以非同步式呼叫設計並實作網路選課系統

Designing and Implementing Web-based Course-taking System based on Asynchronous Invocation

邱國光　　蘇建郡

南台科技大學資訊管理研究所

m9390107@webmail.stut.edu.tw　ccsu@mail.stut.edu.tw

摘要

在分散式物件架構的環境下，需要較久的時間來處理 Client 的請求有時是不可避免的。透過非同步式的呼叫(Asynchronous invocation)將 Client 和遠端服務的耦合度降低，對於會產生延遲的請求提供另一種解決方案。本文以網路選課系統為例探討非同步式呼叫不同的非同步樣式和其應用時機，以及檢視整合訊息服務(Messaging Services)應用在 Web Services 所產生的效果。

關鍵詞：Asynchronous invocation, Messaging Services, Web Services, Web-based Course-taking System.

Abstract

In the context of distributed object frameworks, it is inevitable in some cases that require longer time to deal with requests sent by client. By reducing coupling between client's request and remote process, asynchronous invocation is an alternative solution for latency yielded by requests. With Web-based Course-taking System, for example, this paper discussed different patterns of asynchronous invocation and its occasion and effects brought by integrating with messaging services applying to Web Services.

1. 前言

在一般的分散式物件架構的環境下，有時候會需要較長的時間來處理 Client 的請求，因為在某些情況下，後端系統在處理 Client 的請求會需要繁瑣的程序才可以完成，特別在系統的尖峰時刻處理類似的請求，延遲的現象是無法避免的。一般以同步式設計的系統，在 Client 送出請求至遠端的 Server，Server 在收到請求後會依據請求的內容決定其回應時間，在 Server 處理請求的過程中，以 Window based 的系統而言，其 UI Thread 會因為等候 Server 的回應而被鎖住[8]，致使 Client 的 UI 介面在等候的階段中無法回應使用者的動作，必須要等到 Server 回應之後，Client 才能取回控制權；在 Web based 的系統方面，雖然其 UI 不會被鎖住，但一般的設計大部份是等到 Server 回應之後，使用者才可以做下一個動作，所以若遠端服務處理的時間愈長，使用者等待要做下一個動作的時間也就會愈長。

延遲的問題更會發生在 Web Services 的技術上，特別是在 B2B 的環境[5]，雖然 Web Services[13]在整合異質平台系統的資源有不錯的能力，然而過多的文件描述(SOAP[12] ,WSDL)[11]等)和以純文字的傳送內容，使得執行 Web Services 的時間會比傳統的 RPC 還要長[9,15]。因此透過非同步式呼叫將 Client 的請求和遠端服務的處理程序分開，使得 Client 在送出請求之後，可以立刻取得應用程式的控制權，並且在遠端處理使用者之前的請求同時，使用者仍可以執行別的動作，且不會影響到遠端服務的程序，當遠端處理完之後，系統會主動通知道使用者處理的結果，或者由使用者自行決定何時查看處理結果，所以非同步式呼叫適合解決延遲的問題，進一步結合輕量通訊(Light-weight Communication)[4]的訊息服務機制，以確保資料傳遞的正確性，並且提升非同步式作業的穩定性[1,16]。

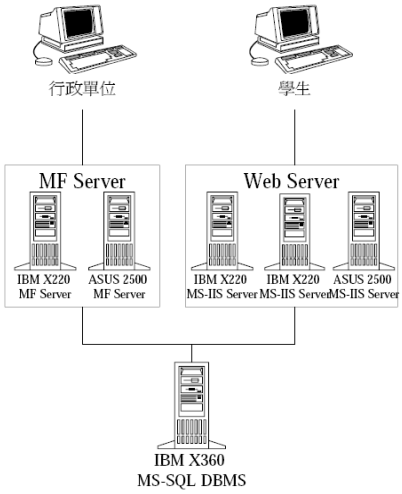

圖 1 學校之網路選課系統架構

學校的網路選課系統的部署方式如圖一[17]，在 Web Server 中有網路選課系統供學生上網選課使用，而 Meta Frame Server 有課務規劃的系統供工作人員使用，Database 則做為兩個前端系統的儲存設備。一般圖 1 的設計方式大多採用同步式的呼叫，假設學生送出選課清單至網路選課系統，網路選課

