行政院國家科學委員會專題研究計畫成果報告
應用於無線環境下之遠端桌面控制系統

The Implementation of a Remote Desktop Mechanism

計畫編號：*NSC 94-2218-E-029-003*
執行期限：94 年 8 月 1 日至 95 年 7 月 31 日
主　持　人：蔡清欉　東海大學資訊工程與科學系
共同主持人：袁賢銘　交通大學資訊科學系
　　廖啟賢　東海大學資訊工程與科學系
計畫參與人員：張順欽　東海大學資訊工程與科學系
　　王聖凱　東海大學資訊工程與科學系
　　張貽勛　東海大學資訊工程與科學系
　　鄭明俊　交通大學資訊科學系

一、摘要

遠端桌面機制(remote desktop software)操作電腦的方便性和實用性，很多軟體已經被廠商實作在現有商用作業系統中，其中最著名的是Microsoft Windows Terminal Services 和 Virtual Networking Computing (VNC)。雖然遠端桌面軟體都有自己的不同實作方法，不過多數的遠端桌面機制多會有偵測螢幕改變、圖形擷取、圖形壓縮和資料傳輸等步驟，本計畫將透過無線網路的環境，實作出一套將用戶端的畫面利用遠端桌面機制傳輸到伺服器端顯示的遠端桌面軟體，如此一來便可達到用戶端電腦共用顯示設備而且節省硬體成本的目的。進而我們可以利用這種影像處理的特性，逆向的改用User Datagram Protocol(UDP)傳輸的特性，將影像廣播由一台電腦傳送到多台電腦，用軟體取代目前電腦教室中的硬體廣播系統。

關鍵字：遠端桌面機制、偵測螢幕改變、影像廣播、圖形擷取、圖形壓縮、資料傳輸

Abstract

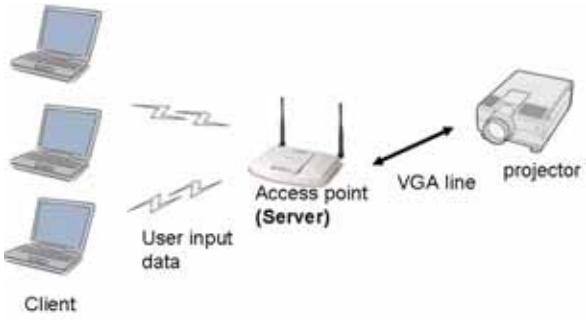
Due to convenience and practicality, many operating system vendors have implemented remote desktop software into their operating systems, the most famous being Microsoft Windows Terminal Services and cross-platform VNC (Virtual Networking Computing). Different remote desktop software has its own implementation method, but all of them have the following processes in common: screen-change detection, image capturing, compression, and transmission. In this report, we discuss the implementation methods of some remote desktop software and propose a novel remote desktop mechanism. Various implementation methods are compared and analyzed. Finally, a remote desktop software is developed based on our experimental results. Derived from characteristics of image processing, the software will be able to reverse User Datagram Protocol (UDP) transmission to broadcast images from a single computer to multiple computers, and will save the cost of broadcast hardware used in computer labs.

Keyword: remote desktop software、detecting screen changes、video broadcasting、capturing image、compression and transmission.

