具可擴充性的協同合作式 Web Caching 共享架構

黃立德, 謝文雄
國立中山大學資訊工程學系
高雄市鼓山區蓮海路 70 號
{lthuang, wshsieh}@cse.nsysu.edu.tw

摘要

本文提出 Web Caching Sharing 共享架構，採用協同合作式的方式，並具有較佳的可擴充性(Scalability)，減少網路頻寬的耗用，縮短使用者等待時間。以往，在階層式 Web Caching 架構上往往會發生連線要求(connection request)過度集中在上層(upper-level)的網路快取伺服器中(cache server)，使得該伺服器成爲整個架構中的瓶頸。而在我們提出的協同合作式架構下，這些共同合作的快取伺服器彼此之間除了共同分享儲存在 cache storage 中的 object，也可互相支援連線要求，分散負載，使得網路負載與網站伺服器均能達到負載均衡(load balancing)，避免過多的連線要求集中在單一快取伺服器上。此外，在階層式架構中，當上層某個快取伺服器發生錯誤，無法提供服務時，也會使得整個階層式架構下層的快取伺服器連帶地無法繼續運作(Single-Point-of-Failure)。而協同合作式 Web Caching 架構則改善了這方面的問題。此外，也提出 Fetching Document Strategy、Request Path Resolution Algorithm 來解決快取分享效率不佳的問題

本文並未針對 Pre-fetching 與 Cache Coherence 或 Server Push 等相關研究主題來加以闡述，而其中 Cache Coherence 將會是我們未來繼續研究的方向。

Keyword: Web Caching, Caching Sharing, Proxy, Squid, WWW, World-Wide-Web, Web, Directory Service, Scalability, Cooperative

1. 前言

在現有的網際網路環境中，隨著越來越多的主機與使用者加入，越來越多的資料引入到這個網路世界中，引起網路使用量遽增，造成網路擁塞不已，World Wide Web (WWW) 更被人戲稱爲 World Wide Wait，因爲使用者在網際網路上游走，必須忍受不停地等待又等待。造成這種現象的主要原因，歸根究底，就是有太多的資料在網路上自資料原始端(Original Web Server)重覆的被傳送至不同使用者的電腦中，而造成網路擁塞、網站伺服器負載過重、傳輸延遲時間(latency)過長，由於以上緣故，而有了 Web Caching 技術的產生[1]，並且廣泛地被使用[2]。雖然如此，但隨著更多的主機、資料與使用者的加入，現有的 Web Caching 技術漸漸顯露出其缺點與不敷使用的窘境，也使得整個網際網路社群愈來愈重視這方面的問題，倘若能對此相關的技術加以改進，造就更高效能的 Web Caching 系統，則應可收立干見影之效，大幅改善網路的使用環境。以 TANet (Taiwan Academic Network) 爲例，雖然 TANet 對國際 Internet 網路頻寬於 1995 年由 512kbps 升級爲 T1(1.544Mbps)，1996 年 5 月再提昇至 2 條 T1，

1998 年 11 月更擴充爲 T3(45Mbps)，網路頻寬不斷擴增[3]，但由 TANet 網路流量統計[4]可知，頻寬仍嫌不足，網路仍滿載，造成嚴重的封包遺失(packet loss)甚麼干擾 DNS、E-mail、News 等網路基本服務的正常運作[5]，所以開始對頻寬使用採取政策性的限制措施[6]。所以我們可以瞭解到，透過 Web Caching 的實行，不只頻寬的浪費的問題可大幅改善，間接地也可減少花費在購買大量頻寬上金錢支出。

Web Caching 的好處在於：(1)減少網路頻寬的耗用。(2)減輕原始網站的負載。

目前廣泛被使用的 Web Caching 技術主要以 Harvest[1]爲主體，配合 1995 年被提出的 ICP (Internet Caching Protocol)[15]以達到 Web Caching 的分享機制。