系統會將清單中的每門課執行數條規則判斷，判斷的過程中會需要相關的資料做為判斷依據，等到判斷完每筆資料後，Web Server 才會回應學生的選課結果。從送出選課清單至得到選課結果的期間，假設學生要查看清單中某門課程的詳細資料，勢必得中斷之前的請求，再送出一個新的請求，為要安全地完成之前的請求，學生必須等到之前的請求完成後，才能繼續做別的瀏覽動作，如果判斷規則愈多，Server 執行的時間也就會愈久，特別是網路選課開放的初期，延遲的現象很明顯[17]。

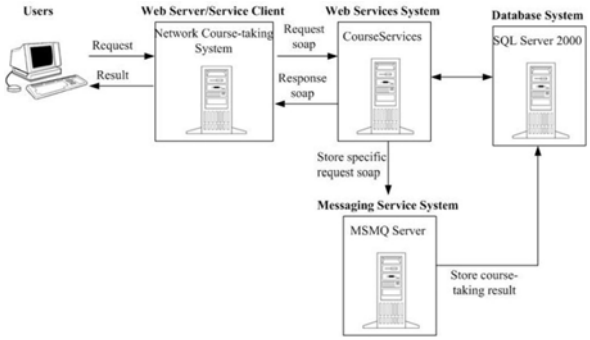

圖 2 本文建議之網路選課系統架構

圖 2 是本文建議的網路選課系統架構，其主要特色是使用 Web Services 跨平台的特性作為服務的介面，將來可以提供服務給不同的平台，如手機等行動裝置，該架構利用非同步式呼叫整合訊息服務機制加速處理使用者的請求以及儲存請求資訊。因此本文首先解釋非同步式作業以及各種非同步樣式的應用時機，接著會說明如何將訊息服務機制整合至非同步作呼叫應用在 Web Services 中，並說明本文的測試結果和建議，最後則為本文的結論。

2. 非同步式呼叫

2.1 非同步式呼叫運作原理

非同步式呼叫可以提供使用者一種非同步式的互動[2]，其互動的達成有如圖 3 所示，當 Primary Thread 執行到 AsyncMethod 時，AsyncMethod 呼叫 Remote Services 的某物件，送出請求後， Primary Thread 即可馬上離開 AsyncMethod 繼續往下執行別別的程式，在這同時，當 Server 收到請求後，系統會從 Thread Pool 取出 Worker Thread 來執行被呼叫的物件的工作，執行完後將結果回傳給 Service Client。以上的作業方式即為非同步式呼叫以 Multi-threading[4,6]配合 Thread Pooling[9]的機制呈現非同步式互動的功能。

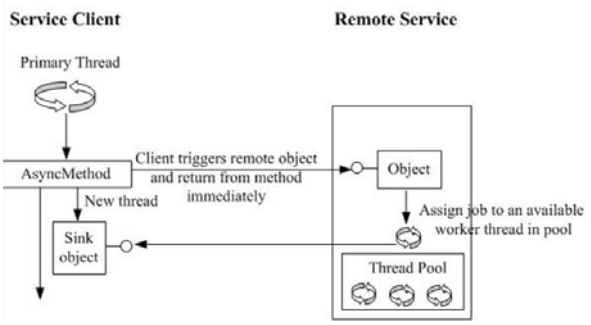

圖 3 非同步式呼叫示意圖

實務上開發者可以自行建立執行緒來執行非同步式作業，然而這種做法在面臨大量使用者同時呼叫時，會讓系統因為產生過多的執行緒而造系統運作不當[3]。所以依據非同步式呼叫的運作原理，原則上在開發的過程中，開發者應該盡量避免自建執行緒來實作非同步式作業。