二、緣由與目的

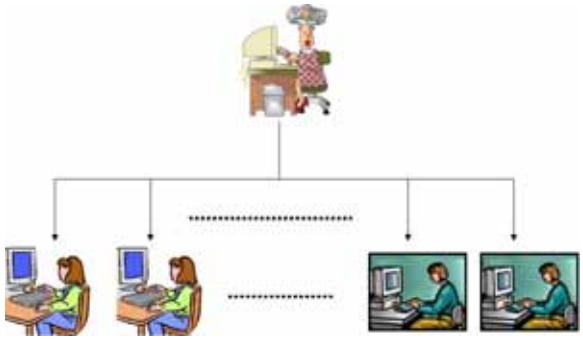
藉由網路連到遠端電腦，使用圖形化界面(而不是命令模式)在遠端電腦上執行更多樣與複雜的工作，例如：遠端網頁瀏覽、網管程式、繪圖、應用程式執行...等。此種遠端電腦執行方式跟 thin client computing 或 network computing 做法類似，所謂的thin client computing system或network computing的架構是由伺服器端和用戶端組成，且通常伺服器端的功能都比用戶端強大。然後透過網路的傳遞，用戶端把要執行的工作交給sever去執行，伺服器端執行完時再把結果傳回給用戶端。就以windows的遠端桌面就是屬於此種架構，其他的還有跨平台的VNC 等，它們各有自己的遠端桌面機制(remote desktop mechanism)。遠端桌面機制是用戶端連接到伺服器，然後由用戶端去操作伺服器端，伺服器端只是在畫面有所變動的時候，才把畫面變動的相關資料傳送給用戶端，然後由用戶端去更新畫面。使用此架構的好處有1)用戶端硬體需求不用很高，即可以執行CPU bound 或 IO bound 的應用程式(如做複雜的數學運算、資料的壓縮)，2)如果用戶端裝置很容易損壞或是資料保護安全性低(例如：機器受到撞擊或是遭竊)，此架構可提供穩定安全的機房，即不會有此顧慮。

依照上述的討論，我們有了一個新的想法，即用戶端畫面可以透過網路直接傳送到伺服器端上顯示，這樣做有什麼好處呢?以一般投影片報告為例，報告者通常需將自己的筆記型電腦帶到投影機旁，連接投影機的VGA線，因此每當要換另一個報告者時將轉換VGA線並且重新設定，所以如果我們能讓用戶端畫面透過無線網路的技術直接傳送到伺服器端上顯示，那麼將可以減少不必轉換VGA線並且重新設定，架構圖如圖一。



圖一 遠端投影架構圖

還有以教學廣播系統為例，教師在主控端裝有控制器，然後利用視訊分流器和每一個學生端連結起來，但此方法將必須要增加硬體成本以及佈線裝置時間，因此若能利用本計劃所提出的構想，讓用戶端畫面可以透過無線網路傳送到伺服器端上顯示的技術，那麼教學廣播系統中的電腦只需要安裝遠端桌面機制的軟體，就能把教師目前的電腦畫面透過無線網路廣播給學生的電腦，因而可以減少硬體成與佈線裝置以及維護時間，架構圖如圖二。



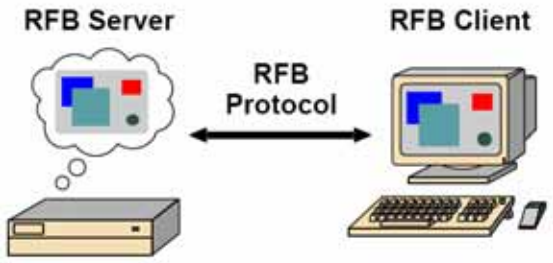
圖二 教學廣播系統架構圖

本計畫中我們探討目前幾個主流的遠端桌面軟體實做方式，並且實做出我們所架構的遠端桌面機制，系統包含三個子系統，分別為用戶端發送系統、伺服器接收與監控管理系統，其中伺服器端可接受用戶端的畫面，並即時顯示用戶端的改變。在此我們先探討1)跨平台Virtual network computing，2)MS windows RDP。

1. 跨平台Virtual network computing：

Virtual network computing (縮寫為VNC)為一套2台互動的遠端控制的軟體，一台是伺服器端程式，另一是viewer（用戶端）程式，允許單一桌面可以被多個人

