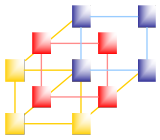
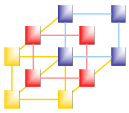
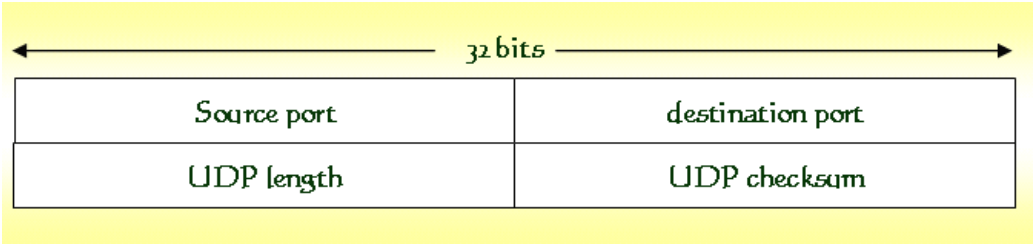


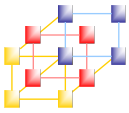
Unit 6
User Datagram Protocol (UDP)



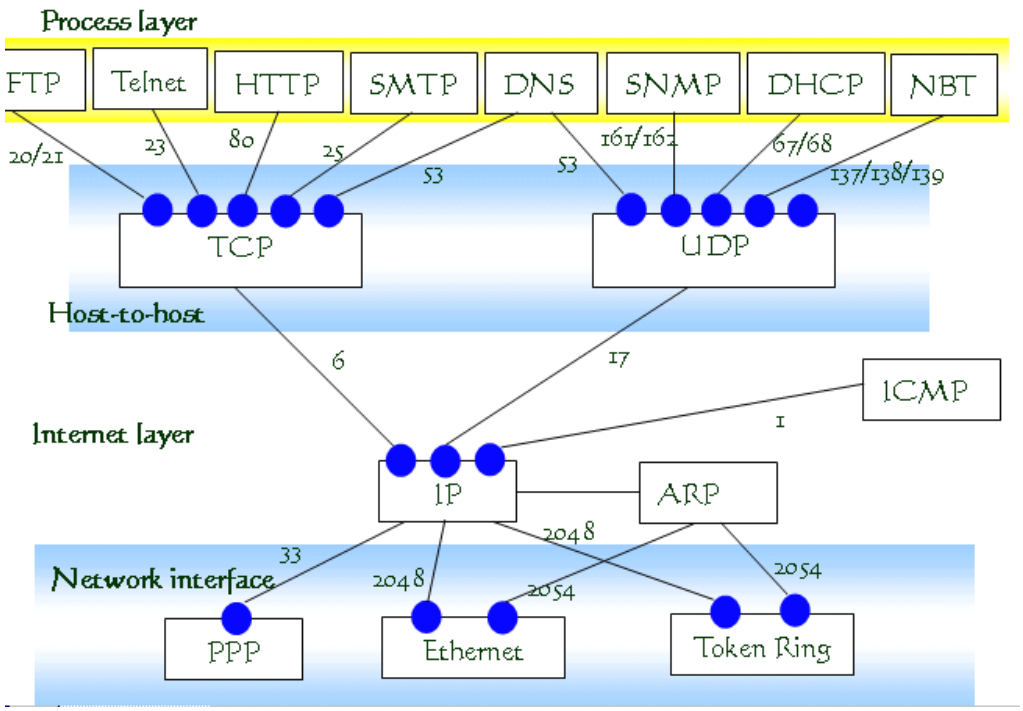
UDP

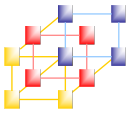
- UDP (user datagram protocol) 定義於 RFC 768，它讓應用程式可以在不需要建立連線 (connection) 的情況下送出封包
- 傳送的片段 (segments) 包括 8 bytes 的標頭 (UDP header) 與隨後的 payload
- UDP header 格式





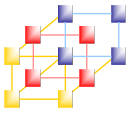
通訊埠(port)





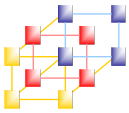
以UDP為基礎的網際網路協定

- BootP (Boot Protocol)
- DHCP (dynamic host configuration protocol)
- SNMP (simple network management protocol)
- TFTP (trivial file transfer protocol)
- DNS (domain name system)
- NTP (Network time protocol)