2.2 使用非同步式呼叫的時機

OASIS 在[5]的報告中提到非同步式 Web Services 的設計是針對當等待回傳結果超過 60 秒，若僅由時間長短決定是否使用非同步式呼叫顯然過於狹隘，從系統開發者的角度而言，以下因素為使用非同步式呼叫的時機[2,4,9,16]：

- 執行的時間過長或無法預測時，
- 系統需要更高的穩定度時，
- 避免執行緒相關的程式寫法，
- 需要同時呼叫多個 Web Services。

從使用者的觀點，以下因素可作參考[4,7]：

- 提供 Client 的應用程式更好執行環境，
- 使用者對執行結果無急迫性的需要，
- 或者使用者不關切執行的結果。

2.3 非同步樣式

表一列出四種實作非同步樣式的特性。

表 1 非同步樣式

非同步樣式	回傳值	通知	資料取得
FIRE AND FORGET	無	否	-
SYNC WITH SERVER	無	是	-
POLLING	有	是	Client 決定
RESULT CALLBACK	有	是	由 Callback 物件通知

2.3.1 FIRE AND FORGET

FIRE AND FORGET[4,9,10]是四個樣式中最容易實作的，因為在 WSDL 文件中，在其描述服務的介面中可指定單向的函式(OneWay Function)，如程式碼 1，因此當使用者呼 Proxy Class 中被標示為 OneWay 的函式，在請求送出之後，Client 的應用程

式可以立即取得控制權，呼叫 FIRE AND FORGET 不會有傳回值，並且請求送出之後，抑不會通知使用者其執行的結果。

```
''' 將SendCourses的方面宣告成OneWay的呼叫
<WebMethod(), Services.Protocols.SoapDocumentMethod(OneWay:=True)> _
Public Sub SendCourses(ByVal courses As CourseCollection)
    Try
        MQHelper.SendCourseMessage(courses, courses.StudNo)
    Catch ex As Services.Protocols.SoapException
        Throw ex
    End Try
End Sub
```

程式碼 1　OneWay Function

FIRE AND FORGET 適合用於通知遠端或觸發遠端某事件，或者使用者不在意執行的結果。因為 FIRE AND FORGET 單向的特性，所以遠端在執行中發生的錯誤是不會回傳給 Client。

2.3.2 SYNC WITH SERVER

類似於 FIRE AND FORGET，SYNC WITH SERVER[9,10]加入通知使用者其請求的資訊已被遠端的服務接收，相同地，SYNC WITH SERVER 不會有回傳值。

2.3.3 POLL ING

```
''' 建立服務的實體
Dim proxy As New Asyn.CourseServices
''' 請求資訊(選修課程清單)
Dim courses As New Asyn.CourseCollection
''' 建立Polling物件，並以非同步呼叫遠端的服務
Dim result As IAsyncResult
result = proxy.BeginSendCourses(courses, Nothing, Nothing)
''' 檢查執行結果是否回傳
While Not result.IsCompleted
    '''當結果未回傳，則可做別的處理
End While
''' 當結果回傳則擷取之
Dim response As Boolean = proxy.EndSendCourses(result)
```

程式碼 2　POLLING

程式碼 2 說明 Client 端的應用程式可以利用 POLLING[3,9,10]的方式呼叫 Proxy Class 中某個以 Begin 開頭的函式，送出請求後，Proxy Class 會回傳 IAysncResult 物件，此時 Client 取得控制權，進而再使用 IAsyncResult 物件查詢執行的結果(圖 4)，結果回傳後，對應於 Begin 開頭的函式，呼叫以 End 為開頭函式即可取得資料。POLLING 在 Client 取控制權後，Client 可依實際需求決定其查詢時機。使用 POLLING 有兩個主要動作，分別是查詢執行結果，以及取得執行結果。

