以即時流量控制之代理伺服器提供彈性的網路服務

游象甫，曾黎明
國立中央大學資訊工程研究所，國立中央大學電子計算機中心
yu@dslab.csie.ncu.edu.tw tsenglm@cc.ncu.edu.tw

摘要

根據教育部電算中心的統計 HTTP 佔 TANet 流量第一位，且持續成長. 爲使網路頻寬使用更有效率, TANet 全面推行使用網路代理伺服器，藉由網頁資源的共用，降低頻寬需求. 網路代理伺服器的確有效增加頻寬的利用率，若代理伺服器能對不同的需求提供不同的服務，則應能進一步提昇網路服務品質. 同時國內多所大學發生使用者以下載程式大量且連續下載電子期刊以致影響其他正常使用者權益的事件，亦突顯網路代理伺服器需要即時有效的使用管理功能，以維護正常使用者的權益. 目前多數代理伺服器只提供簡單的使用者及 Web 伺服器存取管制，無法對異常使用作出反應，雖然 TANet 有些學校管制網路代理伺服器使用量，但只針對使用者的總下載量，並沒有對個別使用者或網站下載量進行控管，另外其監控的時間間隔長達一日，異常的使用行爲需 24 小時後才能被發現阻止，缺乏時效. 在此論文中我們提出一個網路代理伺服器使用量控制系統，可根據不同的需求，提供彈性的管制機制，同時可保障網路資源的公平使用，維護網路智慧財產權.

關鍵字: 網路代理伺器，流量管制，網際網路，WWW, Web

1 簡介

由於 WWW 其支援多媒體及安裝使用簡易的特性，是 Internet 成長最快速也是最重要的服務，幾乎各行各業都有網站供查詢. 根據教育部電算中心統計 TANet 上 HTTP[12, 19] 封包佔所有流量的 85%[9]. 爲使網路頻寬使用更有效率，TANet 推行使用 WWW 網路代理伺服器，(本論文中除特別說明，否則網路代理伺服器或代理伺服器都是指 WWW 網路代理伺服器)，藉由共用網頁降低頻寬需求. 網路代理伺服器有多項好處. 首先，可以降低回應時間讓使用者更快速的瀏覽網頁；其次，被快取(cache)的物件可被多人使用，讓網路使用更加有效；最後，代理伺服器可以分擔熱門網站的負載.

雖然上述優點已經由實際測試而被驗證[18]，但我們認爲代理伺服器還有針對不同的需求提供彈性服務的潛力. 有非常多的研究[]在討論如何對不同的使用者給不同的網路資源，其大部分是從網路層的觀點來提出可能的方法. 若從應用層來考慮，則代理伺服器不失爲一種可能的策略，所以若代理伺服器能控制使用量，則提供網路上優先權服務就不再是不可能的事了.

另一方面，代理伺服器可藉由控制使用量維護所有網路服務提供者及使用者的權益. 當使用者藉代理伺服器連上網站時，網站可以看到代理伺服器而無法知道真正連線的使用者. 此特性使得網站被網頁下載軟體(如 WGET[2])大量存取時，不易偵測，因爲網路代理伺服器會產生非常多的存取，以中央大學的代理伺服器爲例，一天有上百萬次存取，對某些網站可能就有數萬次，所以網站很難識別那些存取是正常或是惡意的. 這種網路濫用行爲會損及他人的權益，對網站而言，網路頻寬被佔用，伺服器負載增加，智慧財產權受侵犯；對使用者而言，網路頻寬與代理伺服器被佔用，等待時間加長，更糟的是如果受攻擊的網站對代理伺服器採取暫停使用的策略，則所有正常的使用者都會受到影響.

1998 年 8 月時中央大學圖書館收到美國物理學會的電子郵件通知校內有不明人士於一個晚上經網路代理伺服器下載約 900 MB 的論文, 希望能查明原因. 對此事件本校迅速查出係一研究生因不想慢慢瀏覽於是利用 Teleport[6] 下載美國物理學會期刊網站中的論文, 其非惡意, 也不是企圖違反智慧財產權. 但這並非單一偶發事件, 隨後在 1999 年台灣其他國立大學也發生類似事件, 甚至造成全校被電子期刊停權.