同時存取。VNC運作原理是以所謂的RFB，即一個給遠端使用者來存取本地端圖形資料的協定，並利用RFB當伺服器端的畫面有所改變時(如圖三)。



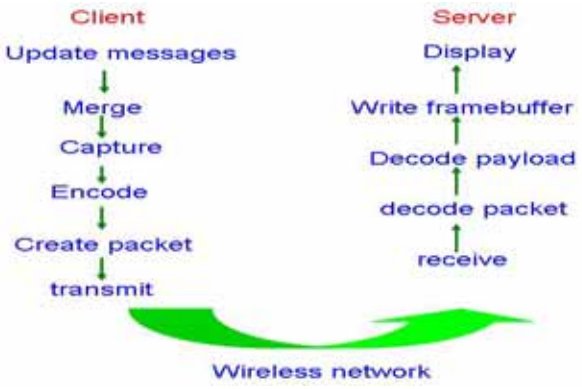
圖三 VNC傳輸原理架構圖

2. MS windows RDP

微軟終端服務是遠端桌面協定Remote Desktop Protocol (RDP)，RDP是依據ITU T-120這一系列的標準所制定的協定。ITU T.120標準為了多點的資料傳輸與即時通訊的應用而建立，該標準由一系列的多點即時通訊與應用協定所組成，並且在多媒體應用方面有不錯的效果，根據T.120標準所實作的產品，有資料的連結、傳輸、接收與合作等功能，例如：程式分享、白板會議和檔案傳輸。終端服務使用TCP通訊協定，通訊埠設定為3389，其運作流程為：在伺服器端，RDP使用自己的繪圖驅動程式處理影像的輸出，並且把影像的資訊分割成網路封包利用RDP的通訊協定傳送到用戶端。而在用戶端，RDP接收到影像資訊的封包後，解析成Microsoft Win32圖形設備介面(GDI)的API函式呼叫。RDP使用多種的技術來減少伺服器端與用戶端之間資料的傳輸量。包括資料壓縮與點陣圖的資料快取等等...。其中點陣圖的資料快取應用在使用了大量點陣圖程式時可以明顯的改善在低頻寬網路環境時的執行效率。

三、研究方法

本計劃主要分為兩個主要部份，首先我們實做遠端無線投影系統(如圖一)，接著我們延伸研究教學廣播系統(如圖二)，兩者在其遠端傳輸的演算技巧皆相同，唯一不同之處乃為傳輸的方式，前者使用TCP，後者則使用UDP。整個系統演算邏輯主要架構在遠端桌面的偵測與壓縮上，我們的基本架構上和一般的遠端呈現協定(remote display protocol)有所不同，一般的遠端呈現協定都是由使用者連到伺服器端，操作伺服器端，再由伺服器端把相關的資料傳給用戶端，由用戶端去顯示，而我們的遠端呈現協定流程剛好跟上述的相反，我們的架構是使用者操作自己的電腦，然後把相關的資料傳給伺服器端，由伺服器端去顯示，而我們所傳的資料僅只有用戶端有改變的部份。我們提出的遠端桌面流程圖如圖四。



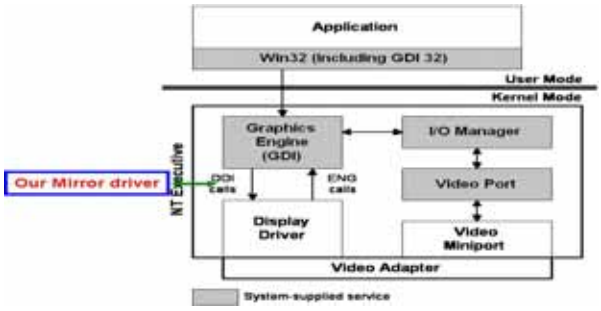
圖四 遠端桌面流程圖

這架構最重要的關鍵就是如何讓用戶端的畫面能即時的顯示到伺服器端。為了提供高品質的即時顯示，我們不使用VNC的被動畫面更新方式(Client pull)，而是使用Client push，這方面的傳輸技巧與RDP、Citrix metaframe等類似。即如果用戶端的畫面改變時，會在極短時間內就把更新的畫面傳給sever端顯示，而不是伺服器端在畫面顯示完後再去詢問用戶端是否有更新畫面，用戶端收到要求訊息時才去檢查是否有新訊息，如果有則去執行相對應的處理動作然後再把資料傳給伺服器端，如果沒有則等待新的更新訊息出現然後再把資料傳給伺服器端，就拿畫面更新頻繁的應用程式如：影片撥放程式來說，由於影片更新訊息產生非常的快，如果每次都要等伺服器端來詢問時才去處理傳送給伺服器端那麼將無法達到即時效果，所以使用此方法是非常沒效率的，除此之外