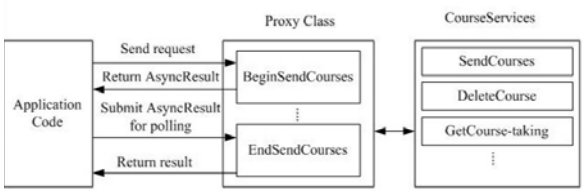

圖 4　POLLING PATTERN

2.3.4 RESULT CALLBACK

```
''' 建立服務的實體
Dim proxy As New Asyn.CourseServices
''' 請求資訊(選修課程清單)
Dim courses As New Asyn.CourseCollection
''' 建立Polling物件
Dim result As IAsyncResult
''' 指定擷取結果的事件程式
Dim callback As New AsyncCallback(AddressOf FetchResult)
''' 以同步式呼叫遠端的服務
result = proxy.BeginSendCourses(courses, callback, proxy)
```

程式碼 3　RESULT CALLBACK

```
''' 事件程式
Private Sub FetchResult(ByVal result As IAsyncResult)
    Dim proxy As Asyn.CourseServices = result.AsyncState
    ''' 擷取結果
    Dim response As Boolean = proxy.EndSendCourses(proxy)
End Sub
```

程式碼 4　Event Code

POLLING 和 RESULT CALLBACK[3,9,10](程式碼 3、4)最主要的差異在於 RESULT CALLBACK 會使用事件驅動(Event-driven)檢查回傳結果，所以 Client 送出請求之前，需先指派 AsyncCallBack 的處理事件，當送出請求後，Client 取得控制權，Client 會由指派的事件查詢執行結果是否回傳，最後再透由事件取得執行結果。RESULT CALLBACK 在請求送出後，透過事件驅動的方式查詢並擷取結果，所以可以利用事件驅動的特性主動通知 Client 取得執行結果。

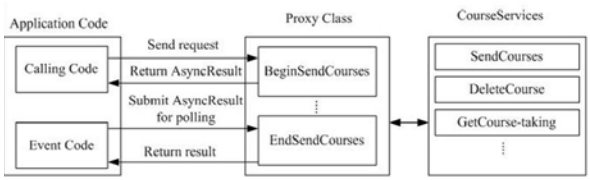

圖 5　RESULT CALLBACK PATTERN

2.4 使用非同步式呼叫的優劣分析

非同步式呼叫主要是針對當執行時間過長或者執行時間無法預測時，透由非同步式呼叫將 Client 的呼叫和遠端服務的處理分開，建立鬆偶合式(Loosely coupling)的架構[14]，使得使用者在送出請求後，即可取得控制權進行別的動作。實作非同步式呼叫可以(1)解決 Client 長時間等待的問題，透由不同的非同步樣式，(2)Client 可依需求選用不同的方式取得資料，(3)另外亦可免除 Client 直接撰寫執行緒。在缺點方面，除了 FIRE AND FORGET 和 SYNC WITH SERVER 外，(1)使用非同步式呼叫會提升撰寫的困難度[2]，由於非同步呼叫使用執行緒，(2)因此在偵錯上不好處理。

3. 整合訊息服務機制

表一所列的四種非同步樣式都必須在請求送

出後，且遠端的服務確實收到請求，不管是否會有回傳值，Client 此時才能夠取得控制權。假設遠端的服務在 Client 送出請求的那一刻非常忙碌，實際上 Client 還是要等，且等到遠端的服務可以接收請求為止；另外，由於非同步式呼叫是採鬆偶合式的架構，來回之間有一段時間，抑或者遠端的服務在收到 Client 的請求後發生故障，能夠持住使用者的請求資訊，並且以最有效率的方式儲存是很重要。所以建立可靠的訊息傳送[1]機制非但可以提升非同步作業的穩定性，亦可增加遠端服務處理 Client 請求的效能。

