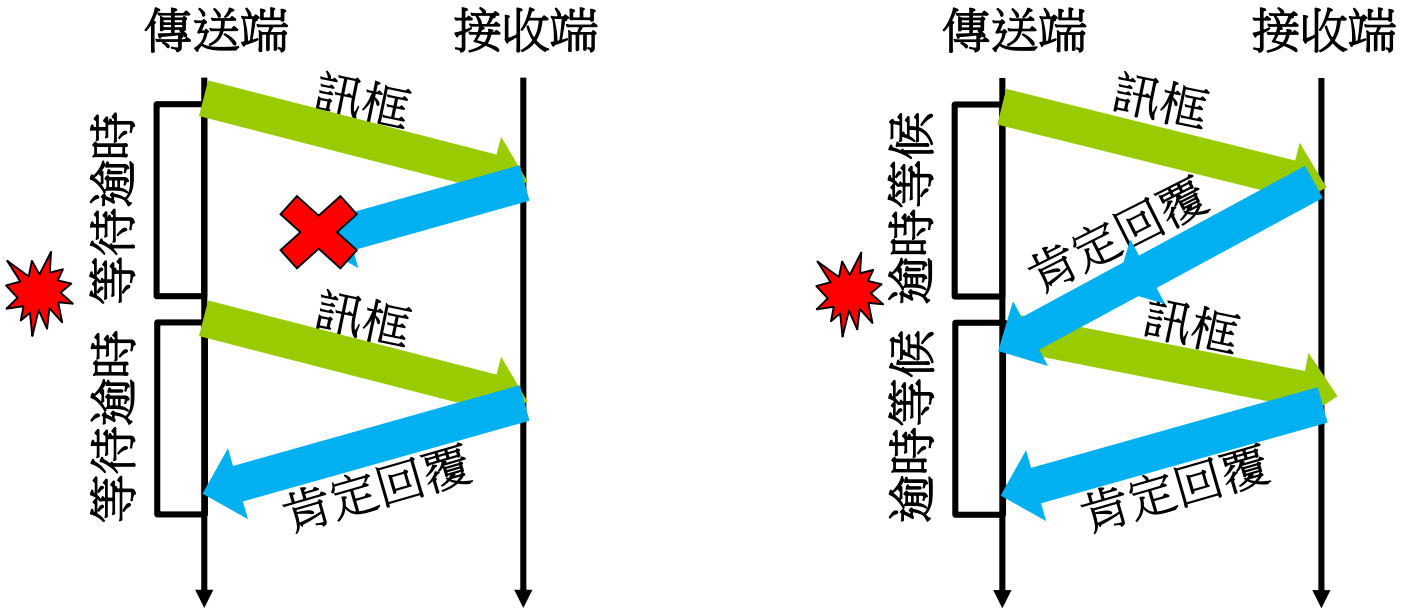


使用停止和等待演算法的四種不同情境

(a) 肯定回覆在等待逾時前送達;

(b) 原始的訊框遺失;

停止和等待協議



使用停止和等待演算法的四種不同情境.

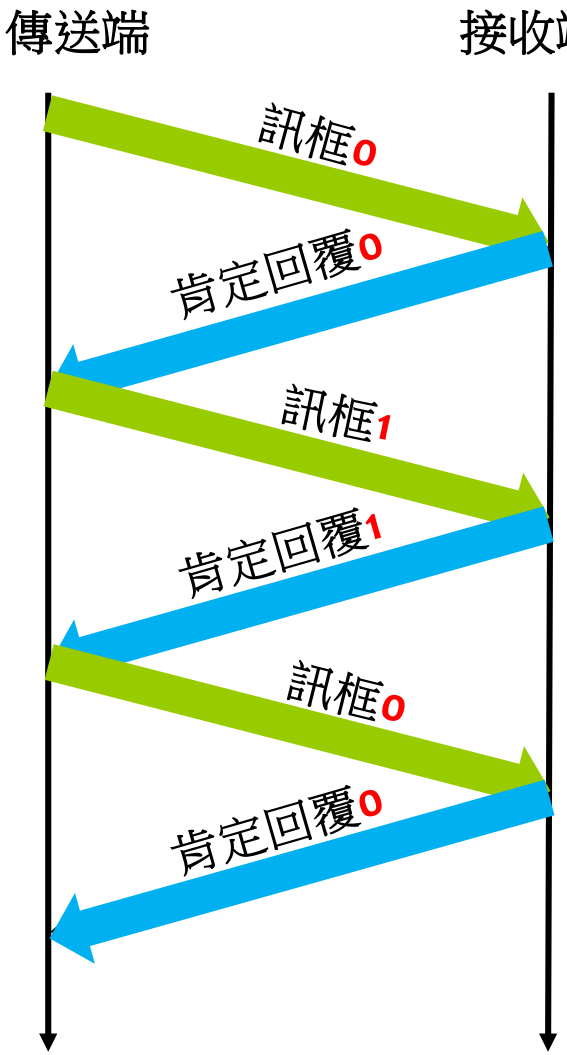
(c) 肯定回覆遺失;

