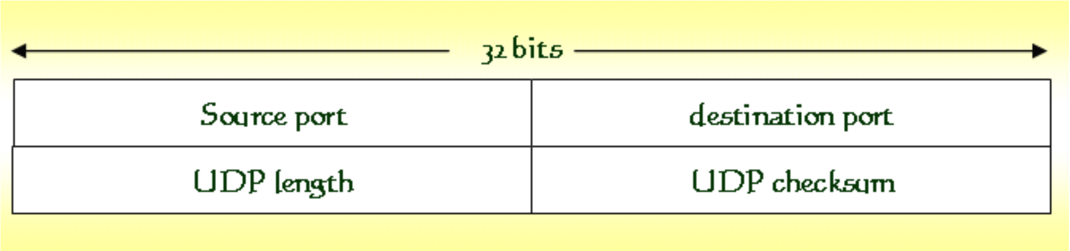
# Unit 5
# User Datagram Protocol (UDP)

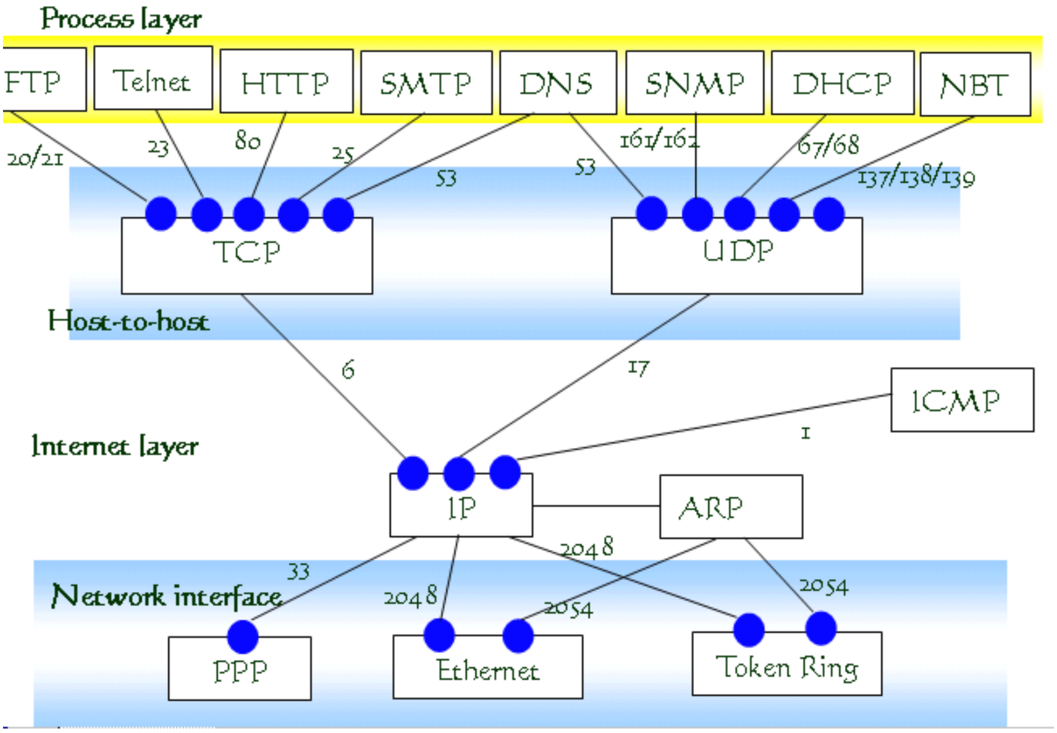
## UDP

- 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



| Source port | destination port |
|---|---|
| UDP length | UDP checksum |

# 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

| 協定 | Port number | Transport 協定 | 涵義 |
|---|---|---|---|
| BootP | 67 | UDP | BOOTstrap 協定(server) |
| BootP | 68 | UDP | BOOTstrap 協定(client) |
| DHCP | 67 | UDP | DHCP 協定(server) |
| DHCP | 68 | UDP | DHCP 協定(client) |
| DNS | 53 | UDP/TCP | Domain name system |
| FTP | 21 | TCP | Server/control |
| FTP | 20 | TCP | Server/data |
| HTTP | 80 | TCP/UDP | HTTP/server |
| NetBT | 138 | UDP | NetBIOS datagram |
| NetBT | 139 | TCP | NetBIOS session |
| SMTP | 25 | TCP | SMTP/server |
| SNMP | 161 | UDP | SNMP/server |
| SNMP | 162 | UDP | SNMP/trap manager |
| Telnet | 23 | TCP | 遠端登入 |
| TFTP | 69 | UDP | 簡易(trivial)FTP |
| WINS | 137 | UDP | Windows Internet Name Service |