快取分享(Caching Sharing)的主要目的就是讓不同快取伺服器彼此間能相互提供 object，以彌補當 cache miss 發生時，除了直接向資料原始網站接取資料，擁有該筆資料的其它快取伺服器就可幫忙服務，就近取得所需的 object，以補其不足，而此時，就稱作 accumulative cache hit，而發生 accumulative cache hit 的比率就稱作 accumulative cache hit ratio。也就是說，提高了 accumulative hit ratio，使用者可在較短時間內取得資料，而不用連到原始網路去接取資料，使得(1)減少使用者等待的延遲時間(latency)。(2)減少 cache miss 的發生，因爲一但發生 cache miss 則必定會增加 latency，並且也可(3)減少網路負載，減少資料在重覆地網路上自原始網路傳送出來。

使用者延遲時間(latency)，又可稱作使用者回應時間(user response time)，或使用者等待時間(user waiting time)。良好的 Web Caching 應該也要能提供較短的 latency，不致讓使用者在接取 object 時，等待過長的時間，才在瀏覽器中呈現出來。

本文中所提出的 Web Caching 架構，提供了良好的快取伺服器所應具備條件，對於 Web Caching 效率的改善主要著重在：
(1) 提高 accumulative cache hit ratio。(2)具可擴充性。(3)負載均衡(load balancing)。(4)有效率的快取分享。(5)網路擁擠(Network Congestion)以及網路頻寬使用(Bandwidth Consumption)的改善。(6)分層簡單的設定管理。

下一節介紹相關的研究成果，有些會應用在我們提出的新架構上。第三節介紹現有 Web Caching Protocol 以及階層式、分散式快取架構。第四節介紹我們提出的協同合作式快取架構，將階層式與分散式的快取架構加以改進。第五節是分析。第六節是本文的結論。在最末節，我們提出幾個方向，可供後續研究的發展延伸。

2. 相關研究

強制性的 HTTP[13]**封包 routing(Forced HTTP packet routing)**會引致網路效率低落。HTTP 封包 routing 又稱 **URL routing**，類似 IP-routing 但卻是使用 URL 來取代 IP 位址，同樣地加入 default routing。 HTTP 封包 routing 的缺點在於需要更換新的路由器與 routing 演算法，並且不能考量到快取伺服器 Application-Level 負載。

Adaptive Cache[8]，**設計建構一個全域性的(Global)網路快取架構。**由於 HTTP 封包是在透過 routing 路徑選擇，經過路由器或 LAN 交換器到達網路快取伺服器。當網路快取伺服器發生 cache miss 時，最後不論是自快取伺服器或原始網站所取得的 object 勢必都會經過 router 兩次，所以針對以上的缺點加以改善。**CGMP** (Cache Group Management Protocol)[8]使用 multicast 來自動發現鄰近的快取伺服器，並且

能夠動態調整彼此之間的關係與架構。

Cache Digest[12]的理念是，快取伺服器彼此間交換 Cache Digest，可以在每隔一定時間進行交換，或是視情況來進行交換，例如：當 False miss, False hit 發生頻率過高時。其中 False miss 指的是，雖然 Cache Digest 利用 Hashing Table 的機制，將 URL 對映到某個數字，例如：n，若某個 URL 的 object 有存在該快取伺服器中，就會記錄 object 相關資訊在 Hashing Table 的第 n 筆欄位中。其它快取伺服器只要將此 Hashing Table 取回，即可不需要送出 ICP Query[15]也不需要等待 ICP Response[15]就可馬上知道該快取伺服器中是否擁有此 URL 的 object，大幅減少 ICP Query / Response。其它還有一些研究成果是與利用 URL 命名區隔的方式來架構全域性的網路快取。

3. 現有的 Web Caching Protocol

Caching Protocol 就是用來掌控快取伺服器群之間分享運作的協定。

HTCP(Hyper Text Caching Protocol)[9]目前還屬於 Internet-Draft 的研擬階段，包含如何發現快取伺服器、object，以及如何管理、監看其運作。

HTTP(HyperText Transfer Protocol) 版本 1.1[14]提供了以下相關 Header 支援 Web Caching 的運作：Cache-Control、Age、Pragma: no-cache、Expires

3.1. 階層式(Hierarchical) Caching

ICP (Internet Caching Protocol)[15]的缺點在於太多的 ICP Query / Response 資料在快取伺服器群之間傳送，明顯地增加了使用者等待時間。而且會造成過多的連線要求集中在單一的快取伺服器上，在階層式快取架構上往往會產生過度集中的問題，使得該快取伺服器成爲整個架構中的瓶頸。並且，當上層的某個快取伺服器發生錯誤，無法提供服務時，也會使的整個階層式架構無法繼續運作