(d) 等待逾時太快發生 (或者 肯定回覆延遲到達)

停止和等待協議

- 若肯定回覆在到達時遺失或延遲了
  - 傳送端等待逾時且重新傳送原本的訊框。
  - 因此，重複的訊框會被傳送
- 如何解決？
  - 使用一位元的序號 (sequence number, 0 or 1)
  - 當傳送端重送 “編號0訊框”，接收端可知此為重傳的編號0訊框, 因此可以忽略它 (接收端依然會給編號0訊框肯定回覆，避免第一份的肯定回覆已遺失所造成的問題)

停止與等待協議



使用一位元序號的停止和等待協議 時間軸

停止與等待協議

- 傳送端在同一個時間只有一個訊框在鏈結上 -- 遠小於鏈結的頻寬大小。
- 舉例來說, 有一個 2 Mbps 頻寬與 40 ms RTT 的鏈結
  - 此鏈結的 延遲 x頻寬 乘積為 80 Kb 或 10 KB
  - 由於使用停止與等待協議, 傳送端在一個RTT時間內只能傳送一個訊框
  - 假設一個訊框大小為 1 KB (1 Kbyte)
  - 則此鏈結的最大吞吐量 (M) 為

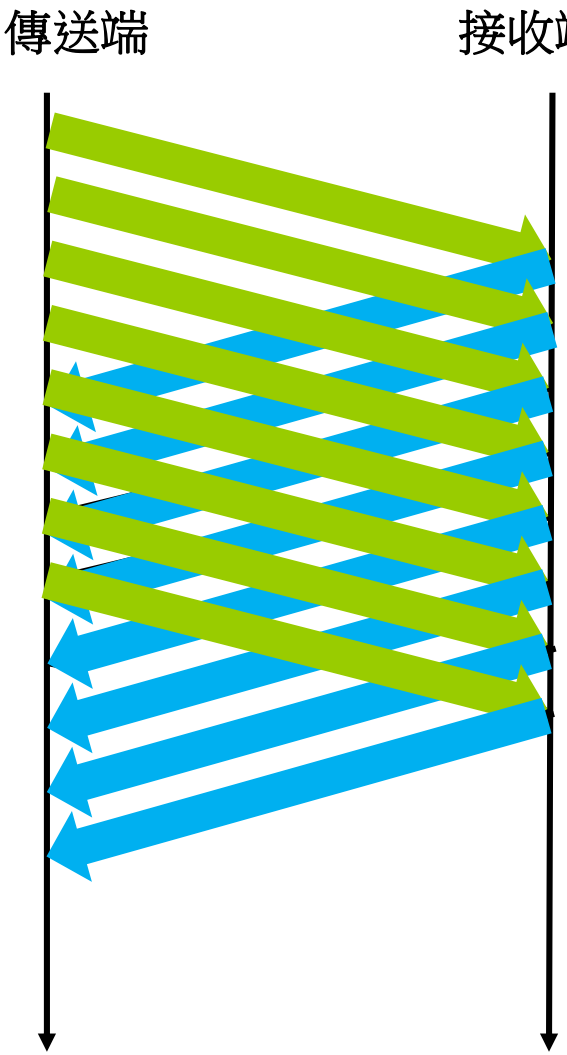
    一個訊框的位元數/ RTT 時間 = 8kb /40ms = 200 Kbps

    或 約為鏈結頻寬 (2Mbps) 的 1/10 (頻寬使用效率不佳)
  - 為了充分使用鏈結頻寬, 在必須停下來等待第一個肯定回覆前, 傳送端需傳送10個訊框。