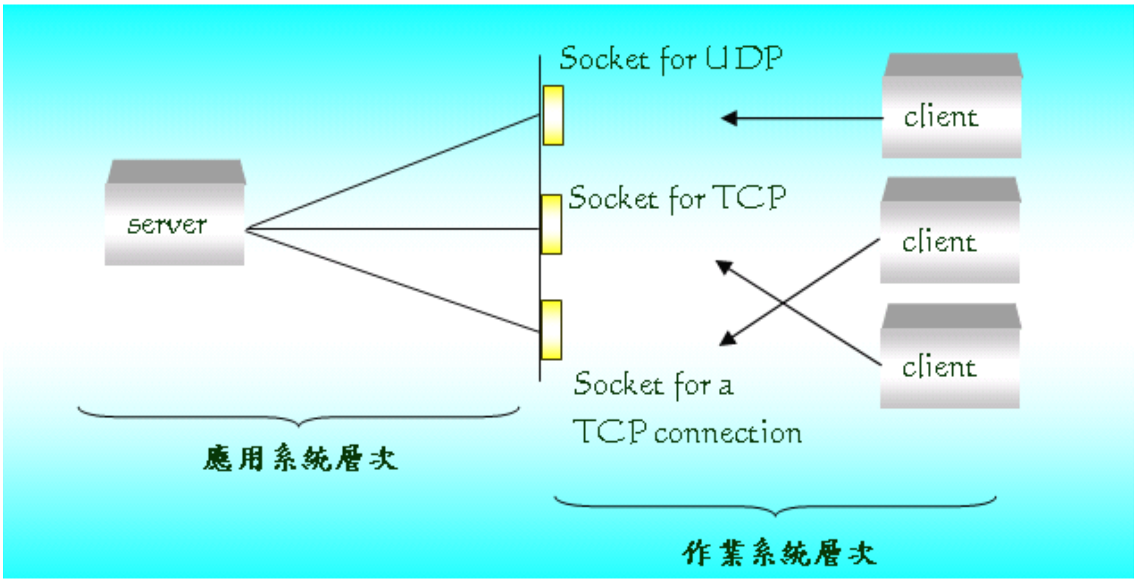
# Client 與 Server 的定義

- 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
  - Server 一般為連線的接收端
    - 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

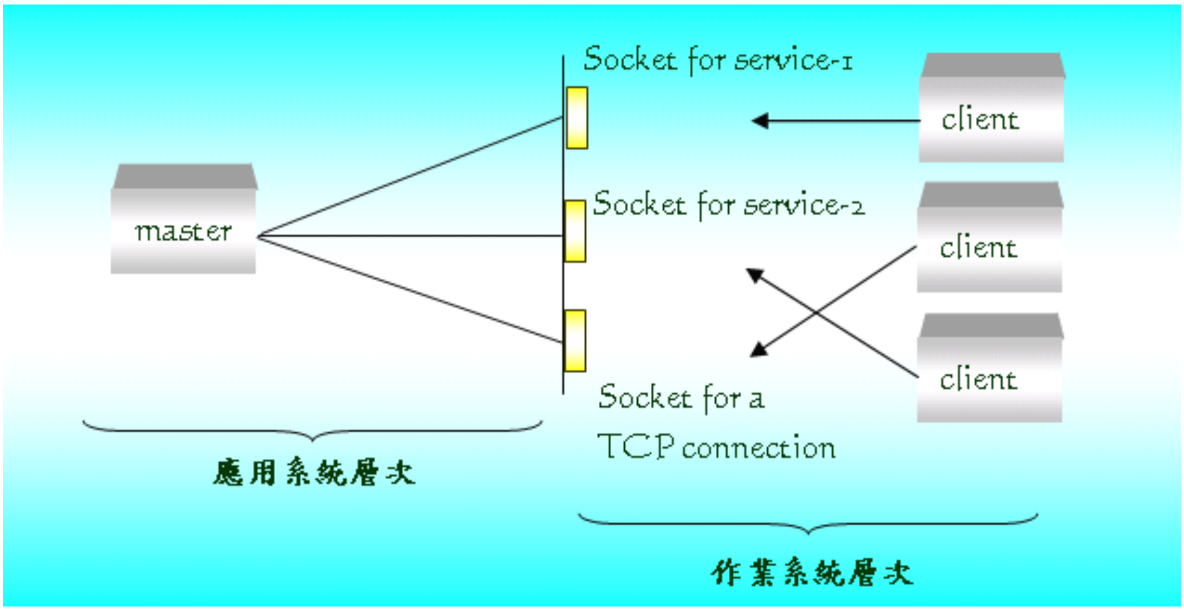
- 多重協定的伺服程式(multiprotocol servers)
- 多重服務的伺服程式(multiservice servers)

