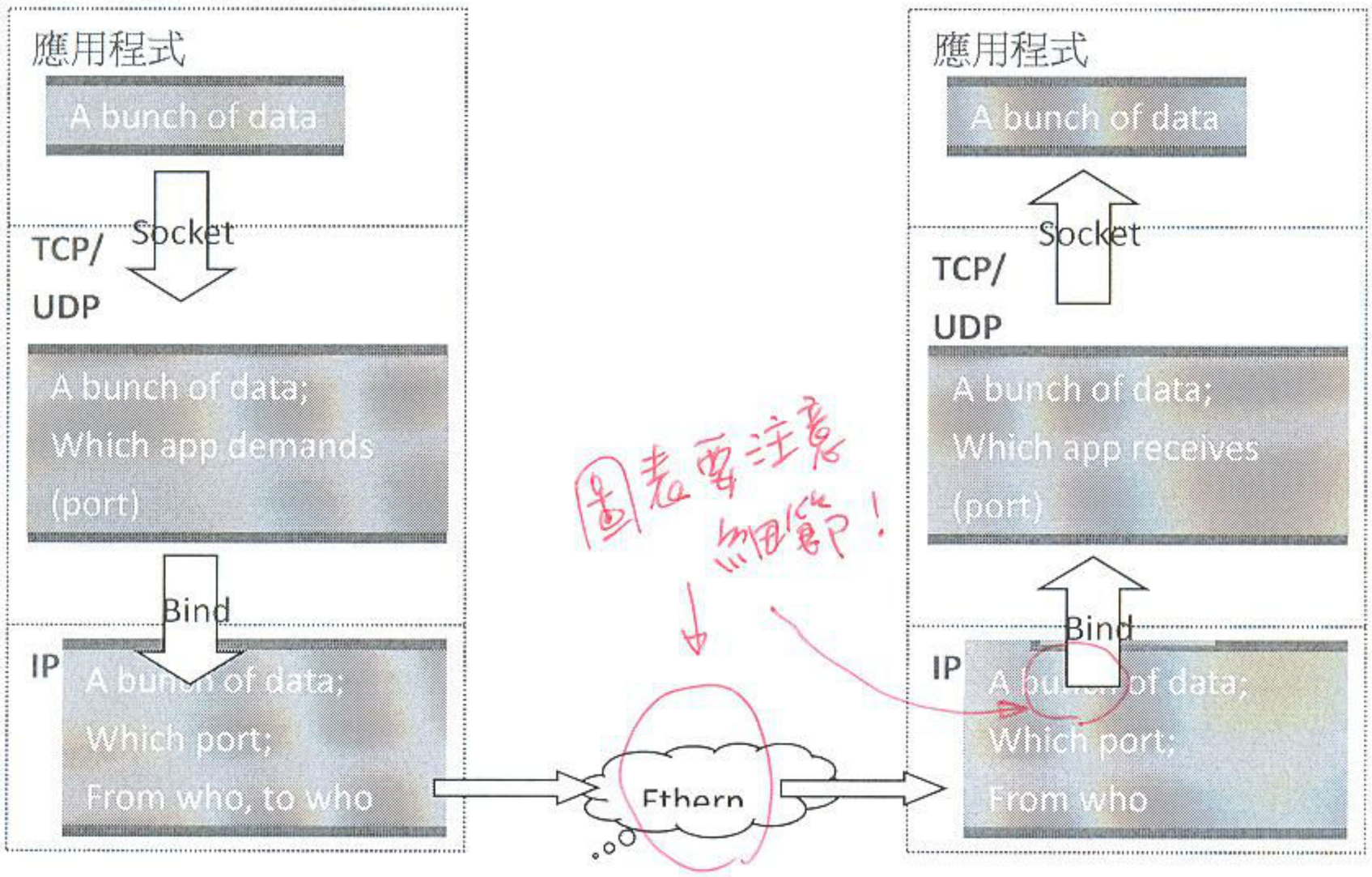
大標題要給編號.

導論(Introduction)

(1)何謂 Socket ; UDP or TCP/IP;Port ?