TANet 上很多大專院校皆有網路流量管制, 但通常屬公共服務的伺服器, 如電子郵件伺服器或代理伺服器不在管制範圍內, 所以使用者可以利用代理伺服器躲避流量管制, 縱然 TANet 有些學校[3, 4, 5] 管制網路代理伺服器使用量, 通常只針對使用者的總下載量, 並沒有對個別使用者或網站下載量進行控管, 另外其監控的間隔長達一日, 異常的使用行爲需 24 小時後才能被發現阻止, 缺乏時效.

爲滿足上述需求及解決已發生的問題, 本論文提出一個可控制網路代理伺服器使用量的系統, 讓代理伺服器能針對不同的需求提供富彈性的服務, 同時保護所有網路使用者的權益, 本系統具備下列功能:

- 可自動產生網路代理伺服器存取控制設定, 包含網站及使用者;
- 可即時對異常使用採取反應並公告, 包含網站每單位時間被下載量管制及使用者每單位時間使用量管制;
- 可將使用者或網站分成不同的群組設定不同的管制策略.

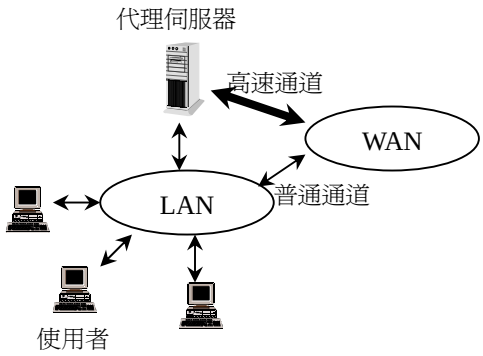
本論文其他章節安排如後, 下一節討論控制代理伺服器使用量的問題及可能的解決方法; 第三節說明本系統的實作; 第四節介紹相關研究; 最後是結論及未來改善方向.

2 問題研究及可行方法

本節會討論如何利用代理伺服器提供彈性的服務以滿足不同的需求或維護使用者權益. 另外, 我們也會說明如何設計具流量管制功能的代理伺服器.

2.1 如何提供彈性的服務

提供彈性服務的策略有很多種, 例如在 WWW 伺服器上建置 prioritized 機制[10, 14, 15], 在此我們僅討論如何在 Internet 讓具控制流量功能的代理伺服器能產生效用. 可以從兩個方面來思考, 第一是代理伺服器的使用是非通透; 第二是代理伺服器的使用是通透. 第一種方式使用者必須在瀏覽器上設定使用代理伺服器, 所以會遭遇使用者不知道或不願意設定使用代理伺服器的問題, 讓代理伺服器無法發揮控制流量的功能. 使用者不知道需要設定或如何設定代理伺服器的問題可以藉由宣導伺服器的好處或教授瀏覽器設定來改善, 這點在 TANet 上已經証明可以克服不再是問題. 但對於不願意使用代理伺器的使用者則必須搭配其他誘因使其樂於使用代理伺服器, 比如代理伺服器有專屬高速頻寬, 如圖一, 過去 TANet 就是利用類似的方式, 讓各區網中心的代理伺服器以 163.28.X.X 的 IP 使用保留頻寬出國, 以高速吸引使用者使用代理伺服器, 現在的作法則改以讓代理伺服器以 ADSL[32] 專線接民營 ISP 出國, 如交大, 中正理工學院等.這種利透策略也証明相當有效, TANet 上大部分使用者均有使用代理伺服器. 不過因爲非通透方式是建立在使用者的配合上, 所以網站也



圖一 以具高速專屬通道的代理伺服器提供彈性網路

需要有自己的管制機制，才能避免被大量下載.

