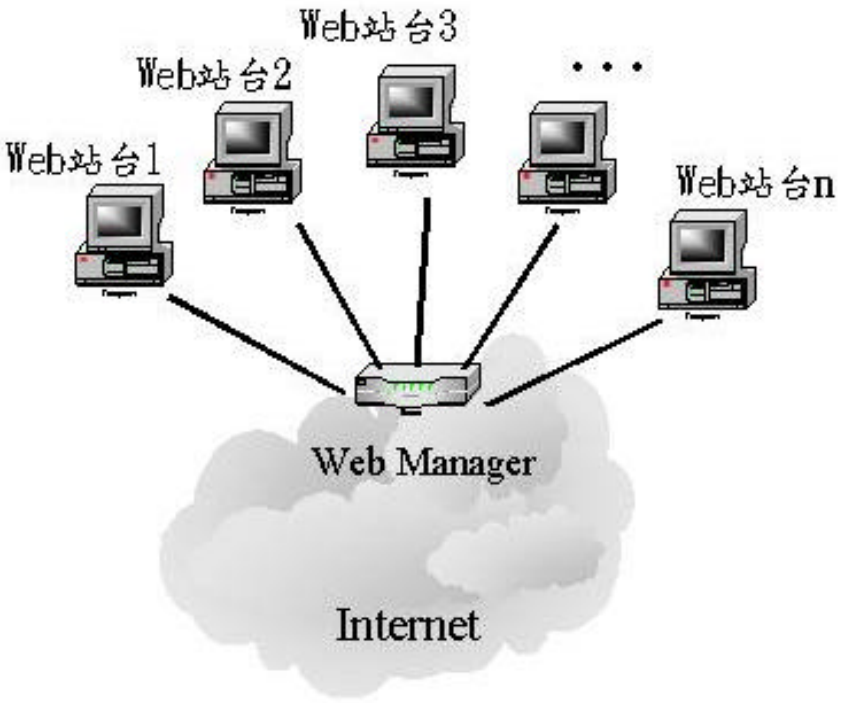
目錄

圖表目錄

1. (簡介)Introduction

由於網路的普及，上網的使用者逐年以等比級數的方式激增，促使了電子商務的興起與蓬勃發展。著名的網站如奇摩站(www.kimo.com.tw)、PC Home(www.pchome.com.tw)等常常有數以萬計的連線同時進入，通常必須提供數台至數十台電腦來服務使用者。當這麼多網站集合在一起時，必須作有效統籌管理。如頻寬控管(Bandwidth Control)、負載平衡(Load Balance)等功能。解決控管問題可分為分散式控管及集中式控管兩種。分散式法則如奇摩站，進入首頁之後，利用超連結方式連結到其他網站存取資料。如首頁網址為 http://www.kimo.com.tw/，證券資料在 http://stock.kimo.com.tw/，新聞資料在 http://news.kimo.com.tw/，利用不同電腦提供不同的資訊。分散式的優點在於依據網站的內容分類來使用不同的電腦服務，透過超連結的方式將使用者連線引導至對應的網站進行服務。此種方式適用於 Internet 網際網路上，其網站可散佈到網路上不同的地方，可在搭配 Mirror Site 的方式，使得 Client 可在最近的站台上存取資料。其缺點為網站可分散在 Internet 上的任何地方，在站台管理上比較困難。另外集中式法則是所有連線與單一指定的系統進行連線，在由此系統依據使用者相關資訊再進行轉接服務。此種方式適合使用在 Intranet 上。其優點為可在此系統上製作許多應用服務，如頻寬控管(Bandwidth Control)，經由在進入 Intranet 網路內部時以頻寬管理系統作為控管的方式，所有的連線必先進入此系統之後再藉由此系統進行轉接的服務，各個連線可依身分等級或存取內容等不同的因素進行頻寬控制，享受不同服務品質，尤其在電子商務的應用上特別合適。電子商務基本商大多需進行身分認證，因為所有連線均需經過此系統的轉接，所以我們可透過此系統提供單一窗口提供此項服務。所有需集中處理的事項可透過此系統提供服務，可降低站台管理的複雜度。我們利用下表作分散式與集中式網站建置的比較

	集中式	分散式
適用範圍	Intranet	Internet
站台管理	容易	不易
頻寬控制	可行	不行
Client 連線相關數據統計	容易統計	不易統計
電子商務的應用程度	高，因為可透過 Web Proxy 進行整合應用。	較低，常因超連結的關係使得 Client 連線在不同網站上跳來跳去，不易掌握 Client 端的動向。

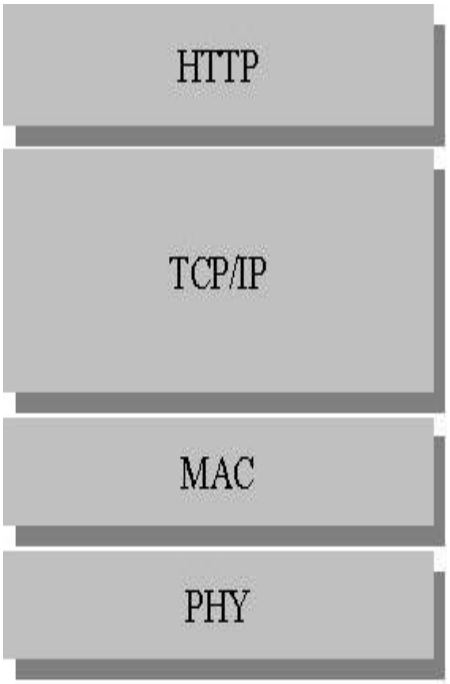


圖表 1-1 網站管理系統架構

因此，我們選擇以集中式的方式設計一個 Web Proxy(**如圖** 1.1)**使得多個** Web **網站能夠作有效的管理以提供電子商務高品質的服務。此系統採用集中式管理法則，站台對外只有單一** IP **位址，對內進行多台站台的管理。**Client **與** Web Proxy **進行連線，經由** Web Proxy **利用** Client **所攜帶的** Cookie **資料進行身分辨識之後將此連線轉接至對應的站台上，此後** Client **與** Web **站台之間的通訊皆經由** Web Proxy **進行轉接。除此之外，可依照** Client **的身分等級不同在進行** Web **站台的負載平衡**(Load Balancing)**，先確認** Client **等級之後，再使用負載平衡演算法選出多個站台中的一台站台進行服務，此選擇的方法乃考慮站台目前的連線狀況為依據，使站台能平均分散進入的流量。**

1.1 HTTP 通訊協定

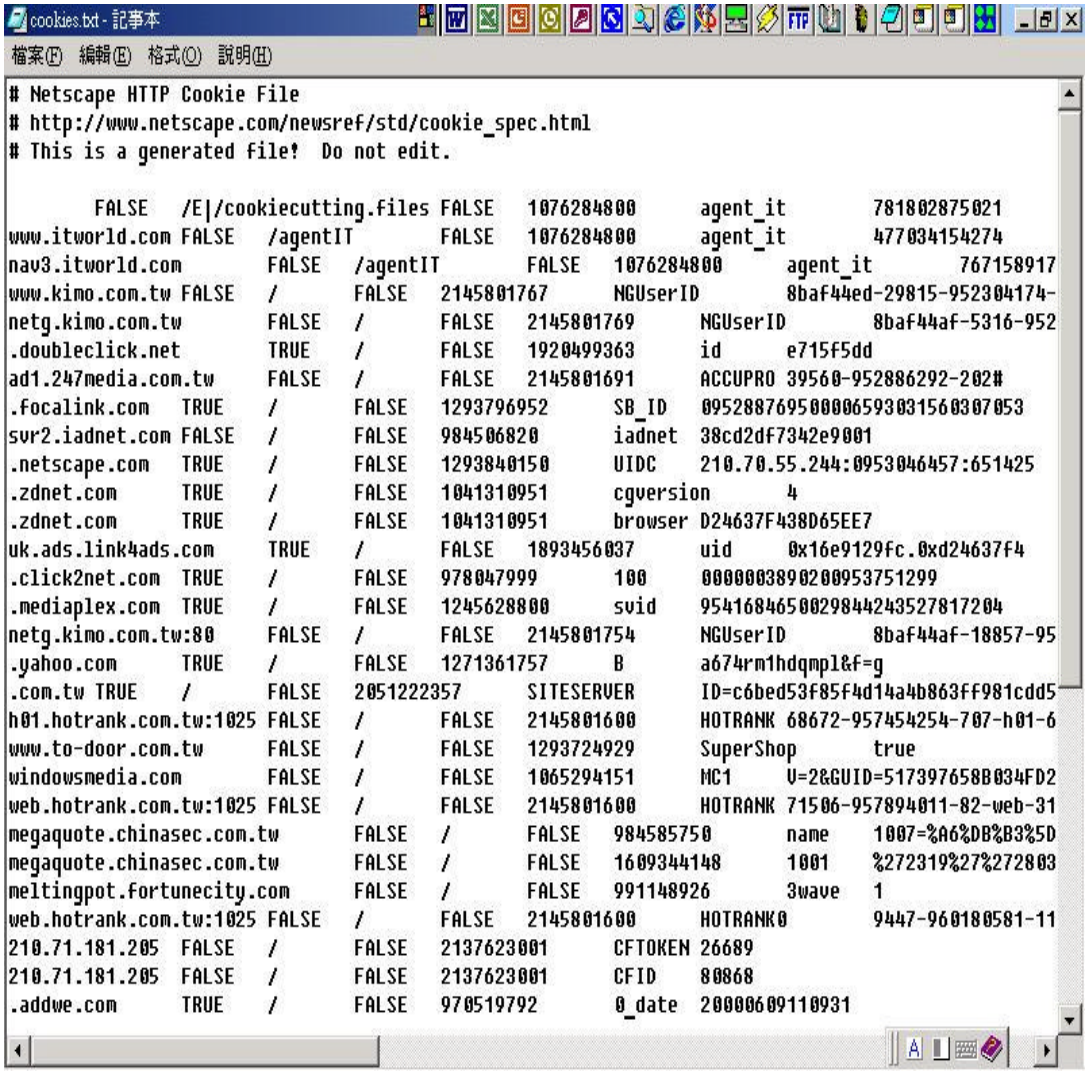
在網際網路上我們使用 TCP/IP(Transmission Control Protocol/Internet Protocol)[20]通訊協定作為電腦與電腦之間的溝通的通訊規則。而在網際網路上最常使用的軟體為瀏覽器。透過瀏覽器我們可以連線至網站上擷取許許多多的資料，包含文字、圖形、影像等。其中，瀏覽器與伺服器之間的溝通方式是使用 HTTP(Hyper Text Transfer Protocol)[1][17]的通訊協定。如圖 1.2 所示，HTTP 是一個應用層通訊協定，它很容易整合其它的通訊協定，如我們可在瀏覽器上使用 FTP(File Transfer Protocol)等功能。瀏覽器透過 Request 的方式向 Web 伺服器要求服務，Web 伺服器使用 Response 方式回應瀏覽器要求。就是這樣一來一往的方式建立了客戶端(Client)與伺服端(Server)之間的互動模式。HTTP 支援多種不同文件格式，如普通的 ASCII 文件為 text/plain，圖片有 image/jpeg 或 image/gif 等用法。HTTP 的特性是無狀態(Stateless)的連線方式。也就是同一個連線上本次存取資料的動作並不會記住上次存取資料的任何相關資訊。換句話說每個存取動作均為獨立的處理。所以如果在一個連線上要記錄此連線相關資訊，則必須使用一些特殊的方法處理，如使用 Cookie。HTTP 採用的 HEAD 會說明傳送的內文的相關資訊以及說明解釋內文的方法。這個方式形成 HTTP 的重要特色。因此，HTTP 的應用式目前網際網路上最常使用的通訊協定之一。



圖表 1-2 HTTP 通訊協定堆疊(Protocol Stack)