網路世界的蓬勃發展，有賴於一套完善的架構，其中 Socket，TCP/UDP，IP，Port 是幾個常見的技術名詞。



圖表要注意細節！

Socket(套接口)：

一如其名地，Socket 代表著一個「插孔」，是軟體程式欲調用網路資源時，向系統 API(應用程式介面)請求對外開的一扇「窗戶」。在需要時打開，在應用程式不需要時關上。就概念上類似於開擋、關檔的操作。

Port(埠口)：

埠口的存在，就相當於高速公路一樣。每一埠的本質都一樣(都是馬路)，但是可能會有約定俗成的一些規範(Well-Known-Ports)— —像是 80 port 是網頁，23 port 是 BBS 常用的 telnet……類比上就像大型車種必須走外線車道，路肩只能走特殊車輛。

這個例子不太對，但是會想要舉例是好習慣.

TCP(傳輸控制協議) & UDP(用戶數據報協議)：

此二種協議爲重要的傳輸協議，兩者的構造以及使用時機不相同。

兩種共通的部分：發送的埠口號、接收的埠口號、校驗碼(檢查和)。

不同的地方在於— —UDP 會有 UDP 封包長度資訊；TCP 則會有 順序編號、確認回覆編號、資料偏移量、緊急指標。

這兩個中文翻譯很糟，不如打英文全名.
通常正式論文是會寫成：英文全名(縮寫).

簡單來說 UDP 提供一個鬆散的溝通方式，屬於廣播性的發送，只會告知轉送者「是誰發的」、「最終是誰該聽到」、「話有多長」。而 TCP 則是一穩定許多的溝同方式。會明確的告知，這「整份文件有多長」、「這封包是文件的哪一段」、「這封包是否帶有緊急資訊」.....

這份報告的段落切的太細了.

IP(網際網路協議)：

IP 的存在就相當於地址。目前現行通用的有 Ipv4, Ipv6 兩種。其中 Ipv4 以 32 位二進制表示，但又時常使用「點十進制」(dot-decimal notation)來書寫。即將 32bit 的位址切成四組 8bit 後，再以十進位表示、並以點號隔開。

然而 IP 封包中，亦會有 4 個 byte 的欄位用來表示上一層是什麼樣的協定(ICMP/TCP/UDP.....)這樣的過程稱作 bind(綁定)。

P.S. 其實 IP 位址的運作亦仰賴實體 NIC(網路控制器，通常爲網路卡)的硬體位址。透過 ARP(位置解析協定)將終端機(有上網能力的機器)與網路上通行的 IP 位址作映射(mapping)。

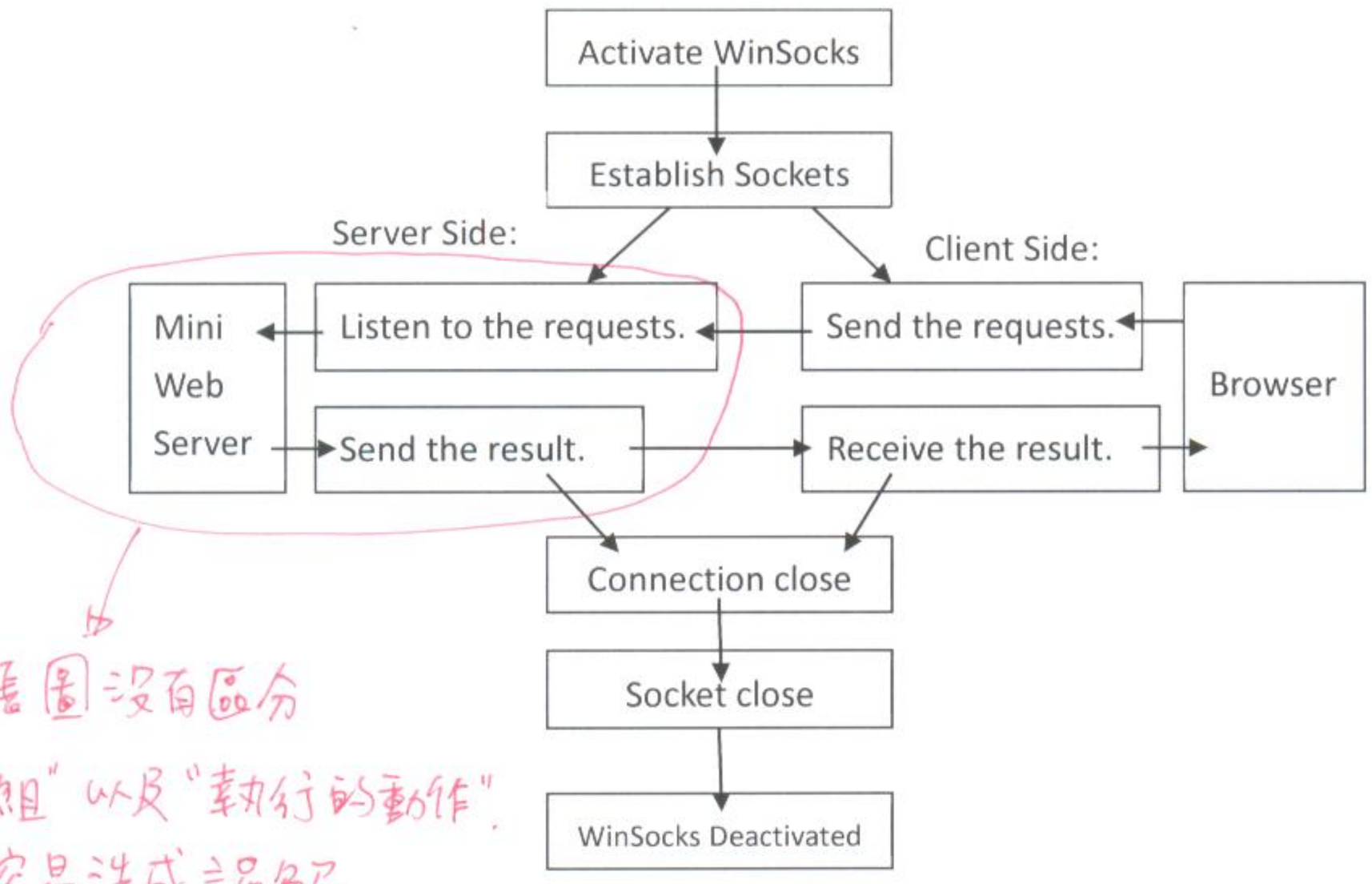
P.S. 不適合用在報告中.