3.2. 分散式(Distributed) Caching

由 Microsoft Inc.所提出的 CARP (Cache Array Routing Protocol)[11]，利用 URL-hashing 的方式將 WWW client 的連線要求分散到所有參與 CARP 運作的快取伺服器群中。由於 CARP 使用 URL-hashing 的方式，來分配連線要求，特定 URL 的 object 會被指定到特定伺服器，而不會考量到 WWW client characteristic，雖然可以造就全域性的快取架構，但效率不佳，且不適合在區域較廣的 Web Caching 分享環境中使用。

3.3. HTTP packet routing

WCCP(Web Cache Control Protocol)[7]是配合 Cisco Inc.發展的 Cache Engine，將 Cache Engine 依附在路由器上，具有 Transparent Web Caching 的特性，Cache Engine 儼然就是快取伺服器，當 cache hit 發生時，便將 object 傳回給連線要求端，否則當 cache miss 發生時，便將連線要求往下個路由器轉送，直到 object 所在的原始網站爲止。缺點是，整個網路上都必須安裝 Cache Engine，因爲其它一般的路由器接收到此連線要求之封包，仍舊會繼續將封包傳遞下去，但此連線要求之封包目的地是 object 原始網站，而不是某個快取伺服器，在這種情況下，雖有傳統的快取伺服器存在網路上，也是徒然無功，無法使用。

4. My Proposed Web Caching Architecture (Cooperative Caching Sharing Architecture)

影響 Web Caching 架構運作效率的因素：

- WWW clients characteristic
- 快取伺服器之負載、hit ratio、記憶體容量、cache storage capacity
- 網路之拓樸、頻寬、congestion

根據以上的因素，我們提出一個解決的方案，針對可擴充性、負載均衡、網路頻寬耗用、簡單的設定管理等，加以改良，形成協同合作式 Web Caching 架構.。

WWW client 端利用 PAC (Proxy Automatic Configuration) 分散連線要求，以使得快取伺服器端能夠達到較佳的負載均衡 (load balancing)。其中 PAC 的使用可配合 URL-hashing 或類似 CARP 的 Score-based 概念。甚至，我們以 cluster 快取伺服器群的方式，依照不同 domain 來區隔快取伺服器所服務的範疇，以加強負載均衡的效果。

以階層式的觀點來看，將全部的快取伺服器群共分爲三層，自最上到最下層分別爲：national 快取伺服器群、regional 快取伺服器群、local 快取伺服器群。最低層的 local 快取伺服器群則以 cluster 的方式運作，彼此之間透過 meta-data 的之換來達成有效率的快取分享，而上層的 regional 快取伺服器群則以 URL-hasing 與服務網域來區分，達成負載均衡，並提供較佳的可擴充性，以形成整體上，具較有效的快取分享(caching sharing)。

在傳統的 Web Caching 架構，採用 ICP Query / Response 的方法，以 multiple unicast 的方式，沒有較好的選擇策略，針對同一層的快取伺服器送出 ICP Query，將近一半的網路頻寬耗用在 ICP Query / Response 的傳輸用量上。針對 ICP Query / Response 加以改進，不再使用 ICP Query / Response 來查詢 object 的所在位址。

在階層式的 meta-data 目錄服務中，每個快取伺服器都保存著自已 cache storage 的 meta-data 資訊以供參考使用。而在同一群組的各個快取伺服器自其它快取伺服器將 meta-data 資訊複製過來，以方便查詢，(1)利用 meta-data 來直接定址 object 在協同合作式快取伺服器群中的所在位址，提高 accumulative cache hit ratio，減少網路傳輸，與減低快取伺服器的負載。(2)漸進式地(Incrementally)更新別台快取伺服器的 meta-data，以減少網路頻寬的耗用。(3)提高了 accumulative hit ratio，使用者可在較短時間內取得資料，而不用連到原始網路去接取資料，也不需要 Query / Response 的時間，減少使用者等待的時間。

