網頁伺服器服務可用性之偵測系統

李政賢　　包蒼龍　　黃立行

大同大學資訊工程學系

摘要

網路技術近年蓬勃發展，各種網路服務也應運而起，隨著網路越來越普及，意味著網路服務必須同時服務更多使用者。現今大多數網路服務伺服器已不再是單一伺服器對外提供服務，而是透過網路連結多台伺服器所建構成的系統，除了傳統觀測頻寬使用率、網路連通性之外，如何以更貼近使用者角度、更全面的方式來量測網路服務品質，是網路管理者需要面對的問題。因此，許多人提出了針對網路伺服器的一些可用性偵測與斷線回報機制，可以在伺服器發生問題的第一時間通知管理人員。

使用舊有的網路層（Network Layer）、傳輸層（Transport Layer）、以及 SWPM（Simply Web Traffic Meter）等技術，偵測 n-tier 架構的網路服務時，可能會發生偵測結果正常但其實被偵測的服務已經發生問題的狀況。有別於基於網路層和傳輸層的偵測機制，本文所提出之網路服務可用性偵測機制，使用了應用層（Application Layer）的偵測技術，能夠偵測 n-tier 架構的網路服務是否運作正常。

關鍵字：斷線回報、可用性偵測

1 前言

由於現今伺服器所能提供的服務越來越完善，且大多數已不再是單一伺服器對外提供服務，而是透過網路連結多部伺服器共同提供服務，這意味著如果伺服器本身或網路連線出了問題，使用者將無法正常使用服務。因此，許多人提出了針對網路伺服器的一些可用性偵測與斷線回報機制[1]，可以在伺服器發生問題的第一時間通知管理人員。

本文將提出一套方法，透過模擬使用者實際在瀏覽器上使用系統服務時，測量操作特定服務所耗費時間，得知系統的服務品質狀況，如此一來，不僅可以更精確貼近使用者感受，也能避免傳統偵測方式無法查覺應用層服務異常的情形，最終會把長時間量測的結果透過網頁方式呈現，讓管理者易於瞭解所管理伺服器的服務狀況。

下一節中，將先介紹常見的網路品質狀態量測方式，針對各種方式的優缺點做簡單的分析和比較。

2 相關的網路品質狀態量測方法

本節將說明相關的網路品質量測方法原理，以及這些方式可能面臨的問題或困難點。

2.1 基於 SNMP 的網路品質監控

許多網路設備都支援 SNMP（Simple Network Management Protocol）協定，網路管理人員只需要利用適當工具，便可自網路設備中取得所需的資訊，例如網路傳輸資料、封包傳遞數、封包錯誤數等。另外，只要在需要被監控的伺服上安裝 SNMP Agent，那麼，管理人員亦可以取得該伺服器的資訊，例如伺服器的處理器負載以及記憶體使用量[3]。

SNMP 雖然功能強大、使用方便，但在低階網路設備上仍可能不被支援，且若欲對伺服器進行監控，必須在每一台伺服器上都安裝 SNMP Agent，一旦管理的伺服器數量龐大，勢必會對管理人員造成困擾。另一方面，透過 SNMP 所取得的資訊，有時候並無法反應出真正的問題，例如，當網頁伺服器和資料庫伺服器的負載皆正常，但網頁程式存取資料庫時卻發生了問題，以致於網頁程式無法正常運作的狀況，就無法透過 SNMP 偵測出來。

此外，SNMP 在使用上必須在網路設備或伺服器上，啟用 SNMP 的服務功能，這也意味著，入侵者可能有機會透過 SNMP 強大的管理功能，對伺服器網路服務造成潛在的安全性風險。

2.2 Ping

Ping（Packet InterNet Grouper）是一種很簡單也易於使用來瞭解網路連接狀況的工具，主要是利用發送 ICMP Echo Request 封包，量測網路上出發地至目的地來回所需時間。但在實務上，不少網路設備對 ICMP 封包，會做加速或攔截阻絕的處理，主

要是因為隨著網路安全議題越來越受到重視，且 ICMP 往往淪為入侵者測試網路連通性的手段，基於安全性的考量，許多網路管理者選擇將設備對 ICMP 的回應關閉，這當然就使得 ping 變得無用武之地。

此外，ping 是在 Layer 3 網路層中進行，只能夠得知兩點之間網路封包傳送與接收是否順暢，如果 server 服務失效的問題是發生在更上層的 Layer4 或 Layer7，極有可能 ping 指令的偵測結果是正常的，但 server 卻無法正常建立連線。