(2)Client & Server Model

有了上述的傳輸協議後，便會有應用以上協議來進行服務的模型。較爲常見的包括了這次要談的「客戶、伺服器模型」外，也有常見的「點對點模型」(Peer to Peer Model)。

其中在本次作業中的，是建構於 Windows Socket 下的 Client & Server model(嚴格來說，是特別化的 Browser & Server 模型，僅僅處理 http 的服務)的應用。

其主要的行爲以圖例表示：



這張圖沒有區分"模組"以及"執行的動作".很容易造成誤解.

程式分析(Program Analysis):

原程式碼 23 到 27 行:

旨在定義出緩衝區大小(8096 個字元)、LOG 檔緩衝大小(bufsize 的左移一單位，相當於兩倍大)、以及記錄訊息的種類(輸入錯誤=>1，錯誤提示=>2、系統資訊=>3)。

29~47:

宣告一結構陣列 extentions[]，並將附檔名正規化、對應到特定的路徑上。

50~87:

定義 syslog()的行爲，主要是針對使用者不合規定的操作進行處理。如非 GET，要求的路徑、檔案不存在或無法解析；並且寫入 log 中。(亦有單純的寫入 log，如處理 get 時，或者純粹 send 出資料時)而最後的 WSAcleanup()會將整個 Socket 砍掉、中止程式。

91~207:

web()是整個程式的核心部分。

最先宣告出一成整數 nbytes = recv(cli_socket, buffer, BUFSIZE, 0)。recv()爲 winsock 中的函式，會回傳接收到的位元組數目；呼叫失敗時會回傳 SOCKET_ERROR。

而透過其它定義有以下行爲— —

(1)當 nbytes<=0 時，即表示 client 作出了未定義或者非法的操作，呼叫 syslog()、寫入記錄中後結束程式

(2)尾端若不是 GET 指令的話，亦會呼叫 syslog 後結束。

(3)視「詢問父目錄」爲非法請求。

(4)無特殊請求下，導入 index.html。

(5)要求的副檔名不在 extentions[]中、或檔案不在時，輸出錯誤訊息。

(6)favicon.ico 不存在時，將此紀錄寫入 log 中。

(7)無以上事項，呼叫 winsock 中的 send()，將資料輸出給 client、並且寫入 log 中。

209~290:

此處定義 server 端的行爲準則。

argc,argv[]兩者爲 C 語言中，對於呼叫程式時的 argument 作出處理的兩個變數。argc 代表有幾個 argument，而 argv 字元陣列則儲存有哪些 argument。