在使用通透代理伺服器[16, 17, 35] 的情況下，瀏覽器不必設定代理伺服器即會自動使用代理伺服器. 建構通透代理伺服器的策略可分爲兩種:

- 封包轉向. 使用 layer 4 switch 或 policy-based router 將瀏覽器的封包轉向至代理伺服器. 通常代理伺服器會直接與 switch 相連，而 switch 會將所有往 port 80 的封包導向代理伺服器. 當 switch 收到往 port 80 的 TCP SYN 封包時會將封包轉向到代理伺服器，而不必更改 IP 或 TCP 的 header 資訊. 此時通常代理伺服器會被設定爲 promiscuous 模式接受任何連線請求而不管其連線目的地爲何，同時也會將送回封包的 source IP address 改爲原來使用者要去的網站位址. 最後當連線建立完成，代理伺服器收到瀏覽器送來的 URL (不含網站位址)，再利用原來從 TCP SYN 得到的目的網站位址組合成一個含網站位址的 URL，接著如正常的代理伺服器一樣開始存取網頁. 使用 router 的方式和 switch 相似，除了 router 只會將往 port 80 的封包轉向至代理伺服器，其餘封包則繞路至 Internet. 利用類似原理發展的通透代理伺服器有[26, 35].
- 封包攔截. 將代理伺服器配置於 router 同段的網路上，利用網路監控軟體攔截 HTTP 封包，將其轉向至代理伺服器，藉此提供具通透性的服務. 當網路監控軟體攔截到連線的第一個 TCP SYN 封包時會將封包轉向到代理伺服器，此時將使用者及其目的網站的 IP 位址及 port 儲存下來，並將被攔截封包的目的位址改爲代理伺服器的位址，接著將修改的封包送回網路，於是接來所有來自於同一個使用者同一個 port 的封包就會被送至代理伺服器. 相反地，若代理伺服器要送封包給使用者時會將封包的 source IP 及 port 改爲原來目的網站的 IP 及 port. 藉此代理伺服器可以在使用者不知情的情況下幫使用者完成 HTTP 請求. 利用類似原理發展的通透代理伺服器有[16, 23, 25].

2.2 如何設計可控制流量的代理伺服器

2.2.1 建立新系統 vs 修改現存系統

可用兩種策略來發展符合需求的代理伺服器: 建立新代理伺服器及修改現存代理伺服器. 兩個方式各有優劣，前者的優點有: 沒有相容於舊系統的包袱、可以完全根據需求發展、沒有智慧財產權問題和擁有完整發展文件及技術，日後修改比較容易. 但缺點爲: 系統開發難度高、需要很多費用及時間. 相反的，修改現存系統的好處爲: 利用現存技術比較容易、節省費用及時間. 其壞處是: 必須牽就修改的系統、可能破壞原設計架構、缺乏完整系統發展文件、有智慧財產權的問題，同時在某些情況下有時不可能修改原系統.

2.2.2 修改程式碼 vs 外加功能模組

若採用 2.1修改現存系統的策略，有兩個方式可以進行: 修改程式碼或外加功能模組. 修改程式碼是取得系統原始碼後直接修改或新增，依這種方式發展系統執行較快，同時對修改的功能沒有限制，但沒有原始程式碼時，則無法進行修改; 另一方面，程式設計師必須必須閱讀大量原始程式碼，且必須考量版本的問題，當原系統更換新版本時，修改的程式碼可能不適用必須修改. 最後，因爲是修改原始碼，智慧財產權問題可能難以避免.