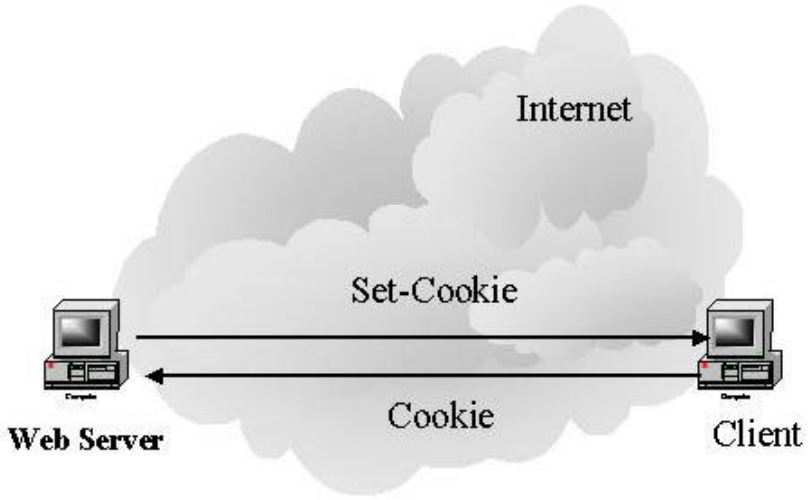
1.2 庫記(Cookie)

伺服器可將有關客戶端的一些相關資料設定到客戶端的硬碟當中，以便當下次客戶端再度連線進入 Web 網站時能夠藉由此條連線所攜帶的資料進行處理，如身分認證等功能，這些資料我們稱之 Cookie(如圖 1.3)[2][3][4][5][6]。Cookie 大都被使用在電子商務中，尤其是網路購物。如將客戶的姓名記錄在 Cookie 中，當下次客戶再度連線時直接攜帶此 Cookie 姓名資料進入網站，就可直接顯示出顧客姓名於首頁畫面，不需使用者作任何 Key In 動作。其實，這樣的 Cookie 資訊可得到一個非常好的優點，那就是將一些非機密性的資料記錄在客戶端上，客戶連線時攜帶此 Cookie 資料直接提供 Web 網站使用，可節省許多客戶輸入這些資料的時間。



圖表 1-3 Cookie 記錄內容範例

在瀏覽器與伺服端對 Cookie **的互動分為兩種主要的方式(如圖** 1.4)**。第一種稱為**”Cookie”**，主要由瀏覽器攜帶** Cookie **資料的動作。第二為**”Set-Cookie”**，主要為** Web **網站將** Cookie **資料設定到客戶端硬碟的動作。**

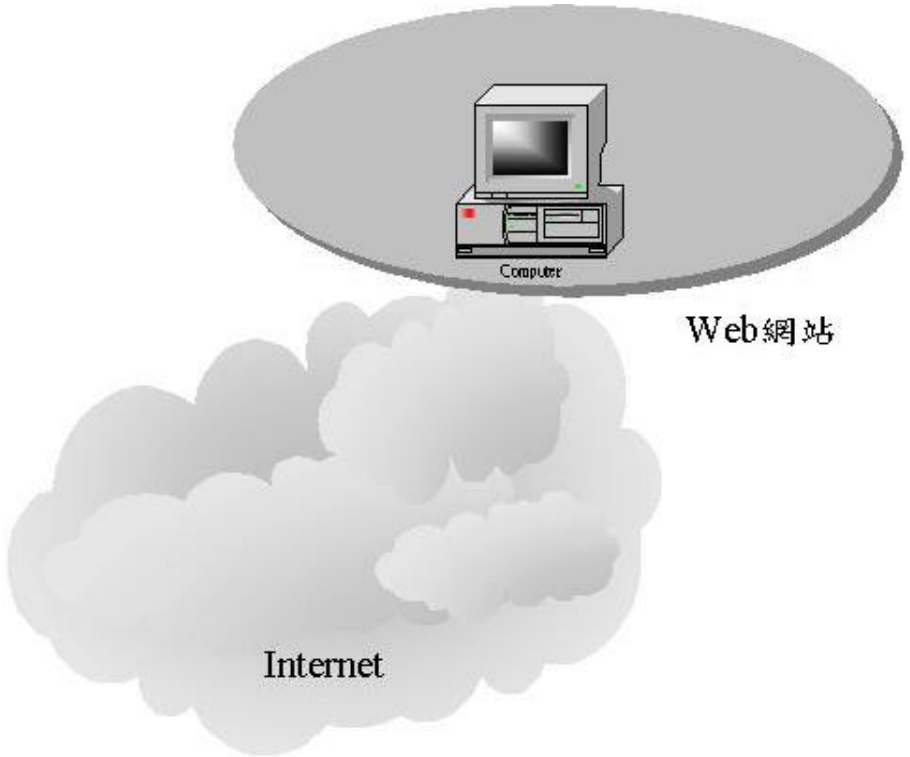


圖表 1-4 Cookie **的存取方法**

1.3 負載平衡的 Web 代理伺服器(Load Balance Web Proxy)

由一台網站服務客戶(如圖 1.5)如果同時上線數不多則沒有什麼問題。但如果有數十萬條連線，則需有效的分散這些連線的客戶才能提供更好的服務品質(如圖 1.6)。負載平衡(Load Balance)[12][13][14]的概念就是希望將大量的連線數分散在不同的電腦，這樣就可提高服務品質。其中，如何分配連線的方法有許多種，最常見的有四種

1 輪詢法則(Round Robin)
2 最少連線數法則(Connection Leastly Used)
3 加權式輪詢法則(Weighted Round Robin)
4 加權式最少連線數法則(Weighted Leasely Used)

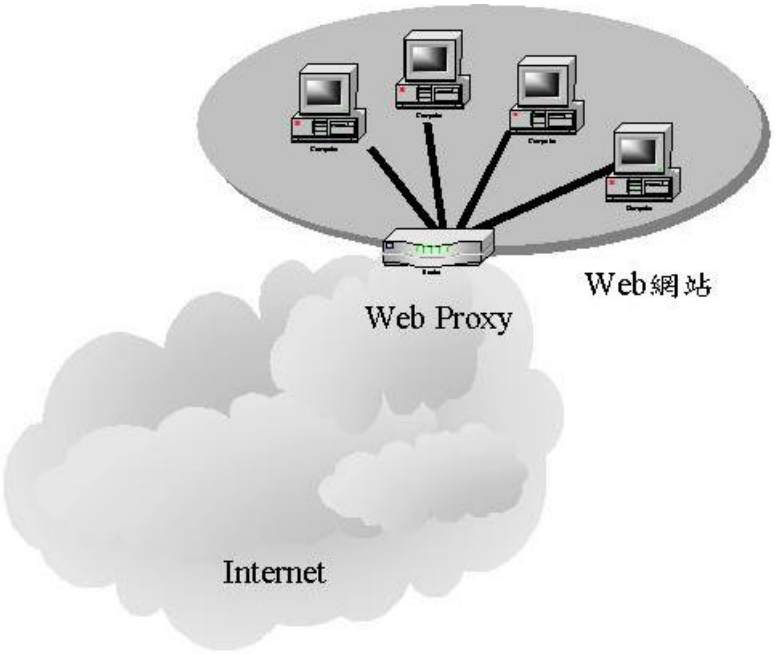


圖表 1-5 一台電腦服務客戶連線

當客戶端要求連線時，負責接收連線並作負載平衡工作的硬體我們稱為 Web Proxy。因為它具有轉接的功能又只提供 Web 服務，所以稱之為 Web Proxy。Web Proxy 主要的工作有三點:

1 轉送(Forwarding)功能
2 過濾(Filtering)功能
3 快取(Caching)功能

透過在 Web Proxy **上提供負載平衡的功能，將使網站能提供出更好更迅速的服務品質。**



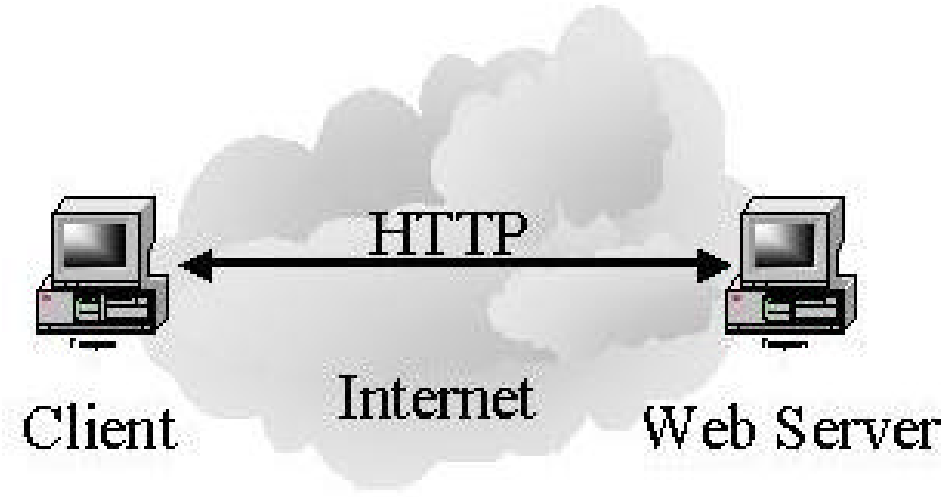
圖表 1-6 **透過** Web Proxy **由多台電腦服務客戶連線**

1.4 論文大綱(Outline of the Thesis)

在本論文中我們分為五章來介紹以 Cookie 為基礎的負載平衡的 Web 代理伺服器。第一章作整個論文的簡介。第二章介紹 HTTP 通訊協定及相關主題。第三章說明本論文所使用的機制。實作說明將在第四章介紹。最後第五章我們做總結。

2. Hyper Text Transfer Protocol

HTTP(Hyper Text Transfer Protocol)**是應用層通訊協定，被採用為網際網路上存取** Web **網站的通訊協定。透過** HTTP **使得我們在網路上傳送文字、圖片、影像等有了標準的方法。有了** HTTP **在網路上的的資訊交換就更多采多姿(如圖** 2.1)。



圖表 2-1 **客戶端透過** HTTP **與** Web **伺服器通訊**

2.1 HTTP 的交易(HTTP Transaction)

當打開瀏覽器打入網址(URL)後進行連線，客戶端就送出請求(Request)給伺服端要求連線。伺服器在客戶端的連線之後便送出首頁的資料給客戶端。接下來我們來說明客戶端送出的請求(Request)及伺服端作出的回應(Response)。

(一) 客戶端請求(Request)

GET / HTTP/1.1
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg,
application/vnd.ms-powerpoint, application/vnd.ms-excel, application/msword, */*
Accept-Language: zh-tw
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 5.01; Windows NT 5.0; DigExt)
Host: 140.114.89.72
Connection: Keep-Alive
Cookie: Password=72; UserName=test

一般來說客戶端送出的訊息如上。再第一列的敘述列中說明客戶端使用 GET 的模式傳送資料給伺服端並且要求從預設站台的主要目錄(/)取得資料，使用的 HTTP 的通訊協定為 1.1 版。第二列是瀏覽器告訴伺服器可接收的文件格式。第三列由瀏覽器告訴伺服器所希望使用的語言，因為伺服器有可能提供多國語言。第四列指出瀏覽器可以接受資料的壓縮模式，可使用 gzip 或 deflate 演算法。第五列由 User-Agent 開頭，代表使用者的瀏覽器版本為 4.0 版。在括弧中說明實際上使用的版本為 5.01 版，NT 環境的作業系統。第六列說明客戶端的 IP 位址。第七列說明伺服端將保持 TCP/IP 的連線直到被通知關閉為止。最後第八列為瀏覽器所送出的 Cookie 資料。其中包含使用者名稱(UserName)及密碼(Password)。由上述的訊息得知，在瀏覽器送出的資料之前會以此訊息作為前導說明瀏覽器的基本狀態。因此伺服端可提供較佳的服務給客戶端。

(二) 伺服端的回應(Response)

HTTP/1.1 200 OK
Date: Mon, 23 July 2000 10:33:12 GMT
Server: Apache/1.3.6 (Unix)
Last-Modified: Fri, 20 July 2000 15:11:24 GMT
Etag: “2d2fc-768-782g4ed3”
Accept-Ranges: bytes
Content-length: 432
Connection: close