行爲模式：

(1)如果使用者呼叫此程式時，有給予多餘的指令(argc>1)，則輸出這項程式的基本功能後，自動關閉程式。

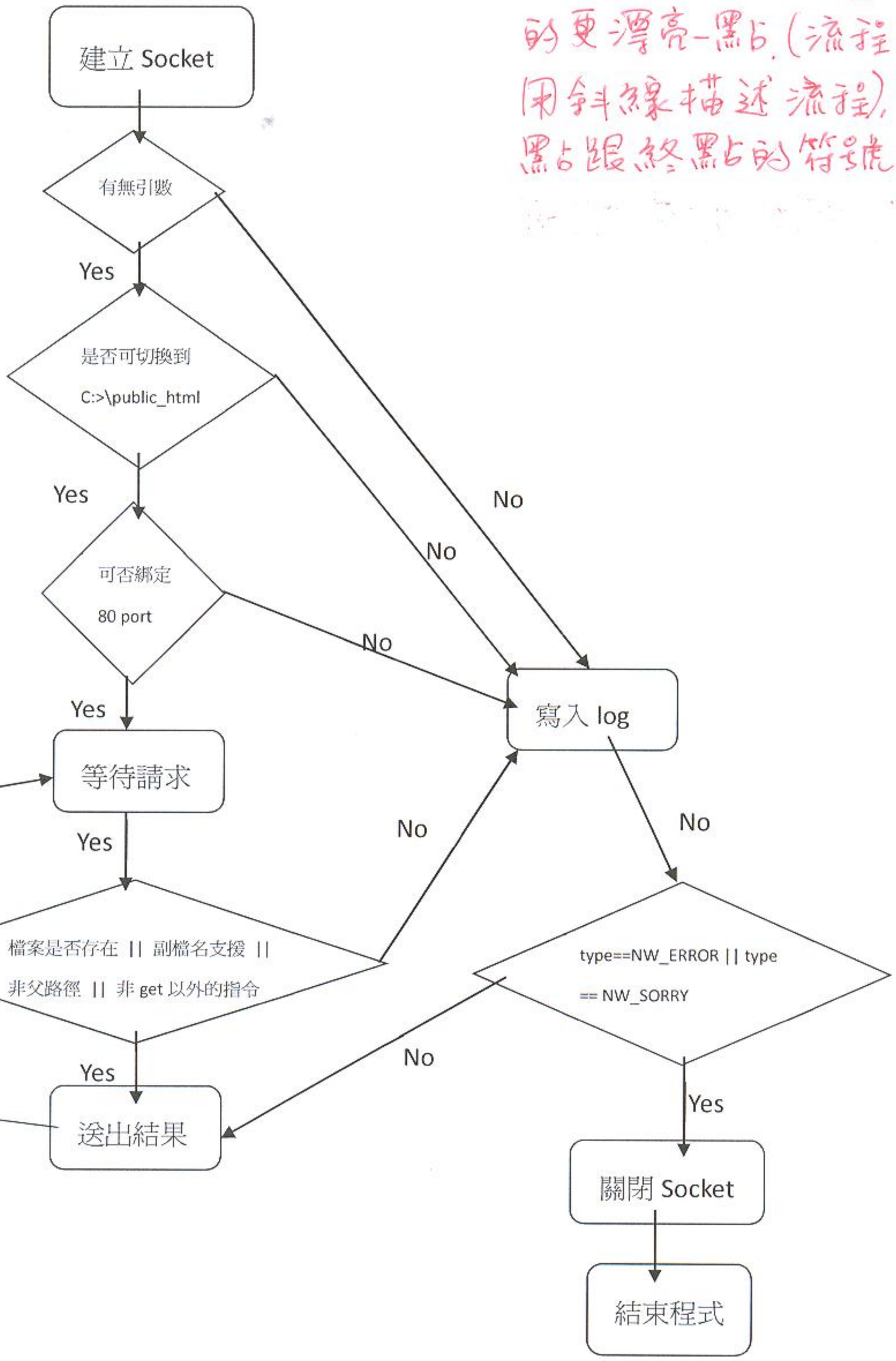
(2)若無法切換目錄到磁碟代號 C 或者第三號磁碟，則輸出錯誤訊息，關閉程式。

(3)如果無法建立 socket、並綁定到 80 port 上時，亦輸出錯誤訊息。

(4)監聽 80port，收到要求時呼叫 web()來處理需求。

原架構可見下圖：

報告中用流程圖分析程式行為是一大優點，但是圖可以再畫的更漂亮一點。(流程圖能夠用斜線描述流程)，而且起點跟終點的符號也沒畫。



排版要多用心，不要把本文和程式碼混在一起用同樣的字型、格式排版。

改進(Improvement):