大綱

- 緒論
- 停止和等候協議
- **滑動視窗協議**
- 滑動視窗協議的問題

滑動視窗協議



滑動視窗協議的時間軸

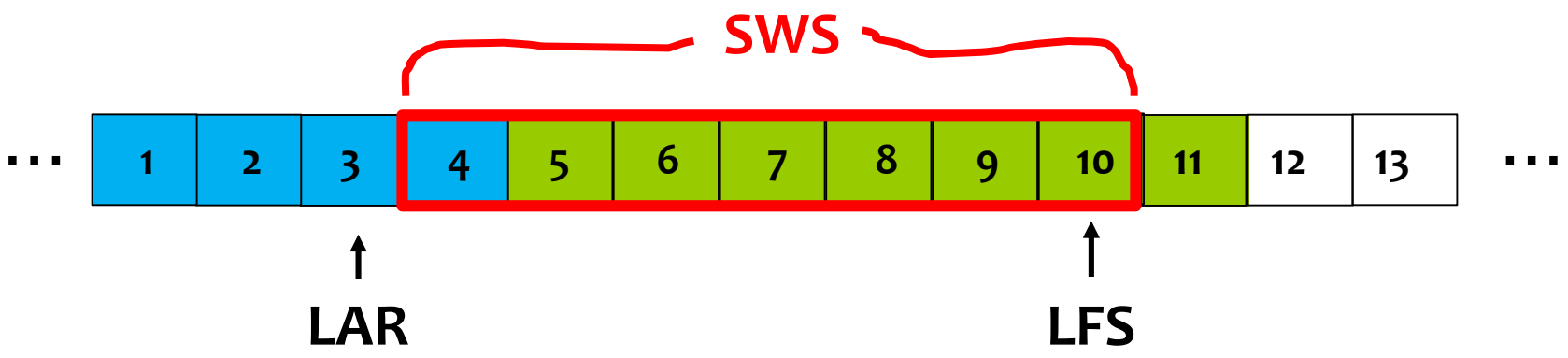
滑動視窗協議

- 傳送端給予每個訊框一個序號 (SeqNum)，假設序號值可成長到無限大。
- 傳送端持續記錄著三個變數
  - 傳送視窗大小 (**SWS**)
    - 傳送端可傳送不同訊框(未收到肯定回覆的訊框)數量的上限
  - 最後收到肯定回覆 (**LAR**)
    - 最後收到的肯定回覆序號
  - 最後傳送的訊框 (**LFS**)
    - 最後送出的訊框序號

滑動視窗協議

- 傳送端也會維持下列的不變條件

  LFS – LAR ≤ SWS



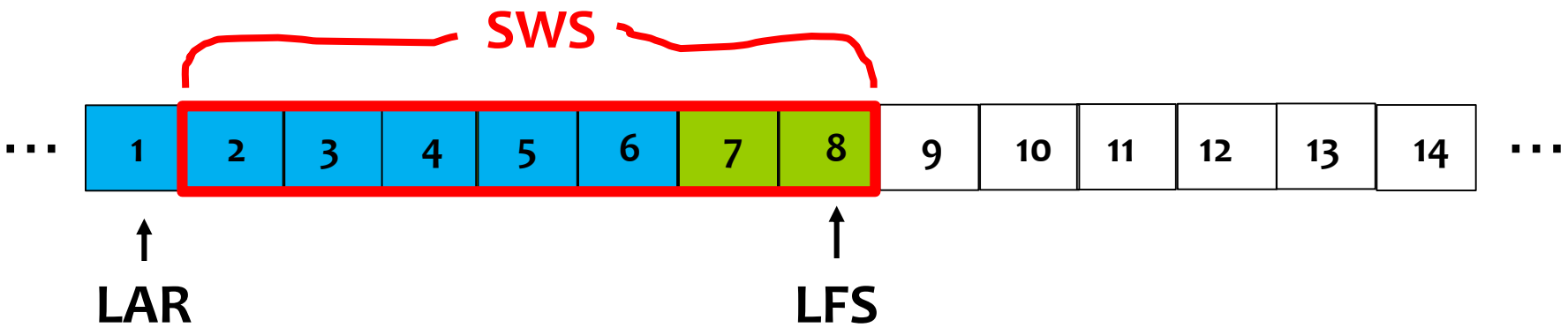
傳送端滑動視窗

滑動視窗協議

- 當傳送端接收到一個肯定回覆
  - 將 LAR 向右移使傳送端可以再傳送另一個訊框
- 傳送端對每個傳送出的訊框會有一個計時器
  - 若計時器在肯定回覆到達前逾時, 則重傳該訊框
- 傳送端需要一個有 SWS 大小的緩衝區放置已傳送出去的訊框，才能在需要重傳訊框時找到訊框