因為每次都要伺服器端傳送訊息來詢問，所以會增加頻寬浪費以及一半的封包來回時間。我們的系統並不像某些系統，以THINC為例，THINC有暫存區(buffer)的功能，其可能會因為網路突然壅塞或是其他原因，因此先將更新資料或指令先暫存起來，為了避免資料的重覆，一些額外的程式來管理此暫存區是必要的。例如：一些重覆的指令和過時的指令必須從暫存區刪除掉。而我們的方法則是當訊息送給伺服器端之後過了一段時間，計算出上一次給伺服器端訊息到目前螢幕有更新過的區域，使用此方法的好處是不需要額外的記憶體來儲存資料，也不須要定時的對暫存區做更新動作。

1. 偵測畫面變化(Update message)

在偵測畫面的改變方面，我們只傳送畫面改變的地方但不含指令，並且以Mirror Drive的方式來偵測與記錄變動畫面。在Windows 2000以後的系統提供一種稱為Mirror driver的裝置驅動程式，它可以將原本的DDI calls完全複製一份相同的需求到Mirror driver，因此Mirror driver可以得知原本的顯示驅動程式到底收到什麼樣的DDI calls。然後我們可以根據DDI calls的參數得知螢幕上那邊被改變。這種技術只會使用非常輕微的CPU資源，安裝後即可使用不用重開機。但只有Windows 2000以上的OS可以使用，我們目前是採用此種方式來偵測。其架構圖如圖五。



圖五 mirror driver架構圖

2. 抓取畫面(Capture)

當我們經由mirror driver得知畫面的更改區域時，接下來的步驟就是如何去把更改區域的畫面抓下來，然後做處理傳給用戶端去顯示出來。那要如果抓圖呢？我們利用GDI API 中的 Bitblt 指令來做畫面的拷貝。使用此方式連續抓取1024x768x16的畫面100次大概只要2.2秒左右，即每秒可抓大概近50次，不過其抓圖時間會隨著抓圖範圍和系統負載有所改變，但此種抓圖方式無法抓取到DirectDraw與Overlay的畫面，像滑鼠游標就無法抓到，但是由於mirror driver會把DirectDraw功能關掉而以GDI指令來繪圖，所以僅無法抓到Overlay的畫面。所以下一小節我們必須去偵測目前滑鼠位置。

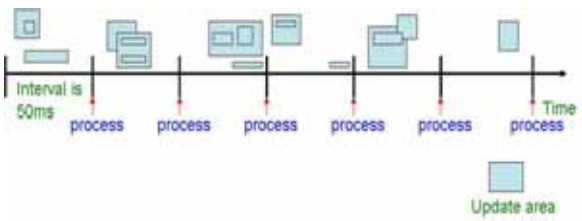
3. 處理滑鼠

對於滑鼠位置的偵測，我們採取polling的方式，此種方式為每隔一段時間去詢問一下目前的滑鼠位置，然後根據目前的取得的滑鼠位置去做相關處理。至於要隔多少時間去詢問呢? 由於人類的眼睛大概只要在每秒有23個frames下，就不會對影片有延遲感覺，而在我們的實驗結果下即使每秒做20次詢問滑鼠位置也不會有延遲感覺。

4. 畫面合併(merge)

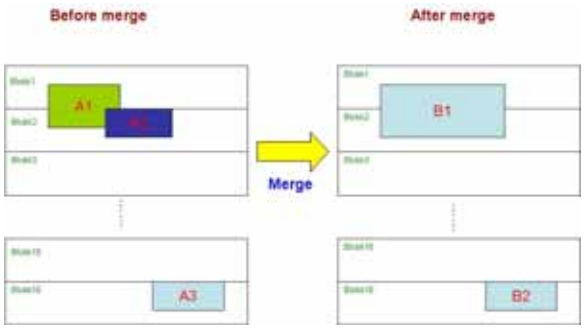
當我們得到畫面更新的訊息時，我們並不會立即把更新資料傳給用戶端，因為這麼做的話將會很耗頻寬和CPU資源，就以撥放投影片來說的話，我們每秒可能獲得將近100個更新的訊息，因此如果我們立即把更新資料傳給用戶端，那麼所需的頻寬和CPU資源是非常可觀的，所以我們還是根據人類的眼睛大概只要在每秒有23個frames下，就不會對影片有延遲感覺，我們以50ms為一間隔時間，在這50ms所得到的更新的訊息，將會被我們做合併的處理，以節省頻寬和CPU資源。