並且透過 URL-hashing 與 redirect 的方式仍可在快取伺服器發生 False hit 發生時加以處理，讓 object 接收偹直接與 object 提供端連線，不需再透過一個多餘的快取伺服器，所以也可減少網路傳輸，與減低快取伺服器的負載。

只有當下層的快取伺服器發生 cache miss，查看同一層快取伺服器的 meta-data 後，仍無法找尋到所要的 object，這時才會向上層快取伺服器送出連線要求，所以我們的分層架構可減少與上層快取伺服器的網路傳輸量。而且通常上層快取伺服器會較靠近網路骨幹，擁有較寬廣快速的網路頻寬。

針對可擴充性來分析，雖然使用 meta-data，需要額外的記憶體來存放，也會增加網路傳輸量(Inter-proxy traffic)，但其關鍵處正是在於 meta-data 的表現格式非常經濟節省、meta-data 的更新頻率[12]與一同分享 meta-data 的快取伺服器數量，所以協同合作式架構仍然可經由控制而具可擴充性 (Scalable)。其中 meta-data 的更新頻率又與 Stale hit、False hit、

False miss 發生頻率有關，因爲 meta-data 的資訊不夠新鮮，不夠即時準確，便會產生 False hit、False miss。雖然架構中仍分爲三個層級的快取伺服器群，針對可擴充性來說，仍會較完全階層式(hierarchical)架構來的好，所以協同合作式架構採用 three-level 快取伺服器群 hierarchies 的方式但並不與可擴充性的本質相衝突，只要我們能夠好好地決定每一層中一同參與合作的快取伺服器數量(Number of proxy sites)即可。

網際網路中，Web Caching 廣泛的被使用，在實際的使用實例中，我們發覺到不當的設定 hierarchies 將會嚴重導致整體運作效率不佳，也直接影響到網路的運作效能，因爲有過多不必要的資料在網路上傳送，如：ICP Query。最好的 hierarchy 設定就是符合網路拓樸(network topology)，但這又是一件很困難的又費力的工作。所以分散式架構被提出來討論與使用，透過一些機制，來達成較少的人工設定管理，雖說如此，所需付出的代價也不低，大多需要 multicast 技術的支援，或是過多的聯繫通訊資料必須要在各快取伺服器間傳送，以致效率不佳，況且，在此分散式架構中，能自動找尋鄰近的快取伺服器，又是一件困難的工作，而且往往並不能提供較好的運作效能，而分散式的好處當然就是在只需非常少的人工設定來維護其運作，或是無全不需要額外的人工設定。

我們只需要簡單的設定與管理，針對各個快取伺服器加以分層，設定上層的快取伺服器所在位址，配合 URL-hashing PAC 分散負載與自動定址出當 cache miss 發生時要將 request 轉送至上層的哪個快取伺服器；藉由 meta-data 交換的機制來加速同一層快取伺服器 之間的 cache object 分享，並減少過多的網路資料傳輸。

就管理策略來說，管理者不需要太繁複的設定與管理，尤其是針對 hierarchies 的設定，簡單但是必要的。(1)管理者必須簡單地去設定 hierarchies 並且以服務網域來區分上層的 cluster 快取伺服器群。(2)使用者只允許直接使用最下層的快取伺服器群。(參考 TANet 針對 HTTP 連線的使用限制策略[6])

4.1. 有效率的快取分享。

4.1.1. Fetching Document Strategy

利用 Fetching Document Strategy，來加強快取伺服器對於選擇接取 object 對象的篩選，而非傳統方式，只以 Parent、Sibling 來區分，以便能夠更有效更快速地傳輸到所需的 object，讓使用者端可感受到較低的 latency。

Fetching Document Strategy 指的就是快取伺服器自 candidate server list 選擇 peer server 去 fetch document 的步驟與所考慮的條件。我們提出，在選擇 fetch document 的對象，應