滑動視窗協議

- 什麼時候傳送端可以移動他的視窗?
  - 當收到編號為 LAR+1 的肯定回覆



傳送端的滑動視窗

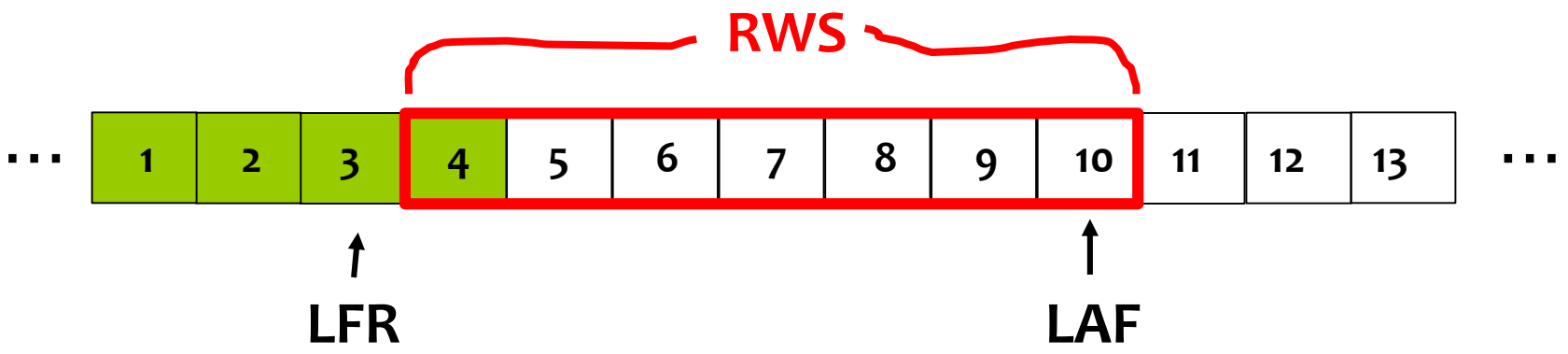
滑動視窗協議

- 接收端持續更新三個變數
  - 接收視窗大小 (RWS)
    - 接收端同時可接收未按照序號到達的訊框數量上限
  - 最大可接受訊框 (LAF)
    - 最大可以接收的訊框序號
  - 最後收到訊框 (LFR)
    - 最後收到的訊框序號

滑動視窗協議

- 接收端會維持下列的條件

$$LAF - LFR \leq RWS$$



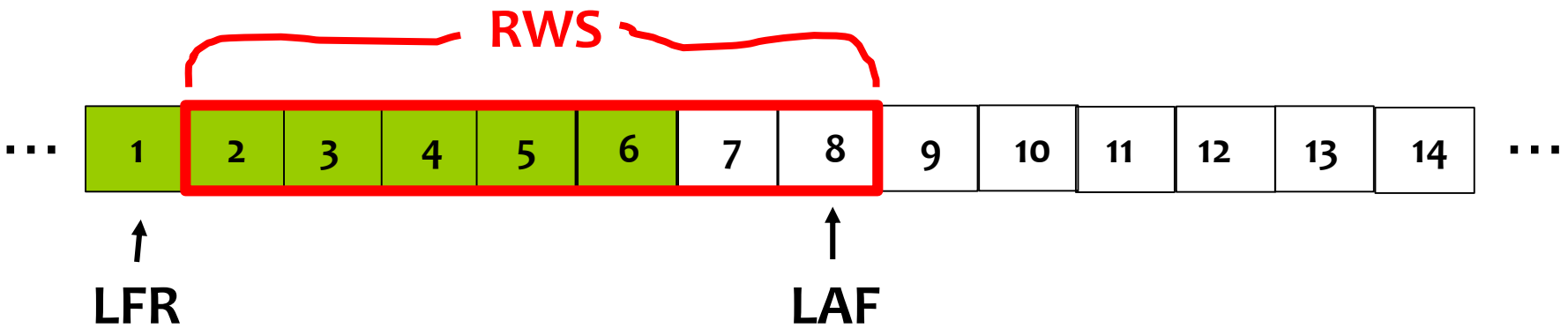
接收端的滑動視窗

滑動視窗協議

- 當序號為 SeqNum 的訊框到達時，接收端會如何處理？
  - 若 SeqNum ≤ LFR 或 SeqNum > LAF
    - 將此訊框丟棄 (此訊框在接收端滑動視窗外)
  - 若 LFR < SeqNum ≤ LAF
    - 接收此訊框

滑動視窗協議

- 什麼時候接收端會滑動他的視窗?
  - 當收到序號 LFR+1 的訊框



接收端滑動視窗

滑動視窗協議

- 舉例來說， 假設 LFR = 1 ， RWS = 7

  → LAF = 8

- 若訊框 4,6,3, 5 先後到達, 雖然他們沒有依照數字順序, 但他們都會先被收起來，因為他們都在接收端視窗內
- 但是沒有肯定回覆會被傳出去因為 訊框2 還沒有抵達
- 最後,訊框2抵達 (重新傳送 或 延遲後到達)
- 接收端送出訊框6肯定回覆