下面是我們依照時間的經過所會得到的更新訊息示意圖如圖六：



圖六 更新訊息示意圖

上圖只是一個簡單的示意圖，通常每50ms可能會有近百個更新訊息，而這些更新訊息有很多的訊息其所指的更新區域不是重複就是相鄰，因此我們必須要對這些更新訊息做一些合併的處理，以減少執行時間，在這裡的執行時間包含了抓圖時間、壓縮時間、傳送時間，而合併的目的主要是在減少抓圖時間。由於一般應用程式其介面都是方形且更新區域也都是由上而下，所以根據這些特性，我們把螢幕由上而下切成16等份的長條形，如果相鄰區域若有更新訊息，則會合併，如圖七。



圖七 切分區塊後的圖形更新示意圖

根據上圖，A1和A2為更新區域且A1和A2所在的區塊是相鄰的，所以我們把A1和A2合併為B1，至於A3因為其所在的區塊其相鄰的區塊沒有更新區域，所以不做合併的動作，因此我們只要對B1和B2這2個區域做處理即可。

5. 影像壓縮

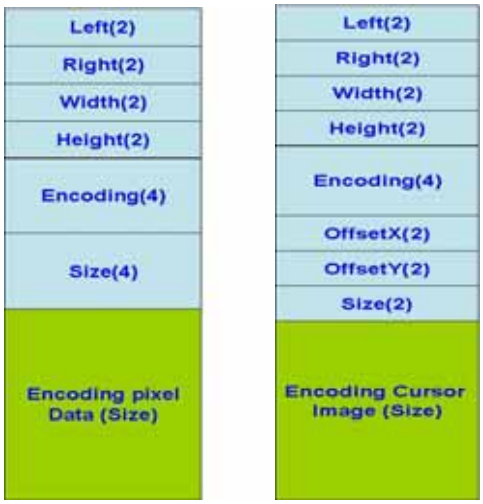
在做完抓圖動作後，接下來我們必須對我們所抓到的資料做壓縮的處理，而我們所用的壓縮方式為minilzo，而minilzo是lzo library的較不耗系統資源的版本，因為我們只用到壓縮和解壓縮並不需要其它額外的特殊功能，所以使用minilzo而不用lzo，且minilzo compile後的程式法只有5KB左右而且是用C寫的能夠跨平台。我們也根據了一些比較壓縮方法的實驗結果如：Archive Comparison Test (http：//compression.ca)和practical compressor test (http：//www.elis.ugent.be/~wheirman/compression/)，而選擇了minlzo，下面是Archive Comparison Test的實驗結果如表一。

表一 Comparison Test的實驗結果表

DOS/Windows Tests				
	Best Compression	Fastest Compression	Quickest Extraction	Best Overall
Text	ENTROPY 0.5	LZOP 1.00w	LZOP 1.00w	SBC 0.860
Executable	RK 1.04.1	LZOP 1.00w	LZOP 1.00w	RAR (Win32) 3.00b5
EXE Packer (Win32)	ASPACK 2.00.1	UPX 0.99.2w	N/A	ASPACK 2.00.1
EXE Packer (DOS)	UPX 0.99.2w	UPX 0.99.2w	EXEPACK 4.06	UPX 0.99.2w
Sound (WAV)	MONKEY'S AUDIO 3.96	LZOP 1.00w	LZOP 1.00w	MONKEY'S AUDIO 3.96
Graphics (TIF)	ERI32 2.1fix	LZOP 1.00w	LZOP 1.00w	ERI32 2.1fix
Calgary Corpus	RK 1.04.1	LZOP 1.00w	LZOP 1.00w	SBC 0.968
Canterbury Corpus	COMPRESSIA 0.96	LZOP 1.00w	LZOP 1.00w	SBC 0.968
Worms 2	RK 1.04.1	LZOP 1.00w	LZOP 1.00w	SBC 0.968