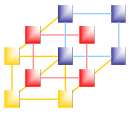
不同應用程式使用的 Port Number

Port Num / Proto	協定	說明
67/UDP	BOOTP	用於動態主機設定協定
68/UDP	BOOTP	用於動態主機設定協定
53/TCP,UDP	DNS	域名服務系統
20/TCP,UDP	FTP	檔案傳輸協定 - 預設資料埠
21/TCP,UDP	FTP	檔案傳輸協定 - 控制埠
22/TCP,UDP	SSH	安全遠端登入協定，用於安全檔案傳輸及埠轉發
80/TCP,UDP	HTTP	超文字傳輸協定或快速UDP網路連接- 用於傳輸網頁
443/TCP	HTTPS	超文字傳輸安全協定
137/TCP,UDP	NETBIOS	NetBIOS NetBIOS 名稱服務
138/TCP,UDP	NETBIOS	NetBIOS NetBIOS 資料報服務
139/TCP,UDP	NETBIOS	NetBIOS NetBIOS 對談服務
25/TCP,UDP	SMTP	簡單郵件傳輸協定- 用於傳遞電子郵件
110/TCP	POP	郵局協定，第3版 - 用於接收電子郵件
161/TCP,UDP	SNMP	簡單網路管理協定
162/TCP,UDP	SNMP	SNMP協定的TRAP操作
69/UDP	TFTP	小型檔案傳輸協定
123/UDP	NTP	用於時間同步



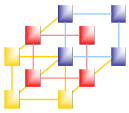
Client 與 Server 的定義

- 一般我們利用連線的方向來定義 Client-Server 連線的角色
  - Client 一般為連線的發起端
    - 通常為要求資源的一方
  - Server 一般為連線的接收端
    - 通常為提供資源的一方
    - Server 執行後會等待 Client 端送來的要求 (Request)
    - Server 在收到 Client 的要求後回執行必要的計算，如果需要，也會將結果回傳給 Client

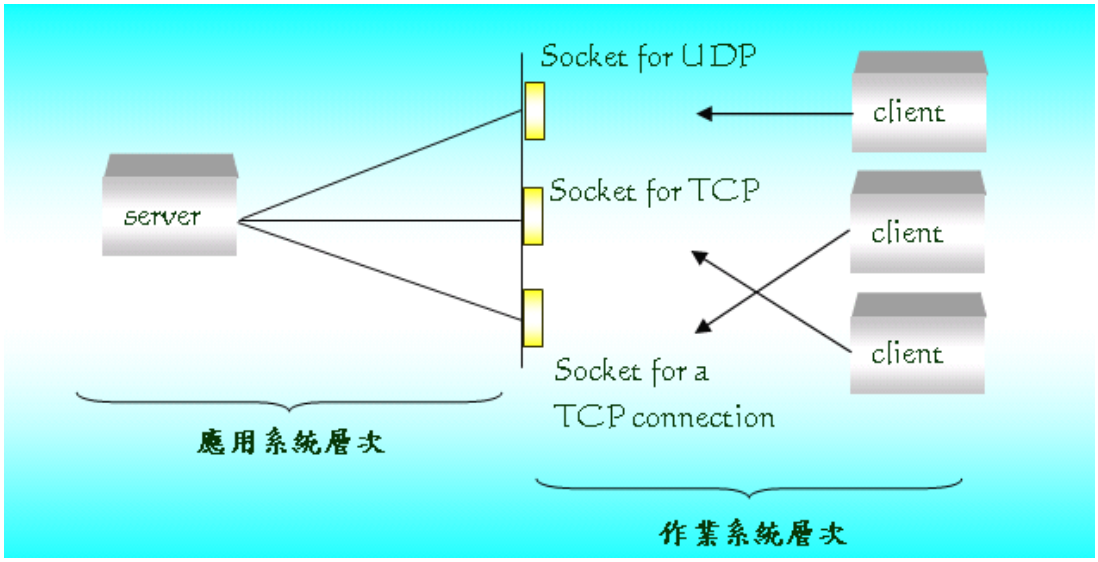


通訊軟體設計的觀點

- 多重協定的伺服程式(multiprotocol servers)
- 多重服務的伺服程式(multiservice servers)