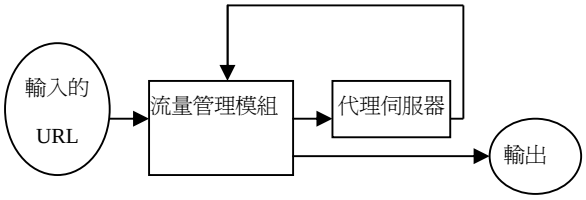
外加功能模組是不變動程式碼但新增模組以加強原有系統的功能. 利用此種方法不需有原始程式即可加強原系統功能，可用來填補商業系統不足之處; 程式設計師不需閱

讀程式碼，可以最適合的程式語言開發，得以加快發展時程，降低成本；若原系統出新版本，外加功能模組較不需隨之更新；最後，因爲沒有引用原始碼，所以可能不會遇到智慧財產權問題. 不過由於不修改原始程式碼，可增加的功能受限，有時可能無法滿足需求；另外，因爲功能模組並沒有完全整合進原系統，所以效率較低.

2.2.3 有關外加功能模組的方式

若利用 2.2外加功能模組的方式使代理伺服器具流量管理的功能，則有三種方式可將網路代理伺服器與流量管理模組連結起來:

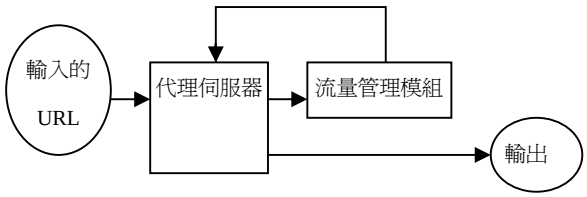
- URL 轉向: 將流量管理模組置於網路代理伺服器之前，即以輸入的 URL 作爲模組的輸入，而模組再利用代理伺服器取得網頁內容，最後由流量管理模組輸出給使用者，在整個過程中流量管理模組可以分析輸入的 URL 決定是否呼叫代理伺服器取得網頁，亦可分析得到的網頁內容統計下載量，進行流量管制，如圖二. 若以 SQUID 爲例，流量管理模組可從 port 3128 取得 HTTP 請求，接著呼叫 SQUID 的 Client 程式[7] 使 SQUID 由網站取得網頁內容，再導回



圖二 以 URL 轉向方式新增流量管理功能

流量管模組分析，決定是否輸出取得的網頁. 另外 SQUID 也提供 URL 轉向的存取介面，也可以外掛程式對處理 URL.

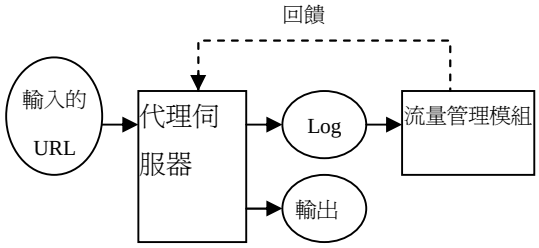
- 網頁轉向: 有些代理伺服器有提供網頁轉向的功能[17]，可將取得的網頁轉向至一個外掛程式處理後，再輸出到瀏覽器，所以我們可以將代理伺服器的輸出轉至管理模組再將結果轉回代理伺服器



圖三 以網頁轉向方式新增流量管理功能

傳給使用者，如圖三.

- 分析 Log: 最後一種是分析代理伺服器產生的 log 以此調整伺服器的運作進行流量控制，如圖四.



圖四 以分析 log 方式新增流量管理功能

2.2.4 其他設計考量

本節討論其他發展網路代理伺服器流量管制時應注意的需求.

2.4.1 即時管制

就像前面曾提到的，雖然 TANet 上有些學校進行網路代理伺服器使用量管制，但管制時間間隔達一日，缺乏時效. 所以我們認爲管制最好能即時，才能在第一時間內對異常情況作出反應. 不過在設計上需考慮網路代理伺服器一般而言非常忙碌，隨時均有大量網頁存取，分析計算各種使用量會耗用相當多的系統資源，降低伺服器效能. 不同的實作技術會有不一樣的方式以避免這種情形,, 若以 2.3 節分析 log 的方法爲例，考慮網路伺服器的輸出量，顯然持續不斷的分析 log 不是最佳的策略，比較有效率的方式是依管理者要求代理伺服器管制下載量的精確度來計算出最合適的監控週期，假設代理伺服器的使用者平均每秒可下載 50 Kbyte，若管理者希望使用量誤差不超過 1 MB，則每 10 秒 (1000/50/2，多除 2 是爲避免最壞的情形: 使用者在前一次統計時非常接近管制量但沒超過比如 0.99 Mbyte，在第二次時又用了 0.99

Mbyte, 總共 1.98 Mbyte 才被管制) 統計下載量一次就可達到即時管制的目的了.