2.3 TCP Socket

由於基於 Layer 3 的量測方式無法完全偵測出伺服器的服務異常狀況，為了解決這樣的問題，乃有利用開啟 TCP socket 方式來偵測網路或伺服器的狀態[1]，以避免伺服器本身的網路連線正常但卻無法提供服務的情況。主要的方式是利用建立 TCP socket 連線來對伺服器特定通訊埠進行測試，好處是不必為了不同的 TCP 服務撰寫不同的測試程式，除了減少開發程式的時間之外，同時也增加系統的通用性。

即便如此，當 TCP socket 連線成功建立時，只是表示對伺服器上的某個通訊埠連接正常，對於後續的網頁伺服器或後端資料庫的服務可用性，並無法保證服務正常，舉例來說，可能資料庫中的某個資料表因不明原因被鎖住，造成其他程式無法存取該資料表，對偵測程式來說，只要成功建立 TCP socket 程式就認為伺服器服務是正常上線，但對使用者來說，感受到的卻是網站出了問題。

2.4 SWPM（Simply Web Traffic Meter）

SWPM 是以 WWW 應用為基礎的網路品質自動化量測系統[2]，用以量測使用者從連接網站至首頁完全呈現所需花費時間。這套量測系統可連線到指定的網站，藉由取回網頁時間、解讀回應之 HTTP header 的狀態碼（ex:[200 OK] 、[404 Not Found]）、分析網頁內容、取回網頁所需其他元件，最後計算全程所需時間，作為網站品質量測的依據。好處是，不會有 ICMP 封包可能被阻絕的疑慮，再者因為 SWPM 系統是基於應用層上進行，當網站於應用層服務發生異常時，能即時反應給管理人員。

不過，這套系統僅能針對 WWW 網站首頁的整體呈現的傳輸時間進行自動化量測，遇到多功能整合性的網站服務，無法以合適的方式分析量測，例如需要連接到後端驗證資料庫進行身份認證、或是需要與使用者互動的網頁表單，這些網站載入首頁頁面是正常的，在 SWPM 系統上看起來也是正常的，但可能後端資料庫發生異常，導致使用者實際操作送出登入或表單資料失敗。

因此，為了解決此一問題，本文提出了另一種機制，同樣也是在應用層上進行，但可以針對網站各種服務進行自動化的操作與記錄花費時間。

3 系統架構與實作

本節將說明本文所提出之基於登入式系統為基礎之網路品質量測系統。主要是利用 PHP/cURL 與網頁伺服器做各種自動化互動並記錄花費時間，可以在網路應用層上模擬使用者操作系統服務時的實際狀況，抑或針對系統上特定服務功能進行監控與服務品質的量測。本文以 PHP/cURL 撰寫自動化登入流程，針對 emoodle 教學系統進行品質量測，利用 php 程式呼叫 curl 函式方式，在背景執行 php script，登入 moodle 系統量測並記錄回應時間。

3.1 PHP/cURL

為了更精準反應使用者實際使用網路及伺服器服務之狀況，並避免前節所提到目前常用量測方法的缺點，我們選擇了自由軟體 cURL[4]，對目標伺服器進行網路品質量測。cURL 支援許多常用的協定，例如 FTP、FTPS、HTTP、HTTPS、SCP、SFTP、TFTP、TELNET、DICT、LDAP、LDAPS、以及 FILE。由於 cURL 是基於應用層執行的程式，可以偵測出網路層正常但應用層服務異常的情況，對於大部份的網站服務，cURL 都可以針對不同需要監控的服務做自動化處理，達到長期量測服務品質的目的。

cURL 提供了獨立執行的 exe 檔，同時也可以和 PHP 結合，我們採取後者的方式-PHP/cURL。主要的原因是 PHP 除了可以撰寫網頁程式之外，亦可以 script 方式在背景執行，有助於針對不同服務開發各種偵測程式，以及長時間不間斷地進行量測工作。同時，透過 PHP 呼叫執行 cURL 函式，方便對接收到的資料進行解析處理，並可將結果直接存到資料庫中。

本文以登入 moodle 教學平臺為例，記錄自載入登入網頁至登入成功載入重導頁面所耗費時間，以達到偵測應用層服務的目的。

3.2 PHP/cURL 自動化登入量測程式設計

在設計量測程式前，當然必須先清楚整個系統服務的架構與流程，像是登入頁面的網址、頁面中各個需要用到的表單欄位名稱、登入成功會導到什麼網址、網頁是否有使用 cookie…等等。因為這些資訊對 cURL 來說，就是用來代替一位真正的使用者的眼睛、雙手，告訴 cURL 該怎麼來操作系統、如何判斷處理回傳的資訊。在這一個階段裡，我們先利用獨立的 cURL.exe 程式搭配適當參數，以人工方式針對目標伺服器進行測試與分析，得到確切的相關資訊之後，再把這些資訊設定改寫成 PHP/cURL script。圖 1 即為針對銘傳大學 moodle 系統首頁自動化登入量測程式設計的流程圖，圖 2~圖 4 則為 PHP/cURL 程式碼片段。