6. 資料編碼

當所抓到的資料壓縮完之後，接下來就是把資料傳送給伺服器端，讓伺服器端去把畫面顯示出來，為了讓伺服器端了解用戶端所傳資料的內容，我們使用了下面的傳輸格式(括號的部分為使用的位元組數)，下面是螢幕更新資料的封包格式以及滑鼠更新資料的封包格式如圖八。



(a) (b)

圖八 (a) 螢幕更新資料的封包格式,
(b) 滑鼠更新資料的封包格式

在一開始相同部分，總共有12個Bytes，前8個Bytes用來描述這個畫面更新的位置是發生在哪裡，Left代表該區域的左上角X座標，Top代表改區域的左上角Y座標，Width代表區域的寬度，Height代表區域的高度。後4個Bytes代表Payload部分的資料是採用何種編碼方法，和辨別是螢幕更新資料還是滑鼠更新資料。Left，Top，Width和Top個用2個Byte是因為螢幕解析度為1024*768，所以只要用2Byte即可

表示，而編碼用4Byte是為了以後的擴充性，至於Size方面，在Screen update的資料是用4Byte，因為螢幕解析度為1024*768，像素為16bits，所以至少要有1024*768*16的type大小來表示，在此為了memory alignment的方便我們使用4byte表示，滑鼠的Size大小為2byte，是滑鼠大小為32*32，像素為16bits，所以其大小為32*32*16，所以用2Byte表示即可，另外滑鼠還有多2個欄位OffsetX和OffsetY，是因為在Windows中，我們在得到滑鼠座標時，其真正影像開始之處並不是那個位置而是會有偏移情形，下面是我們抓取一般Windows預設的滑鼠外觀，其大小為32*32，與抓取立體銅色滑鼠，其大小為32*32如圖如圖九。

圖九 (a) 預設的滑鼠外觀,
(b) 抓取立體銅色滑鼠

由上面可以得知他們得起始位置是不一樣的，所以每次要去畫滑鼠時必須要再重定位，否則可能會因為滑鼠不同，顯示出來的位置也會有所不同。當伺服器端收到從用戶端傳送的資料時，伺服器端會根據Encoding欄位將Payload的部分編碼，然後將編碼後的資料先畫到我們的暫存區上，等到更新訊息一到再把暫存區中的資料拷貝到影像緩衝區上，這樣是為了避免每次收到新的資料時就把資料立即寫到影像緩衝區上而造成的閃爍情形。

7. 傳輸方式

本計劃實做兩個系統，其最大的分別就在此步驟。遠端無線投影系統使用TCP將壓縮後的資料傳輸到Server端，由於是一對一的傳輸，故採TCP保證資料的完整性。而研究教學廣播系統則因為一對多，所以使用UDP廣播的方式來傳輸影像壓縮資料，並針對UDP之封包遺失做特別的處理。不論是遠端無線投影系統或研究教學廣播系統，傳送到Server端的影像資料都需再經過解壓縮與重組再繪到螢幕上。以下將介紹兩種不同的傳輸方式。

(A)遠端無線投影系統（使用 TCP）

在我們的系統中，用戶端和伺服器端會建立2個連線，分別為control和data，在一開始時用戶端和伺服器端和先建立control connection來向伺服器端註冊連線，再由伺服器端通知用戶端可以傳送資料，等到要換人報告時伺服器端可以終止他的所有連線。因此control connection的功能有1)建立雙方溝通連線，2)建立資料連線，3)註冊，4)通知用戶端開始，5)結束用戶端。