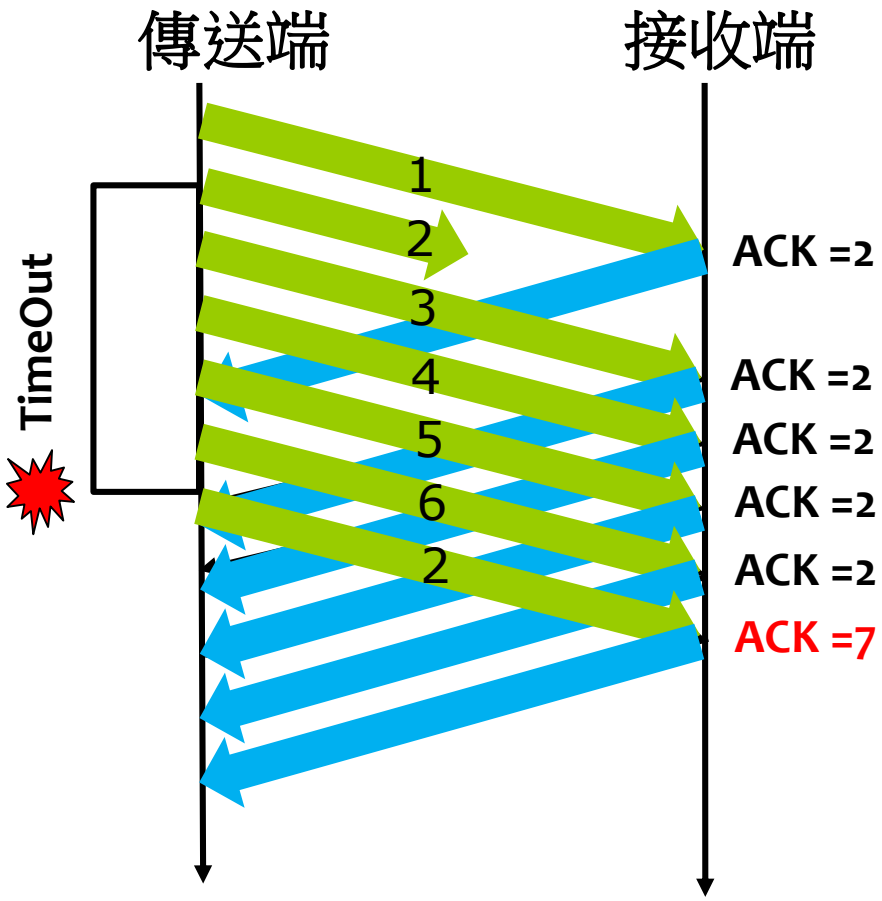
  將 LFR 改為 6

  和 LAF 改為 13 (視窗滑動)

滑動視窗協議

- 累積式肯定回覆 (Accumulative ACKs)
  - **SeqNumToAck** 表示未收到肯定回覆的最大序號，因此所有編號小於 **SeqNumToAck** 的訊框都已經收到。
  - 即使收到較高序號的訊框，接收端還是會送出序號為 **SeqNumToAck** 的肯定回覆
    - 這種稱為累積式的肯定回覆
  - 接收端會設定值
    - **LFR = SeqNumToAck -1** 與
    - **LAF = LFR + RWS** (視窗滑動)

滑動視窗協議



累積式肯定回覆範例 (SeqNumToAck = 2)

大綱

- 緒論
- 停止和等候協議
- 滑動視窗協議
- 滑動視窗協議的問題

滑動視窗協議的問題

- 當等待逾時發生傳送資料的量就會減少
  - 由於傳送端無法使它的視窗往前滑動
- 當封包遺失發生，這個架構將無法再使管線保持滿載狀態
  - 需要偵測出封包遺失的時間越長，伺服器所遇到的問題就會越多。
- 如何改進
  - 否定回覆 (NAK)
  - 附加肯定回覆 (Additional Acknowledgement)
  - 選擇性肯定回覆 (Selective Acknowledgement)

滑動視窗協議的問題

- 否定回覆(NAK)
  - 當接收端收到訊框3會送出訊框2的否定回覆 (以累積式肯定回覆為例)
    - 然而這是沒有必要的，因為傳送端的等待逾時機制已經可以知道上述情境
- 附加肯定回覆
  - 當收到訊框3時接收端多傳送一個訊框2的肯定回覆
    - 傳送端使用多餘的肯定回覆作為訊框遺失的線索
- 選擇性肯定回覆
  - 接收端每收到一個訊框就會發出該訊框的肯定回覆而不是序號最高的肯定回覆
    - 接收端會分別發出訊框 3,4,5,6 的肯定回覆
    - 傳送端知道訊框2 遺失
    - 傳送端可以保持管線滿載 (更高的複雜度)

滑動視窗協議的問題

- 如何選擇視窗大小？
  - SWS 很容易能夠計算
    - Delay(延遲) × Bandwidth(頻寬)
  - RWS 有較多的設定方式
    - 兩個基本的設定
      - RWS = 1

        接收端沒有緩衝區提供給順序錯誤的訊框
      - RWS = SWS

        接收端可以儲存傳送端送來的訊框
  - 如果讓 RWS > SWS 合理嗎？

滑動視窗協議的問題

- 有限的序號
  - 訊框序號會被放在標頭欄位
    - 有限大小
      - 3 位元: 可表示八個可能的序號: 0, 1, 2, 3, 4, 5, 6, 7
    - 這些序號必須循環使用

滑動視窗協議的問題

- 如何判別擁有相同序號的不同訊框？
  - 可使用的序號數量必須大於被允許同時在網路上傳送的訊框數量 (outstanding frames)
    - 停止和等待協議：只有一個訊框可以同時在網路上傳送
      - 2 個不同的序號 (0 和 1)
    - **MaxSeqNum** 表示可以使用的序號上限值
    - **SWS + 1 ≤ MaxSeqNum**
      - 這樣子會有問題嗎?
      - 決定於 **RWS**
      - 若 **RWS = 1,** 則正常
      - 若 **RWS = SWS,** 則會出現問題