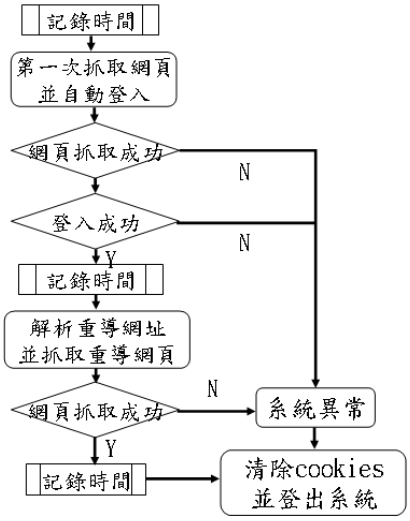

圖1　php/cURL 自動化登入流程

```
<?php

function L7_ping($url, $timeout) {

        //每次偵測都隨機產生一個cookie檔
        $cookie = mt_rand().".cookie";
        $total_time = $timeout;
        $dynamic_timeout = $timeout;

        //準備抓網頁
        $ch = curl_init($url);
        curl_setopt($ch, CURLOPT_POST, 1);
        curl_setopt($ch, CURLOPT_POSTFIELDS, "foo=test&bar=test2");
        curl_setopt($ch, CURLOPT_SSL_VERIFYPEER, FALSE);
        curl_setopt($ch, CURLOPT_RETURNTRANSFER, TRUE);
        curl_setopt($ch, CURLOPT_COOKIEJAR, $cookie);
        curl_setopt($ch, CURLOPT_COOKIEFILE, $cookie);
        curl_setopt($ch, CURLOPT_HEADER, 1);
        curl_setopt($ch, CURLOPT_TIMEOUT, $dynamic_timeout);
```

圖2　php/cURL 抓取網頁程式片段

圖 2 中的程式碼，預先將需要的資訊設定好，例如要抓取的網頁網址、測試用的登入帳號及密碼、模擬 cookies 暫存、設定 timeout 時間等資訊，以便執行抓取網頁的功能。

```
        //如果成功，http header 會是 303，且內含 redirect  的目的地
        if ($output != false && $info['http_code'] == 303 && $total_time < $timeout) {

                //解析要 redirect 到哪個 url
                $begin_tag = "location.replace('";
                $begin_position= strpos($output, $begin_tag) + strlen($begin_tag);
                $end_tag = "');";
                $end_position = strpos($output, "');");
                $length = $end_position - $begin_position;
                $url = substr($output, $begin_position, $length)."\n";
```

圖3　php/cURL 判斷登入是否成功程式片段

因為該網站會將使用者重導數次，因此程式必須解析每一次抓回的資訊，直到成功登入為止。圖 3 的程式碼片段是用來判斷是否登入成功的關鍵，首先針對第一次抓取回來的網頁進行 http header 內容解析，如果解析第一次登入後的 http code 數值為 303 即表示登入成功，同時因為我們在圖 2 程式碼第四行中設定了 timeout 時間為 20 秒，如果超過 20 秒才收到回應，我們仍判定為 timeout 登入失敗。此外，依據正常流程，若成功登入後系統會自動將使用者重導至另一頁面，因此必須同時符合這兩項條件，才能判定為成功登入；反之的任何情況皆視為登入失敗。

```
50              //準備抓下一個網頁
51              $ch = curl_init($url);
52              curl_setopt($ch, CURLOPT_SSL_VERIFYPEER, FALSE);
53              curl_setopt($ch, CURLOPT_RETURNTRANSFER, TRUE);
54              curl_setopt($ch, CURLOPT_COOKIEJAR, $cookie);
55              curl_setopt($ch, CURLOPT_COOKIEFILE, $cookie);
56              curl_setopt($ch, CURLOPT_HEADER, 1);
57              curl_setopt($ch, CURLOPT_TIMEOUT, $dynamic_timeout);
58
59              //抓網頁 -- 第二次 -- 進入登入成功後的第一個頁面
60              $output = curl_exec($ch);
61              $info = curl_getinfo($ch);
62              curl_close($ch);
63
64              //結算到目前共花了多少時間
65              list($end_usec, $end_sec) = explode(" ", microtime());
66              $end_time = ((float) $end_usec + (float) $end_sec);
67              $total_time = round($end_time - $start_time, 2);
68              $dynamic_timeout = ceil($timeout - $total_time);
69
70              //如果成功，http header 會是 200，且內含登出的 link
71              if($output != false && $info['http_code'] == 200 && $total_time < $timeout
72                      && (strpos($output, "http://test.url/logoff.php?session_key=") !== false)) {
```