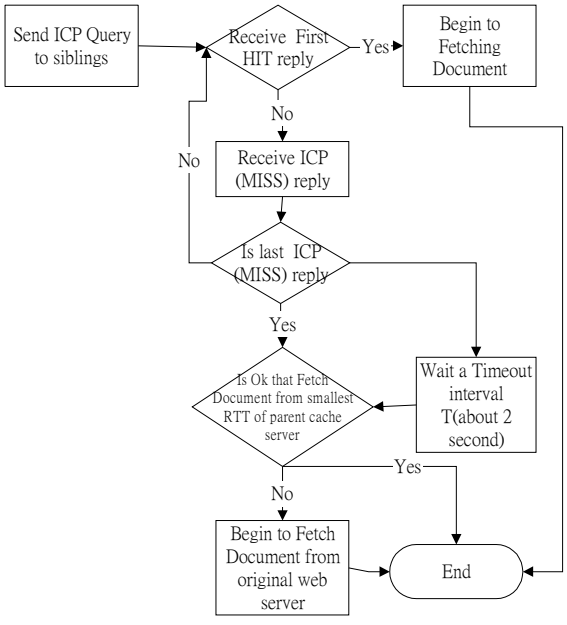

Figure1. Traditioanl Fetching Document Strategy

考慮的選擇：

- 若 Sibling 有此 object 時，則向 Sibling 抓取 object，當 Sibling 沒有此 object 時才向

Parent 送出離線要求

- 自本地快取伺服器(或 WWW client)到另端快取伺服器的 RTT(Round Trip Time)以及另端快取伺服器的負載
- 自本地快取伺服器(或 WWW client)到原始網站的 RTT(Round Trip Time)以及原始網站的負載
- 由於 Latency 包含 RTT、Connection Time、Transmission Time[10]。所以若 object 是大檔案，則選擇自較近的鄰近快取伺服器取回 object；若 object 是小檔案，則選擇自 Application-Level 負載較低、花費較低傳輸時間(Transmission time)的鄰近快取伺服器取回 object
- 快取伺服器可針對 object age 去訂立可忍受的範圍，所以在 meta-data 中應包含 object age 以提供其它快取伺服器足夠的資訊加以判別

4.1.2. Cache storage management

利用 Cache Storage Management 來有效地使用有限的 cache storage。針對快取分享來說，cache storage 是有限的且不應該浪費的，雖然我們允許有重複的 object 存在於不同的快取伺服器中，以增高 cache hit，讓快取伺服器可就近服務，而不需要自較遠的距離或花費較長的時間來完成傳輸。但 cache storage 是有限的且不應該浪費的，該如何避免有重複的 object 存在於不同的快取伺服器中，又那個快取伺服器應該儲存這份 object。我們提出以下的策略：

- 若 sibling 沒有此份 object，則本地快取伺服器則不將此份 object 儲存
- 反之，則本地快取伺服器將此份 object 儲存

當 Remote Hit ratio 提高時，便會增加網路頻寬的使用，所以熱門的 objects 應該要能夠慢慢遷移到離 object 要求端較近快取伺服器，這樣的遷移過程可以用 piggyback 的方式，再 object 要求端送出連線要求給較近的快取伺服器後，若發生 cache miss，則向其它快取伺服器要求，待接收到回應與該 object 時便將之儲存在 cache storage，而原存有該 object 的其它快取伺服器則應該將 object 移除，以完成遷移動作。

4.1.3. Request Path Resolution Algorithm

參考由 Microsoft Inc.所提出應用在 Microsoft Proxy Server 產品上的 CARP (Cache Array Routing Protocol)。

將最下層的快取伺服器群透過相互傳送 meta-data 來達到 cache sharing。上層的快取伺服器群 則透過 URL-hashing 或 score-based 的方式來分散 load, 並可減少 meta-data 的 broadcast。與 CARP 最大的不同在於，在最低層的快取伺服器群之間，並不使用 URL-hashing 或 score-based 的方式來定址 object 的所在位址，而是透過每個快取伺服器自同一層的其它快取伺服器所得到的 meta-data 來得知 object 的所在位址。這樣的優點在於，使用 URL-hashing 會難以控制 object 的擺放位址是有利用於減少 latency，而且難以提供良好的負載均衡，因爲有可能 URL-hashing 所得到的位址在集中在某個快取伺服器或是 object 接收端自該處接取 object 需要花費較長的時間。當下層的快取伺服器發生 False hit 時，便會將連線要求連同此連線的 Session-ID 往上層快取伺服器轉送，並使用 HTTP redirect[13]的方式通知 object 接收端，使得 object 接收端能轉向上層快取絲服器接取 object。