# Multiprotocol Server的架構



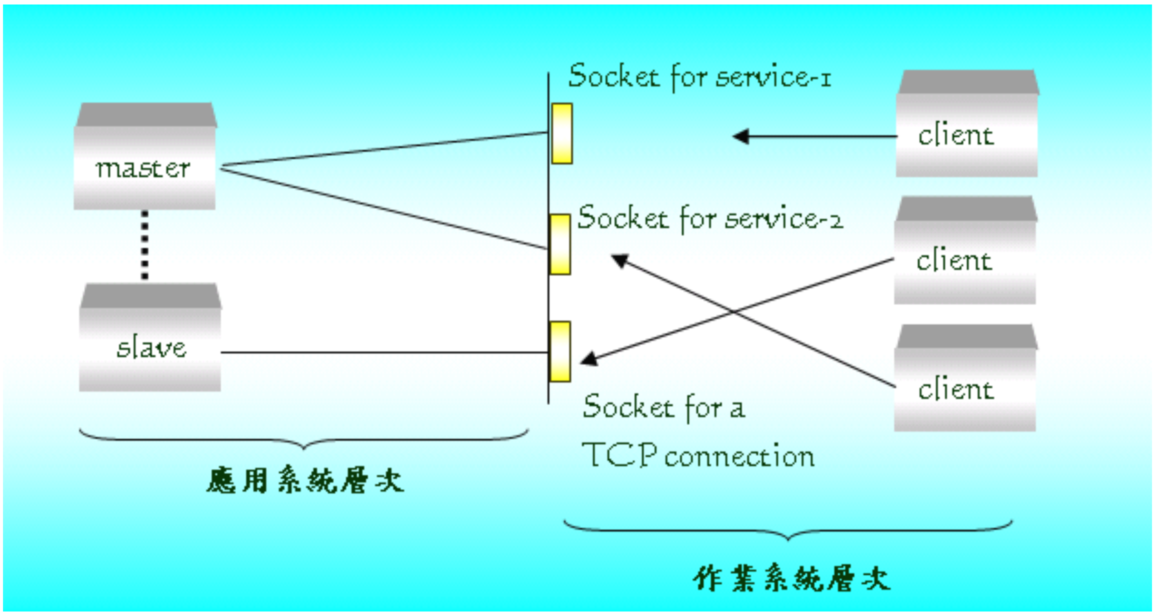
一個 Server 程式同時處理多個同訊協定

# 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

UDPClient1, UDPServer1
UDPClient2, UDPServer2

# Java對於UDP的支援

- java.net.DatagramPacket類別
  - 將資料封裝成 UDP packet – 稱為 Datagram
- java.net.DatagramSocket類別
  - Send: 使用 DatagramSocket 將 Datagram 送出
  - Receive: 從 DatagramSocket 接收 Datagram
  - 每一個 Datagram 為獨立個體

# RFC 1035
# Network Time Protocol (V3)

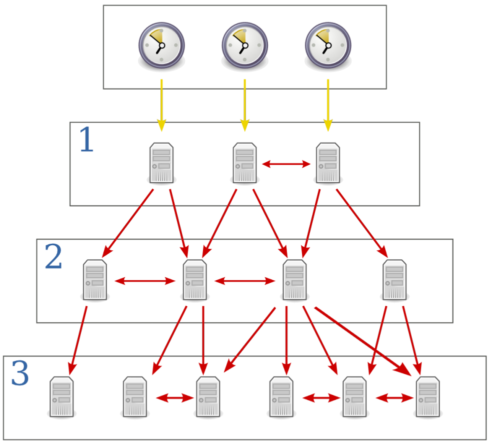
- Defined in RFC 1305
- NTP was designed by David L. Mills of the University of Delaware.
- Based on UDP, port number 123
  - Simple connectionless service
  - No guarantee of delivery
  - No detection of lost packets
  - Can also use broadcasting or multicasting
- Message authentication is optional
  - 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

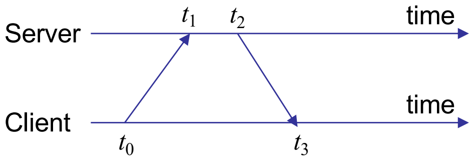
# Clock Strata

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



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

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

- (1) – (2)