滑動視窗協議的問題

舉例來說,有8個序號 0, 1, 2, 3, 4, 5, 6, 7

RWS = SWS = 7 = 8-1

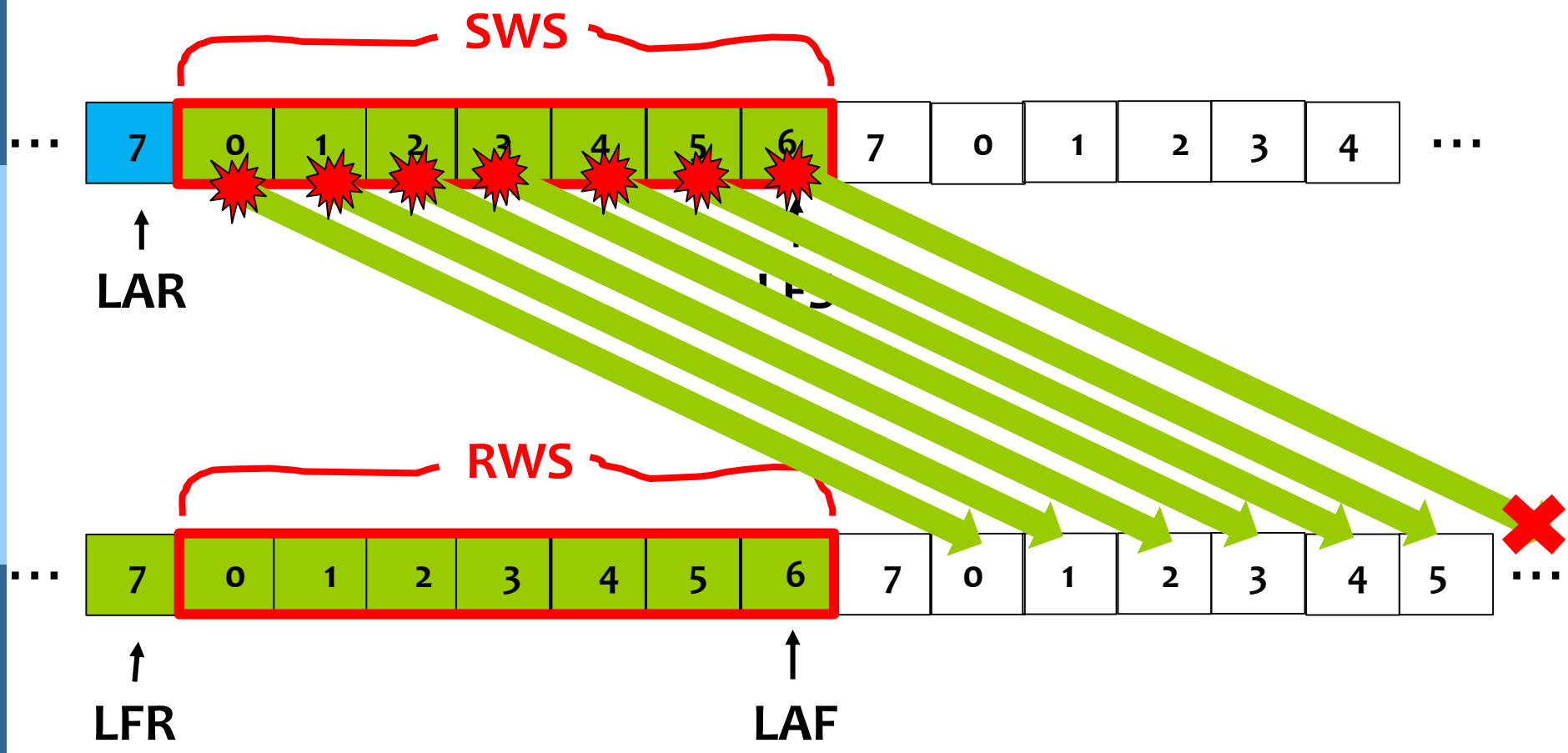
1. 傳送端送出訊框 0, 1, … , 6
2. 接收端收到訊框 0, 1, … ,6
3. 接收端傳送肯定回覆 0, 1, … , 6

   但肯定回覆 (0, 1, … , 6) 遺失了
4. 傳送端發生等待逾時且重送 0, 1, …,5, 6
5. 接收端期待收到7, 0, …., 5

➔ 訊框0-5 會被接收端接受 (但這些是重複的訊框 !!)