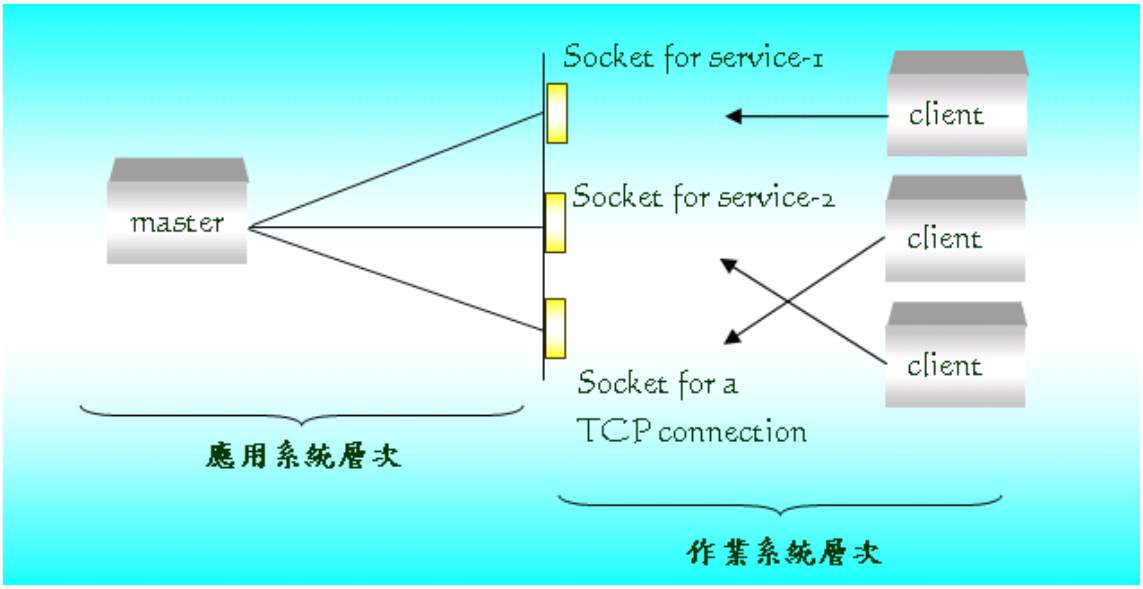


Multiprotocol Server的架構



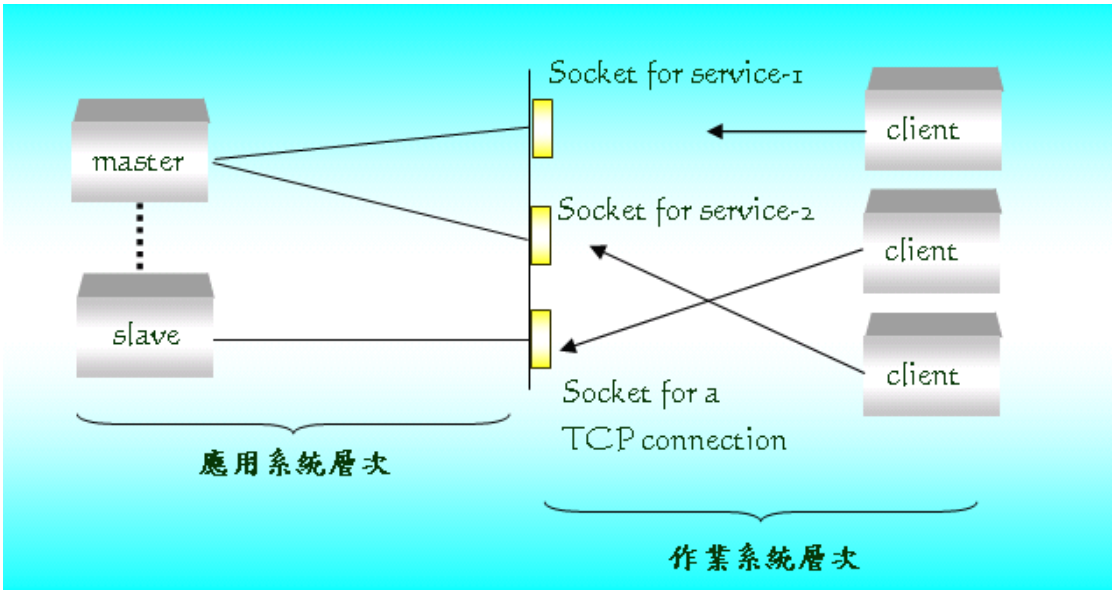
一個 Server 程式同時處理多個同訊協定

Connection-oriented multiservice server的架構

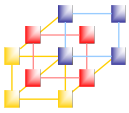


一個程式同時提供多個 Server 服務

Concurrent, connection-oriented multiservice server的架構

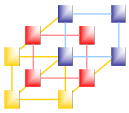


Master 接受 client 的請求後產生一個 thread 來處理client 的請求



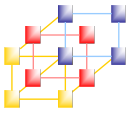
Datagram Sockets Functions

- `socket.recvfrom(`*`bufsize`*`[, `*`flags`*`])`
  - Receive data from the socket
  - The return value is a pair (bytes, address)
    - *bytes* is a bytes object representing the data received
    - *address* is the address of the socket sending the data.
- `socket.sendto(`*`bytes`*`, `*`address`*`)`
  - Send data to the socket.
  - The socket should not be connected to a remote socket, since the destination socket is specified by *address*



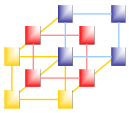
RFC 1035
Network Time Protocol (V3)

- Defined in RFC 1305
- NTP was designed by David L. Mills of the University of Delaware.
- Based on UDP, port number 123
  - Simple connectionless service
  - No guarantee of delivery
  - No detection of lost packets
  - Can also use broadcasting or multicasting
- Message authentication is optional
  - Authenticator added to message
- All versions of NTP can use IPv4
  - New version (v4) supports IPv6



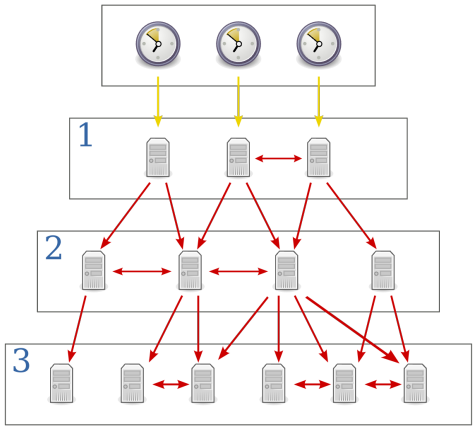