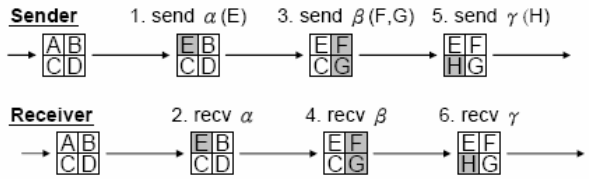
Control 連線是開始和結束才會用到的，大部份時間都是data連線，所謂的data連線是傳送更新資料用的，其資料內容如上所示，而Control連線和data連線所用得傳輸協定都是用TCP/IP，在Control 連線時其每次所傳的資料都是很重要的不可以被丟掉，因此必須用可信賴的傳輸協定而不能用udp，至於data連線雖然其資料並不是那麼重要(因為我們只要顯示最新的畫面，所以如果這次更新區域的資料被傳丟了，只要下一次又傳送了相同區域的更新資料時，那麼上次傳丟的資料就不重要了)，可是還是要用可信賴的傳輸協定不能用UDP，這是因為如果我們可以讓資料傳丟的話，那麼我們要能保證過不久之後會有相同更新區域的資料會被再一次傳送過來，否則那塊區域的資料將無法和用戶端同步，還有如果我們使用UDP，那麼所傳送的資料可能會無法按照順序抵達，因此如果有一個區域分別在時間T1和T2有被更新，且T1比T2的資料舊，若再傳輸過程中，T1比T2晚被伺服器端收到，此時T2可能已經被畫上去了，當伺服器端再收到T1時，伺服器端並不知此已經是一個過時的封包，所以伺服器端還是以一般封包處理方式處理它，因此我們反而得到一個舊的畫面。

當伺服器端收到資料時，做完編碼之後就是把資料顯示螢幕上，在此我們使用了linux 2.2 kernel 之後支援的frame buffer device，使用frame buffer device的好處是可以將硬體的細節隱藏起來，所以我們只要

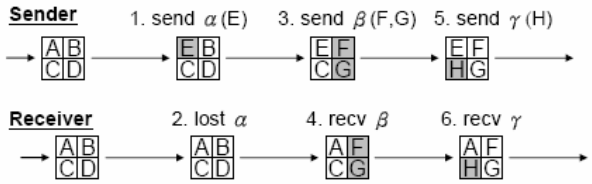
將資料寫到 frame buffer device即把影像顯示出來。

(B)研究教學廣播系統（使用 UDP）

這個系統修改了UDP的傳輸架構，主要原因有二：其一、UDP封包常因到達時間的先後次序關係，讓我們接到過期的、也就是非即時的影像封包。而且也常常造成封包遺失的現象。而系統並沒有重送的機制。其二、TCP並不適用在一對多的網路連線。當TCP用在一對多的通訊上，它必須付多與接收端同倍數的網路頻寬。而UDP也並不用在傳輸即時畫面的更新上。所以我們針對標準的UDP做了一些變動，我們稱它為MDP。

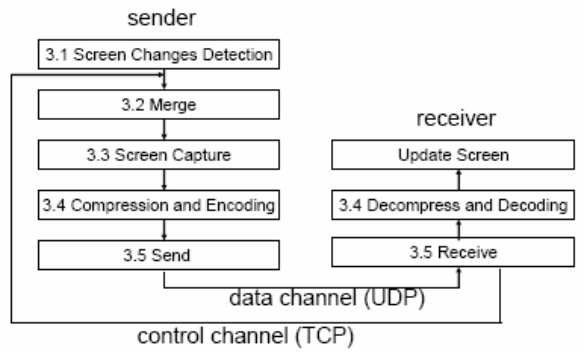


圖十 正常的封包傳輸與接收情況



圖十一 錯誤的封包傳輸與接收情況

首先上圖十為一標準的接收情況，傳送端送出的封包資料依序為ABCDEFGH，而接收端也依序接收為ABCDEFGH。圖十一為錯誤的接收情形，我們發現傳送端送出的封包資料依序為ABCDEFGH，而接收端也依序接收為ABCDFGH，遺失了E封包。為了解決這個問題，此時我們的MDP機制即發揮作用，我們利用control channel將遺失的封包資訊告訴傳送端，傳送端接收到接收端遺失的影像封包畫面時，他會去檢查到底是遺了那一塊，並將此遺失的部份記錄在mirror driver中，這樣一來，封包不用傳送，根據我們的演算法，這遺失的部份將重新被處理，待下回再傳送最新的畫面到接收端，以上就是我們的MDP演算法，其流程示意圖如圖十二。