訊框 6 會被丟棄 (但這是正確的訊框)

滑動視窗協議的問題



問題發生在 SWS + 1 ≤ MaxSeqNum 與 SWS = RWS

MaxSeqNum = 8, SWS = RWS = 7 的時候

滑動視窗協議的問題

為了防止此問題,

若 **RWS = SWS**

SWS < (MaxSeqNum + 1)/2

滑動視窗協議的問題

舉例來說, 有8個序號

0, 1, 2, 3, 4, 5, 6, 7

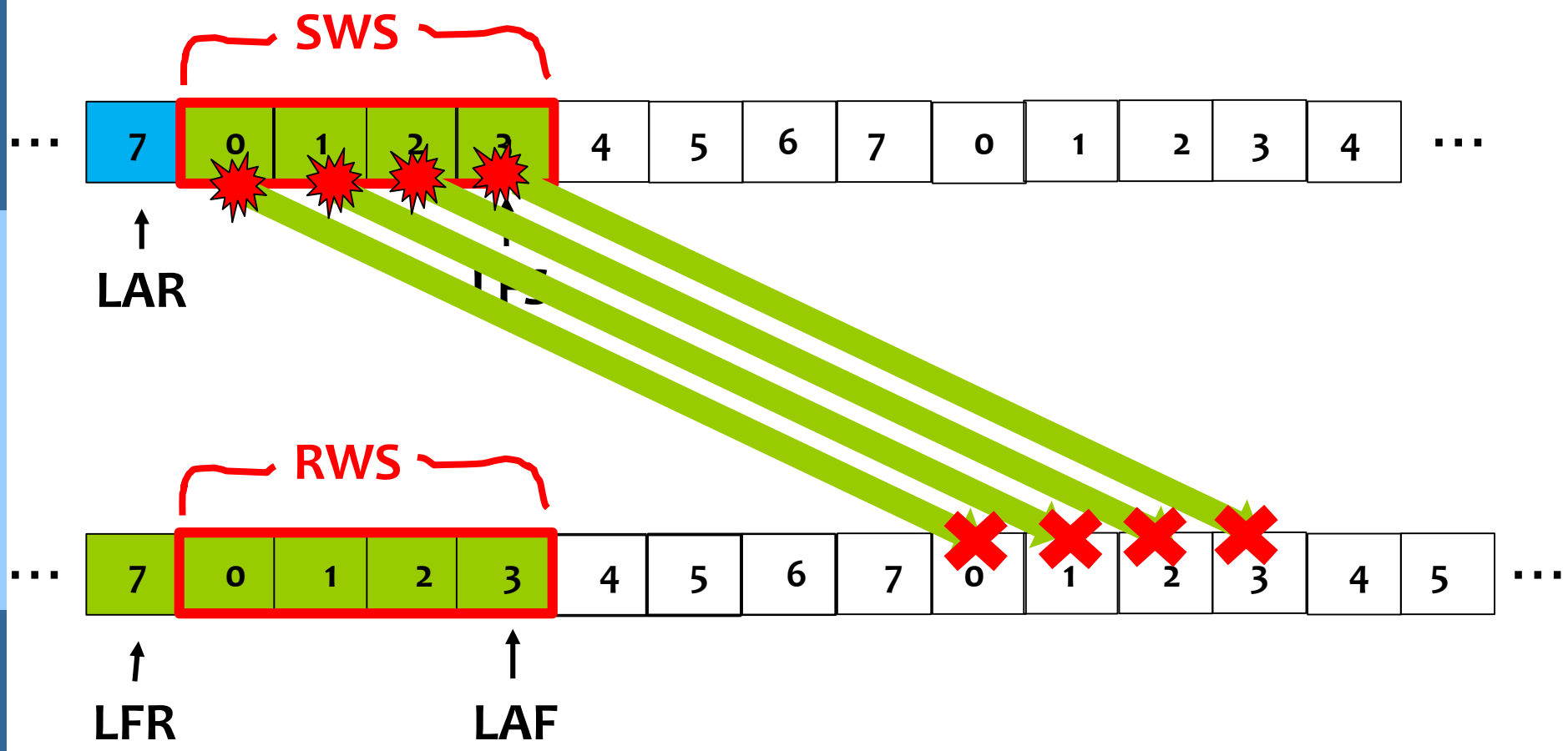
RWS = SWS = 4 < (8+1)/2 = 4.5

1. 傳送端送出訊框 0, 1, 2, 3
2. 接收端收到訊框0, 1, 2, 3
3. 接收端肯定回覆 0, 1, 2, 3

   但肯定回覆 (0, 1, 2, 3) 遺失了
4. 傳送端發生等待逾時且重送 0, 1, 2, 3
5. 傳送端期待收到 4,5,6,7

➔ 所有 0-3 訊框將會被丟棄 (這就正確了 !!)

滑動視窗協議的問題



範例 SWS < (MaxSeqNum+1)/2

MaxSeqNum = 8, SWS = RWS = 4 < (8+1)/2

滑動視窗協議的問題

- 滑動視窗協議提供下列三個特性
  - 可靠的傳輸
  - 維持訊框順序
    - 每個訊框都有序號
    - 順序錯誤的訊框會被置於緩衝區
  - 流量控制
    - 接收端可以藉由設定RWS的值調節傳送端的速度
    - 防止傳送端傳送過多訊框將接收端塞爆