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

NTP_Client.java

# 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)

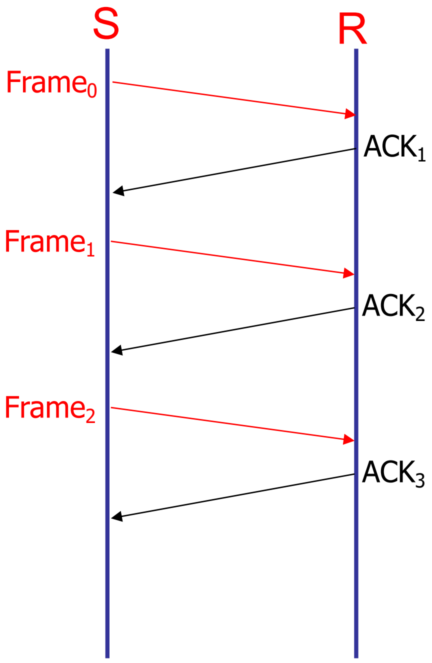
- Reference Timestamp
  - This field is the time the system clock was last set or corrected, in 64-bit time-stamp format.
- Originate Timestamp
  - This value is the time at which the request departed the client for the server, in 64-bit time-stamp format.
- Receive Timestamp
  - This value is the time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

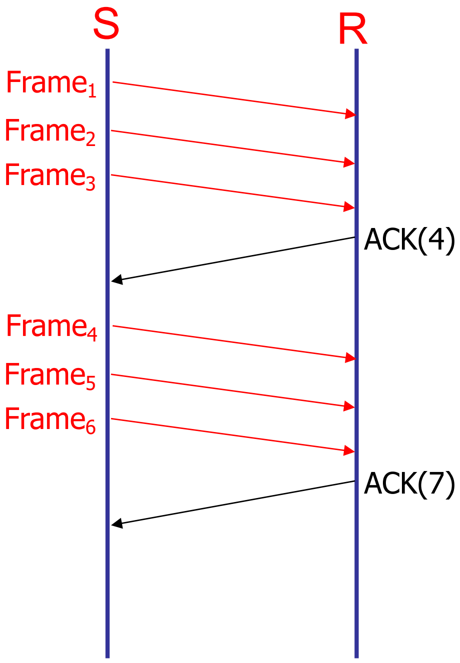


## Stop-and-Wait Flow Control (2/2)

- 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 $2^k$



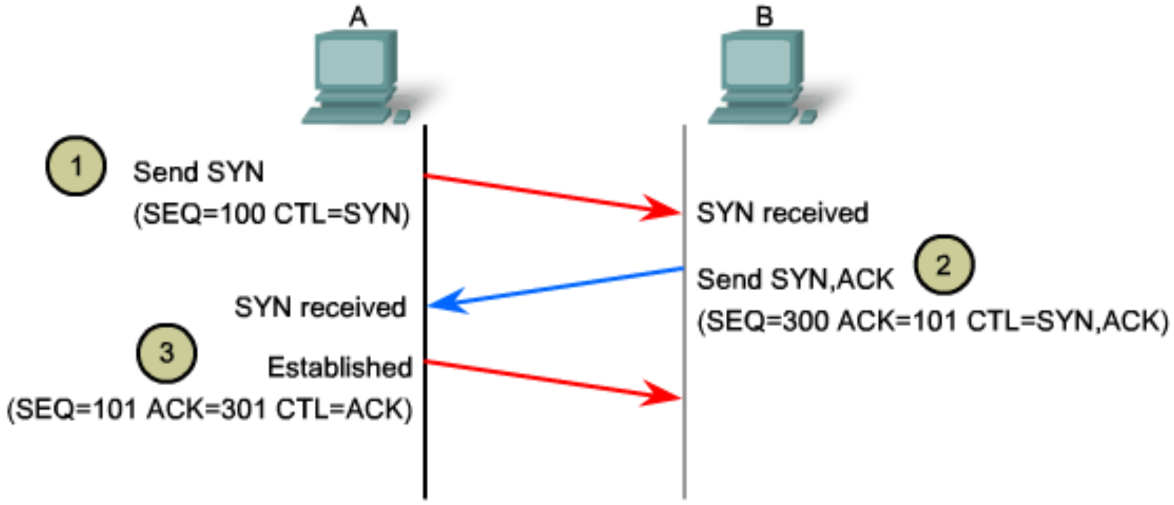
## Sliding Window Enhancements

- 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)

UDPClient3, UDPServer3
SAWSocket

# 實作 Stop-and-Wait (1/3)

- Connection Establishment



A

B

1 Send SYN
(SEQ=100 CTL=SYN)

SYN received

Send SYN,ACK 2
(SEQ=300 ACK=101 CTL=SYN,ACK)