NTP Timestamp

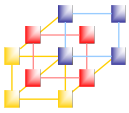
- Reference scale is UTC (Coordinated Universal Time)
  - 最主要的世界時間標準，其以原子時秒長為基礎，在時刻上儘量接近於格林威治標準時間 (GMT)
- Time parameters are 64 bits long
  - A 32-bit part for seconds
  - A 32-bit part for fractional second
  - A time of 0.0 is usually treated as an error
  - NTP uses an epoch of January 1, 1900
    - The first rollover occurs on February 7, 2036
- Advance notice of leap second (潤秒)
  - Leap second at end of today when set
  - Positive leap second in progress
  - Transmit 23:59:59 twice



Clock Strata

- NTP is a hierarchical, semi-layered system of time sources
  - Each level of this hierarchy is termed a *stratum* and is assigned a number starting with zero for the reference clock at the top.
  - The upper limit for stratum is 15; stratum 16 is used to indicate that a device is unsynchronized.





Clock Synchronization Algorithm

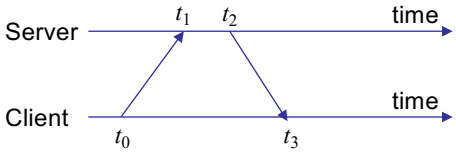
- Time offset θ
  - The difference in absolute time between the two clocks
- The round-trip delay δ
  - $\delta = (t_3 - t_0) - (t_2 - t_1)$

$$t_0 + \theta + \delta/2 = t_1 \qquad (1)$$

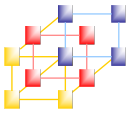
$$t_2 - \theta + \delta/2 = t_3 \qquad (2)$$

- (1) – (2)

$$\theta = \frac{(t_1 - t_0) - (t_3 - t_2)}{2}$$

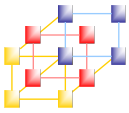


3-NTPClient.py



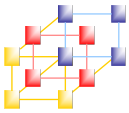
NTP Message Format (1/6)