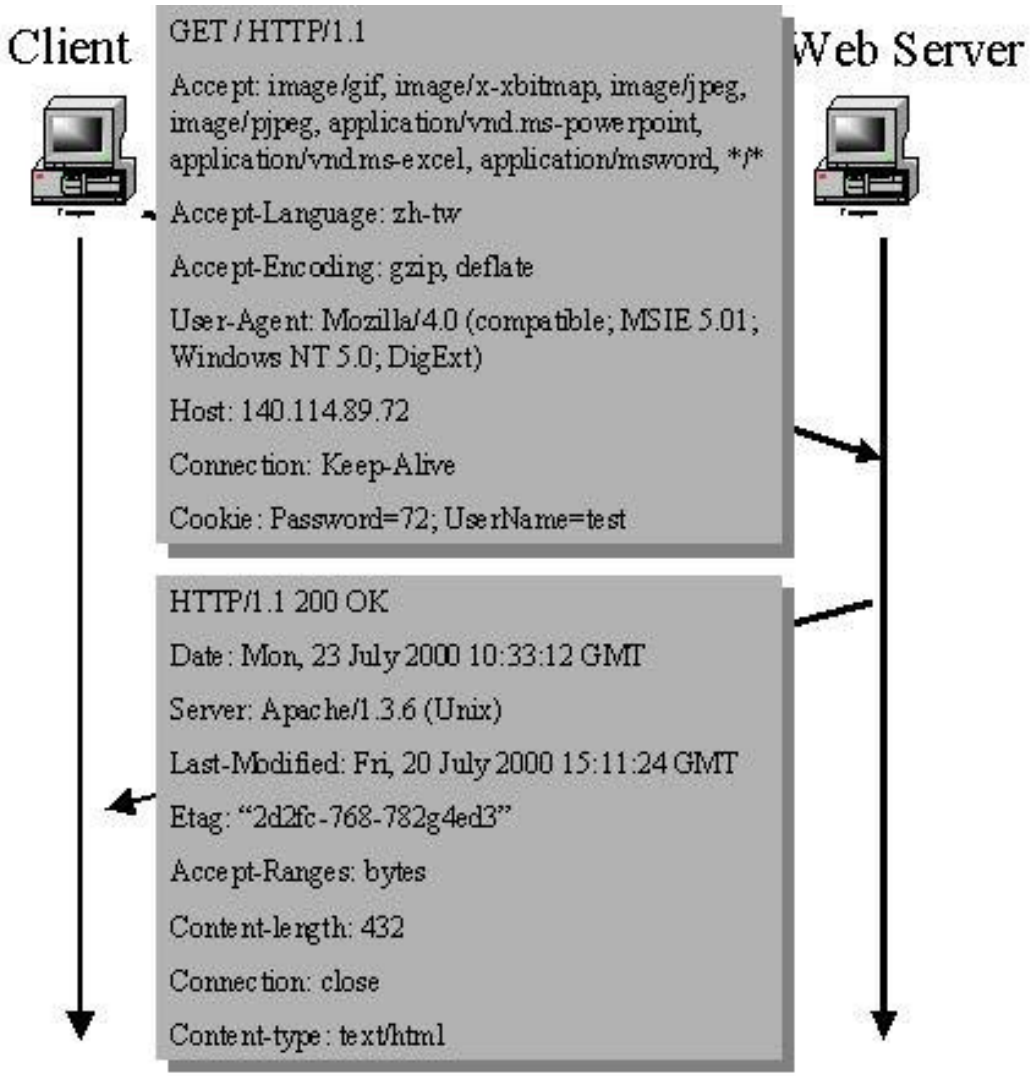
```
Content-type: text/html
Set-Cookie: Password=72; UserName=test

<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=big5">
<title>Cookie Example</title>
</head>
<body>

<h1>Cookie Example</h1>
<hr>
<form method="post" action="reg.asp" name="frmUserData">
                    請輸入姓名： <input type="text"
                                 name="txtUserName"
                                 size="20"
                                 value=""><p>
                    請輸入 Password： <input type="text"
                                 name="txtPassword"
                                 size="40"
                                 value=""><p>
<font color=red>請填寫以上表單的資料，並按下「登入」按鈕</font><p><input
type=submit value='登入'>
<input type=reset value='重寫'>
</form>
</body>
</html>
```

當伺服器收到瀏覽器的要求時，伺服端作出回應的動作。第一列指出伺服端使用 HTTP/1.1 **的版本以及回應碼為** 200**，**200 **的意思為找到了首頁的資料，在最後**"OK"**的意思為首頁資料將傳送給客戶端。第二列說明伺服器現在的時間。第三列說明伺服端所使用的** Web **伺服器的環境平台為** Apache/1.3.6 **在** Unix **系統上執行。第四列最近一次客戶端要求修改伺服端文件的時間。一般來說此項目的為處理有關** Cache **的問題上使用。第五列說明一個** entity tag**。此項用法為伺服器提供一個唯一的識別號給客戶端，此項目的用在** Caching **上。第六列說明伺服端有能力一次傳送部分資料給瀏覽器，不見得一次傳送全部的資料。此種方式有利於傳送資料庫的資料，如以記錄**(Record)**為單位傳送而不是一次傳送完所有資料庫內的資料。第七列說明伺服端傳送的內容長度。第八列說明此連線將在傳送完**

資料之後關閉。第九列說明資料內容的文件型態，text/html 指出此內容為 HTML 的文件。最後第十列為伺服端設定 Cookie 資料給客戶端。當伺服器作出反應(Response)時會加上此訊息作為回應標頭，使得客戶端能夠順利的解讀所收到的資料(如圖 2.2)。



圖表 2-2 客戶端與伺服端之間的要求(Request)與反應(Response)

2.2 標頭(Heads)

在上述的 HTTP Transaction[7]中我們可以發現，無論客戶端或伺服端在作出要求或反應時，在傳送資料之前都會先行送出標頭(Head)。這些標頭最主要說明隨之而來的資料應如何解讀並說明瀏覽器或伺服器的一些基本特性。而這些標頭都是以純文字的方式跟著 HTTP 的通訊協定之後傳送。接下來我們說明 HTTP 的四種標頭型態。

1. 一般標頭(General Header)
2. 要求標頭(Request Header)
3. 回應標頭(Response Header)
4. 個體標頭(Entity Header)

1. 一般標頭(General Header)

一般標頭說明一般性的事務，可被瀏覽器及伺服器雙方使用。如日期、連線狀態等。如 Cache-Control、Connection、Data 等指令。

2. 要求標頭(Request Header)

由瀏覽器使用。主要說明瀏覽器的架構及所希望伺服器傳送的文件格式。如 Accept-Charset、Accept-Language、Cookie 等指令。

3. 回應標頭(Response Header)

由伺服端使用。主要說明伺服端的架構及 URL 的相關資訊。如 Accept-Ranges、Age、Etag 等指令。

4. 個體標頭(Entity Header)

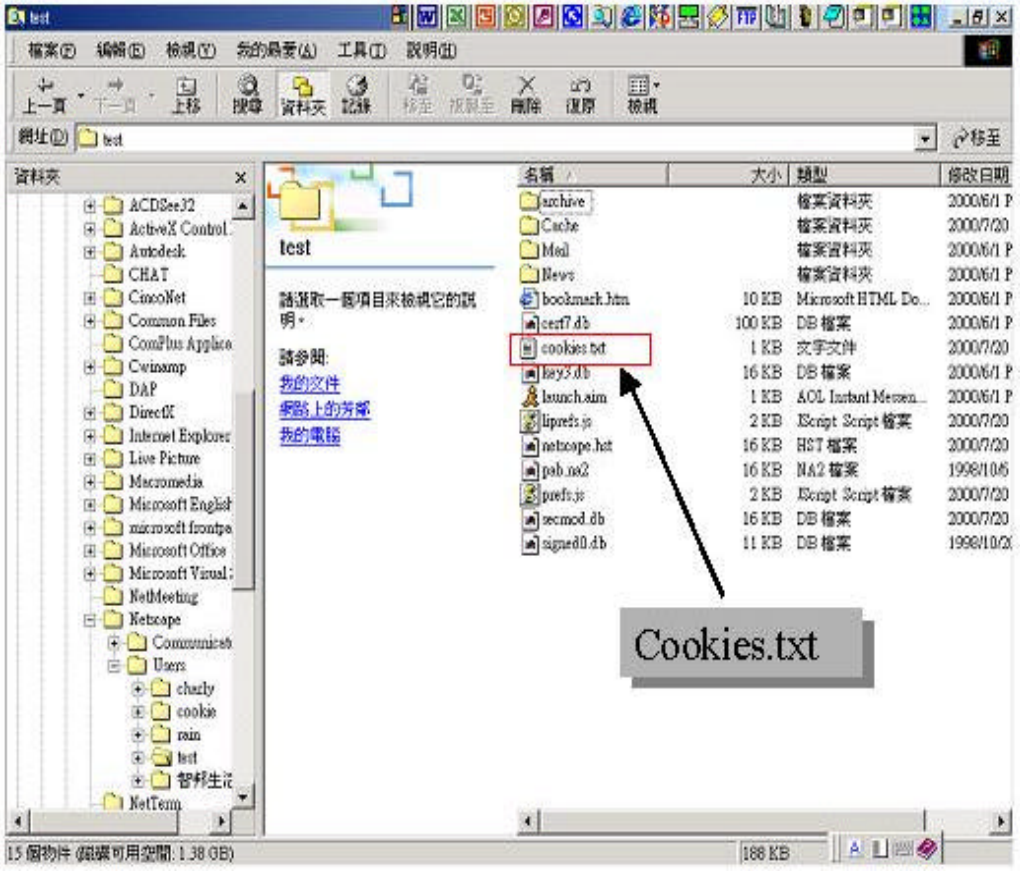
主要說明瀏覽器與伺服器之間傳送文件的格式。此種格式最常被伺服器使用來回應客戶端的 GET 或 POST 要求動作。如 Content-Encoding、Content-Length、Content-Type 等指令。

2.3 庫記(Cookies)

Cookie[9][10]是由伺服器設定給客戶端硬碟的一些有用的資料。Cookie 的使用方法分成兩種: 第一是"Cookie"，第二是"Set-Cookie"。"Cookie"的動作是由客戶端所使用。主要將存在硬碟中有關網站的 Cookie 資料內容送往伺服端。另外"Set-Cookie"由伺服器所使用的方法，主要將一些資料設定到客戶端的硬碟中。Cookie 資料的設定與所使用的瀏覽器有關。我們以最常見的 Netscape 領航員(Navigator)及 Microsoft 的探險家(Explorer)來作說明。

1. **領航員**(Navigator)**的做法**

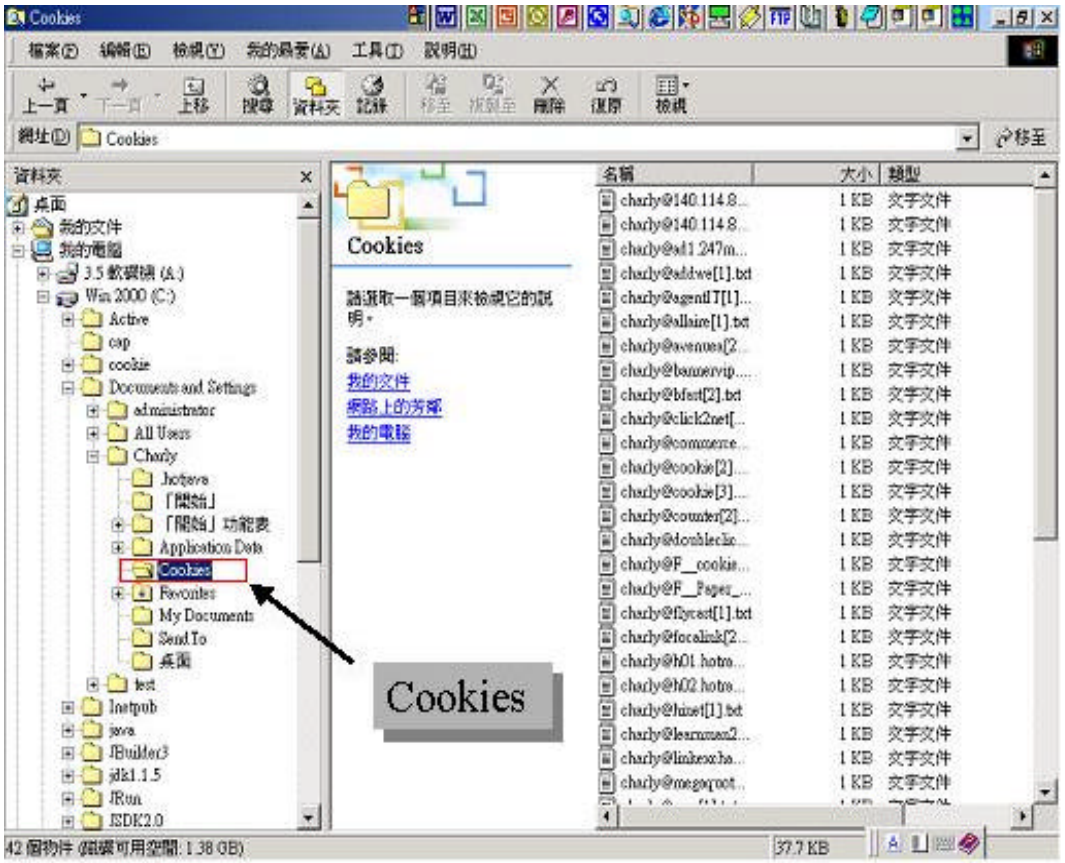
當開啟領航員(Navigator)**瀏覽器時會被要求選擇使用哪一個帳號進入瀏覽器。當初在瀏覽器建立使用者帳號時，瀏覽器就替每一個使用者建立個別的目錄及檔案來記錄每個使用者自己的資料。如** Navigator **的目錄如建立在** C:\Program Files\Netscape **，則使用者** test **將被建立一個使用者目錄** C:\Program Files\Netscape\Users\test(**如圖** 2.3)**。其目錄下的** Cookies.txt **檔案即為伺服端記錄** Cookie **資料給使用者** test **的檔案所在。**