2.4.2 自我調適

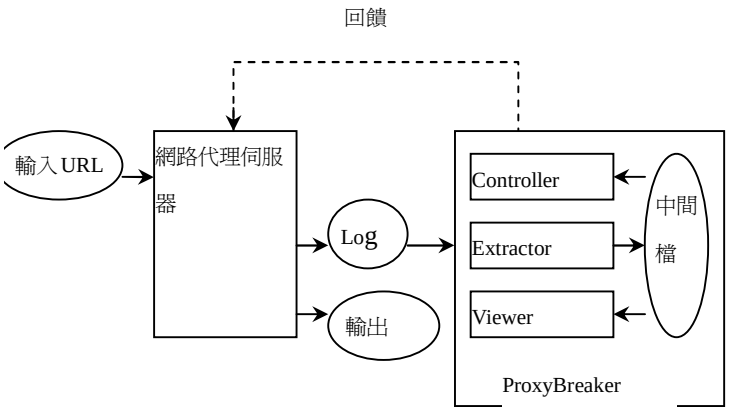
因爲網路環境變化非常快, 常常在不同的時間有不同情況, 例如, TANet 流量一般是從晚上開始上昇, 至午夜到達高峰, 然後下降, 至早上 5 點左右最低. 在設計流量控制系統時, 必須儘量使其可根據環境的改變自動調整, 以減輕流量控制對系統造成的額外負載, 舉例來說, 管制機制可自動於晚上 8 點網路擁擠, 傳輸速度較低時, 將監控的時間間隔調大, 使代理伺服器效能較不受影響; 同時又自動於早上 5 點傳輸速度較快時, 將監控的時間間隔調小, 避免有漏網之魚.

2.4.3 友善使用者介面

雖然系統的目標是使代理伺服器可對使用量進行控制, 然而對管理者或使用者而言, 一個具親和力的使用者介面仍然是必要的. Web 介面無疑是非常好的選擇, 對 Internet 的使用者而言, 瀏覽器易於安裝及使用, 而且跨平台, 幾乎所有提供視窗環境的作業系統都有支援, 因此一個好的流量管制系統基本上應該提供以 HTTP 存取的使用者介面.

3 系統實作

我們已完成一個網路代理伺服器即時流量控制系統, 稱爲 ProxyBreaker, 本節將介紹其架構與實作.



圖五 ProxyBreaker 與網路代理伺服器的連結

3.1 ProxyBreaker 的設計

根據前面的討論, 考量發展的成本, 因此 ProxyBreaker 是採用修改已存在網路代理伺服器的策略, 我們選擇修改 Internet 上常見的代理伺服器 SQUID[7, 31, 34]. 因爲 SQUID 常常更新版本, 若修改原始碼必須花時間產生不同的 patch file, 以維持與 SQUID 版本一致, 所以我們不考慮修改原始碼而以外掛流量控制模組的方式, 讓 SQUID 能進行使用量控管. 最後爲使我們的控制模組容易相容於其他網路代理伺服器, ProxyBreaker 以 2.3 節中分析 log 方式與網路代理伺服器連結, 進行流量管制, 如圖五.

3.2 ProxyBreaker 的架構

ProxyBreaker 分成三部分: Extractor, Controller 及 Viewer. Extractor 負責從 log 萃取必須的資料, 同時將資料存入中間檔 (middle file). 接著 Controller 分析中間檔後, 依據管理者的設定自動產生 SQUID 的設定檔 (squid.conf), 藉以進行使用量管制. Viewer 則提供 Web 介面供使用者查詢有關的管制資訊.

