行政院國家科學委員會補助專題研究計畫成果報告

※ ※※※※※※※※※※※※※※※※※※※※※※※
※
※ ※
※ 分散式多媒體物件導向應用程式架構 ※
※ ※
※ ※※※※※※※※※※※※※※※※※※※※※※※
※

計畫類別：¦個別型計畫　　□整合型計畫
計畫編號：NSC　89-2213-E-032-011-
執行期間：88 **年** 8 **月** 1 **日至** 89 **年** 7 **月** 31 **日**

計畫主持人：莊 淇 銘
共同主持人：王 英 宏

本成果報告包括以下應繳交之附件：
- **□赴國外出差或研習心得報告一份**
- **□赴大陸地區出差或研習心得報告一份**
- **□出席國際學術會議心得報告及發表之論文各一份**
- **□國際合作研究計畫國外研究報告書一份**

執行單位：淡江大學資訊工程學系

中　華　民　國 89 **年** 8 **月** 20 **日**

行政院國家科學委員會專題研究計畫成果報告

分散式多媒體物件導向應用程式架構

Distributed Multimedia Object-Orient Application Framework

計畫編號：NSC 89-2213-E-032-011-
執行期限：88 年 8 月 31 日至 89 年 7 月 31 日

主持人：莊淇銘　淡江大學資訊工程學系
共同主持人：王英宏　淡江大學資訊工程學系
計畫參與人員：林宏儒　淡江大學資訊工程研究所
計畫參與人員：簡嘉祐　淡江大學資訊工程研究所

一、中文摘要

由於多媒體電腦的發展和電腦網路的盛行，使得多媒體應用程式在各個軟體應用領域裡大量的增加。但是由於多媒體物件格式的複雜，加上分散式多媒體物件的管理、傳送、展示和同步等技術需求，使得整個軟體開發週期變長，軟體開發成本相對的增加。而其中有些程式碼其實是可以重複的使用，而不需要重新開發的。而應用程式架構的目的是在提供了一個應用程式的骨架，縮短應用程式開發時間。所以我們針對分散式多媒體的特性，設計了分散式多媒體應用程式架構(**DMOOAF**：Distributed Multimedia Object-Oriented Application Framework)**來解決這個問題。這篇論文使用了物件導向的技術，將目前一般的物件導向應用程式架構加以特殊化**(Specialize)**，並且加入了一些針對多媒體物件的特性所需要的一些類別階層**(Classes Hierarchy)**。並且使用了抽象類別以增加其跨平臺跟可移植性。為了達到多媒體高互動的特性，論文中使用了** SmallTalk-80 **所提出的** MVC(Model-View-Controller)**架構，來作為應用程式架構的基礎. 另外加入了多媒體展示**(Multimedia Presentation)**類別階層、多媒體同步**(Multimedia Synchronization)**類別階層、可調式服務品質**(Flexible Quality of Service)**類別階層等，使得開發多媒體應用程式的週期能縮短，並且具有增加可重用性、容易擴充等特性，以符合多媒體物件格式和展示方式的快速變化。**

關鍵詞：應用程式架構、多媒體、分散式多媒體

Abstract

Because of the prevailing of multimedia computer and the computer network, the multimedia applications are greatly applied in many areas. Due to the complication of multimedia objects and the demand of distributed technology for multimedia objects, the whole period of software development extends and the cost of software development equally increases. However, some program codes can be reused and doesn’t need re-coding. The purpose of application framework offers programmer in the skeleton of application and the shorter time of application developing. Therefore, we develop the Distributed Multimedia Object-Oriented Application Framework (DMOOAF) in order to solve the problem. The thesis quotes the object-oriented technology specialize the current object oriented application framework and add some class hierarchies according to the properties of multimedia objects. Furthermore, We use abstract classes to improve the across-platform and portability. To reach the high interaction of multimedia, we use Model View Controller provided by SmallTalk-80, as the basis architecture of application framework. In addition, we add multimedia presentation classes、 multimedia synchronization classes、Flexible Quality of Service classes, such that the cycle of software development can be reduced and have the properties of reusability and extensibility to suit the rapid changed of the multimedia object formats and the method of presentation.