圖表 2-3 Navigator **記錄** Cookie **的檔案**

2. **探險家**(Explorer)**的做法**[8]

Explorer **開啟時並不會讓使用者登入帳號，因此使用登入作業系統時的帳號為分辨** Cookie **的方法。如使用** charly **帳號登入** Win 2000 Profession **作業系統，在** C:\Documents and Settings\test\Cookies(**如圖** 2.4)**有存放** charly **使用者的** Cookie **資料。**



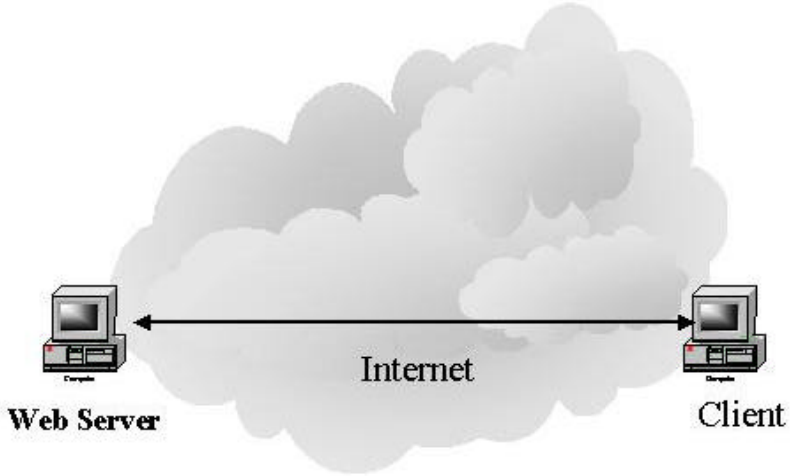
圖表 2-4 Explorer **記錄** Cookie **的目錄**

3. 以 Cookie 為基礎的負載平衡 Web 代理伺服器 (Cookie-Based Load Balancing Web Proxy)

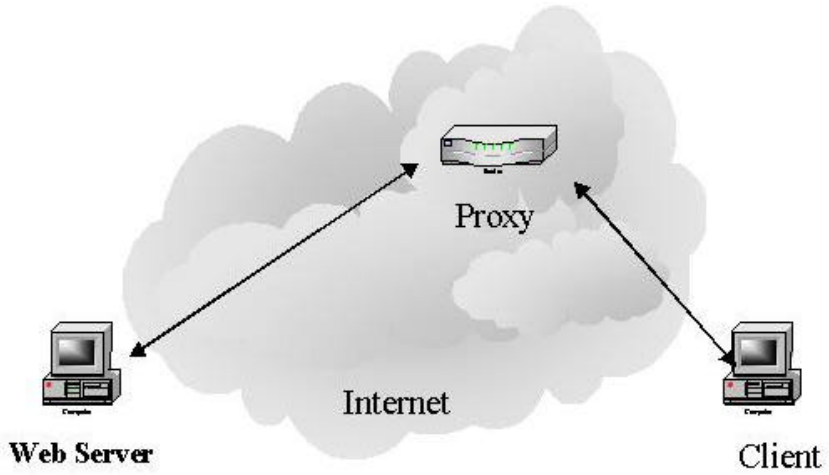
在本章中，我們將說明以 Cookie 為基礎的負載平衡(Load Balancing)Web 代理伺服器(Proxy)。首先，我們先描述目前的 Proxy 的模型。然後我們將提出負載平衡的 Web 代理伺服器的模型。接下來將說明 Cookie 在本系統中所扮演的角色。四種常被用來作為負載平衡的方法將說明比較。最後，我們將提出利用 Cookie 作為身分認證及優先等級來運用在 Web Proxy 上作為平衡負載的機制。

3.1 以 Cookie 為基礎的負載平衡 Web 代理伺服器模型(Cookie-based Load Balancing Web Proxy Model)

一般來說，在網際網路上我們使用瀏覽器連上網站時可分為直接與網站連線(如圖 3.1)及透過代理伺服器(Proxy)連線(如圖 3.2)兩種。其中，為考慮連結速度等因素，我們會在瀏覽器上設定使用代理伺服器的選項，藉由代理伺服器我們可享受較快的連線服務。



圖表 3-1 瀏覽器直接連線網站

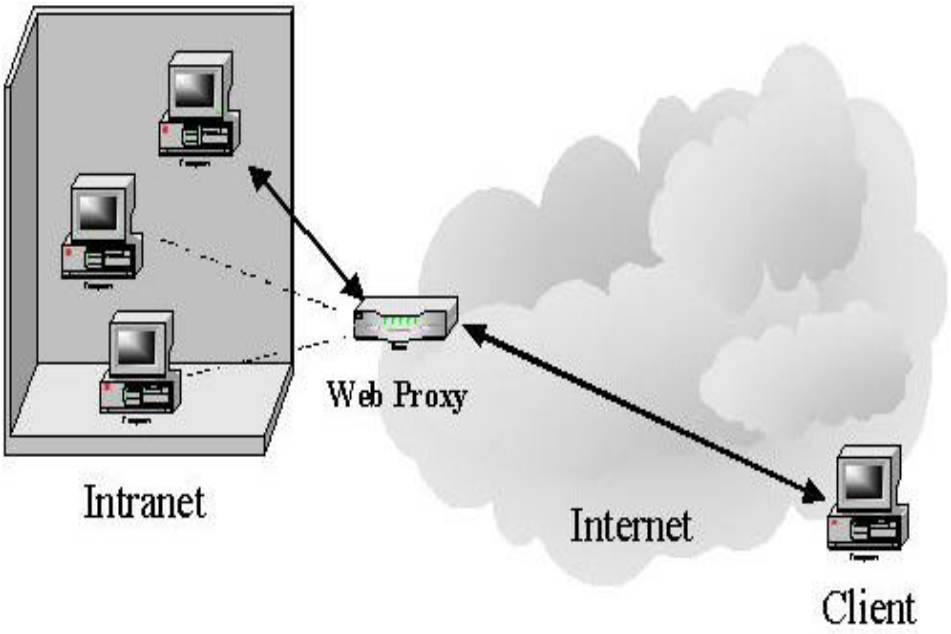


圖表 3-2 網際網路 Web Proxy

雖然代理伺服器被廣泛應用於網際網路之上，但也有不少的企業自行建立代理伺服器於 Intranet 中(如圖 3.3)，除了進行代理功能之外，也加上了防火牆的功能，以杜絕非法的使用者入侵而使 Intranet 癱瘓。除此之外，還有許許多多的其他的附加價值的功能陸續的與代理伺服器整合而形成一套更完整的服務系統，如頻寬控制功能等。在 Web 伺服器與瀏覽器進行連線交換資料時，伺服器可將這些有用的資訊紀錄在客戶端中，使得這些資訊可在以後客戶端再次連線時由伺服端飲用而提供出更多樣化的服務，這些有用的資訊我們稱之 Cookie。在本系統當中，我們特別將代理伺服器應用於 Intranet 內中，並且加強他的功能。我們使用了 Cookie 作為身分認證及客戶等級劃分，並且加上了負載平衡(Load Balancing)的功能，使得透過本系統的服務之後可達成將客戶端的連線可依其身分等級不同而連線至指定的電腦上，而可獲取不同等級的服務。

3.2 庫記(Cookie)

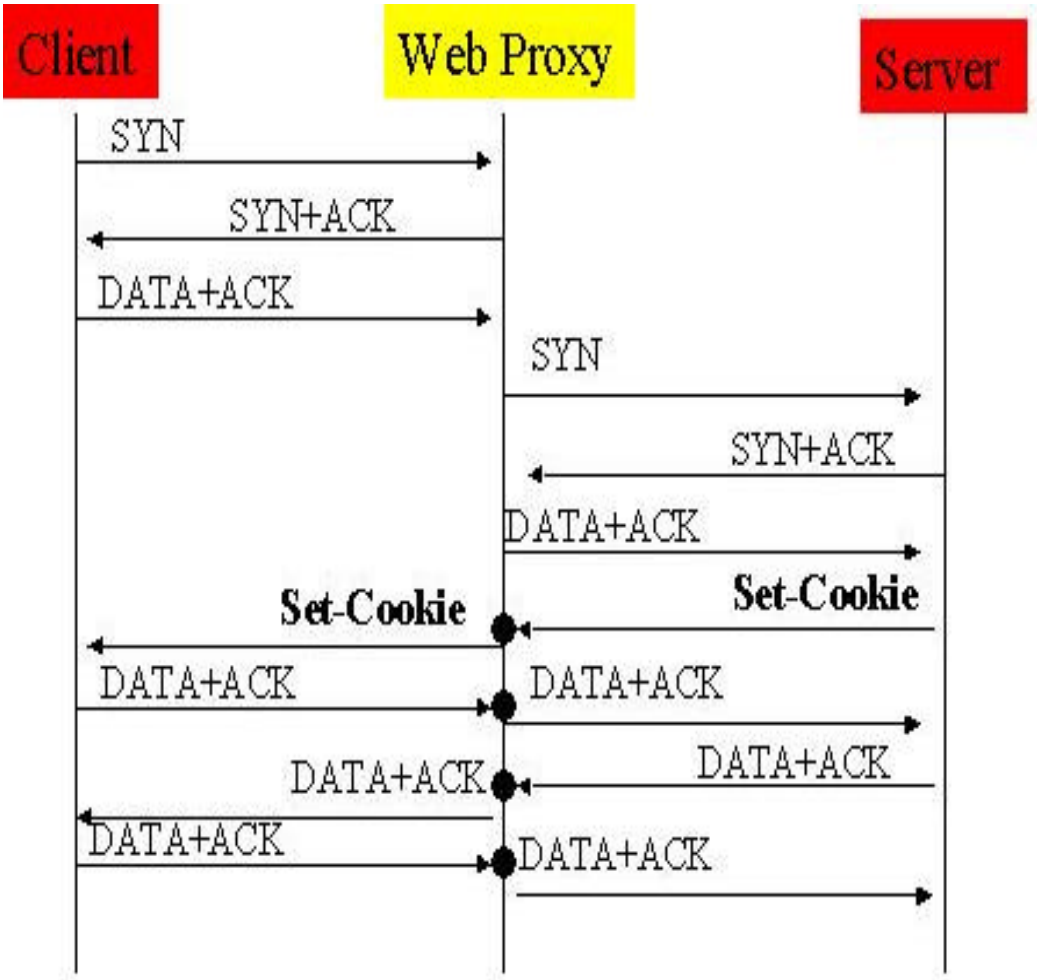
Cookie 是在網際網路上 Web 伺服器用來將一些有用的資料記錄在客戶端的方法。其中最常用來記錄有關身分相關的資訊。在本系統中我們提出使用 Cookie 客戶端的身分及優先等級。使得客戶端再次連線時可透過 Cookie 直接由 Web Proxy 進行身分認證及優先等級確認，於資料庫中比對 Cookie 資料找到對應的網站，進而進行轉送服務的工作。