以微軟的 MSMQ(Microsoft Message Queuing Service)為例，其底層的運作採用非同步作業模式[4]，因此當遠端的服務收到 Client 端的請求時，服務僅只要將請求資訊儲存至 MSMQ Server 即可，如此可提升服務在短時間內處理大量的請求資訊，並且還可以透過 MSMQ Server 調節後端伺服器(例如：資料庫系統)的工作量[16]，另外，當後端的機器停擺時，MSMQ 亦支援離線作業，等到後端的機器回復正常，MSMQ 自動地會將訊息傳送至後端處理，MSMQ 預設是以 FIFO 的方式處理訊息，然而 MSMQ 允許使用者更改處理訊息的方式。因此，藉由訊息服務機制建立可靠的訊息傳送機制，在資料傳遞的正確性和穩定性提供不錯的架構。

4. 模擬之網路選課系統測試及架構說明

4.1 模擬網路選課系統之方式

網路選課系統在開放初期會接收到很多使用者送出的請求，使得系統在處理選課上會產生延遲的現象，因此本文模擬此情境，分別以不同的呼叫次數同時向圖 2 之 Web Server 發出請求，每一個請求均有選課清單資訊，包含了學號、課程代碼，以及學分，這些資訊會經由 CourseServices 先暫存在 MSMQ Server。在規則判斷方面，MSMQ Server 會有一支監看程式，當偵測到有請求資訊送至 MSMQ Server 時，監看程式會將請求資訊取出，並且擷取選課清單，開始執行規則判斷，判斷的結果會存到 SQL Server 2000，事後使用者可以查詢其結果，所以依據圖 2 的架構，本文實測之並檢視其效果。

4.2 測試結果及說

本文選擇同步式、CALLBACK 和 POLLING 三種樣式實施測試，測試目的主要是求出何種的搭配方式可以提供最好的服務品質，以做為服務的部署依據。測試方法是以不同的呼叫次數，求得 Client 可以在多短的時間內取得控制權，以及服務可以在多短的時間內儲存 Client 的請求資訊，時間的測量單位是毫秒，(1)表示在不同的呼叫次數下，Client 可在多短的時間內取得控制權的，(2)則為在不同的呼叫次數，服務可在多短的時間內儲存請求資訊。

$$\text{Response Time} = \sum_{\text{Invoke}_i=1}^{\text{Invoke}_i=N} \left(\text{EndInvokeTime}_i - \text{BeginInvokeTime}_i\right) \quad (1)$$

BeingInvokeTime：發出請求的時間
EndInvokeTime：發出請求後取得控制權的時間

$$\text{Process Time} = \sum_{\text{Process}_i=1}^{\text{Process}_i=N} \left(\text{EndProcessTime}_i - \text{BeginProcessTime}_i\right) \quad (2)$$

BeginProcessTime：儲存請求資訊的時間
EndProcessTime：儲存請求資訊的結束時間

為提供更快速的反應，本文比較同步式和非同式的反應時間，以決定 Client 使用何種方式呼叫服務，在儲存請求訊息方面，本文比較 MSMQ 和 SQL Server 2000 的儲存時間，以決定何種儲存體可以在最短的時間內儲存使用者的請求資訊，進而可以更快速地回應使用者其請求處理的狀態。

測試中使用兩個測試平台，第一個平台會以不同的同步模式和樣式送出不同次數的呼叫，以模擬多個使用者對服務同時呼叫，其硬體規格為 CPU P4 1.8GHz，642MB RAM，使用 Windows XP SP2 作業系統，並以 Visual Studio .NET 2003 搭配.NET Framework v1.1 做為測試工具；另一測試平台其 CPU 為 P3 500，512MB RAM，使用 Windows 2000 Professional SP4 作業系統，此平台有 Microsoft SQL Server 2000 資料庫系統，以及 Microsoft Message Queuing Service 2.0。

