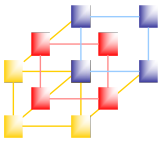
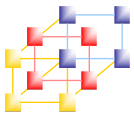
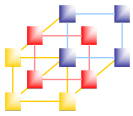


Unit 8
Non-blocking Socket



Non-Blocking Socket 簡介

- 在 UNIX 系統下，使用者建立一個 BSD socket 之後，該 socket 的預設狀態是「阻攔」（Blocking）模式
- 使用者可以利用 ioctl() 函式來變更為「非阻攔」（Non-Blocking）模式

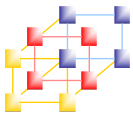


Blocking 模式

- 使用者在呼叫了某一個 blocking 函式之後，程式必須等待該函式呼叫完成（或失敗），並且回返（return）之後才能再繼續執行下一個指令
- Example

```
n = recv( socket_num, data, 50 );
memcpy( buf, data, n );
```

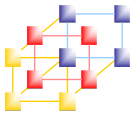
  - 在 Blocking 模式下，如果對方沒有傳送資料過來的話，那麼這個程式將會一直被阻攔在 recv()，而不會執行到 memcpy()
  - 等對方資料過來後，才會自 recv() 呼叫回返，然後執行 memcpy() 函式



Non-Blocking 模式 (1/2)

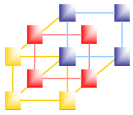
- 使用者呼叫了一個 non-blocking 函式之後，不管要求的條件或狀況是否成立或完成，都會馬上自該函式呼叫回返，接著執行程式的下一個指令
- Example

```
n = recv( socket_num, data, 50 );
if  /* 收到資料 */
   memcpy( buf, data, n );
else     /*沒有收到資料 */
   ...
```