圖4　php/cURL 登入成功抓取重導網頁程式片段

由於我們的目標是針對整個 moodle 的登入系統服務品質進行量測，因此必須再記錄登入成功後，載入重導網頁所花費的時間，這樣才是一般使用者完成系統登入系統所感受到的整個流程。因此，圖 4 的程式碼再次利用 cURL 抓取重導的網頁，並依據回傳的 http code 數值和網頁是否包含登出 link，來判定重導網頁是否可正常載入。若成功抓取重導網頁，則記錄下抓取成功的時間，至此即為整個登入流程所花費的總時間。本文即據此數據來評估 moodle 登入系統服務的網路品質。

4　量測結果與結論

PHP/cURL 是實際去模擬一個 client 向伺服器要求提供服務，透過這種方式，不再只能量測網頁載入時間，而可以在同一次的流程裡，同時針對 web server、網頁程式、後端 DB server 進行測試，比先前介紹的 SWPM 系統功能更加完整。和 SWPM 系統一樣，PHP/cURL 系統是在應用層上進行，針對被設定監控的服務，透過各種標準的通訊協定（本文採用 HTTP protocol）對伺服器進行量測，即使在網路層正常而應用層異常的情況，仍然可以察覺並發出警告。同時，也可以在 web server 服務正常但 DB server 失效的情況，發揮作用。

PHP/cURL 的自動化量測網頁伺服器方式，也彌補了單靠 SNMP 取得監控伺服器或網路設備資訊(如 CPU 使用率、記憶體使用率、網路頻寬使用率)，來判定伺服器服務是否正常的不足之處。與 ICMP 封包相較之下，亦不會有被管理者阻絕的問題，因為 PHP/cURL 量測系統服務的方式就如同使用者操作系統一般。

除此之外，雖然利用開啟 TCP socket 連線方式[1]偵測網路設備或伺服器的上線狀態，能夠偵測出網路層正常但機器本身無法提供服務的情況，但仍然無法保證整個網站在應用層的服務可用性，而 PHP/cURL 可以針對網站在應用層的服務是否正常提供服務，來進行自動化量測，可在使用者感受到網站出了問題時， 確保管理人員能在第一時間得知系統發生異常。

此外，為了方便管理人員能夠清楚知道 PHP/cURL 量測的結果，透過 Open Flash Chart 將量測結果繪製成如圖 5 的二十四小時監控圖。程式設定每 25 秒對 moodle 首頁執行一次自動化登入動作，並記錄所花費的時間。從圖 5 中可以很輕易地知道哪些時段系統服務變得較緩慢，或是被判定 timeout，而不再只限於 SNMP 所能提供的制式網路資訊。

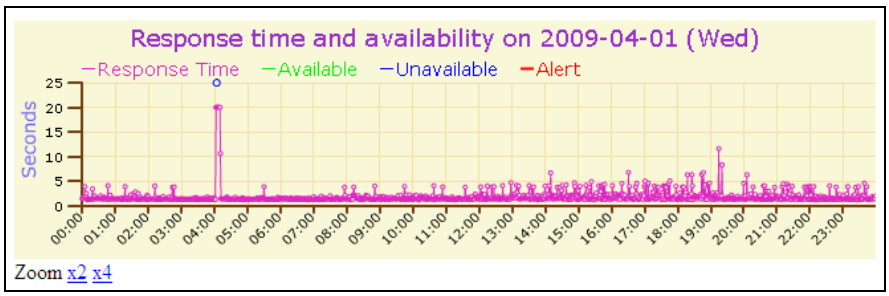

圖 5 登入 moodle 教學平台所花費時間之統計圖

5 未來工作

本文所提出的機制可以依照系統架構不同，針對不同的服務功能進行監控與量測，像是學生在 moodle 平台進行作業繳交功能、或是教師透過網頁上傳檔案的網路品質。此外，考慮到目前的網路服務失效或斷線警報發送機制，並沒有辦法判別伺服器是因為忙碌或是其他原因暫時無法提供服務，或是真的發生異常狀況。未來計劃將 PHP/cURL 量測所得到的結果，與同一時段內記錄 SNMP 所回報的伺服器資訊(CPU 使用率、記憶體使用率、網路頻寬使用率)結合，進行進一步的分析，期望能找出一合適的模式來作為系統是否真正失效的依據，進而使用在服務失效或斷線警報發送與否的機制上。

6 參考文獻

[1] 包蒼龍、黃立行、陳建伯，改良之網路及伺服器服務斷線偵測回報機制，TANET 2007。

[2] 賴守全、郭文曲，以應用層為基礎之網路品質量測，TANET 2006。

[3] 戴江淮、賴正延、姜玲鳳，網路斷線自動警報系統，TANET 2004。

[4] PHP: cURL - Manual, http://tw.php.net/curl