3.2.1 Extractor

SQUID 的 log[7] 是一個循序文字檔, 每一次存取都會留下一筆記錄. 因爲原始 log 包含非常多資料, 但不都是流量管制所需要, 爲減少處理資料量, Extractor 將原始 log 轉成中間檔供 Controller 分析. Extrator 每隔一段時間就讀取 log, 只保留其中有關 HTTP 存取的部分, 取出 time, client address, bytes, peerstatus/peerhost 等四個欄位資料後, 將相同的使用者及網站下載量加總後, 逐筆儲入中間檔. 中間檔有兩種, 一種是以使用者對網站的下載量, 另一種爲網站對使用者的被下載量 (中間檔的格式如圖六). 爲加速 Controller 處理的速度, 每次讀取 log 時就會建立新的中間檔, 換句話說, Extractor 會產生多個中間檔, 每個中間檔只代表某一段時間

的資料，而且其檔名會包含資料的起始時間以加速搜尋檔案速度.

WWW server	Client	Status	Byte	Frequency

網站被下載量記錄

Client	WWW server	Status	Byte	Frequency

使用者下載量記錄

WWW server: 網站位址
Client: 使用者位址
Status: 網路代理伺服器快取存取狀態碼
Byte: 被下載量或下載量
Frequency: 存取次數
圖六 ProxyBreaker 中間檔格式

由於 Extractor 每隔一段時間才會讀取 log, 因此時間間距會影響 ProxyBreaker 的最大可能誤差，最大可能誤差是使用者最大下載速度 X 時間間距 X 2, 即若使用者最大下載速度爲 20 Kbyte/sec, 時間間距爲 50 秒, 則 ProxyBreaker 管制的最大可能誤差爲 2 MB, 使用者可能超用 2 MB 才被發現. 在實務上，時間間隔會依據使用者平均下載速度來設定，像 2.4.1 所提及，若網路不擁塞，使用者下載速度快，則時間間隔須設定較小，反之，若網路擁塞，下載速度慢，則時間間隔可爲較大值. 值得注意的事是若時間間隔設定較小會讓 SQUID 重新設定的次數增加，影響其服務效能.

此外，Extractor 會檢查網路代理伺器的磁碟空間是否足夠，自動清除 log 及過期的中間檔. 雖然 log 已被化簡產生新的資料，可大幅縮短分析的時間，但尚未解決 log 檔佔用大量磁碟空間的問題. 因此 Extractor 會壓縮過期的 log 約成爲原大小的十分之一，同時清除過期中間檔. 除此之外在磁碟空間低於臨界點時，會自動清除 log. 最後，Extractor 會檢查網路代理伺服器快取磁碟空間是否不足，如果低於臨界值則會以電子郵件告知系統管理者，若沒有得到有效處理，會自動呼叫網路代理伺器清除快取中的檔案.

3.2.2 Controller

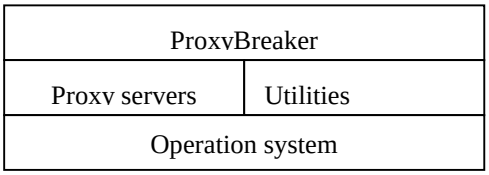
Controller 的功能在於根據管理者預先設定的流量進行即時管制. 首先利用不同的中間檔建立兩種雜湊表，一個是以使用者-網站爲鍵，值爲使用量的雜湊表; 另一個爲以網站-使用者爲鍵，值爲下載量的雜湊表. 接著根據管制條件找出違規的使用者，最後 Controller 會自動產生對應網路代理伺服器(即 SQUID)的 ACL (Access Control List), 並且更新其設定檔，同時讓新的設定生效.

3.2.3 Viewer

Viewer 負責提供介面供使用者或管理者查詢管制情況. 爲容易使用及跨平台，Viewer 主要提供 Web 查詢介面，利用 CGI 程式讀取 Extractor 產生的中間檔中各項使用量資訊，讓使用者可查詢其使用量況狀同時可知道其是否已被暫時停止連線至某些網站.