	0	2	5	8	16	24	31 (bit)
0	LI	VN	Mode	Stratum	Poll	Precision	
4	Root Delay						
8	Root Dispersion						
12	Reference Identifier						
16	Reference Timestamp (64)						
24	Origin Timestamp (64)						
32	Receive Timestamp (64)						
40	Transmit Timestamp (64)						
48	Optional						



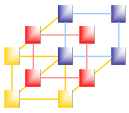
NTP Message Format (2/6)

- LI: Leap Indicator (2 bits)
  - This field indicates whether the last minute of the current day is to have a leap second applied
  - The field values follow
    - 0: No leap second adjustment
    - 1: Last minute of the day has 61 seconds
    - 2: Last minute of the day has 59 seconds
    - 3: Clock is unsynchronized
- VN: NTP Version Number (3 bits)



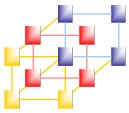
NTP Message Format (3/6)

- Mode: NTP packet mode (3 bits)
  - The values of the Mode field follow:
    - 0: Reserved
    - 1: Symmetric active
    - 2: Symmetric passive
    - 3: Client
    - 4: Server
    - 5: Broadcast
    - 6: NTP control message
    - 7: Reserved for private use
- Stratum: Stratum level of the time source (8 bits)
  - 0: Unspecified or invalid
  - 1: Primary server
  - 2 - 15: Secondary server
  - 16: Unsynchronized



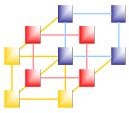
NTP Message Format (4/6)

- Poll
  - Poll interval (8-bit signed integer) 2 value of the maximum interval between successive NTP messages, in seconds.
- Precision
  - Clock precision (8-bit signed integer)
  - The precision of the system clock, in $\log_2$ seconds.



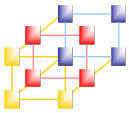
NTP Message Format (5/6)

- Root Delay
  - The total round-trip delay from the server to the primary reference sourced. The value is a 32-bit signed fixed-point number in units of seconds, with the fraction point between bits 15 and 16. This field is significant only in server messages.
- Root Dispersion
  - The maximum error due to clock frequency tolerance. The value is a 32-bit signed fixedpoint number in units of seconds, with the fraction point between bits 15 and 16. This field is significant only in server messages.
- Reference Identifier
  - For stratum 1 servers: a four-character ASCII code that describes the external reference source (refer to Figure 2)
  - For secondary servers : the 32-bit IPv4 address of the synchronization source



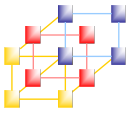
NTP Message Format (6/6)

- Reference Timestamp
  - This field is the time the system clock was last set or corrected, in 64-bit time-stamp format.
- Originate Timestamp
  - This value is the time at which the request departed the client for the server, in 64-bit time-stamp format.
- Receive Timestamp
  - This value is the time at which the client request arrived at the server in 64-bit timestamp format.
- Transmit Timestamp
  - This value is the time at which the server reply departed the server, in 64-bit time-stamp format.



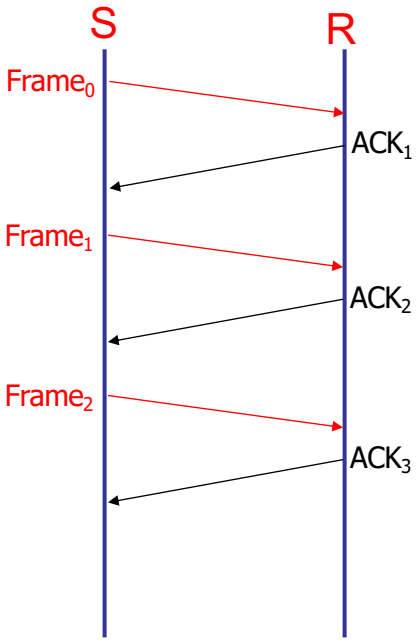
Flow Control

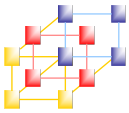
- Ensuring the sending entity does not overwhelm the receiving entity
  - Preventing buffer overflow
- Flow control
  - Stop-and-wait
  - Sliding window



Stop-and-Wait Flow Control (1/2)

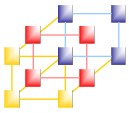
- Source transmits frame
- Destination receives frame and replies with acknowledgement
- Source waits for ACK before sending next frame
- Destination can stop flow by not send ACK
- Works well for a few large frames