這個程式最讓人起疑竇的的部分在於錯誤處理上，只要任何一個步驟出了錯，便大部分會直接進入 syslog()中的

```
if (type == NW_ERROR || type == NW_SORRY)
    {WSACleanup();exit(1); }
```

此一段落中，並且強制結束程式。

個人認爲應該將錯誤切成「致命性錯誤」(例如說無法將建立 socket)，以及一般使用者輸入不合法指令的「可容許錯誤」。故將 NW_SORRY 跟 NW_ERROR 作出不同的處理— —

(1)NW_ERROR 爲 Socket 初始化時發生問題，故應該要讓使用者能重新設置綁定的 port、或者路徑

(2)NW_SORRY 則是使用者要求了不被允許的操作，應該要輸出 HTTP 403 forbidden 頁面，而非連 server 端都停止運行。

將上述程式碼改寫爲：

```
if (type == NW_ERROR)
    WSACleanup();
```

並在 main function 中建立一個 check flag，並將 socket 初始化的過程置於 while(~check_flag){//Initializing Socket}之中，便可以達到要求。

另外則是此 mini web server 只能綁定 80port，目錄也只預設爲 C:>\public_html\。但是我們在其他 windows 上面執行時，很有可能沒有權限進行這些存取。所以應該要善用 argv[]陣列的操作，例如說規定 argv[1]爲 port number，argv[2]則是 server 架設者欲進行映射(mapping)的目錄。

很好的作法！

故

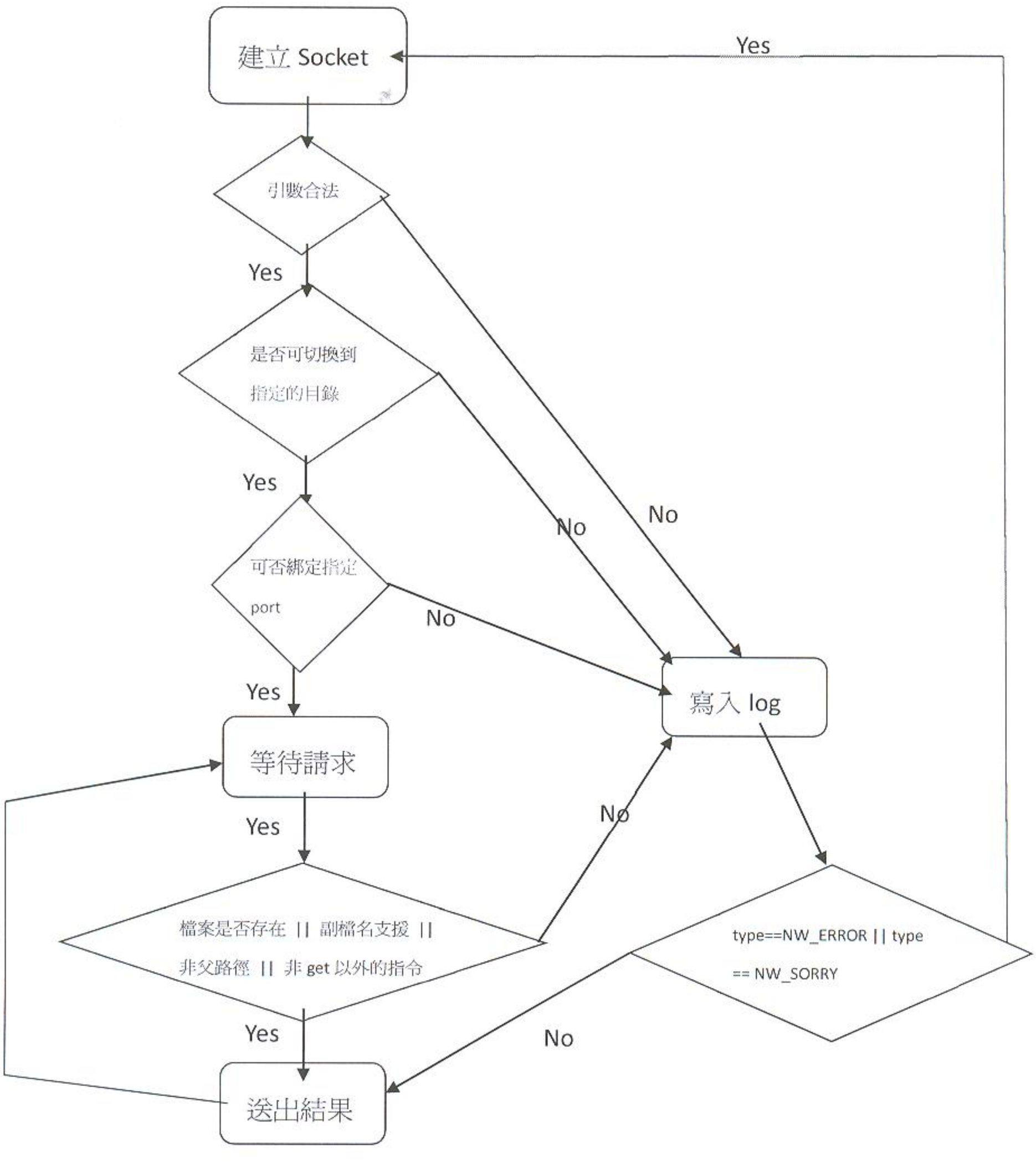
(1)修改原本的 port 定義：port = atoi(argv[1]);