圖表 3-3 以 Cookie 為基礎的 Web Proxy 模型

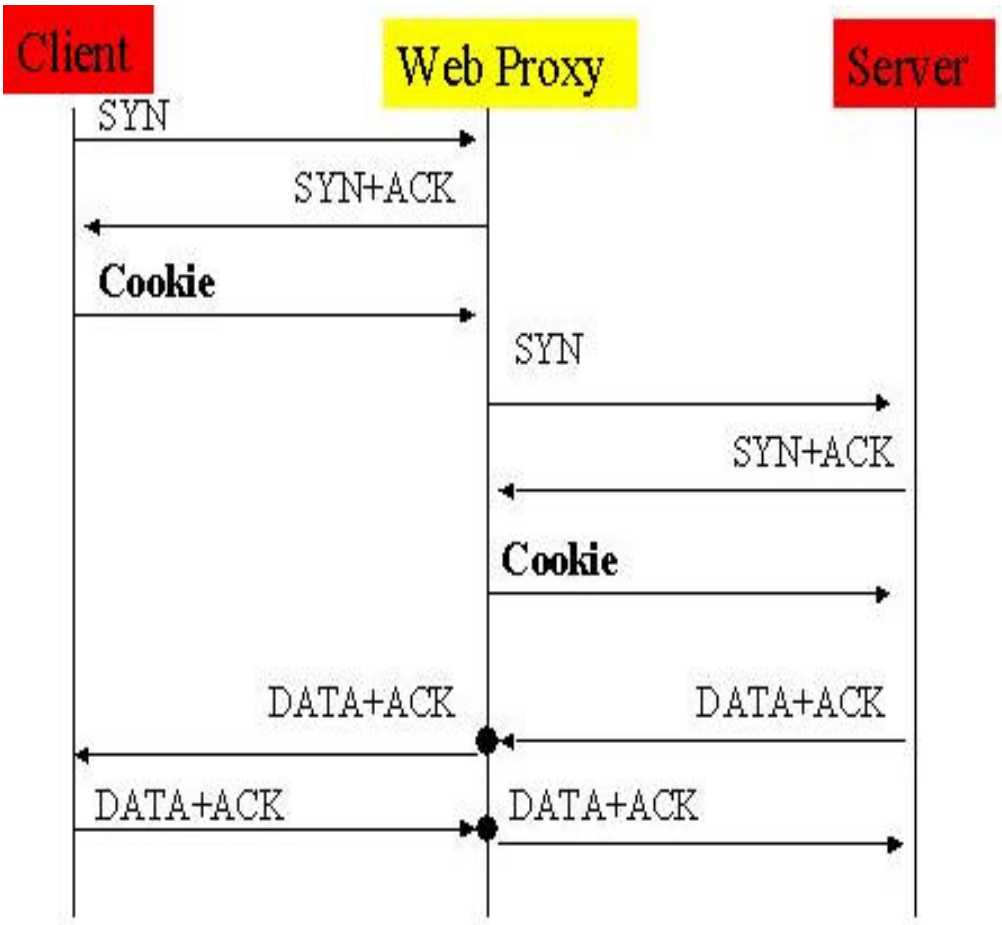
在 Cookie[22][24][25][26][27]的使用上分成由伺服端在客戶端設定資料的動作，我們稱之 Set-Cookie。另外由伺服端要求客戶端傳送已存在在客戶端上的資料稱為 Cookie。接下來我們說明在網際網路的 TCP/IP 通訊協定之上是如何透過我們的 Web Proxy 系統進行 Cookie 資料的處理。

如圖 3.4 所示，一開始客戶端與伺服端是透過單一的網址與 Web Proxy 連線。當客戶端的第一次連線上 Web Proxy 時，並沒有在自己的硬碟記錄中找到 Web Proxy 所記錄的 Cookie 資料，所以在建立 TCP/IP 的連線之後並無攜帶任何有關的 Cookie 資料。因此，在 Web Proxy 中並沒有擷取到任何 Cookie 的資料，故將此客戶端連線轉向到預設的站台。此時，預設站台需確認客戶端身分之後做出 Set-Cookie 的動作，將客戶端的身分等級透過 Set-Cookie 的動作記錄到客戶端上。此時，客戶端就擁有此網站的身分等級 Cookie 的資料。在下次連線時會攜帶此 Cookie 資料進行身分等級辨識。



圖表 3-4 Set-Cookie **時序圖**

在取得網站的 Cookie **資料之後再度連線時，如圖** 3.5 **由客戶端透過** TCP/IP **的連線之後，客戶端會自動由本身的** Cookie **資料中取得有關此網站的相關資訊並攜帶進入** Web Proxy**。此時** Web Proxy **擷取由客戶端送出的** Cookie **資料並進行資料庫比對動作以確認客戶端的身分等級。在比對結束之後，**Web Proxy **就依照搜尋結果將此連線轉向到指定的** Web **網站上，並且將此** Cookie **的資料轉送到** Web **網站上。此後，客戶端與** Web **網站就可以透過** Web Proxy **進行資料互動交換。**



圖表 3-5 Cookie **時序圖**

3.3 負載平衡(Load Balancing)

在 Web Proxy 中，當收到客戶端的連線時，我們將此連線轉至預設的 Web 站台，以提供網站服務功能。而此時客戶端並不知道究竟是哪一台網站被選擇作為連線的標的。而 Web Proxy 所扮演的角色即為在資料庫上依據客戶端的身分等級比對相對應的網站，並進行連線轉送的功能。而為了提高服務速度即分散用戶端的流量，於是我們使用多台網站來服務同一身分等級的客戶，如此可提升服務品質。因此我們在 Web Proxy 上提供了負載平衡(Load Balancing)的功能。將同等級的客戶連線分散在數台網站上服務。而較常見的負載平衡的方法有四種;

1. 輪詢法則(Round Robin)
2. 最少連線數法則(Connection Leastly Used)
3. 加權式輪詢法則(Weighted Round Robin)
4. 加權式最少連線數法則(Weighted Leasely Used)

3.3.1 輪詢法則(Round Robin)

在 Web Proxy 擷取到客戶端的 Cookie 資料之後，我們可找出此客戶的身分等級可服務的網站數目及網址。當有多台網站可服務此客戶時，我們可採用輪詢法則。此方法將客戶端連線依照可服務的網站台數輪流服務。如有甲、乙、丙三台網站可服務客戶等級 A，第一條等級 A 的客戶將被網站甲服務，接下來的第二條客戶等級 A 的連線將由網站乙來負責。同理第三條進來的客戶等級 A 將被網站丙服務。接下來又輪到網站甲，以此類推。此種方式為網站平均負擔客戶端的服務。

3.3.2 最少連線數法則(Connection Leastly Used)

因為網站服務的客戶時間不定，因此有些客戶可能使用網站的時間短，有些卻很長。為使網站真正可平均負擔客戶端的連線流量，因此最少連線數法則(Connection Leastly Used)將尋找在可服務的網站中目前連線數最少的網站進行服務。此方法在 Web Proxy 上需記錄哪些客戶端連線已經關閉。因為在某些情形下無法獲知客戶端是否繼續使用連線，所以在 Web Proxy 上我們設定客戶端連線 Time out 時間。若客戶端在此時間內未能再度存取網站資料，則中斷此連線。因此此方法可提供較好的負載平衡功能。

3.3.3 加權式輪詢法則(Weighted Round Robin)

在輪詢(Round Robin)法則中，我們採用輪流的方式與提供同等級的網站進行連線，此方法乃唯一非常公平的法則。而因應不同服務的需求，在輪詢法則上再提供加權(Weight)功能，使得在相同等級不同網站可依據其加權可有不同的連線服務。如假設有三台網站 A、B、C。其中 A 的加權為 1。B 的加權為 2。C 的加權為 3。則新進入的連線第一條會接至 A。第二及第三條連線則轉接至 B。第四至第六條則轉接至 C。以此循環輪流。此即為加權式輪詢法則。

3.3.4 加權式最少連線數法則(Weighted Leasely Used)

在考慮將新連線轉接至目前連線數最少的網站時，我們加上了加權(Weight)的功能，使得相同等級內的數個網站可依照不同的因素提供不同的服務。如有三台同等級的網站 A、B、C。A 有一條連線及加權為 1。B 有兩條連線及加權為 3。C 有三條連線及加權為 4。則 A 的計數為 A 的連線數乘以 B 的加權在乘以 C 的加權得到 12。同理 B 可得計數 8。C 可得計數 9。比較之後取出最小的計數為 B。因此，新連線將轉接至網站 B。

3.4 Web 代理伺服器(Web Proxy)

當我們使用瀏覽器時設定代理伺服器時是希望首頁顯示資料的速度能快些。而 Web 代理伺服器[23]能夠使客戶端的連線得到較快速的反應，進而使得瀏覽器下載首頁速度相對較快。我們將 Web 代理伺服器的功能區分成下列三項功能:

1. **轉送**(Forwarding)**功能**
2. **過濾**(Filtering)**功能**
3. **快取**(Caching)**功能**

3.4.1 轉送(Forwarding)功能

當 Web 代理伺服器接受到從客戶端的連線時，最主又的功能要將客戶連線依照所希望連線的網址(URL)進行轉送功能。而將網站所做的反應也轉送至客戶端。在本系統中我們不僅進行轉送功能，而能透過客戶端連線上的 Cookie 資料進行分級，並且進行負載平衡(Load Balancing)的功能。而與傳統的代理伺服器(Proxy)最大的不同點是一般網際網路使用代理伺服器均是要將客戶端連線透過代理伺服器進行轉接，而本系統之 Web Proxy 在客戶端的概念中即為目的網址。也就是客戶端連線所使用的目的地 IP(或 URL)就是 Web Proxy。透過單一位址的服務，使用者可因身分等級不同而在 Web Proxy 上被轉接到相對應的網站。如同公司對外使用一個代表的電話號碼，當有超過一個電話 Call in 時則電話系統自動跳號並轉至指定的分機是一樣的道理。而在 Web Proxy 後方將有數台至數十台以上的電腦進行服務客戶的功能，一但利用 Cookie 確認身分等級後，便可轉接至指定的網站加以服務。

3.4.2 過濾(Filtering)功能

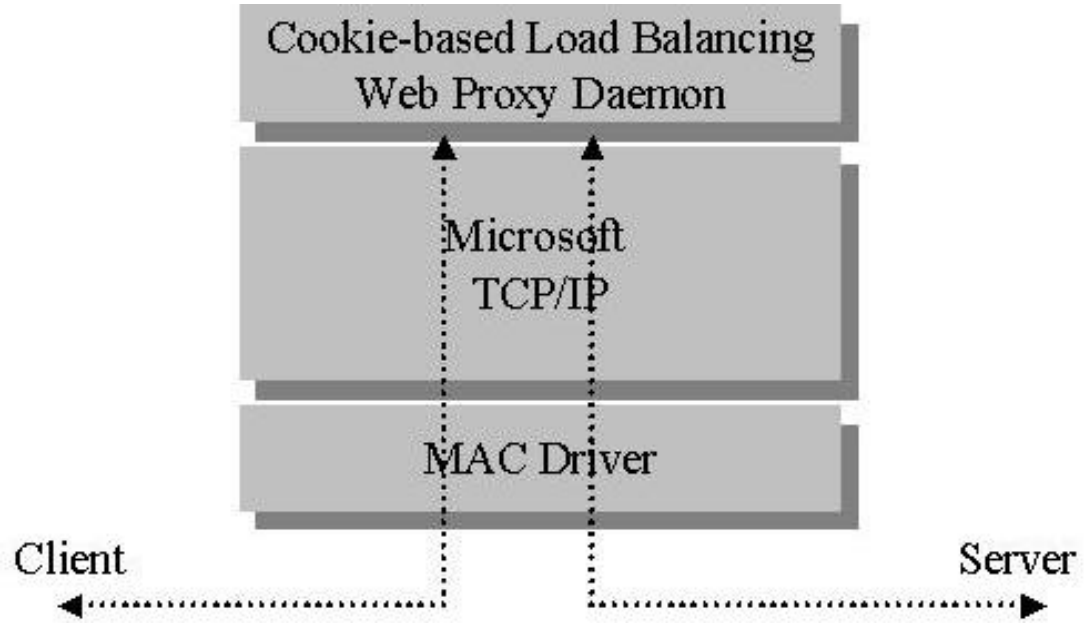
當 Proxy 收到一個新連線的時候會去判斷此連線是否為有效連線，如果此連線身份不明，一般 Proxy 如有防火牆功能，會將此連線關閉而拒絕服務。而我們的 Web Proxy 判斷身份的依據是客戶端連線進入時是否有攜帶 Cookie 資料。因為 Cookie 是我們判斷此連線身分優先等級的依據。如沒有攜帶 Cookie，則將此連線轉接到預設的網站上進行身分登入，則下次再度連線時就具有身分依據。如果有攜帶 Cookie 資料，則依據資料庫比對轉接至對應的網站進行服務。因此，過濾功能在本系統中是採用轉接至預設伺服器進行身分登入，如無身分則無法進行進一部的網站服務。