表 2 是對 SQL Server 2000 所測得的數據，表 3 則是對 MSMQ 的測試數據。CALLBACK 和 POLLING 欄位的兩個時間分別是 Client 可在多短的時間內取得控制權，以及服務可在多短的時間內儲存請求資訊，同步式的時間是包含使用者取得控制權和儲存 Client 的請求資訊的時間。

從兩張表來看，整體上 MSMQ 優於 SQL Server 2000，在回應的時間方面，非同步式優於同步式，然而在處理時間方面則相反，把 CALLBACK 和 POLLING 欄位的兩個時間相加和同步式比較，MSMQ 的同步式欄位會有最好的表現。

表 2 SQL Server 2000 之同步和非同步測試

呼叫次數	同步式	CALLBACK Response/Process	POLLING Response/Process
500	813ms	62ms/1016ms	220ms/9991ms
1000	1796ms	80ms/2604ms	330ms/13210ms
2000	3328ms	220ms/58786ms	1322ms/15603ms
3000	4765ms	310ms/8201ms	2003ms/18800ms
4000	6149ms	500ms/15091ms	2974ms/21153ms
5000	7641ms	591ms/19047ms	4346ms/22520ms

表 3　MSMQ 之同步和非同步測試

呼叫次數	同步式	CALLBACK Response/Process	POLLING Response/Process
500	120ms	50ms/421ms	91ms/1813ms
1000	250ms	90ms/891ms	240ms/2995ms
2000	491ms	170ms/1682ms	420ms/5857ms
3000	772ms	271ms/2554ms	531ms/9282ms
4000	1062ms	381ms/3455ms	711ms/12135ms
5000	1362ms	531ms/4516ms	841ms/14092ms

4.3 網路選課系統架構之建議

學校開放學生上網選課是有時間限制的，學生必須在開放的時間內完成選課的動作。一般網路選課系統會有很多的選課規則要判斷，在學生送出選修課程清單至網路選課系統，系統可能需要檢查是否超修、衝堂等規則，在判斷每條的規則中，需要不同的資料做為依據，如某門課程現有的選課人數、開課系別和班級等資料，若同時有大量的請求到達，以及網路選課系統要對每個請求要做很多的規則判斷，在這樣的情形之下，特別是在初期開放選課的段階，延遲的現象一定會發生的，當延遲發生時，學生必須等待 Server 的回應才可以做下一個動作。

4.4 系統架構比較

[17]提出幾個解決延遲問題的方案，例如登記抽籤、分配點數，或者是先選先取，這些解決方案大都是以改善選課作業流程為主，雖然可有效解決延遲的問題，然而在選課初期還是會有延遲的現象，即然延遲的現象會發生，本文所提之解決方案是從系統的架構面著手，著重於使用不同的技術和系統部署方式來解決延遲的問題，由於不是從作業流程面解決問題，本文之架構亦適合解決不同領域的延遲問題。

4.5 本文系統架構說明

由表 2 和表 3 測試的結果，本文決定以非同步式呼叫處理 Client 的請求動作，使 Client 的應用程式可以快速取得控制權，並且以同步式的方式使用 MSMQ 儲存使用者的請求資訊(學生選課清單)，以加速儲存使用者請求資訊。由於.NET Framework 不支援 SYNC WITH SERVER 的同步樣式，本文以 RESULT CALLBACK 樣式實作非同步式呼叫。

本文選用 Web Services 做為服務的介面是因其跨平台的特性，因應將來系統擴充的可能性和 3G 時代的來到，除了一般的 Desktop、Laptop 外，現在手機、PDA 上網的情形也漸漸普遍了，因此 Web Services 是預備未來可以服務不同的 Devices，另外，Web Services 本身亦支援非同步作業模式，所以，以 Web Services 實作非同步作業是很容易的，圖 2 中的 MSMQ Server 是儲存學生的選課清單，MSMQ Server 有一支監看程式，此程式預設每隔 5 秒會檢查是否有訊息送達，若連續 2 秒之內無偵測到訊息，則停止檢查動作。當有訊息送至 MSMQ Server 時，監看程式會將訊息取出，取出之後開始做選課規則判斷，所有判斷的結果，均會儲存至 MS SQL Server，事後 Service Client 可以取得結果。