圖十二 MDP 傳送流程示意圖

四、結語

我們在設計和實現一個在無線網路環境下的圖像呈現系統時，有許多必須去探討的相關問題。我們在製作這個系統的時候，我們也去探討了許多不同的方法，以及每個處理步驟當中的技術細節，並且解釋了如何從中去選擇一個最適合的方案來使用在我們的架構中。

在這個環節下，這個呈現系統的連線速度可以達到20 fps，這個速度已經足以應付大多數的工作，除了播放電影之外。

在這個計劃中，我們已成功應用我們的畫面偵測擷取演算法在一對一的無線投影系統與一對多的研究教學廣播系統中，接下來我們最大的目標就是減少資料傳輸的頻寬，讓整個投影與廣播更順暢。

五、參考文獻

[1].Ricardo A. Baratto, Jason Nieh, and Leo Kim: "THINC: A Remote Display Architecture for Thin-Client Computing." Technical Report CUCS-027-04,Department of Computer Science, Columbia University, July 2004.

[2] Bernd O. Christiansen, Klaus E. Schauser, Malte Munke: "A Novel Codec for Thin Client Computing." Data Compression Conference 2000: 13-22

[3] S. Jae Yang, Jason Nieh, Matt Selsky, and Nikhil Tiwari: "The Performance of Remote Display Mechanisms for Thin-Client Computing", Proceedings of the 2002 USENIX Annual Technical

Conference, Monterey, CA, June 10-15, 2002, pp. 131-146
[4] Albert Lai, Jason Nieh, Andrew Laine, and Justin Starren: "Remote Display Performance for Wireless Healthcare Computing", Proceedings of the Eleventh World Conference on Medical Informatics (Medinfo 2004), San Francisco, CA, September 7-11, 2004, pp. 1438-1442.
[5] Bernd O. Christiansen, Klaus E. Schauser: "Fast Motion Detection for Thin Client Compression." DCC 2002: 332-341
[6] Jason Nieh, S. Jae Yang, and Naomi Novik, "A Comparison of Thin-Client Computing Architectures", Technical Report CUCS-022-00, Department of Computer Science, Columbia University, November 2000.
[7] S. Jae Yang, Jason Nieh, Matt Selsky, and Nikhil Tiwari, "The Performance of Remote Display Mechanism for Thin-Client Computing", Proc. of the 2002 USENIX Annual Technical Conference, Monterey, CA, June 10-15, 2002, pp. 131-146
[8] H. Balakrishnan, V. N. Padmanabhan, S. Seshan, and R. H. Katz, “A Comparison of Mechanisms for Improving TCP Performance over Wireless Links”, IEEE/ACM Transactions on Networks, 1997.
[9] Tsun-Yu Hsiao, Ming-Chun Cheng, Hsin-Ta Chiao, Shyan-Ming Yuan, “FJM: A High Performance Java Message Library”, IEEE International Conference on Cluster Computing 2003, Hong Kong, Dec 1-4, 2003, pp.460-463
[10] Sair, S., Sherwood, T., Calder, B., “A decoupled predictor-directed stream prefetching architecture”, IEEE Transactions on Computers, Vol 52, Issue 3, pp.260-276
[11] Microsoft Windows DDK Document
[12] Archive Comparison Test (http://compression.ca)
[13] Practical compressor test (http://www.elis.ugent.be/~wheirman/compression/)
[14]VNC (http://www.uk.research.att.com/archive/vnc/)
[15] Microsoft Terminal Services (http://www.microsoft.com/)
[16] minilzo (http://www.oberhumer.com/opensource/lzo/)
[17] MSDN (http://msdn.microsoftcom)
[18] Microsoft DirectX 9.0c
[19] Micrsoft Windows Media Series SDK Document
[20] FramebufferDevice Programming Tutorial (http://www.sm5sxl.net/~mats/src/unix/graphics/fbtest/framebuffer.html)