3.4.3 快取(Caching)功能

透過 Proxy 伺服器能夠得到更快速的服務主要原因是因為 Proxy 伺服器具有快取功能。在 Proxy 伺服器上許多連線常會存取同一網站資源。在上一個連線存取某一網站不久後另一條連線也要存取相同網站。因此Proxy只要在尚未 time out 的條件下即可將此網站的資料直接傳給新連線而不用再到指定的網站抓取資料。尤其此種方法用在存取外國網站時更容易見到其效益。在 Web Proxy 中，我們也提供快取功能，使得使用者能透過此一功能提供更為快速的服務。

3.5 以 Cookie 為基礎的負載平衡機制(Cookie-based Load Balancing Mechanism)

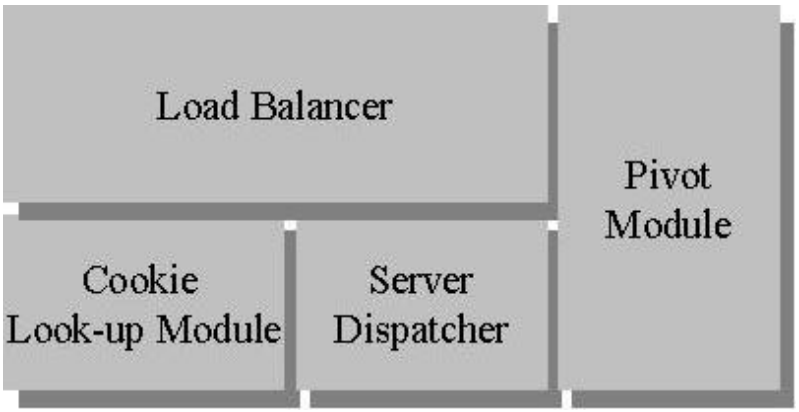
首先，我們先說明以 Cookie 為基礎的負載平衡機制。如圖 3.6 所示，Web Proxy 系統架構在 TCP/IP 之上。因為瀏覽器透過 HTTP (Hyper Text Transfer Protocol)通訊協定與網站連線。Web Proxy 先傾聽 80 埠，因為這是 HTTP 的預設埠。當有客戶端連線進入時，則 Web Proxy Daemon 會接收客戶的連線並進行轉接。而網站伺服端如有反應則也是透過 Web Proxy Daemon 進行轉接。



圖表 3-6 Web Proxy Daemon Protocol Stack

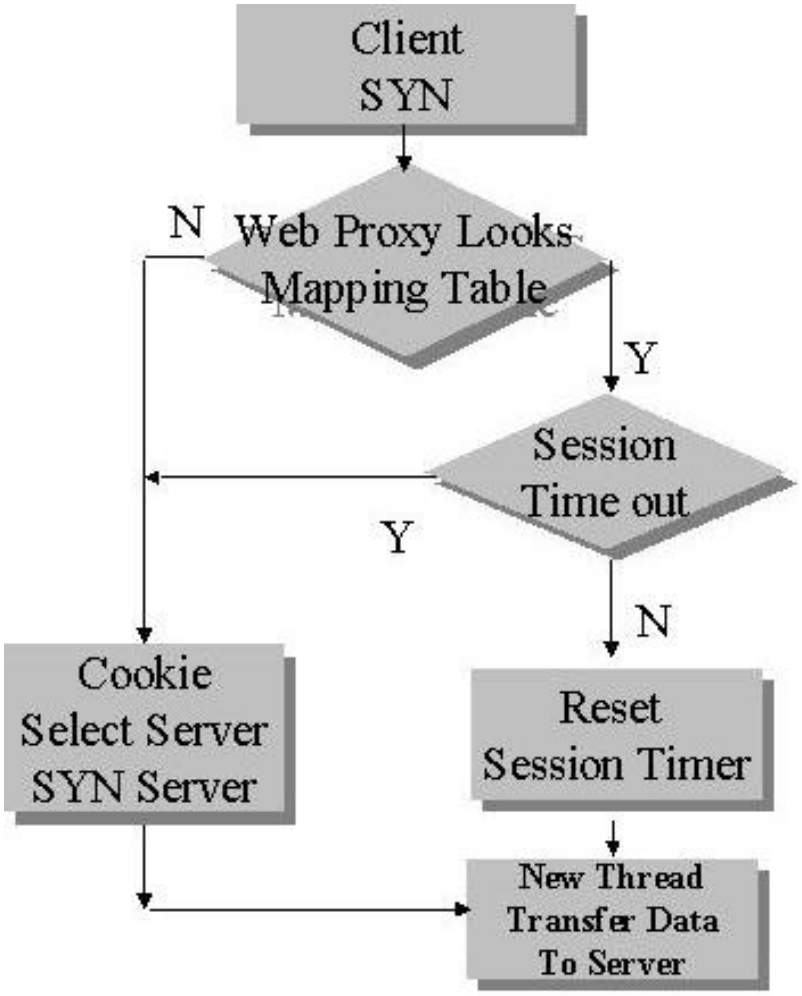
接下來我們將 Web Proxy Daemon **分成四個主要模組(如圖** 3.7)**。**

1. **庫記查詢模組**(Cookie Look-up Module): **負責向客戶端取得** Cookie **中的使用者等級，將此資訊傳向** Load Balancer**。**
2. **負載平衡器**(Load Balancer): **由** Cookie Look-up Module **取得等級後，在此等級的電腦群組中選取對應的** Web Server**。**
3. **伺服器派遣器**(Server Dispatcher): **由** Load Balancer **取得對應的** Web Server IP **後，負責與此** Web Server **進行連線。**
4. **樞紐模組**(Pivot Module): **在連線建立後負責進行客戶端與伺服端之間的資料轉送。**



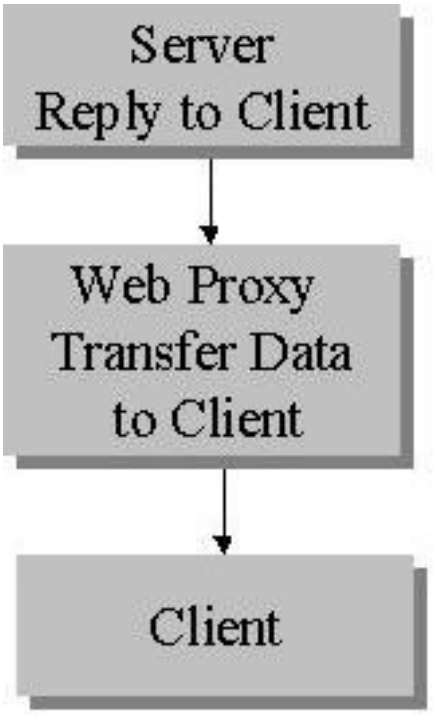
圖表 3-7 Web Proxy Daemon **模組**

在 Web Proxy Daemon 起動後，則本系統開始正式運作。如圖所示，我們將整個流程分成兩個部分: 一是客戶端經 Web Proxy 連線伺服端。二是伺服端經 Web Proxy 回應客戶端。在圖 3.8 中，客戶端使用瀏覽器透過單一 IP(或 URL)連線 Web Proxy，而此時在 Web Proxy 上正執行 Web Proxy Daemon 正傾聽 80 埠，等待客戶端連線。當 Web Proxy Daemon 接受客戶端連線的請求後，隨即檢查客戶端是否有攜帶 Cookie 資料，裡面記載客戶端的身分等級。如果沒有的話，則將此連線轉接至預設的網站服務。此網站可立刻要求客戶端進行身分登入。當登入手續完成之後，網站伺服端就會在客戶端上設定 Cookie 資料作為客戶端的身分等級。等到下次客戶端再度連線 Web Proxy 時就可攜帶 Cookie 資料作為身分辨識。然而當 Web Proxy Daemon 發現客戶端連線有攜帶 Cookie 資料時，此時剖析 Cookie 資料的內容並檢查此連線是否已經存在，如已經存在但連線尚未 time out 則將此連線繼續轉接至對應的網站繼續通訊。若連線尚未存在或者連線存在但已經 time out，則啟動負載平衡(Load Balancing)功能進行選擇適當的伺服器來服務。此時 Web Proxy Daemon 會產生一個新的執行緒(New Thread)來服務一條新連線。透過這樣的程序，客戶端可以與網站伺服端進行聯繫。



圖表 3-8 客戶端連線伺服端流程圖

另一方面如圖 3.9 所示，伺服端為回應客戶端的請求，所以必須回應客戶端。於是在 Web Proxy Daemon 必須記錄這條連線的客戶端相關資料，以便將回應資料能準確的送往對應的使用者。

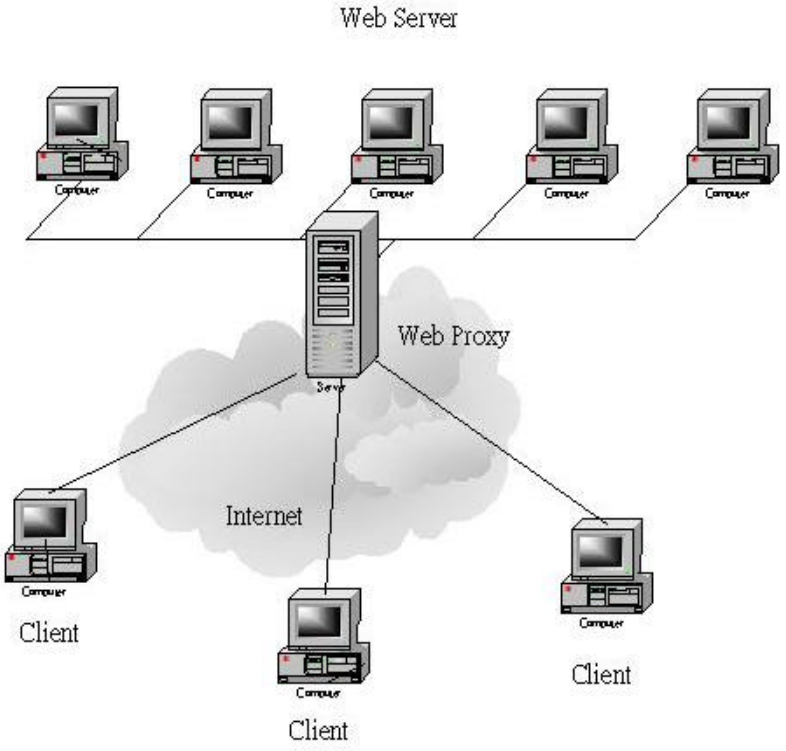


圖表 3-9 伺服端反應客戶端流程圖

4. 實作(Implementation)

在本章中，我們將介紹以 Cookie 為基礎的負載平衡的 Web 代理伺服器[27]並且以 JAVA 語言[18][19]進行實作。

4.1 環境介紹(Environment)