表 4 是本文以圖 2 之架構模擬的網路選課系統測試結果，由於該架構難以測得儲存時間，因此僅有 Client 取得控制權的時間。比較表 3 和表 4 之 CALLBACK 欄位，表 4 整體上會比表 3 花較多的時間取得控制權，那是因為 SOAP Requet 比一般的請求的容量要大，所以才會造成此現象。圖 6、圖 7、圖 8、圖 9、圖 10 為本文模擬網路選課系統之操作畫面。

表 4　模擬之系統測試結果

呼叫次數	RESULT CALLBACK
500	78ms/-
1000	125ms/-
2000	148ms/-
3000	422ms/-
4000	500ms/-
5000	578ms/-

4.6 本文架構之優劣分析

在實際測試圖 2 之架構，很明顯地，該架構的確可以提供 Client 應用程式較好的執行環境，相較一般的同步式的系統，本系統可以在遠端服務處理之前的請求過程中，使用者仍可以做別的動作，例如查詢某課程現有的選課人數、開課老師等資訊，等到遠端的服務處理完結果之後，系統會自動通知使用者取得選課結果，由於 Web Services 的跨平台特性，其服務可以擴及到不同的平台環境，如手機、PDA 等系統，使得使用者有可以選擇不同的方式使用網路選課系統。

相較於圖 1 的架構，實作圖 2 架構的困難度會提高許多，那是由於使用到不同的技術，如 Web Services、訊息服務機制等，以及整合這些不同的技術，在偵錯方面也不易處理，因為不同的功能服務分散不同的機器，因此當有錯誤發生時，錯誤會很難追溯。

5. 結論

一般系統對延遲現象的解決方案就是讓使用者繼續等，非但造成使用者使用上不方更，也造成系統效能瓶頸，以及使系統更不穩定，本文以非同步式呼叫整合訊息服務機制，結合 Web Services 跨平台特性來解決網路選課系統的延遲問題，經實際測試以及模擬後，有效地解決延遲的問題，並且提

供 Client 應用程式更好的執行環境。為因應未來的需求，以 Web Services 做系統服務的介面，使得可以服務不同的系統平台，如手機、PDA 等，以達到行動運算的能力。