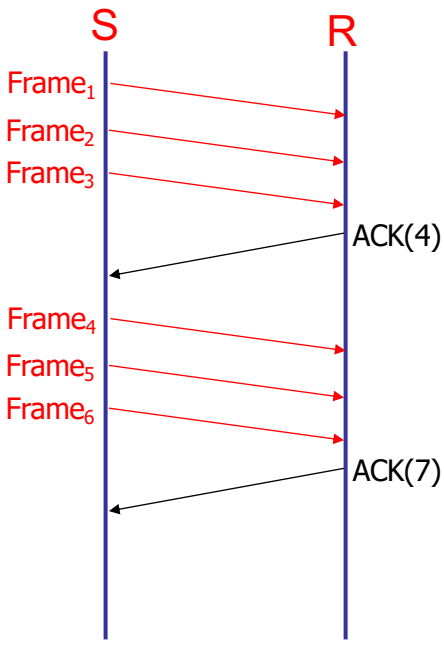
Stop-and-Wait Flow Control (2/2)

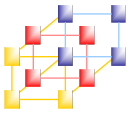
- Large block of data may be split into small frames, because
  - Limited buffer size
  - Errors detected sooner (when whole frame received)
  - On error, retransmission of smaller frames is needed
  - Prevents one station occupying medium for long periods
- Stop and wait becomes inadequate



Sliding Windows Flow Control

- Allow multiple frames to be in transit
- Receiver has buffer W long
- Transmitter can send up to W frames without ACK
- Each frame is numbered
- ACK includes number of next frame expected
- Sequence number bounded by size of field (k)
  - Frames are numbered modulo 2^k

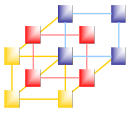




Sliding Window Enhancements

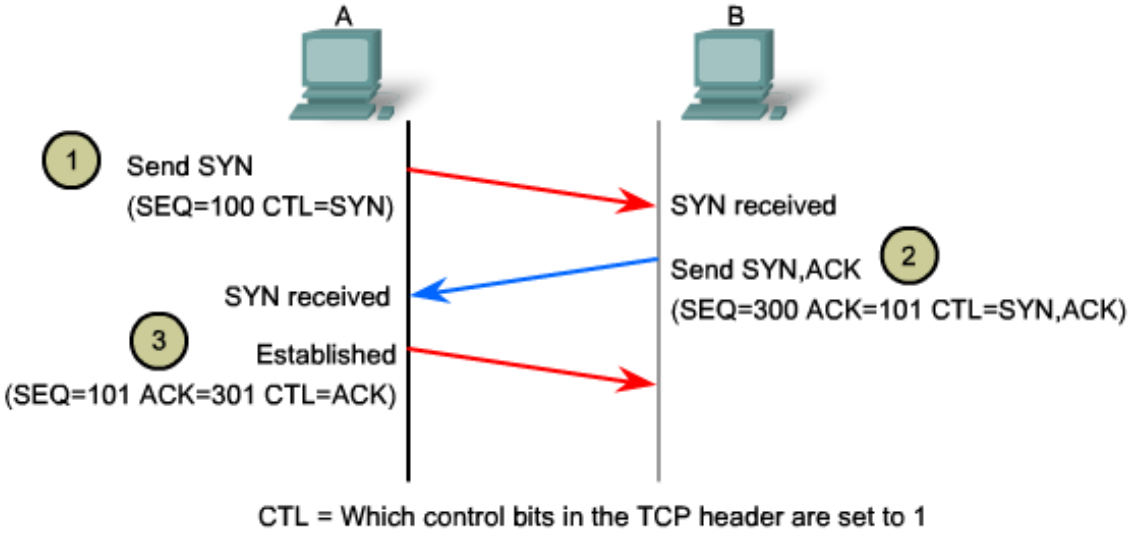
- Receiver can acknowledge RNR frames to forbid further transmission
  - RNR: Receive Not Ready
  - Must send a normal acknowledge to resume
- Piggybacking – each data frame includes a field that holds the sequence number for ACK
  - If no data to send, use a separate acknowledgement frame
  - If data but no new acknowledgement to send, send last acknowledgement number again, or have ACK valid flag (TCP)

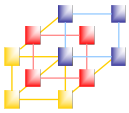
4-SAWServer.py, 4-SAWClient.py
SAWSocket.py