以上論述的方法，所形成的缺點是：

快取伺服器必須花費更多的 CPU cycles 來進行 Fetching Document Strategy 與 Cache Storage Management 與 Request Path Algorithm Resolution。

5. 分析

國立中山大學快取伺服器位於 TANet 網域上，採用 Cluster 的運作模式，並依照所服務的連線要求之目的地來區分，共分爲三台快取伺服器主機，位址分別爲如下：

由以上數據，可知 ICP Query / Response 的數量龐大，佔的比率頗高，並且以上快取伺服器已根據服務區域區分開來，並保留頻寬給 TANet 與 Internet 出國使用。若能多加利用 meta-data 資訊應可有較高的 accumulative hit ratio。

6. 結論

本文提出一個有效的、具可擴充性的 Web Caching 分享架構。透過 meta-data 的使用，不需要再使用 Query / Response message 來查詢 object 的所在位址，配合 Fetching Document Strategy、Cache Storage Management、Request Path Resolution，所以不但減輕快取伺服器的負載，也減少網路的傳輸量。當快取伺服器與原的負載減輕，網路的傳輸減少，也就是網路較不擁塞，使用者自網路上接收 object 的延遲等待時間也就減少。在整個架構中，也考量到可擴充性的問題，配合增加快取伺服器的數量可以容納更多的使用者與資料網站在 Web Caching 架構中有效地運作。

7. 未來目標與工作

雖然我們提供了一個良好的 Web Caching 分享架構，但主要是針對快取伺服器端的支援，對於使用瀏覽器的眾多使用者而言，設定快取伺服器的位址與配合快取伺服器的策略運作，整體上來看，還是無法提供很便利的使用環境。若能將 Transparent Web Caching 的觀點加入在我們提出的協同合作式架構中，就能具有更佳的彈性與便利。使用者就不需要去記憶他需要將 proxy server 做如何的設定。

本文中並未對 Cache Coherence 作相關的討論。雖然使用者透過快取伺服器能夠得到較好的使用效率，較低的等待時間，但往往發現傳送回來的資料過舊或與原網站不一致，甚至在其它快取伺服器還存在不同的資料版本，所以我們希望能夠在原本的架構中加入 Freshness Distribution Protocol 來解決這樣的問題。

在 HTTP (HyperText Transfer Protocol) 版本 1.1 中提到 Persistent Connection (持續性連線)，對同一個使用者而言，在一短時間內於網站上瀏覽，不斷地自網站上接取資料，HTTP 主要架構在 TCP (Transmission Control Protocol) 的基礎上，不斷的建立連線、接收資料、關閉連線，對於網路的使用與網站伺服器而言，都會導致較差的使用效益，於是提出持續性連線的方法，在不重新連線的情況下，可接取多次資料接收。但 Web Caching 在這方面的支援尚未有較顯著的使用。這是我們未來的計劃。若考量到使用上的安全性，快取伺服器不應將具有私密性的 object 儲存至 cache storage 中，或將 object 曝露給未經授權的對象，對於接取此類具有私密性 object 時，應該告知快取伺服器。

參考文獻

[1] Harvest, <URL:http://harvest.cs.colorado.edu/>,<URL: http://harvest.transarc.com/>