圖 7　網路選課系統—選取課程

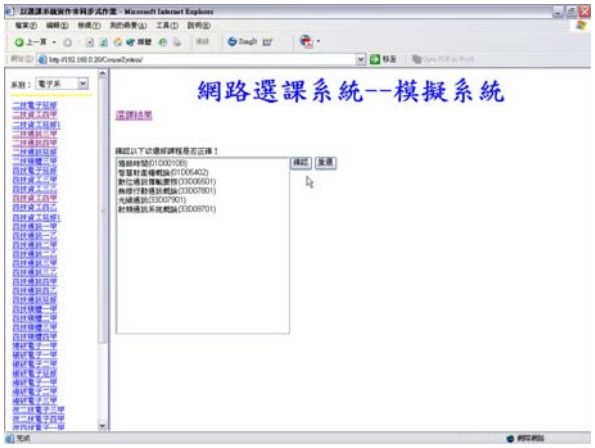

圖 8　網路選課系統—確認選取課程

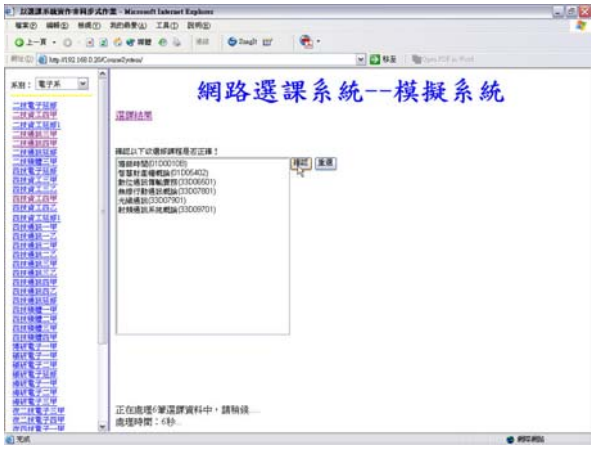

圖 9　網路選課系統—以非同步呼叫送出選取課程

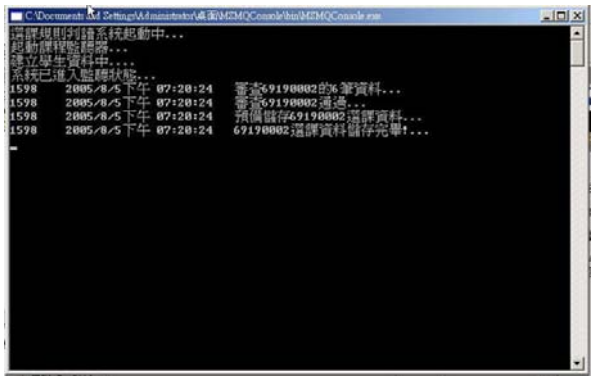

圖 10　網路選課系統—MSMQ Server 的監看畫面

參考文獻

[1] Jeffrey Hasan, Expert Service-Oriented Architecture in C#, Apress, 2004

[2] Macro Brambilla, Stefano Ceri, Mario Passamani and Alberto Riccio, "Managin Asynchronous Web Services Interactions", Proceeding of the IEEE International Conference on Web Services, pp.80-87, July 2004

[3] Matt Powell, "Asynchronous Web Services Calls over HTTP with the .NET Framework", MSDN Magazine, September, 2002

[4] Matthew MacDonald, Distributed Applications: Integrating XML Web Services and .NET Remoting, Microsoft Press, 2003.

[5] OASIS, "Asynchronous Transactions and Web Service", http://xml.coverpages.org/async.htm, October 2003.

[6] Richard Grims, ".NET Delegates: Making Asynchronous Mehothd Calls in the .NET Enviroment", MSDN Magazine, August 2001.

[7] Scott Seely, Eric A. Smith and Deon Schaffer, Creating and Consuming Web Services in Visual Basic, Addison Wesley, 2002.

[8] Ted Pattison, "Basic Instincts: Asynchronous Method Execution Using Delegate", MSDN Magazine, January 2004.

[9] Uwe Zdun, Markus Voelter and Michael Kircher, "Design and Implementation of an Asynchronous Invocation Framework for Web Services",The International Conference on Web Services - Europe 2003, Erfurt, Germany, September, 2003.

[10] Uwe Zdun, Micheal Kircher and Markus Volter, "Remoting Patterns: Design Reuse of Distributed Object Middleware Solutions", IEEE Internet Computing, Vol. 8, No. 6, pp.60-68.

[11] W3C, "Web Services Description Language(WSDL) 1.1", W3C Note, http://www.w3.org/TR/wsdl, Mar 2001.

[12] W3C, "Simple Object Access Protocol(SOAP) 1.1", W3C Note, http://www.w3.org/TR/2000/NOTE-SOAP-2000

0508/, May 2000.

[13] W3C, "Web Services Architecture", W3C Working Group Note, http://www.w3c,org/TR/ws-arch/#introduction, February 2004.

[14] 李昇暾、詹智安。JAVA Web Services 實務程式設計。旗標。2004。

[15] 周立德、程上傑。 “Web 服務的服務品質架構”。第十五屆物件導向技術及應用研討會。2004 九月。

[16] 彭靖灝。.NET Web Services 分散式應用程式設計。旗標。2004。

[17] 蘇建郡、蔡昇宏、劉毓芬。“由網路選課 LOG 資訊分析選課制度對系統負載之影響”。2003 台灣網際網路研討會。2003 十月。