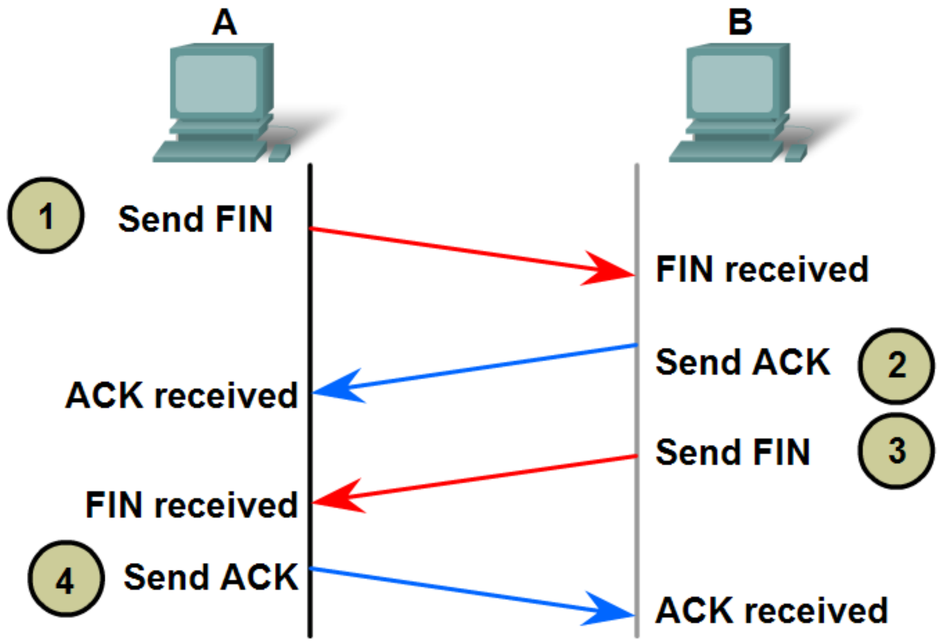
SYN received

3 Established
(SEQ=101 ACK=301 CTL=ACK)

CTL = Which control bits in the TCP header are set to 1

# 實作 Stop-and-Wait (2/3)

- Connection Termination



A

B

1 Send FIN

FIN received

Send ACK 2

ACK received

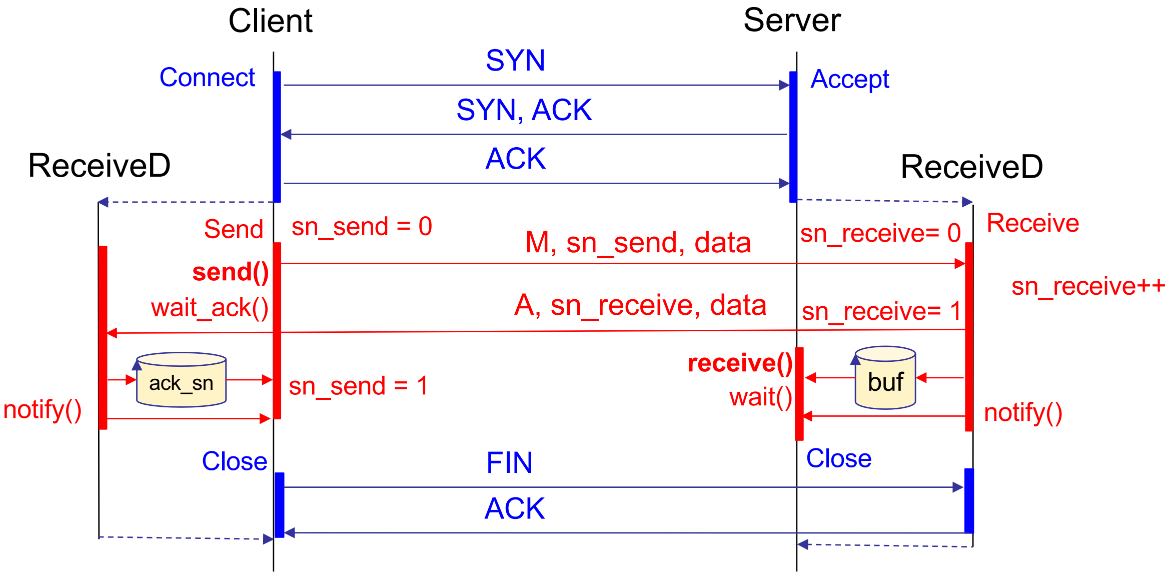
Send FIN 3

FIN received

4 Send ACK

ACK received

# 實作 Stop-and-Wait (3/3)



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