[2] Squid, <URL:http://squid.nlanr.net/Squid/>
[3] 台灣學術網路(TANet)概述, <URL:http://www.edu.tw:81/tanet/tan-intro/2.html>
[4] 國際 T3 流量, <URL:http://gumpism.edu.tw/tanet/backbone/taiwan-t3-internet.html>
[5] 楊素秋, 劉大川, 許乃文, "ATM 高速網路管理技術探討", TANET98 研討會
[6] TANet 技術小組決議, <URL:http://cache.edu.tw/proxy/proxy-res.html>
[7] Cisco Inc., "Web Cache Commumnication Protocol(WCCP) v2", <URL:http://www.cisco.com/univercd/cc/td/doc/product/software/ios120/120newft/120t/120t3/wccp.htm>
[8] Adaptive Web Caching: Towards a New Global Caching Architecture
[9] ICP Working Group, Paul Vixie, "Hyper Text Caching Protocol (HTCP/0.0)", IETF Internet-Draft, March 1998
[10] Pablo Rodriguez, Christian Spanner, Ernst W. Biersack, "Web Caching Architecture: Hierarchical and Distributed Caching", The 4th International Web Caching Workshop, San Diego, California, March 31-April 2, 1999
[11] Microsoft Inc., "CARP White Paper", <URL:http://www.microsoft.com/proxy/documents/CarpWP.exe>
[12] Li Fan, Pei Cao, Jussara Almeida, and Andrei Z. Broder. Summary Cache: A Scalable Wide-Area Web Cache Sharing Protocol. In Proceedings of ACM SIGCOMM, September 1998
[13] Network Working Group, "Hypertext Transfer Protocol -- HTTP/1.0", RFC1945, May 1996
[14] Network Working Group, "Hypertext Transfer Protocol -- HTTP/1.1", RFC2068, Jan 1997
[15] D. Wessels , K. Claffy , "Internet Cache Protocol(ICP), version 2", RFC2186, Sep 1997

■ 表一、proxy.nsysu.edu.tw 負責 .tw 網域的資料傳送及其使用狀況分析

(記錄時間：Mon, 09 Aug 1999)

Allocated memory	Cache storage	Number of clients accessing cache	Mean object size	CPU Time usage
501421 KB	104.18 GB	5285	20.49 KB	33.88%
HTTP requests per minute	ICP messages per minute	Server requests	Server kbytes in	Server kbytes out
2922.9	1445.1	http 20.019353/sec ftp 0.033332/sec other 0.056665/sec	http 75.944213/sec ftp 19.586034/sec other 0.083331/sec	http 9.929679/sec ftp 0.006666/sec other 0.039999/sec
ICP pkts	ICP queries	ICP replies	ICP kbytes	Hit ratios
sent 7.693085/sec recv 7.693085/sec	sent 0.000000/sec recv 7.693085/sec	sent 7.693085/sec recv 0.000000/sec	sent 0.536649/sec recv 0.569982/sec	By bytes (70.1%) By requests (65.9%)

■ 表二、cproxy1.nsysu.edu.tw 負責 .com 網域的資料傳送及其使用狀況分析

(記錄時間：Mon, 09 Aug 1999)

Allocated memory	Cache storage	Number of clients accessing cache	Mean object size	CPU Time usage
505014 KB	200.21GB	6080	28.30 KB	27.32%
HTTP requests per minute	ICP messages per minute	Server requests	Server kbytes in	Server kbytes out
1700.4	2404.6	http 8.219934/sec other 0.209998/sec	http 86.012641/sec other 6.356616/sec	http 4.006634/sec other 0.033333/sec
ICP pkts	ICP queries	ICP replies	ICP kbytes	Hit ratios
sent 15.866539/sec recv 15.866539/sec	sent 6.033285/sec recv 9.833254/sec	sent 9.833254/sec recv 6.033285/sec	sent 1.279990/sec recv 1.293323/sec	By bytes (55.7%) By requests (48.4%)

■ 表三、ccproxy2.nsysu.edu.tw 負責其它網域的資料傳送及其使用狀況分析

(記錄時間：Mon, 09 Aug 1999)

Allocated memory	Cache storage	Number of clients accessing cache	Mean object size	CPU Time usage
504487 KB	135.85 GB	6084	27.81 KB	20.07%
HTTP requests per minute	ICP messages per minute	Server requests	Server kbytes in	Server kbytes out
1195.6	1698.0	http 6.219901/sec ftp 0.006667/sec	http 53.342482/sec ftp 3.373280/sec	http 3.259948/sec ftp 0.003333/sec
ICP pkts	ICP queries	ICP replies	ICP kbytes	Hit ratios
sent 6.673227/sec recv 6.413231/sec	sent 0.519992/sec recv 6.153235/sec	Sent 6.153235/sec Recv 0.259996/sec	sent 0.503325/sec recv 0.506659/sec	By bytes (52.2%) By requests (59.9%)