3.3 ProxyBreaker 的實作

本系統已實作完成，在 Pentium II 400, 10/100 Mbps 網路卡的個人電腦上執行. 作業系統是 FreeBSD[21], 網路代理伺服器是 SQUID 2.3-STABLE, WWW 伺服器是 Apache 1.36[11]. 使用 perl 程式語言開發，實作架構如圖七，中央大學圖書館的網路代理伺服器已利用 ProxyBreaker 進行流量管制,



Operation system: FreeBSD

Proxy servers: SQUID

Utilities: mail, Apache WWW server

圖七 ProxyBreaker 實作架構

目前提供下列功能:

- 自動產生網路代理伺服器存取控制設定, 包含網站及使用者管制;
- 自動對異常使用者進行即時管制並公告, 包含網站每單位時間被下載量及使用者每單位時間使用量管制;
- 可將使用者或網站分成不同的群組設定不同的管制策略;

- 自我偵測網路代理伺服器是否正常運作，回報異常情形並自動進行校正.

4 相關研究

網路代理伺服器的管理軟體眾多，在此僅介紹與 SQUID 相關的軟體. MRTG (Multi-Router Traffic Grapher)[28] 是 Internet 普遍用來顯示網路流量的軟體，用 C 及 Perl 語言所撰寫，可在多種平台上執行，其可根據所提供的資料，如 SNMP 的 MIB[29] 資訊，自動產生 HTML 檔及相關的 GIF 檔，讓網路的狀況以圖形化的方式呈現. 另外 MRTG 也可顯示系統的資訊，例如伺服器負載，hit ratio, request rate 等.

Calamaris[13] 是分析 SQUID 的 log 計算各項統計數據並以網頁顯示的軟體，提供的資訊可供管理者調整其網路代理伺器效能. 其他類似的軟體有 Anemone[1], pwebstats[22], PY_Squid_Stats[30], WebLog[27] 及 Squeezer[24]. 最後介紹兩種 SQUID 的網頁過濾器 (filter): SquidGuard[8] 及 Squirm[20], 由於 Internet 上暴力及色情網頁日益增加，爲保護青少年的身心，所以在網路代理伺服器上安裝過濾器以攔阻內容不宜的網頁，兩種軟體提供的功能類似，可以限制使用者存取網站或網頁、禁止下載含某些字串的 URL 及將使用者分群管制的功能. 根據 Jann-Perng Tseng[33] 的研究 SquidGuard 遠較 Squidrm 來得有效率. 表一是各軟體功能的比較.

表一 有關 SQUID 網路管理軟體的比較

	ProxyBreaker	Calamaris	SquidGuard
Access control	V		V
Flow control	V		
URL filter			V
Proxy status report	V	V*	V
模組外掛方式	分析 log	分析 log	URL 轉向

* Calamaris 各項統計較深入.

5 結論

爲使網路頻寬的使用更有效率，TANet 推行使用網路代理伺服器，藉由網頁的共用降低頻寬需求. 網路代理伺服器的確有效增加頻寬的利用率，若代理伺服器能對不同的需求提供不同的服務，則應能進一步提昇網路服務品質. 同時國內有數所大學的圖書館因使用者以網頁下載程式大量且連續下載電子期刊而被警告甚至停權，因此我們提出一個網路代理伺服器使用量控制系統，ProxyBreaker，可根據不同的需求，提供彈性的管制機制，同時可保障網路資源的公平使用，維護網路智慧財產權. 未來我們希望能繼續改善本系統使其能提供 differentiated service.