Keywords: Application Framework, Multimedia,
Distributed Multimedia

二、緣由與目的

由於電腦網路的迅速發展和普及，網路應用程式的發展，已成為不可避免的趨勢。近幾年來，各式各樣的應用程式都朝向網路化、分散式來發展。其中最明顯的，莫過於多媒體應用程式。從最近全球資訊網(World Wide Web)**到** CSCW(Computer-Supported Cooperative Wok)**乃至於多人網路電腦遊戲，都和電腦網路有著密不可分的關係。因此，提供一個良好的分散式架構給程式設計師使用，也成為多媒體應用程式架構不可或缺的一部份。**

分散式物件導向多媒體應用程式架構的網路通訊部份，是採用 Windows Socket **來達成，我們將整個** Windows Socket **加以分類，並予以抽象化，使其成為一類別階**

層，變成應用多媒體應用程式架構的一部份。

三、多媒體物件導向應用程式架構

Socket最早是在Berkeley Unix中出現，做為TCP/IP的網路應用程式設計的介面(API：Application Programming Interface)。它是由四十六個函式(Functions)所組成。其中包括了轉換(Conversion)函式、資料庫(Database)函式、Socket函式和針對視窗特性增加的函式。由於後來其他的UNIX作業系統都採用BSD(Berkeley Unix Distributed)的標準來發展。所以，Socket也成為最通用的網路程式設計的介面。後來支援微軟Microsoft Windows作業系統，它的Windows Socket也就是Winsock也是出於BSD，並由各軟體廠商針對視窗的特性加以擴充。

Winsock API早期是用加入標準的DLL(Dynamic Linking Library)檔來達成，不同廠商所發展的Winsock都不盡相同，都是以配合自己發展的網路應用程式產品為主。後來Microsoft發表的一些作業系統(如：Windows 95)，才將網路的功能加入。因此，Microsoft的Winsock API才成為視窗環境底下，網路應用程式的標準。

為了因應不同的網路傳輸需求，Winsock主要分成兩類：Datagram及Stream。這兩個種類在Socket初始時就必須加以指定，並且不能加以修改。

TCP(Transmission Control Protocol)架構於IP之上，主要目的是提供傳輸無錯誤的大量資料。並提供錯誤的偵測、資料的排序和重傳等機制，以保證資料能完全無誤的抵達傳輸的目的地。TCP在傳送資料前會先建立連線，將所要傳送的資料切成小塊，再將每個資料小塊加上檢查碼和排序的資訊，將其包裹成封包(Packet)後，再透過IP傳遞，用戶端(Client)收到封包後，會根據封包內的資訊加以檢查，並且排序，若有錯誤則要求伺服端(Server)重傳。

UDP(User Datagram Protocol)的目的則主要在於用來傳送少量的資料，它只比IP多了一層錯誤檢查功能。UDP使用簡單的檢查方式來檢查送達的封包是否正確，對於錯誤的封包便直接丟棄，並不要求重送。而且也不能保證封包的次序有無錯誤或遺失。因此，UDP只能用來做簡單、「單向」的資料傳送，在收發資料前也不需要先建立連線。由於UDP比TCP少了一段等待連線的時間，還有一些繁複的檢查手續。因此，它的速度比TCP快，適用於網路應用程式做單一查詢，或是一些允許低正確性的網路傳送。

Port是一個整數值，由於TCP和UDP都透過IP將資料送到目的地，而IP將資料送到目的地之後，IP的責任就結束了。但是在多工的作業系統底下，我們還要指定資料是屬於哪一個應用程式的，Port的目的就是將TCP/IP或UDP/IP所送達的資料，交由所屬的網路應用程式。不論是透過TCP或UDP傳送，都必須配合著Port和IP位址，將資料送達正確且目的地唯一的應用程式。

在分散式環境裡面，客戶端(Client)與伺服端(Server)架構是最常使用的架構。當然還有點對點(Peer-to-peer)架構等。但在網路程式設計裡，「伺服端」的意義指的是等待「客戶端」連線，並提供服務的一個程式，而要求服務的程式則稱為「客戶端」。在一部機器上可以同時執行「客戶端」與「伺服端」程式，而任何一個程式也可以同時執行數個執行緒(Thread)，同時扮演「客戶端」與「伺服端」的角色。

Stream(TCP/IP) Socket的伺服端程式設計方式，是先由socket( )函式開啟，再用bind( )函式將它與本端位址結合在一起，結合完畢之後再用listen( )函式等待客戶端的要求連接，最後再以accept ( )函式表示接受連線。

Stream(TCP/IP) Socket的客戶端程式設計方式大致相同，先用socket( )開啟，再用connect( )指定IP及port要求連接伺服端。

Datagram(UDP/IP) Socket的連接方式與Stream Socket相近。Datagram Socket的伺服端程式設計方式，也是先由socket( )函式開啟，再用bind( )函式將它與本端位址結合在一起，結合完畢之後不需要用listen( )函式等待客戶端要求連接，就可以直接使用。Datagram Socket的客戶端程式，也不需要再用connect( )指定IP及port要求連接伺服端。所以，在sendto( )和recvform( )每一次傳送資料時，都必須指定目的地的位址。