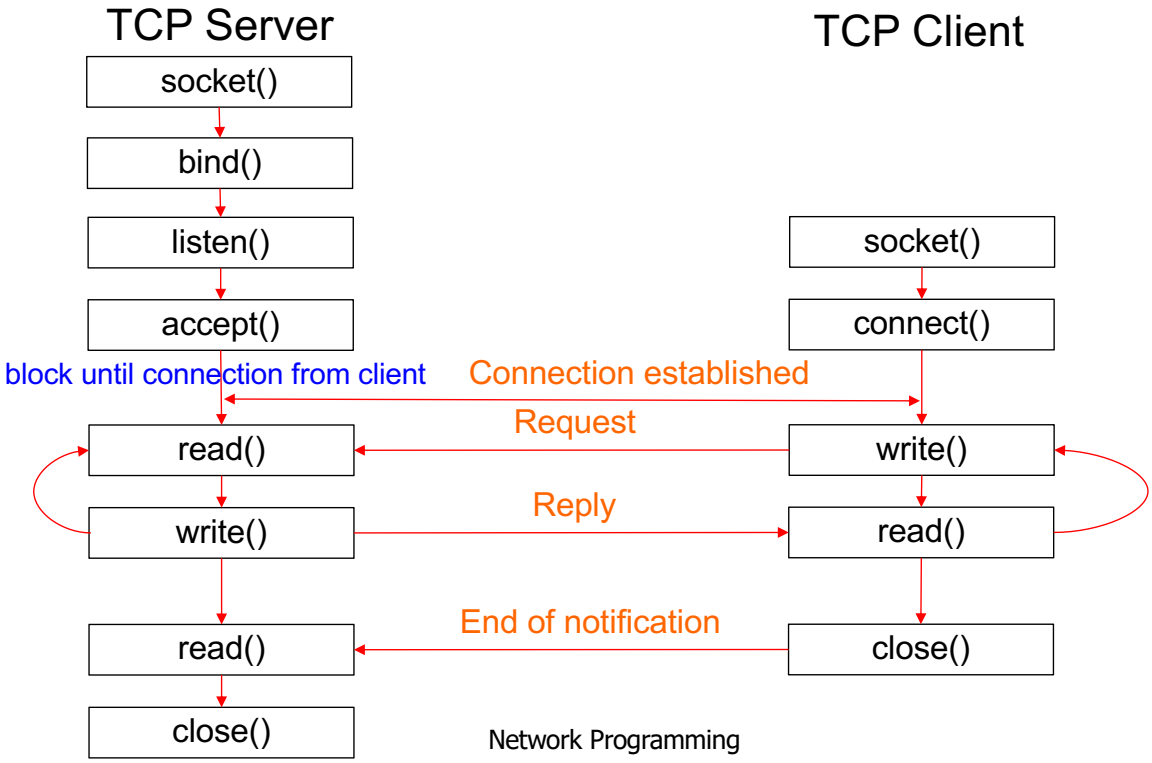


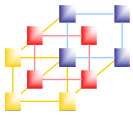
Non-Blocking 模式 (2/2)

- 在呼叫 recv() 函式時，不管對方是否已經送資料來了沒有，程式都不會被阻攔在 recv() 上
  - 如果此時的確有資料了，那麼就會被放在 data 這個暫存區（buffer）
  - 如果沒有資料，那麼就會回返錯誤的訊息 (python 會產生一個 BlockingIOError exception)
- 由於 non-blocking 模式在呼叫 recv() 時，並不一定有資料，且不知道什麼時候對方會送資料來，所以在使用 non-Blocking 模式時，程式必須常常再次呼叫 recv()，以檢查資料是否來了 (polling)



TCP 網路應用程式模型

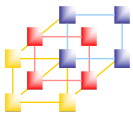




Non-blocking Socket

- Instead of timeouts, socket can set non-blocking
  - `socket.setblocking(False)`
- Future send(),recv(), connect(), accept() operations will raise an exception if the operation would have blocked

1-NonblockingServer.py
1-NonblockingClient.py



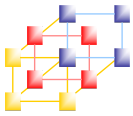
Non-blocking Server (1/3)

- 建立一個 TCP Server Socket 並將此 Socket 設成 non-blocking 模式

```
srvSocket = socket.socket(socket.AF_INET,
  socket.SOCK_STREAM)
srvSocket.setblocking(False)
```

- 將所建立的 scoket 綁定於 port

```
srvSocket.bind(('', PORT))
srvSocket.listen(backlog)
```

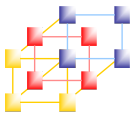


Non-blocking Server (2/3)

- 接受連線

```
try:
    client, (rip, rport) = srvSocket.accept()
    break
except BlockingIOError:
    pass
```

  - 這一個動作將以 non-blocking 方式進行
    - 如果有進來的連線要求，accept() 將回傳 socket
    - 如果沒有連線的要求，將產生一個 BlockingIOError 的 exception

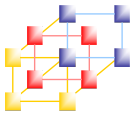


Non-blocking Server (3/3)

- 接收訊息

```
try:
    client_msg = client.recv(recv_buff_size)
    break
except BlockingIOError:
    pass
```

  - recv() 仍以 non-blocking 方式進行
    - 如果有收到訊息，recv() 傳回訊息
    - 如果沒有訊息，將產生一個 BlockingIOError 的 exception



Selector (1/2)

- Selector 提供一個機制來註冊我們有興趣的 I/O events，並提供方法讓我們了解那些已註冊的 events 已發生
- 建立一個 selector

```
Selector selector = Selector.open();
```

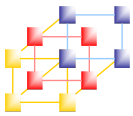
- 註冊有興趣的 I/O events

```
SelectionKey key = ssc.register(selector, EVENT-TYPE);
```

- 等待 event 發生(blocking mode)

```
int num = selector.select();
```


2-MultiPortServer.py
2-MultiPortClient.py



select — Waiting for I/O completion

- r, w, x = select.select(*rlist*, *wlist*, *xlist*[, *timeout*])
  - rlist: wait until ready for reading
  - wlist: wait until ready for writing
  - xlist: wait for an “exceptional condition”
  - Empty iterables are allowed
  - The optional *timeout* argument specifies a time-out as a floating point number in seconds.
    - When the *timeout* argument is omitted the function blocks until at least one file descriptor is ready
    - A time-out value of zero specifies a poll and never blocks.
  - The return value is a triple of lists of objects that are ready: subsets of the first three arguments.