(2)改善原本的路徑設定：

```
if( !strncmp(argv[2],"/"     ,2 ) || !strncmp(argv[2],"/etc", 5 ) ||
            !strncmp(argv[2],"/bin",5 ) || !strncmp(argv[2],"/lib", 5 ) ||
            !strncmp(argv[2],"/tmp",5 ) || !strncmp(argv[2],"/usr", 5 ) ||
            !strncmp(argv[2],"/dev",5 ) || !strncmp(argv[2],"/sbin",6) )
 {
```

```
        (void)printf("ERROR: Bad top directory %s !! ",argv[2]);
        exit();
}
if(chdir(argv[2]) == -1)
{
        (void)printf("ERROR: Can't Change to directory %s !!",argv[2]);
        exit();
        }
```

修改後的架構見下圖：

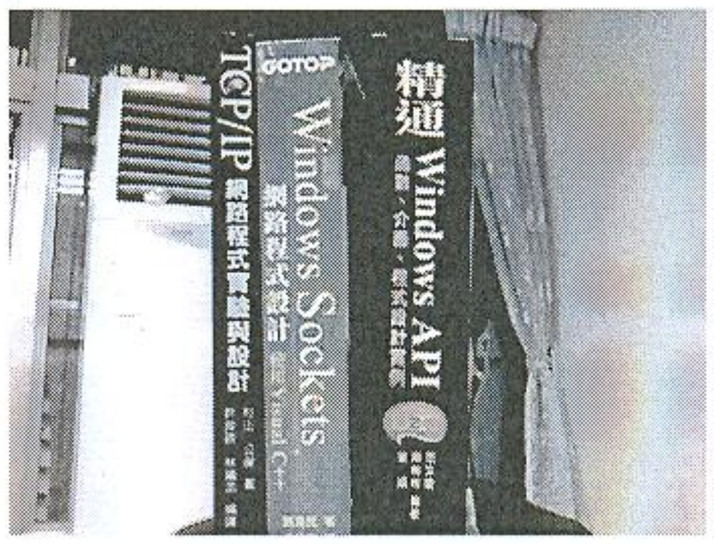

建立 Socket
Yes
引數合法
Yes
No
是否可切換到
指定的目錄
Yes
No
No
可否綁定指定
port
No
Yes
寫入 log
等待請求
Yes
No
檔案是否存在 || 副檔名支援 ||
非父路徑 || 非 get 以外的指令
type==NW_ERROR || type
== NW_SORRY
Yes
No
送出結果

結論(Conclusion):

經過這次的作業後，我更加深入理解了 Windows Socket 的操作；並且爲此也去查閱了一些 TCP/IP 上的實作方式，以及各式封包的構成要件。

而且 mini web server 真的是一個很方便的小程式，以往要架網站時，即便是最方便的 Lighttpd 都還是需要經過一定的程度的安裝與設定，大小也不見得真的多小(免安裝的 Lighty2go 約 12.5Mb)。但透過這支小程式，我們可以迅速的架起一個小型的網站，不管是要交換檔案、亦或是緊急備援系統，都可以在最精簡的狀況下執行。(總比直接讓使用者看到 404 not found 好:P)

參考書目：《精通 Windows API》
《精通 Windows Sockets 網路程式設計》
《TCP/IP 網路程式實驗與設計》



報告最後都應該要列出參考資料，但參考資料有一定的格式，
最起碼要列出：①作者 ②書名 ③出版社 ④出版年月。