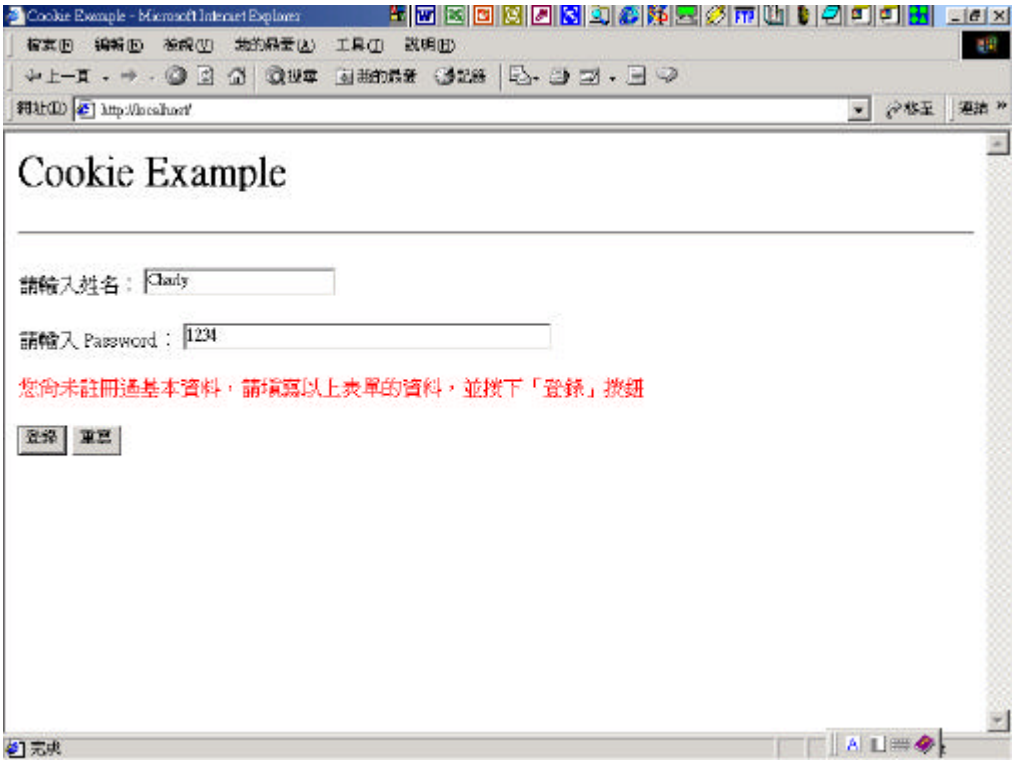


圖表 4-1 Web Proxy 環境架構

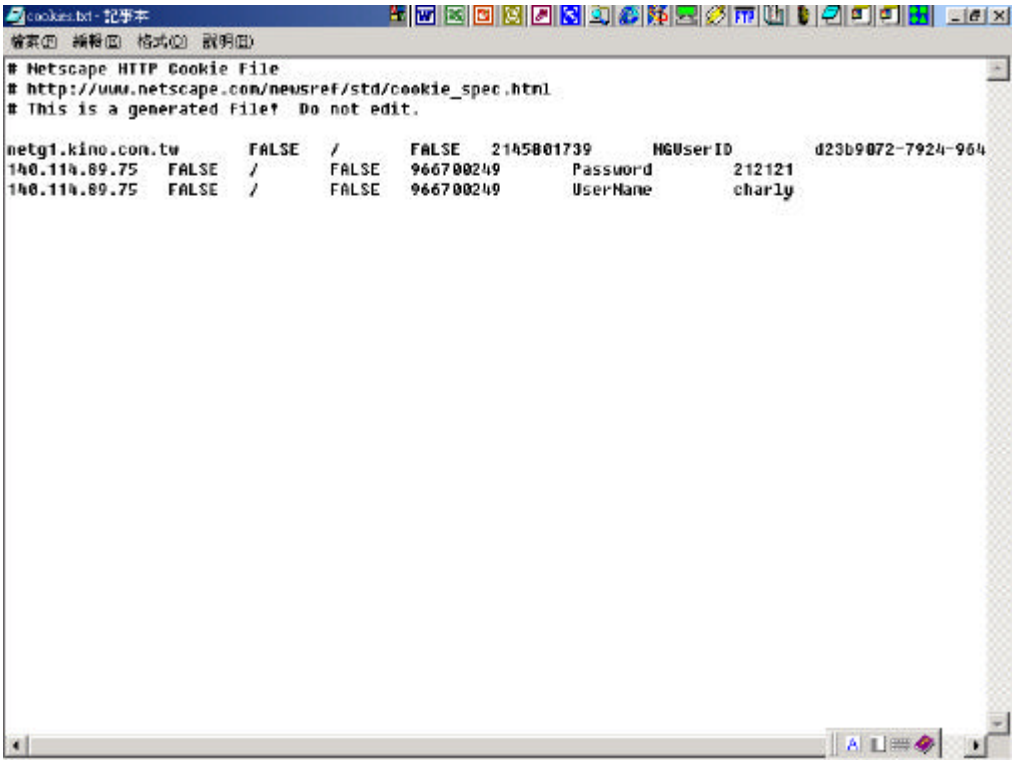
一般來說，Web Proxy 大都用在 Internet 上。可能放在網際網路上的任何地方。而客戶端只要設定好 Web Proxy 的 IP 位址之後就可以透過 Web Proxy 進行資料轉送的動作。在本系統中，如圖 4.1 我們將 Web Proxy 使用在 Intranet 或在區域網路上。在 Web Proxy 之後利用 LAN 連接了許多 Web 網站。而外面的使用者必須透過 Web Proxy 才可以存取這些 Web 網站。建置 Web Proxy 硬體我們使用 NT 伺服器上安裝 Web Proxy Daemon 軟體以進行服務。並在 NT 伺服器上安裝兩張網路卡。一張對外面的 Internet，另一張連接內部的網站。我們可痛過在網際網路上的任何一台電腦當作客戶端連線到本系統預設的網址(或 IP 位址)，此位址為 Web Proxy 的位址。這樣就可以透過本系統來進行服務。

4.2 庫記(Cookie)

Cookie 是用來辨識身分等級的資料。因此，Cookie 的設定及處理非常的重要。在本系統當中，我們在網站站台上設計了一個登入程式，此登入程式由客戶端進入之後登入自己的帳戶及密碼後完成(如圖 4.2)[21]。當結束註冊手續之後，Web 伺服端將此帳號的等級經由 Set-Cookie 的方式填寫到客戶端的 Cookie 中(如圖 4.3)。因此，客戶端就會記錄有關自己的身分等級資料。等到下次再度連線網站時，就會攜帶屬於自己身分等級的 Cookie 資料。當資料經過 Web Proxy 時，就會被擷取下來，由 Cookie 上的訊息便可知此連線的使用者身分等級為何，不需再經使用者由鍵盤輸入任何有關自己身分等級的相關資料。這樣可節省許多客戶端連線網站的延遲時間(Latency Time)。那麼如果多個使用者使用同一台電腦連線 Web 伺服器會不會造成身分混由不清?答案是不會。因為當 Web 伺服端設定 Cookie 給客戶端時，Cookie 進入客戶端時，客戶端會依當時使用者的身分將 Cookie 設定到該用戶的 Cookie 資料中。也就是說 Cookie 資料是以使用者身分來進行設定，而不是使用電腦為設定對象。因此每個用戶皆有自己的 Cookie 資料，在進行連線網站時 IP 位址取出自己的 Cookie 資料串送到 Web 伺服端以進行身分辨識。

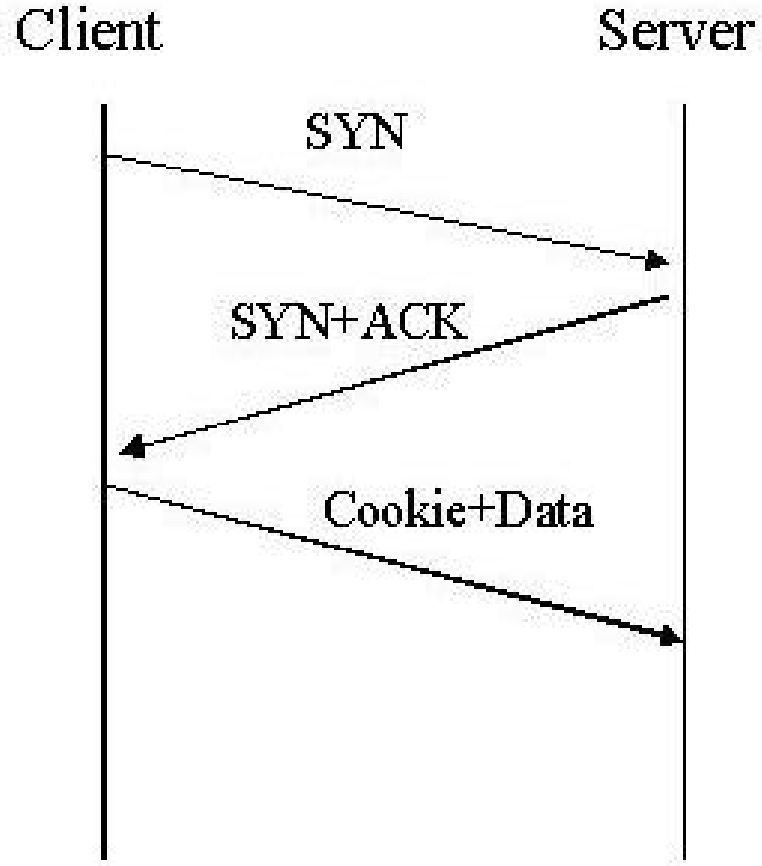


圖表 4-2 客戶端的註冊登入



圖表 4-3 客戶端的 Cookie 資料

當 Web Proxy **接受到客戶端的連線之後，首先會進行** TCP/IP **的同步**(Synchronization)**動作。在經過** Handshaking **的同步之後，客戶端如有之前此網站所設定過的** Cookie **資料時就會自動攜帶此** Cookie **資料進入** Web Proxy(**如圖** 4.4)**。**Cookie **資料釋放在** HTTP **通訊協定** Header **之後的純文字檔，所以要擷取** Cookie **資料只要在** HTTP **的** Header **後面將進入的純文字一一的放進緩衝區**(Buffer)**中，再加以剖析裡面的敘述即可。其中，因為** HTTP **在連線網站時會攜帶許多有關** Payload **的資料，所以剛放進** Buffer **內的** Cookie **資料是以**"Cookies" **字串起頭，緊接著就是** Cookie **的資料內容(如圖** 4.5)**。其內容是以**"**名稱-內容值**" **出現，所以要處理** Cookie **的資料是非常容易的。**



圖表 4-4 **客戶端與伺服端的握手**(Handshaking)**原理**

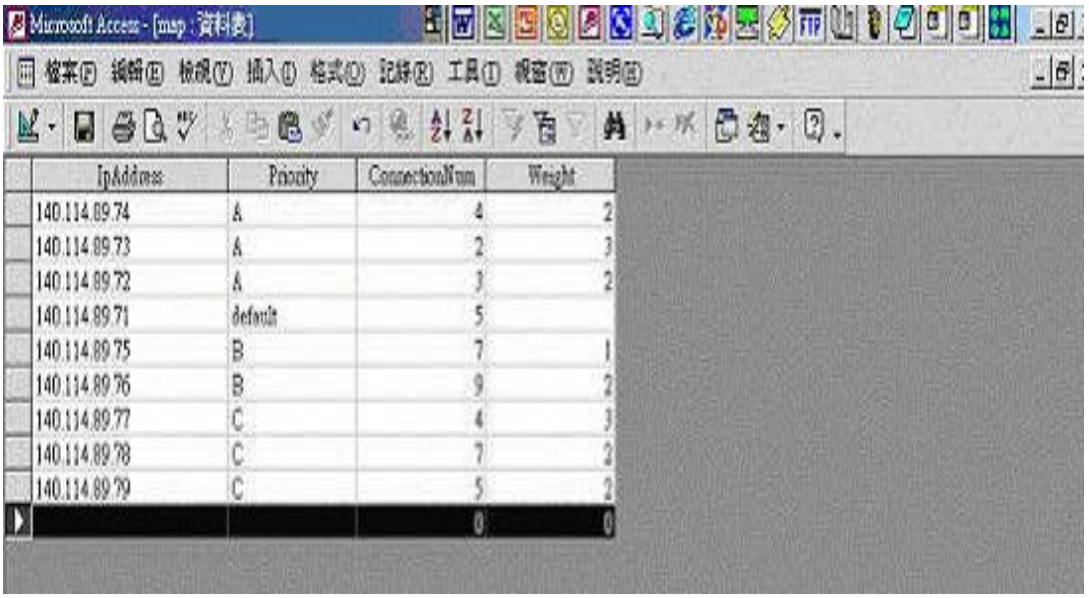
圖表 4-5 客戶端送出的 Cookie 資料

4.3 負載平衡(Load Balancing)

PriorityIpAddress

欄位	型態	大小(位元組)	備註
PriorityIp	Char	7	分組等級
IpAddresss	Char	15	分組等級內的電腦 IP
ConncetionNum	Int	4	連線數
Weight	Int	4	加權比重