實作 Stop-and-Wait (1/3)

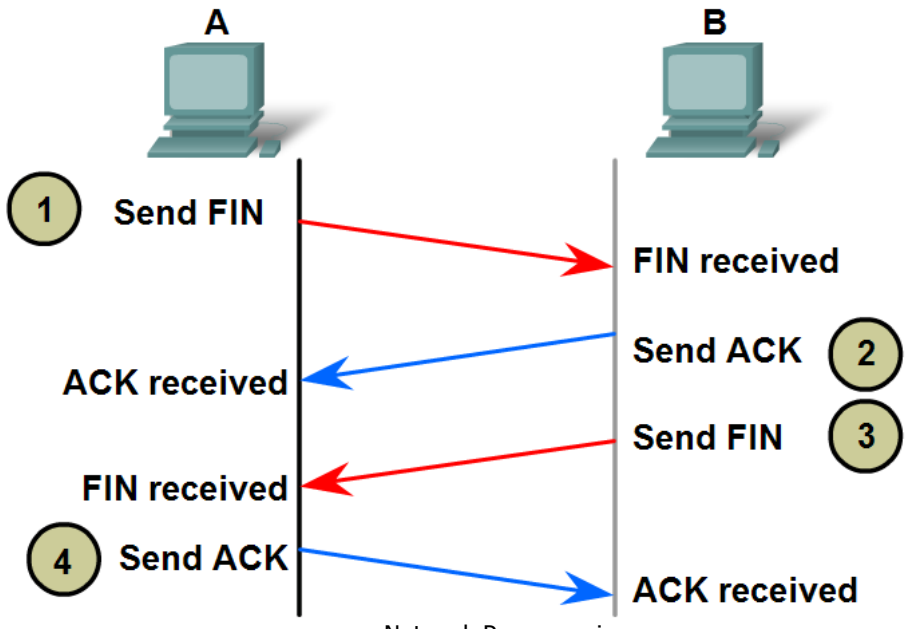
- Connection Establishment

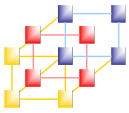




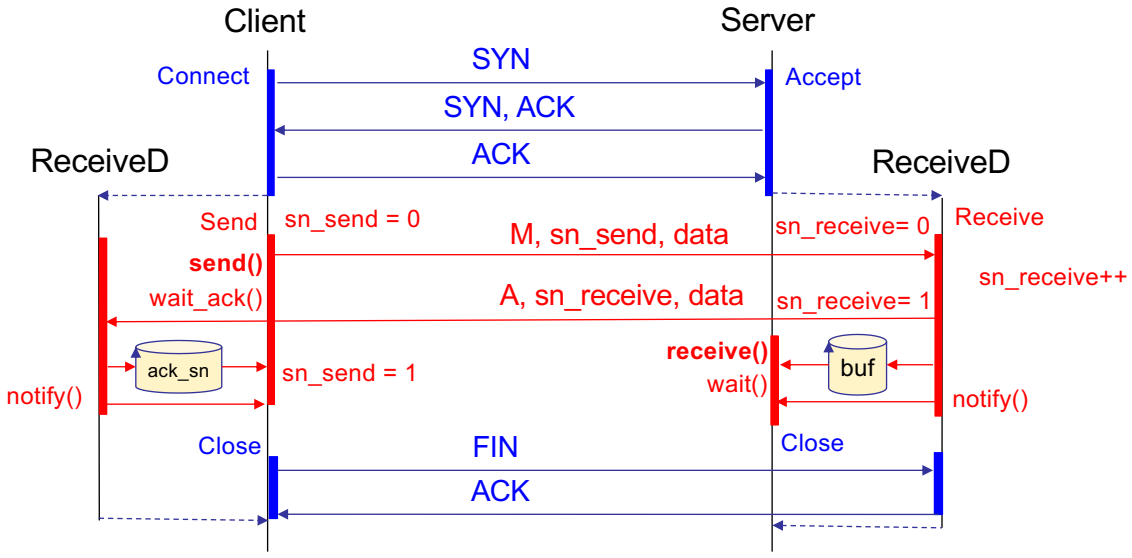
實作 Stop-and-Wait (2/3)

- Connection Termination





實作 Stop-and-Wait (3/3)




```
// buf[0] = (M)essage/(A)ck, buf[1-4] = SN
```