在Windows的Winsock環境下，Socket可以切換成三種模式：Blocking、Non-Blocking和Asynchronous。

Blocking模式就是當程式呼叫一個blocking函式時，它必須等待回應，當收到

回應成功或失敗後，才能繼續下一個動作。在 Unix、Windows95 和 Windows NT 等優先權式（Preemptive,強制）多工作業系統底下並沒有太多的問題，可以使用多個執行緒(Multi-threads)來避免系統資源浪費在等待上。但是在像 Windows3.1 等非優先權式多工，則必須使用 select( )函式來預先做檢查。但是這種迴圈式、沒有效率的 polling 作法，也不是很好的解決方案。所以在 DMOOAF 中，我們並不提供這種方式。但是在 Socket 剛被建立起來時，它的初始模式是 blocking 模式。

Non-Blocking 模式和 Blocking 模式不一樣的地方是它並不會等待函式是否完成其條件，都會馬上自該函式返回，程式設計師可以從回傳值中判斷函式是否呼叫成功。所以這種模式底下就必須用一個迴圈不停的呼叫或判斷，直到函式執行成功或放棄嘗試為止。這種方式也浪費很多系統資源，缺乏效率，DMOOAF 並不採用。要使用 Non-Blocking 模式，可以透過 ioctlsocket( )函式呼叫，將 socket 設定成為 Non-blocking 模式。

Asynchronous 模式則是由作業系統處理函式的呼叫情況。當函式呼叫完成時，再透過作業系統通知程式。透過 WSAAsyncSelect( )我們可將 Socket 變成 Asynchronous 模式，並且指定在特定的事件發生時通知程式。同時我們也必須指定是由哪一個視窗接收系統訊息。這種方式也就是 Windows 所謂的事件驅動方式(Event-Driven)。這種方式比較有效率，而且不會浪費時間在等待上，對於真正的優先權式多工作業系統也是非常有效率的。

為了減少系統的資源使用與對於 Winsock 訊息的集中處理，網路應用程式在啟動時，就啟動一個隱藏視窗作為 Winsock 訊息控制中心(圖 1 WSAHWindow)。另外，必須要將藉由呼叫 GetLastError( )函式的類別（圖 1 WSAError）予以獨立出來，GetLastError( )函式是用來取得呼叫 Winsock 函式發生錯誤時的「最後一次」錯誤代碼。由於可能取得別的物件案例(Instance)的錯誤代碼，違反了物件導向的精神。

WSAWinSocket(圖 1)除了提供設定和釋放 Socket 的成員函式，並且定義了視窗訊息處理函式。而 WSAStream 和 WSADatagram(圖 1)則分別採用 TCP 與 UDP 連接，程式設計師繼承其類別後不可再更改。

WSAWinApp(圖 1)則是為了簡化網路應用程式的設計，所提供的一項類別。若一個應用程式同時使用 Client 或 Server，則可以將其定義為多重繼承。

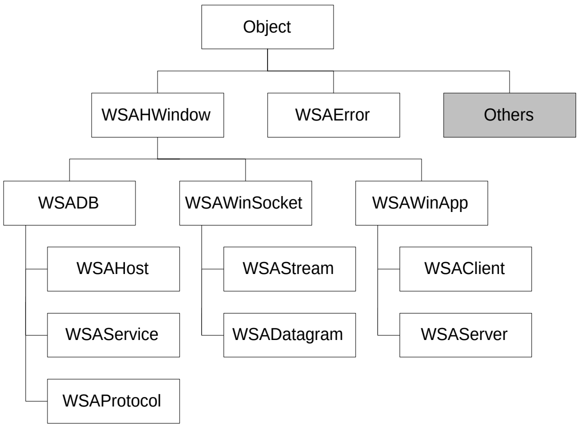

圖 1 Winsock 類別階層

我們針對在第一年計劃中所提出的 Connector 加以擴充，加入了 Network Connector (圖 2)。它本身使用了 Winsock 類別階層，再加以更近一步的封裝，只提供給 Component 間做為連接用。

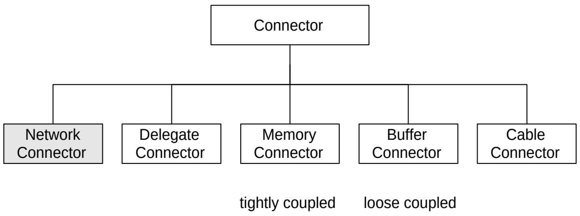

圖 2 Network Connector

加入了 Winsock Framework 和 Network Connector 使得能透過網路存取多媒體物件的功能之後，分散式多媒體物件導向應用程式架構才能顯得更為完備。