圖表 4-6 **負載平衡**(Load Balancing)**表格**



圖表 4-7 Web Proxy **的資料庫資料內容**

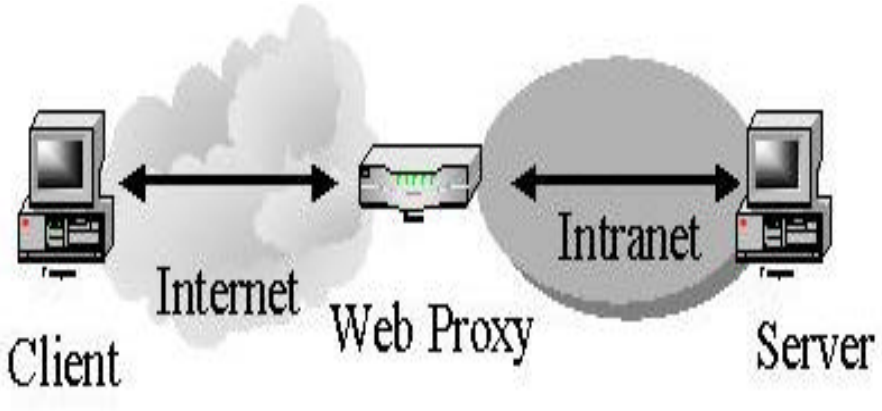
當 Web Proxy **接受客戶端**(Client)**的連線要求時，首先確認連線的資料中是否有攜帶** Cookie **的資料。如果沒有** Cookie **的資料，那麼** Web Proxy **上的負載平衡器**(Load Balancer)**就會將客戶端連線轉接至預設的站台。如果客戶端連線有攜帶** Cookie **資料，則由** Load Balancer **先行解譯** Cookie **上所攜帶的資料。如果所攜帶的資料具有正確的身分等級資料，則** Load Balancer **啟動** Cookie Look-up Module **進行資料庫查詢(如圖** 4.6 **及圖** 4.7)**。**

當確認使用者等級所對應的 Web **網站** IP **時,** Load Balancer **可採用四種負載平衡**(Load Balancing)**的方式進行選取在此等級上相對應可服務的數台** Web **站台中的一台來進行服務。此四種方法如下**:

1. **輪詢法則**(Round Robin)
2. **最少連線數法則**(Connection Leastly Used)
3. **加權式輪詢法則**(Weighted Round Robin)
4. **加權式最少連線數法則**(Weighted Leasely Used)

我們以最少連線數法則為例。如圖所示，假設客戶連線經身分等級確認後為 B **及則在資料庫中有兩台網站為** B **級，一為** 140.114.89.75**，連線數為** 1**。另一台為** 140.114.89.76**，連線數為** 2**。依照最少連線數法則選擇目前最少連線數的站台** 140.114.89.75**。接下來再由** Server Dispatcher **進行與站台** 140.114.89.75 **連線。**

4.4 Web 代理伺服器(Web Proxy)



圖表 4-8 Web Proxy **模型**

在 Web Proxy **上的** Cookie Look-up Module **確認了客戶端的** Cookie **資料之後，透過** Load Balancer **進行** Balancing **演算法的處理之後，利用** Server Dispatcher **進行與指定網站的連線，**Web Proxy[15]**此時扮演著轉接資料的角色。首先接受客戶端的連線後，**Server Dispatcher **會在與網站建立連線完畢後將客戶端與** Web **伺服端資料互動的動作交給了** Pivot Module **來處理。**Pivot Module **負責將客戶端的要求送往伺服端，也將伺服端的回應轉至對應的客戶端(如圖** 4.8)**。**Pivot Module **負責這樣的轉接的功能。轉接的方式是將客戶端的連線資料先接收至** Web Proxy **中，**Web Proxy **再建立一條往指定站台的新連線。也就是從客服端到伺服端經由** Web Proxy **必須建立兩條** TCP/IP **連線才可以完成通訊的功能。此原因乃是我們的** Web Proxy **的軟體是建立在** TCP/IP **之上，也就是應用層軟體，透過** Socket[16] **的方式處理有關客戶端與伺服端的通訊問題。**

5. 結論與未來的工作(Conclusion and Future Work)

網際網路的便利造就了許多著名的網站，網路的普及使得電子商務成為現在最熱門的網路話題。我們主要提供 Intranet 建置網站時分散處理客戶連線的一個機制，並且實作一個以 Cookie 為基礎的負載平衡的 Web 代理伺服器。透過客戶連線時攜帶的 Cookie 資料，我們可以利用上面的身分等級資料作為負載平衡(Load Balancing)的依據進行連線分散給多台網站服務，藉由此項方法提供更快速、更高更好的服務品質。而我們將此系統建置在 Intranet 的門口，使用單一 IP 位址面對客戶，接受客戶的連線後再轉接至對應的網站進行服務。如此可大大提升網站服務品質。在本系統中，我們利用連線數來作 Load Balancing，並且站台語等級的配對已經固定，如某一等級進入大量連線，其它等級僅有少量連線，在本系統中則無法提供動態的引用其它等級的站台進行支援。所以動態的使用其他等級的站台將是我們未來繼續研究的一項課題。另外我們使用連線數進行 Load Balancing 的依據，因為所有連線請求服務均需先行連線至本系統，所以我們可將使用連線的分配法則進一步的在將上頻寬控制(Bandwidth Control)。也就是進入的連線不僅導向至對應等級的站台進行服務，而且我們在進行站台連線時並進行頻寬分配，也就是我們可指定分配多少頻寬給進入的連線使用。此種方法必須在製作頻寬分配模組進行此項工作，此為未來繼續研究的另一項課題。

Cookie 經常被使用在身分辨識中(Identification)，為了減低首頁的反應時間及降低 Web 伺服端資料庫記錄的資料，所以 Web 伺服端將 Client 身分等級及其他相關的使用者資料記錄在 Client 中，下次 Client 連線時會主動攜帶這些 Cookie 資料進入 Web 伺服端，我們可在本系統攔截這些 Cookie 資料進行數據統計分析，以了解 Client 的行為模式。如在電子商務的應用上，可將 Client 端的消費商品、金額、使用站台的時間等資料使用 Cookie 的方式記錄在 Client 中，當 Client 再度連線時就可擷取這些資料進行分析並在首頁上推薦使用者有哪些符合消費行為的新進商品以促進使用者購買慾望。另外我們可進一步使用 Cookie 的功能於 Web Proxy 上，可將分散式與集中式的功能合為一，先透過固定的 IP 由客戶端連線上主要的 Web Proxy 上，由連線上的 Cookie 資料在進行轉接的其它的 Web Proxy 上，再由新的 Web Proxy 提供本系統服務，。此項功能可應用於 N 層次(N-tier)網際網路(Internet)之上，先進行分散式法則進行第一次分類，再由新的 Web Proxy 進行集中式的 Load Balancing。因此，利用 Cookie 的資訊進行各種服務是網際網路上的一個重要的應用。

所以在未來的工作(future work)中，我們可以繼續利用 Cookie 繼續延伸幾個問題。第一是 Cookie-based session tracking，也就是 Persistence。當我們在 Web Proxy 上進行 Load Balancing 時，Client 端進入 Web Proxy 時會被轉接至對應等級內的某一站台，因為 HTTP 是無記憶狀態(Stateless)的通訊協定，所以在 Client 再度進行 Request 時，Web Proxy 在進行轉接時有可能轉接至相同等級不同的站

台，此種方式對電子商務的應用上可能會有某些問題的存在。如牽涉到信用可刷卡問題時，最好使用同一台站台服務。此項問題可在 Web Proxy 接收到連線時插入一些預設的編號給各連線當作 Cookie，這樣 Web Proxy 就可辨識每一條進入的連線而將連線轉接至相同的 Web 伺服器。第二是電子商務的應用。如網路購物 Online Ordering Systems，可將客戶所選的物品利用 Cookie 記錄下來，如果客戶端尚未完成訂購手續而發生網路斷線時，藉由 Cookie 可在回覆斷線前購物車的內容。另外 Cookie 可應用在 Site Personalization。當使用者進入網頁之後，可選擇網頁上哪些功能西往使用，哪些功能不想使用。Web 伺服器可將這些資訊記錄在 Cookie 中，當 Client 再度連線時，則 Web 站台可利用 Cookie 上記載的資訊顯示 Client 端所希望顯示的畫面。也就是依據使用者的喜好只顯示個人化的畫面。除此之外，Cookie 常用來作身分辨識(Identification)及作為流量分類(Classify Traffic)等作用。最後，Cookie 也可用在 Search Engine 上，因為常常我們搜尋資料常常都會在不同的網站搜尋同一個 Keyword，我們可透過 Cookie 的資訊協助搜尋，這也是目前在 Internet 上的一個熱門話題。

Bibliography

[1]. R. Fielding, J. Gettys, J. Mogul, H. Frystyk and T. Berners-Lee, "Hypertext Transfer Protocol – HTTP/1.1", Internet Request for Comments 2068, January 1997.

[2]. D. Dristol, L. Montulli, "HTTP State Management Mechanism", Internet Request for Comments 2109, February 1997.

[3]. Netscape Inc., "Persistent Client State HTTP Cookies". [Online]. Available WWW: http://home.netscape.com/newsref/std/cookie_spec.html.

[4]. TED SCHROEDER, "'Cookie Cutting' keeps traffic moving", [Online]. Available WWW: http://www.nwfusion.com/news/2000/0221tech.html.

[5]. Mark Gibbs, "Confounded cookies leave a strange taste", [Online]. Available WWW: http://www.nwfusion.com/archive/1999b/1115gearhead.html.

[6]. Bob Trott, "Microsoft bakes cookie management", [Online]. Available WWW: http://www.nwfusion.com/news/2000/0721cookiemgmt.html.

[7]. Barry D. Bowen, "How popular sites use cookie technology", http://www.netscapeworld.com/netscapeworld/nw-04-1997/nw-04-cookies.html, Apr 1997.

[8]. Ann Harrison, "Cookie data in IE may be vulnerable to snooping", [Online]. Available WWW: http://www.nwfusion.com/news/2000/0512cookie.html.

[9]. Mark Gibbs, "Who are you? (take a cookie)", [Online]. Available WWW: http://www.nwfusion.com/archive/1999b/1122gibbs.html.

[10]. G. Apostolopoulos, D. Aubespin, V. Peris, P. Pradhan, D. Saha, "Design, Implementation and Performance of a Content-Based Switch", INFOCOM '00, Tel Aviv, Israel, March 2000.

[11]. JEFF CARUSO, "Resonate sticks its hand in the 'cookie' jar", [Online]. Available WWW: http://www.nwfusion.com/archive/1999/82432_12-06-1999.html.

[12]. Jeff Caruso, “Cookies are the rage in load-balancing switches”, [Online]. Available WWW: http://www.nwfusion.com/newsletters/lans/0124lan2.html.

[13]. JEFF CARUSO, “Foundry extends server load-balancing reach”, [Online]. Available WWW: http://www.nwfusion.com/archive/2000/84794_01-17-2000.html.

[14]. CHANDRA KOPPARAPU, “Persistence methods key for e-comm”, [Online]. Available WWW: http://www.nwfusion.com/news/tech/0424tech.html.

[15]. D. Malts, P. Bhagwat, “TCP Splicing for Application Layer Proxy Performance”, IBM Research Report RC 21139, March 1998.

[16]. M. Leech and D. Koblas, ‘SOCKS Protocol Version 5”, IETF Request for Comments 1928, April 1996.

[17]. Clinton Wong, “HTTP Packet Reference”, O’ REILLY, May 2000.

[18]. Elliotte Rusty Harold, “Java Network Programming”, O’ REILLY, June 2000.

[19]. David Flanagan, “JAVA IN A NUTSHELL”, O’ REILLY, April 2000.

[20]. W. R. Steven, “TCP/IP Illustrated Volume 1”, Addison-Wesley, New York, 1996.

[21]. **廖信彥**, “Active Server Pages **應用大全**”, **博碩文化**, **台北**, 1999.

[22]. Cookie Central, [Online] Available WWW: www.cookiecentral.com.

[23]. Oskar Pearson, “Squid A User’s Guide”, [Online]. Available WWW: http://squid-docs.sourceforge.net/latest/html/book1.htm.

[24]. Vipin Samar, “Single Sign-On Using Cookies for Web Applications”, Proceedings of the IEEE 8th International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises 1998.

[25]. David M. Kristol, “Cookie Specification”, [Online]. Available WWW:: http://portal.research.bell-labs.com/~dmk/cookie.html

[26]. Viktor Mayer-Schönberger, "The Internet and Privacy Legislation: Cookies for a Treat?", [Online]. Available WWW: http://www.wvjolt.wvu.edu/wvjolt/current/issue1/articles/mayer/mayer.htm

[27]. Network Associates, Inc., "Vulnerabilities in the Apache httpd", [Online]. Available WWW: http://www.nai.com/nai_labs/asp_set/advisory/02_apachemod_adv.asp

[28]. Electronic Privacy Information Center , "The Cookies Page", [Online]. Available WWW: http://www.epic.org/privacy/internet/cookies/