參考文獻

[1] http://anemone.electricc.com/

[2] http://gnu.sinica.edu.tw/manual/wget/html_mono/wget.html

[3] http://proxy.nctu.edu.tw

[4] http://proxy.nsysu.edu.tw

[5] http://proxy.ntu.edu.tw

[6] http://www.btsoftware.com/index.html

[7] http://www.squid-cache.org/Doc/FAQ/

[8] http://www.squidguard.org/

[9] “The TANet Traffic”, The Ministry of Education Computer Center Newsletter, No. 8807, pp. 83-114, July 1999.

[10] Abdelzaher, T.F. and Bhatti, N., “Web server QoS management by adaptive content delivery”, IWQoS '99: 1999 Seventh International Workshop on Quality of Service, pp.216 –225, 1999.

[11] Apache development group, Apache web server 1.31, http://www.apache.org.

[12] T. Berners-Lee, R. Fielding, H. Frystyk, HyperText Transfer Protocol HTTP/1.0, RFC 1945, IETF HTTP WG, May 1996.

[13] Cord Beermann, http://cord.de/tools/squid/calamaris/

[14] Chandra, S., Ellis, C.S. and Vahdat, A.,

"Differentiated multimedia Web services using quality aware transcoding", INFOCOM 2000, Vol. 2, pp. 961 –969, 2000.

[15] Xingping Chen and Prasant Mohapatra, "Providing Differentiated Service from an Internet Server", Proceedings of Eight International Conference on Computer Communications and Networks, pp. 214 –217, 1999.

[16] Ariel Cohen, Sampath Rangarajan and Navjot Singh, "Supporting Transparent Caching with Standard Proxy Caches", The 4th International Web Caching Workshop, 1999.

[17] Peter Danzig, "NetCache architecture and deployment", Computer Networks, Vol. 30, Issue 22-23, pp. 2081-2091, 1998.

[18] Peter Danzig, Mike Schwartz, and Richard Hall, "A Case for caching file objects inside Internetworks", 1993 ACM SIGCOMM, 1993.

[19] R. Fielding, J. Gettys, J. Mogul, H. Frystyk and T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2068, January 1997.

[20] Chris Foote, "Squirm – A Redirector for Squid", http://www.wenet.com.au/squirm/.

[21] FreeBSD, http://www.freebsd.org

[22] Martin Gleeson, http://martin.gleeson.com/pwebstats/

[23] A. Heddaya, "DynaCache: weaving caching into the Internet", The 3rd International Web Caching Workshop, Manchester, England, June 1998.

[24] Maciej Kozinski, http://www.geocities.com/maciej_kozinski/w3cache/squeezer.html

[25] P. Krishnan, Danny Raz and Yuval Shavitt, "Transparent En-Route Caching in WANs", The 4th International Web Caching Workshop, 1999.

[26] Ulana Legedza, John Guttag, "Using network-level support to improve cache routing", Computer Networks, Vol. 30, Issue 22-23, pp. 2193-2201, 1998.

[27] Mark Nottingham, http://www.mnot.net/scripting/python/WebLog/

[28] T. Oetiker, D. Rand, MRTG: Multi Router Traffic Grapher, http://ee-staff.ethz.ch/~oetiker/webtools/mrtg/mrtg.html.

[29] D. Perkins, E. McGinnis, Understanding SNMP MIBs, Prentice-Hall, Englewood Cliffs, NJ, 1997.

[30] David Ramahefason, http://casimir.easynet.fr/

[31] Squid Internet Object Cache, http://squid.mlanr.net/Squid

[32] Andrew S. Tanenbaum, Computer Networks, 3rd edition, Prentice Hall , Inc., USA, 1996.

[33] Jann-Perng Tseng and Huei-Huang Chen, "The Implementation and Evaluation of the Proxy Server's Access Control System", The Proceedings of TANet'99 Conference, 1999.

[34] D. Wessels and K. Claffy, "ICP and the Squid Web Cache", IEEE Journal on Selected Areas in Communications, Vol. 16, No. 3, pp. 345-357, 1998.

[35] B. Williams, "Transparent web caching solutions", The 3rd International Web Caching Workshop, Manchester, England, June 1998.