總結

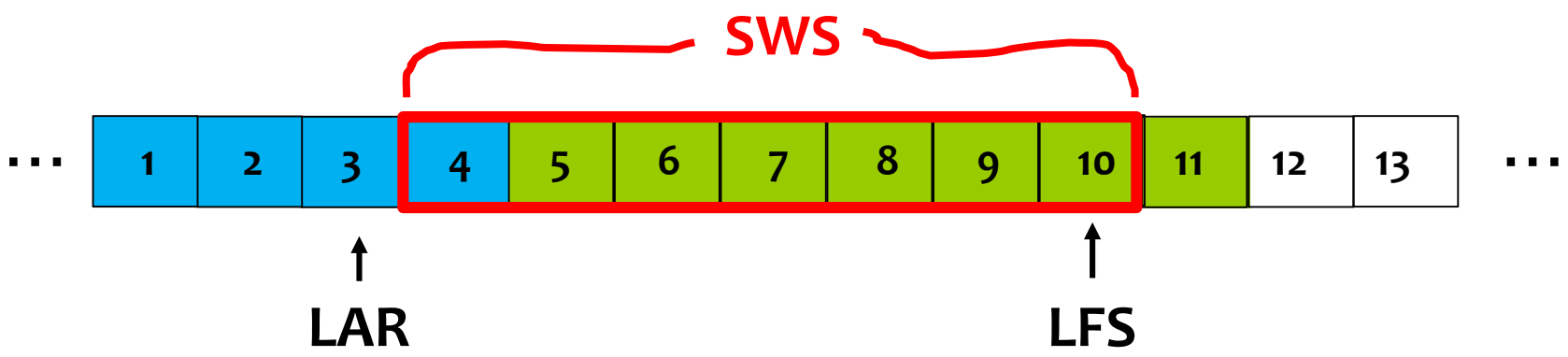
- 可靠傳輸是在通訊鏈上使用兩個基礎技術
  - 肯定回覆
  - 等待逾時
- 停止和等待協議是一個可靠的協議, 但是效能還是不夠好
  - 同時只有一個訊框可以在網路上傳送
  - 接收端可能收到重複的訊框

總結

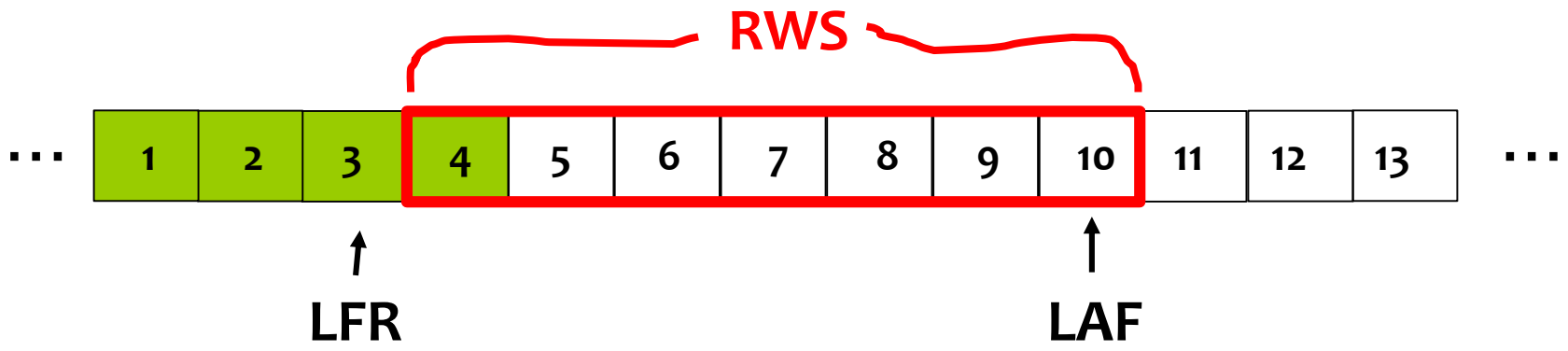
- 滑動視窗協議是一個可靠且高效能的協議
  - 序號會被附加在所有訊框上
  - 同時可以多個訊框在網路上傳送(使得管線滿載)
  - 傳送端
    - 傳送端視窗大小 (SWS) = 延遲 x 頻寬
    - 最後收到肯定回覆 (LAR)
    - 最後送出訊框 (LFS)
  - 接收端
    - 收到視窗大小 (RWS) = 1 or SWS
    - 最大可接受資訊窗號碼 (LAF)
    - 最後收到訊框 (LFR)

總結

- 傳送端維持條件: **LFS – LAR ≤ SWS**



- 接收端維持條件: **LAF – LFR ≤ RWS**



總結

- SWS 和 RWS 的值可能會不同
- 當RWS = SWS, 最好 SWS < (MaxSeqNum + 1)/2 , 否則接收端可能還是會收到重複的訊框
- 回覆 (ACK) 機制是一個選項
  - 否定回覆 (NAK)
  - 累積式肯定回覆 (Accumulative ACKs)
- 滑動視窗協議提供三個特性
  - 可靠的傳輸
  - 維持訊框順序
  - 流量控制 (接收端決定 RWS)